ИСП
Институт Системного Программирования
им. В.П. Иванникова
Российской Академии наук

ISSN 2079-8156 (Print)
ISSN 2220-6426 (Online)

Труды Института Системного Программирования РАН

Proceedings of the Institute for System Programming of the RAS

Том 30, выпуск 6

Volume 30, issue 6

Москва 2018

ISSN 2079-8156 (Print), ISSN 2220-6426 (Online)

Труды Института системного программирования РАН

Proceedings of the Institute for System Programming of the RAS

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областях системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.

Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

- операционные системы;
- компиляторные технологии;
- базы данных и информационные системы;
- параллельные и распределенные системы;
- автоматизированная разработка программ;
- верификация, валидация и тестирование;
- статический и динамический анализ;
- защита и обеспечение безопасности ПО;
- компьютерные алгоритмы;
- искусственный интеллект.

Журнал издается по одному тому в год, шесть выпусков в каждом томе.

Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.

Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.

The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

- Operating Systems.
- Compiler Technology.
- Databases and Information Systems.
- Parallel and Distributed Systems.
- Software Engineering.
- Software Modeling and Design Tools.
- Verification, Validation, and Testing.
- Static and Dynamic Analysis.
- Software Safety and Security.
- Computer Algorithms.
- Artificial Intelligence.

The journal is published one volume per year, six issues in each volume.

Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

УДК004.45

Адрес: 109004, г. Москва, ул. А. Солженицына, дом 25.
Телефон: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Сайт: http://www.ispras.ru/proceedings/

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Web: http://www.ispras.ru/en/proceedings

Содержание

Table of Contents

Принципы построения межведомственного центра коллективного пользования общего назначения в модели программно-определяемого ЦОД

[1] *Б.М. Шабанов <shabanov@jscc.ru>*
[2] *О.И. Самоваров <samov@ispras.ru>*
[1] *ФГУ ФНЦ НИИСИ РАН,*
117218, Москва, Нахимовский просп., 36, к.1
[2] *Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. Центр обработки данных (ЦОД) является наиболее прогрессивной формой предоставления вычислительных ресурсов, когда необходимо обеспечить обслуживание широкого круга пользователей. В статье рассматривается один из подходов к созданию ЦОД – концепция программно-определяемой инфраструктуры. Программно-определяемой является такая инфраструктура ЦОД, в которой все ее ключевые элементы – вычислительные ресурсы, сеть, системы хранения данных и пр. виртуализованны и предоставляются пользователям как сервисы с заданным качеством. Показано, что реализация предлагаемой инфраструктуры позволяет обеспечить возможность каждому пользователю продуктивно решать свои задачи за приемлемое время с приемлемым уровнем затрат. В статье формулируются общие требования, предъявляемые к центрам обработки данных межведомственного уровня, предлагаются методы создания программно-определяемого ЦОД (развертывание вычислительных платформ, обеспечивающих максимальное переиспользование аппаратуры, обеспечение поддержки выполнения программ разных классов задач и пр.) и описываются реализации инфраструктур данного класса в конкретных проектах.

Ключевые слова: программно-определяемые ЦОД; облачные вычисления; распределенные вычисления; виртуализация

DOI: 10.15514/ISPRAS-2018-30(6)-1

Для цитирования: Шабанов Б.М., Самоваров О.И. Принципы построения межведомственного центра коллективного пользования общего назначения в модели программно-определяемого ЦОД. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 7-24. DOI: 10.15514/ISPRAS-2018-30(6)-1

1. Введение

В настоящее время наиболее эффективной формой предоставления вычислительных ресурсов для обслуживания широкого круга пользователей является центр обработки данных (ЦОД) [1].

ЦОД межведомственного уровня имеет следующие особенности:

- обслуживаются запросы на исполнение потока задач, относящихся к разным классам;
- допускаются запросы от пользователей разного уровня подготовки;
- класс задачи и уровень подготовки пользователя определяются в момент обращения в межведомственный ЦОД.

При этом к межведомственному ЦОД предъявляется дополнительное требование: по необходимости обеспечить каждого пользователя возможностью продуктивно решать свои задачи. Продуктивностью мы называем возможность качественного решения задачи за приемлемое время и на приемлемом уровне затрат.

Качество решения задачи с точки зрения предоставляемых возможностей включает удобный интерфейс, обеспечивающий возможность точно описать требования задачи к аппаратуре ЦОД, а также возможности ЦОД адекватно обеспечить эти требования.

Приемлемое время складывается из времени решения задачи, времени ожидания на организацию платформы по требованиям пользователя и времени предоставления доступа к платформе.

Приемлемый уровень затрат связан с эффективной формой организации ЦОД, когда аппаратура максимально переиспользуется при выполнении разного класса задач при соблюдении условия качественного решения.

Задача статьи состоит в том, чтобы конкретизировать общие требования в плане организации межведомственного ЦОД, а именно ответить на следующие вопросы:

- методы развертывания вычислительных платформ межведомственного ЦОД, обеспечивающие максимальное переиспользование аппаратуры;
- используемая аппаратура, способная эффективно обеспечить поддержку выполнения программа из классов задач;
- используемый стек системного программного обеспечения.

2. Общие требования межведомственного центра коллективного пользования общего назначения

Основные классы задач, на поддержку выполнения которых приходят запросы в ЦОД [2]:

- вычислительные задачи (MPI, CUDA, OpenMP);

- задачи искусственного интеллекта (Machine Learning);
- встроенные вычисления (Embedded Computing);
- задачи обработки графики;
- управление и анализ больших данных.

Определим общие требования, предъявляемые к центрам коллективного пользования межведомственного уровня. Центр коллективного пользования должен:

- обеспечить возможность поддержки решения максимального числа классов задач;
- обеспечить эффективное использование аппаратуры (консолидация вычислителей, систем хранения, каналов сетей, обеспечить возможность переиспользования одной аппаратуры под разные задачи, с учетом особых требований к системному окружению);
- обеспечить возможность быстрого доступа пользователя к аппаратуре, настроенной в соответствии с требованиями его задачи (доступ к аппаратуре на уровне готовых платформ, приложений)
- обеспечить возможность доступа к аппаратуре в сервисной модели (доступ по запросу)
- обеспечить заданный уровень качества обслуживания;
- обеспечить высокий уровень продуктивности;
- обеспечить безопасность на всех уровнях.

3. Концепция программно-определяемого ЦОД

Современным подходом к созданию вычислительной инфраструктуры ЦКП, который позволяет удовлетворить всем поставленным требованиям, является использование концепции программно-определяемого центра обработки данных [3], [4], [5].

Программно-определяемым является центр обработки данных в котором все элементы вычислительной инфраструктуры – сеть, системы хранения данных, вычислительные ресурсы, приложения и пр. виртуализованны и предоставляются как сервисы с заданным качеством.

Виртуализация является ключевой технологией, формирующей программно-определяемый центр обработки данных. Можно выделить три основных блока, из которых строится программно-определяемый ЦОД.

- Виртуализация серверов позволяет скрыть особенности аппаратной реализации конкретного физического сервера, предоставляя разным пользователям доступ к его вычислительным ресурсам (вычислительным ядрам процессора, оперативной памяти и пр.) в унифицированном виде, обеспечивая при этом полную изоляцию их вычислений.
- Виртуализация сети позволяет управлять сетевыми ресурсами путем

разделения доступной широты пропускания на независимые каналы, которые могут назначаться или переназначаться по запросам конкретным виртуальным серверам в реальном режиме времени.

- Виртуализация систем хранения данных позволяет сформировать по запросу из нескольких физических, объединенных сетью хранилищ, единое устройство хранения данных, которое может быть назначено конкретному виртуальному серверу.

На основе этих блоков строятся сервисы более высокого уровня – уровня специализированных платформ: виртуальный высокопроизводительный кластер (HPC-кластер), виртуальный кластер хранения и обработки больших данных (BigData-кластер), виртуальная платформа машинного обучения, виртуальная ферма визуализации и пр.

На базе виртуализированных вычислительных ресурсов и специализированных платформ могут быть развернуты приложения или прикладные информационные системы, предлагающие решения в конкретных прикладных областях: web-лаборатории инжиниринговых компаний, платформы анализа социальных сетей, платформы промышленного Интернета вещей и пр.

Одним из основных преимуществ такого подхода является полная автоматизация всех рутинных функций, связанных с операционным управлением ИТ-инфраструктурой за счет возможности реализовать специальный уровень, отвечающий за бизнес-логику управления виртуализированными ресурсами. Этот уровень обеспечивает трансляцию запросов на развертывание по заданным параметрам конкретных вычислительных сервисов программно-определяемого ЦОД и предоставление к ним доступа.

Такая организация ЦОД позволяет обеспечить высокую степень гибкости (переиспользование аппаратных средств для выполнения приложений с различными требованиями к настройкам системного окружения) по сравнению с традиционным подходом, а также использовать эластичность – возможность равномерно распределять текущую вычислительную нагрузку независимо от особенностей организации физической ИТ-инфраструктуры, модели облачных вычислений.

Высокий уровень масштабируемости программно-определяемого ЦОД позволяет прозрачно наращивать вычислительные ресурсы, системы хранения, каналы обмена данными по мере возрастания требований к их объему.

4. Архитектура программно-определяемого ЦОД

Архитектуру программно-определяемого ЦОД можно разделить на три логических уровня: уровень аппаратуры, уровень виртуализации, уровень управления.

4.1 Уровень аппаратуры

Уровень аппаратуры программно-определяемого ЦОД включает в себя оборудование (сервера, системы хранения, GPU ускорители, сети обмена данными и пр.), которое по своим характеристикам удовлетворяет требованиям классов задач конечных пользователей, решаемых в ЦОД. Такая аппаратура, как правило, состоит из широко распространенных компонентов общего назначения разных производителей, что значительно снижает общую стоимость ИТ-инфраструктуры и позволяет исключить риски, связанные с технологической зависимостью от единого поставщика.

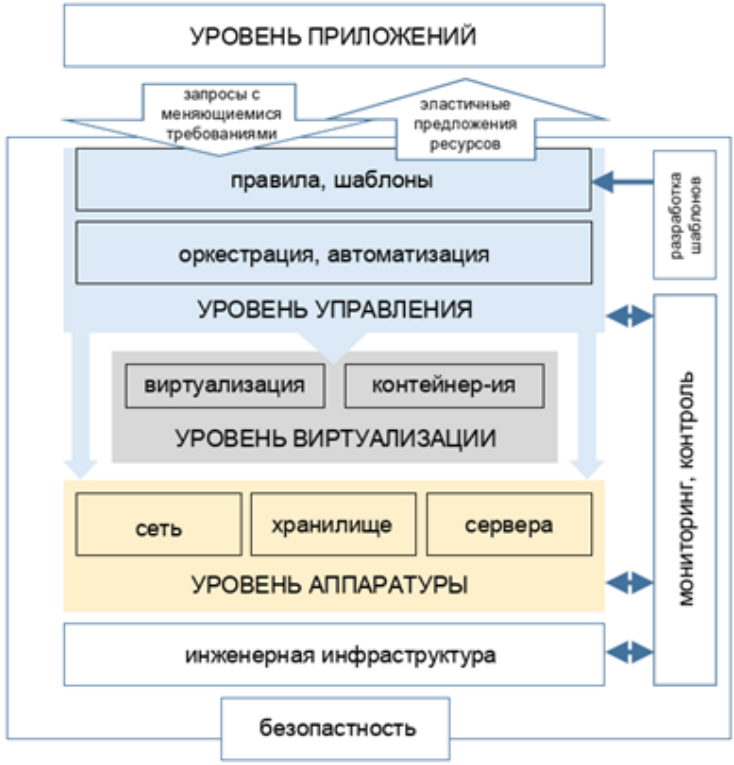


Рис. 1. Архитектура программно-определяемого ЦОД
Fig. 1. Software-defined data center architecture

В зависимости от классов решаемых задач отдельные серверы объединяются в Bigdata-кластеры, фермы визуализации, аппаратные платформы поддержки машинного обучения и пр. в соответствии с требованиями к базовой архитектуре данных вычислителей. Например, для решения задач анализа больших данных в модели in-memory в узлы BigData-кластера устанавливается большой объем оперативной памяти (от 512 GB).

Для поддержки решения задач класса искусственного интеллекта физический уровень ЦОД должен иметь в своем составе сервера с шиной обмена данным NVLink и графическими ускорителями типа GPU NVIDIA V100.

Компоненты хранения могут быть реализованы как SAN, NAS или DAS хранилища данных. В зависимости от класса решаемых задач в качестве физических устройств хранения могут использоваться как дисковые (HDD), так и твердотельные накопители (SSD) и их сочетание. Например, для решения задач, связанных со сбором и аналитикой больших данных, на серверах локально устанавливаются дисковые подсистемы обладающие значительным объемом (от 16 TB), производительные и надежные RAID-контроллеры, а для высокопроизводительных вычислений в области нефти и газа требуется высоконадежное (с глубиной репликации данных 3) распределенное файловое хранилище, обеспечивающее параллельный ввод-вывод и имеющее в своем составе как дисковые разделы общего назначения на накопителях тира NL-SAS, так и разделы, обладающие сверхпроизводительными характеристиками на основе накопителей SSD.

Сетевые компоненты программно-определяемого ЦОД включают физическое оборудование для обеспечения обмена данными между всеми серверами и системами хранения данных. Аппаратное обеспечение может включать в себя коммутаторы, маршрутизаторы, шлюзы и любые другие компоненты, необходимые для поддержки решения в ЦОД всех классов задач. Так, например, для поддержки высокопроизводительных вычислений узлы, входящие в состав HPC-кластера, должны быть дополнительно соединены высокоскоростной сетью Infiniband с низкой латентностью и задержками, а сервера Big-Data-кластера должны быть соединены сетью с высокой пропускной способностью (от 40 Gbs).

4.2 Уровень виртуализации

Уровень виртуализации является ключевым в организации программно-определяемого ЦОД. На данном уровне реализуется слой абстракции, отделяющий от физического сервера вычислительные ядра процессора и память, которые в дальнейшем предоставляются, как пулы логических вычислительных компонент. Таким образом, физический уровень аппаратуры скрыт от приложений, и для своего выполнения они используют исключительно виртуализованные процессоры и память.

Ядром уровня виртуализации является гипервизор – программа, реализующая одновременное, изолированное выполнение виртуализированных ресурсов на одной и той же аппаратуре. Гипервизор также обеспечивает перевод набора команд виртуальной машины в набор команд целевой аппаратуры.

Виртуализация хранилищ также абстрагирует основные физические устройства хранения, скрывая от приложений специфику базового оборудования и представляя по запросу доступное хранилище (блочное или объектное) в виде логического пула ресурсов.

Виртуализация сетей позволяет разделять физическую пропускную способность на независимые каналы, которые в дальнейшем можно назначать или переназначать, подключая виртуальные сервера для конкретных рабочих нагрузок в режиме реального времени.

В целом, уровень виртуализации делает возможным гибкое управление унифицированными ресурсами, что может быть использовано для автоматического развертывания по запросу сложных программно-аппаратных платформ для решения конкретных прикладных задач с учетом всех их требований.

4.3 Уровень управления

Кроме виртуализации вычислительных ресурсов, инфраструктура программно-определяемого ЦОД также включает в себя уровень, обеспечивающий оркестрацию (координацию взаимодействия нескольких элементов виртуализированной ИТ-инфраструктуры) и автоматизацию операций развертывания и управления сервисами.

Данный уровень обеспечивает централизованное управление всеми виртуализированными ресурсами программно-ориентированного ЦОД и может распределять эти ресурсы в соответствии с возникающими во время выполнения приложений рабочими нагрузками, переназначать ресурсы по мере изменения требований и пр. Данные функции реализуются через предоставляемые данным уровнем инструментальные средства.

Уровень управления также реализует возможности по проактивному мониторингу, оповещению и планированию выполняемых операций, по поддержке работоспособности сервисов всех уровней. Это дает администратором ЦОД возможность контролировать все операции по изменению конфигурации ИТ-инфраструктуры, следить за уровнем производительности сервисов, собирать и анализировать данные по использованию ресурсов.

Кроме того, уровень управления реализует бизнес-логику транслируя требования и запросы приложений в инструкции API, которые выполняют операции оркестровки и автоматизации. Через API-интерфейсы программный слой уровня управления формирует элементы ИТ-инфраструктуры, необходимые для выполнения приложений конечного пользователя в

соответствии с их требованиями, а также реализует политики соглашения об уровне обслуживания (SLA).

4.4 Функциональные и не функциональные требования

Вышесказанное обобщим в виде таблиц функциональные и нефункциональных требований, предъявляемых к реализации программно-определяемого ЦОД.

Табл. 1. Функциональные требования
Table 1. Functional requirements

Требование	**Описание**
Виртуализация	Реализация программно-определяемого ЦОД должна включать решения, обеспечивающие виртуализацию и контейнеризацию вычислительных ресурсов, систем хранения данных, сети.
Мониторинг, сбор и анализ событий, управление и планирование ресурсов	Непрерывный мониторинг работоспособности и загруженности облачной инфраструктуры, сбор и анализ случаев возникновения сбоев, планирование и управление перераспределением ресурсов.
Самообслуживание и оркестрация	Автоматизация управления (включение в каталог, развертывание, предоставление доступа, контроль) сервисами и виртуальными ресурсами программно-определяемого ЦОД.
Управление потоками заданий	Обеспечение возможностей автоматической организации и управления потоками заданий, при которой запросы на выполнение сервисов выполняются как бизнес-процессы в соответствии с набором процедурных правил, которые могут требовать подтверждений, модификаций, настроек ограничений доступа, делегирования и пр.
Администрирование облачной инфраструктурой	Обеспечение возможностей администрирования облачной инфраструктуры, которые включают прозрачное расширение ресурсов: добавление систем хранения, вычислителей, агрегированных каналов связи и пр.
Управление шаблонами виртуальных ресурсов	Обеспечение возможностей создания описаний/шаблонов виртуальных ресурсов, контроля версий, сравнения описаний, регистрации описаний в репозитории.
Управление шаблонами сервисов	Обеспечение возможностей создания сервисов/шаблонов сервисов, контроля версий, сравнения шаблонов, регистрации шаблонов в репозитории.
Контроль доступа	Обеспечение возможностей создания учетных записей пользователей, групп пользователей и назначения прав доступа к различным возможностям облачной инфраструктуры; создания разделов арендаторов

	ресурсов ЦОД и их администраторов, учетных записей разработчиков ресурсов и ученых записей конечных пользователей.
Миграция виртуальных машин	Средства миграция приложений, виртуальных машин и шаблонов с описаниями между облачными ЦОД, объединенными в федерации и публичными облачными инфраструктурами.
Миграция профилей безопасности	Средства миграции профилей безопасности описаний настроек ограничений сетевых сервисов (firewall) между облачными ЦОД, объединенными в федерации и публичными облачными инфраструктурами.
Масштабирование сетей Network Extension	Сохранение настроек свойств сети виртуальных машин (уровни L2, L3) при миграции между облачными ЦОД, объединенными в федерации, и публичными облачными инфраструктурами.
Управление каталогом ЦОД	Система управления и обмена каталогами описаний ресурсов, сервисов, правил, шаблонов между облачными ЦОД, объединенными в федерации, и публичными облачными инфраструктурами.

Табл. 2. Нефункциональные требования
Tab. 2. Non-functional requirements

Требование	**Описание**
Резервное копирование и восстановление	Система интегрального резервного копирования и восстановления
Простота инсталляции	Снижение сложности развертывания
Простота управления	Снижение сложности управления инфраструктурой и облачными приложениями
Технологическая безопасность	Снижение рисков технологической зависимости от единого поставщика решений
Масштабируемость	Прозрачное масштабирование компонент на всех уровнях с увеличением числа пользователей, сервисов, нагрузки
Безопасность	Реализация профилей безопасности данных пользователей
Надежность, доступность и удобство обслуживания	Высокая доступность и отказоустойчивость управления на всех уровнях ЦОД

5. Реализация модели программно определяемого ЦОД

Модель использования программно-определяемого ЦОД предусматривает варианты, представленные на рис. 2.

Управляющий ЦОД управляет развертыванием и конфигурированием программно-определяемого ЦОД, реализует политики безопасности и доступа к ресурсам, обеспечивает сбор и анализ данных по использованию ресурсов, отвечает за расширение аппаратных возможностей по хранению данных и вычислительных мощностей. *Управляющий ЦОД* предоставляет возможности инфраструктуры арендаторам ресурсов через создание разделов, управление которыми передает *администраторам арендаторов ресурсов*.

Администратор арендатора ресурсов управляет выделенными ему мощностями программно-определяемого ЦОД, настраивает права доступа.

Администратор сервиса разворачивает вычислительные сервисы на предоставленной ему инфраструктуре, следит за их работоспособностью.

Конечный пользователь использует сервис для решения своих прикладных задач. Разработчик сервиса используя возможности программно-определяемого ЦОД принимает участие в создании сервиса, его развитии, обеспечивает техническую поддержку конечных пользователей.

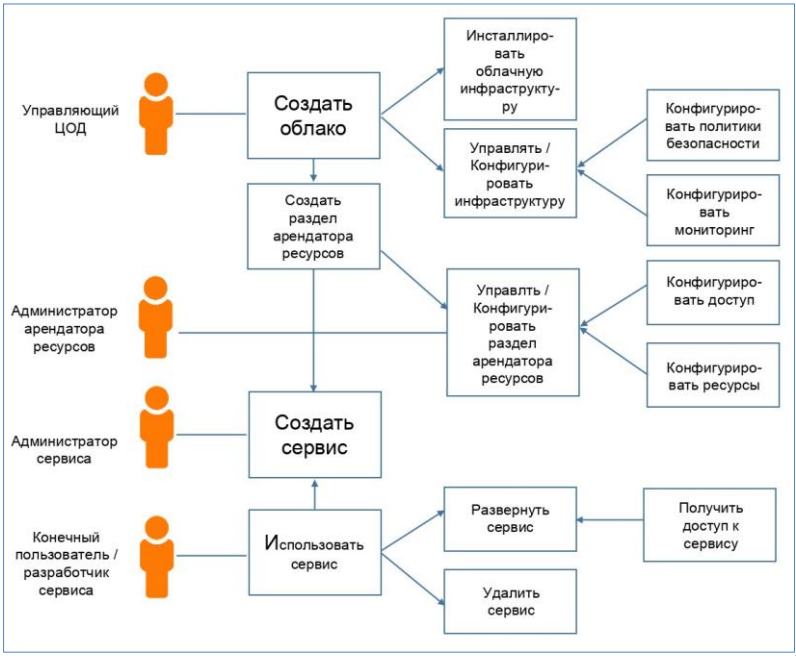


Рис. 2. Упрощенная модель использования программно-определяемого ЦОД
Fig. 2. Simplified model of using software-defined data center

5.1 Функциональная модель программно-определяемого ЦОД

Опишем функциональную, компонентную и операционную модели (см. рис. 3, 4, 5) программно-определяемого ЦОД на примере реализации сервис-ориенированного программно-аппаратного комплекса, направленного на решение широкого класса задач.

Модуль параллельных вычислений (домен HPC) предназначен для выполнения высокопроизводительных программ с поддержкой парадигм MPI, OpenMPI и представляет собой высокопроизводительную вычислительную кластерную систему.

Модуль виртуализации (облачный домен) представляет собой среду облачных вычислений и предназначен для поддержки решения следующих задач:

- предоставление унифицированных вычислительных виртуализированных ресурсов (regular zone, bigdata zone) – виртуальные машины, виртуальные блочные и объектные хранилища, виртуальные сети, виртуальные кластеры хранения и анализа больших данных, виртуальные кластеры среды iot.
- предоставление унифицированных контейнеризированных ресурсов (ml zone, vdi zone) – контейнеры вычислений Machine Learning, контейнеры вычислений CUDA, контейнеры визуализации результатов научных расчетов, контейнеры виртуальных рабочих столов.

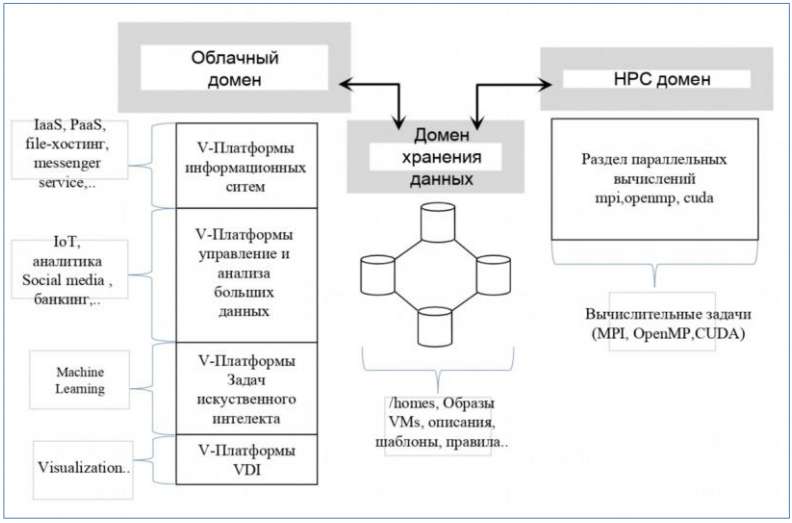


Рис. 3. Упрощенная функциональная модель программно-определяемого ЦОД
Fig. 3. Simplified functional model of software-defined data center

Модуль долгосрочного хранения данных (storage domain) обеспечивает надежное, эффективное долгосрочное хранение данных пользователей. Реализован на базе распределенной файловой системы Gluster.

5.2 Компонентная модель программно-определяемого ЦОД

Программный стек программно-определяемого ЦОД формируется на базе свободных стандартов и программных технологий индустриального качества с открытым исходным кодом, что в совокупности позволяет обеспечивать высокую гибкость, технологическую независимость и безопасность конечного решения.

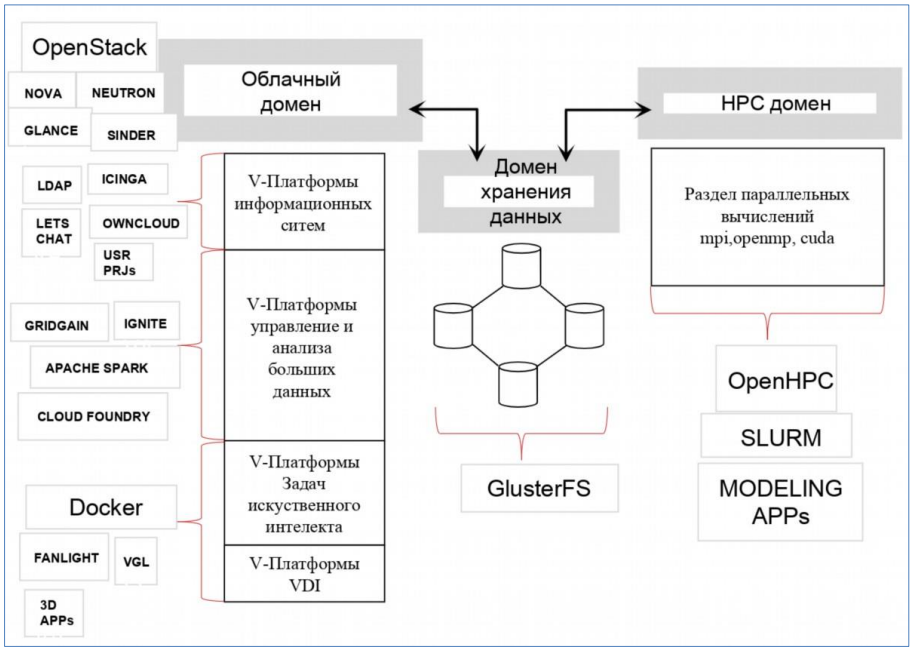


Рис. 4. Упрощенная компонентная модель программно-определяемого ЦОД
Fig. 4. Simplified component model of software-defined data center

В качестве основы управления и оркестрации виртуальных ресурсов программно-определяемого ЦОД может быть взят открытый пакет OpenStack. Виртуальные вычислительные ресурсы реализует компонент Nova (управление вычислительным облаком). Виртуализацию сетевых ресурсов обеспечивает компонент Quantum. Виртуализацию аппаратуры обеспечивают гипервизоры: XenServer/XCP, KVM/QEMU, LXC, ESX. Для случаев, когда требуется нативный доступ к реальной аппаратуре, используется драйвер OpenStack, реализующий метод BMD. Автоматизацию развёртывания и

управления приложениями в рамках изолированных контейнеров, реализует компонент Docker.

Виртуализованное хранилище реализуют пакет Gluster и компонент Cinder (виртуальные блочные устройства хранения) пакета OpenStack. Объектное хранилище реализовано с помощью компонента Glance пакета OpenStack. Функции аутентификации, авторизации конечных пользователей, разработчиков, администраторов арендаторов ресурсов реализованы с помощью компонента Keystone, входящего в состав пакета OpenStack. Интерфейс, обеспечивающий доступ к функциям ЦОД, реализует компонент Horizon (компонент пакета OpenStack).

Вычислительные платформы, предназначенные для решения прикладных задач, также реализуются с использованием СПО. Так, в качестве системного программного обеспечения HPC-кластера используется операционная система Linux CentOS и стек утилит из набора OpenHPC (http://openhpc.community/). Обеспечивается развертывание вычислительных узлов с помощью единого образа, поддерживаются централизованное управление, администрирование и мониторинг. В качестве планировщика задач используется пакет SLURM (поддержка пакетного и интерактивного режима запуска приложений, гибкое планирование и аккаунтинг ресурсов и пр.)

Приложения и платформы для решения пользовательских задач реализуются на основе IaaS как сервисы более высокого уровня (PaaS, SaaS) с использованием соответствующих платформ и SDK (ignite, gridgain, apache sparck, cloud foundry и др.). Номенклатура таких сервисов может быть легко расширена. Сервисы уровня IaaS используются как основа для развертывания информационных систем.

5.3 Операционная модель программно-определяемого ЦОД

Аппаратура высокопроизводительной вычислительной системы (hpc-кластер) состоит из управляющего CTR_NODE и вычислительных узлов COMPUT_NODE, сети управления CTR_NET на базе технологии GigEthernet), вычислительной сети COMPUT_NET на базе технологии Infiniband. HPC-кластер интегрирован с модулем долгосрочного хранения данных с помощью сети хранения (10 GigEthernet).

Совокупная мощность однородного вычислительного поля составляет 1440 вычислителей.

Для виртуальной среды OpenStack используются 4 сервера, обеспечивающие управление и общесистемные функции OS_CORE, и 12 серверов, формирующих виртуальные вычислительные ресурсы OS_COPMUTE. Для контейнеризированной среды Docker используются 2 сервера с графическими ускорителями типа NVIDIA V100 и NVIDIA GRID K2.

Контейнеризированная среда предназначена для приложений, эффективность выполнения которых чувствительная к виртуализации, случаев

предоставления прямого доступа к аппаратуре, выполнения тяжелых графических приложений, требующих использования ускорителей для отображения 3D сцен и т.д. Хранилище данных имеет объем сырого дискового пространства в 92 TB в разделе SSD. Все аппаратные модули ЦОД соединены с системой долгосрочного хранения по сети хранения, реализованной на базе 10 Gigabit bEtnernet.

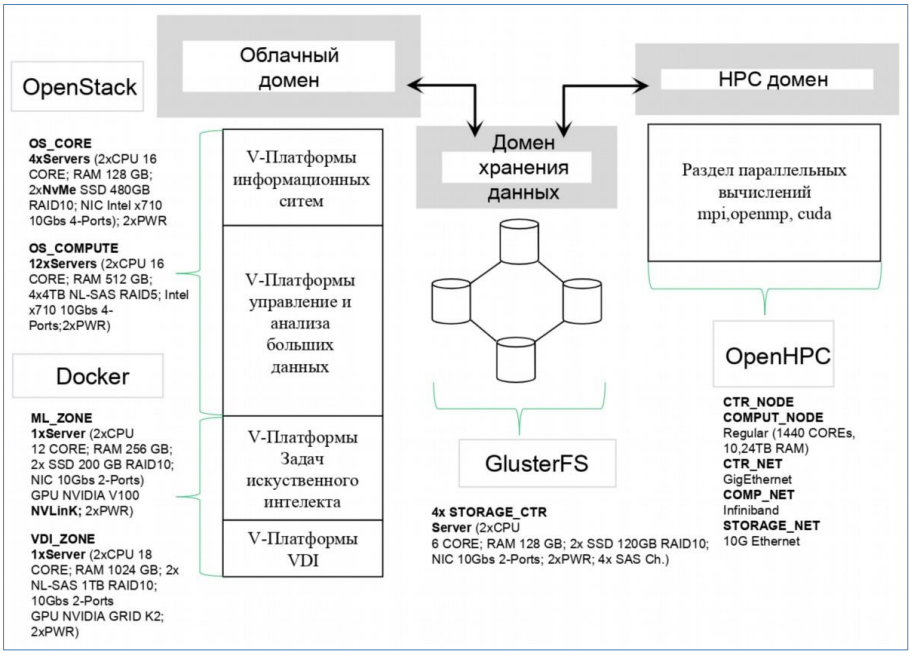


Рис. 5. Упрощенная операционная модель программно-определяемого ЦОД
Fig. 5. Simplified operating model of software-defined data center

6. Заключение

Описанный подход был успешно внедрен при реализации ряда проектов.

- Вычислительная инфраструктура программы «Университетский кластер» – совместная программа ИСП РАН, МСЦ РАН и компании HP, направленная на поддержку параллельных и распределенных вычислений [6]. На базе описанной концепции создана вычислительная инфраструктура с помощью, которой участники программы выполняют научные исследования, промышленные разработки, реализуют образовательные проекты по обучению студентов и аспирантов. Инфраструктура предназначена для организации на ее основе

масштабируемых облачных сервисов и предметно-ориентированных виртуальных лабораторий, доступных широкому кругу пользователей. Среди сервисов, создаваемых в рамках программы, можно выделить следующие:

1) обеспечение выполнения параллельных программ на высокопроизводительных вычислительных системах с распределенной или общей памятью;
2) разработка программных продуктов: средства управления проектом, система отслеживания ошибок и управления версиями;
3) поддержка проведения конференций, лекций, семинаров, лабораторных работ в режиме онлайн.

- Испытательный стенд международного проекта OpenCirrus, в рамках которого проводятся исследовательские работы в области разработки стека системного программного обеспечения для облачных сред. Центрами компетенции проекта стали ИСП РАН и МСЦ РАН [7].
- Проблемно ориентированная web-лаборатория решения задач механики сплошных сред UniCFD [8], [9], [10]. Web-лаборатория UniCFD продемонстрировала свою высокую эффективность при выполнении исследовательских и индустриальных проектов, а также при решении образовательных программ. Так, на базе web-лаборатории UniCFD разработан ряд учебных курсов («Основы использования свободных пакетов SALOME, OpenFOAM и ParaView при решении задач МСС»; «Расширенные возможности пакета OpenFOAM»; учебный трек «Пакет OpenFOAM – платформа для решения задач МСС»; «Использование свободных пакетов для создания расчётных сеток в задачах МСС»). Учебные курсы и лабораторные работы для обучения студентов проводятся на кафедрах ФН2, СМ2, СМ3 МГТУ им. Н.Э.Баумана. С 2014 года прошли обучение более 1000 слушателей из более 70 образовательных и научно-исследовательских учреждений и промышленных предприятий.

Список литературы

[1]. Joseph M. Hellerstein. Programming a Parallel Future. EECS Department University of California, Berkeley, Technical Report No. UCB/EECS-2008-144, November 7, 2008

[2]. Krste Asanović, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams and Katherine A. Yelick. The Landscape of Parallel Computing Research: A View from Berkeley. EECS Department University of California, Berkeley, Technical Report No. UCB/EECS-2006-183 December 18, 2006

[3]. Mike Perks. Reference Architecture: VMware Software Defined Data Center. 15 August 2018 [Электронный ресурс], страница LENOVOPRESS, режим

доступа: https://lenovopress.com/lp0661-reference-architecture-vmware-software-defined-data-center-thinksystem, дата обращения 20.10.2018

[4]. FUJITSU White Paper [Электронный ресурс]. Software-Defined Data Center – infrastructure for enterprise digital transformation . Режим доступа: https://sp.ts.fujitsu.com/dmsp/Publications/public/wp-sddc-infrastructure-for-enterprise-digital-transformation-ww-en.pdf, дата обращения 20.10.2018

[5]. EMC Reference Architecture [Электронный ресурс]. Federation Software-Defined Data Center, Foundation Infrastructure Reference Architecture. Режим доступа: https://www.emc.com/collateral/TechnicalDocument/h13378-evp-sddc-ra.pdf, дата обращения 20.10.2018

[6]. Арутюн Аветисян, Сергей Гайсарян, Виктор Иванников, Олег Самоваров. «Университетский кластер»: интеграция образования, науки и индустрии". Открытые системы. СУБД, № 05, 2010

[7]. Арутюн Аветисян, Олег Самоваров, Сергей Гайсарян, Эшсоу Хашба. OpenCirrus, российский сегмент. Открытые системы. СУБД, № 05, 2011

[8]. M. Kraposhin, O. Samovarov, S. Strijhak. Web Laboratory UniHUB in the Scope of Program «University Cluster». In Proc. of the Workshop «Multiphysical Modelling in OpenFOAM», Riga, October 20-21, 2011

[9]. М.В. Крапошин, О.И. Самоваров, С.В. Стрижак. Особенности реализации Web-лаборатории механики сплошной среды на базе технологической платформы программы «Университетский Кластер»». Труды международной суперкомпьютерной конференции с элементами научной школы для молодежи «Научный сервис в сети Интернет: экзафлопсное будущее», 2011, стр. 473-475

[10]. Крапошин М.В., Самоваров О.И., Стрижак С.В. Опыт использования СПО для проведения расчетов пространственной гидродинамики промышленного масштаба. Труды конференции «Свободное программное обеспечение», 2010, стр. 44-46.

Building the Software Defined Data Center

[1] *B.M. Shabanov <shabanov@jscc.ru>*
[2] *O.I. Samovarov <samov@ispras.ru>*
[1] *Scientific Research Institute of System Development,*
117218, Moscow, Nakhimovskiy Avenue, 36, bld.1
[2] *Ivannikov Institute for System Programming of the RAS,*
109004, Russia, Moscow, Alexander Solzhenitsyn st., 25

Abstract. Data Center is the most progressive form of providing computing resources when it is necessary to provide services to a wide range of users. The paper discusses one of the approaches to the creation of a data center – the concept of software-defined infrastructure. Software-defined is such a data center infrastructure, in which all its key elements – computing resources, network, data storage systems, etc. – are virtualized and provided to users as services with a given quality. It is shown that the implementation of the proposed infrastructure allows each user to productively solve their tasks in a reasonable time with an acceptable level of costs. The paper formulates the general requirements for inter-departmental data processing centers, proposes methods for creating software-defined data

center (deployment of computing platforms that maximize hardware reuse, provide support for the execution of programs of different classes of tasks, etc.) and describes the implementation of infrastructures of this class in specific projects.

Keywords: software-defined data center; cloud computing; distributed computing; virtualization программно-определяемые ЦОД; облачные вычисления; распределенные вычисления; виртуализация

DOI: 10.15514/ISPRAS-2018-30(6)-1

For citation: Shabanov B.M., Samovarov O.I. Building the Software Defined Data Center. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 7-24 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-1

References

[1]. Joseph M. Hellerstein. Programming a Parallel Future. EECS Department University of California, Berkeley, Technical Report No. UCB/EECS-2008-144, November 7, 2008

[2]. Krste Asanović, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams and Katherine A. Yelick. The Landscape of Parallel Computing Research: A View from Berkeley. EECS Department University of California, Berkeley, Technical Report No. UCB/EECS-2006-183 December 18, 2006

[3]. Mike Perks. Reference Architecture: VMware Software Defined Data Center. 15 August 2018 [online], страница LENOVOPRESS, available at: https://lenovopress.com/lp0661-reference-architecture-vmware-software-defined-data-center-thinksystem, accessed 20.10.2018

[4]. FUJITSU White Paper [online]. Software-Defined Data Center – infrastructure for enterprise digital transformation. Available at: https://sp.ts.fujitsu.com/dmsp/Publications/public/wp-sddc-infrastructure-for-enterprise-digital-transformation-ww-en.pdf, accessed 20.10.2018

[5]. EMC Reference Architecture [online]. Federation Software-Defined Data Center, Foundation Infrastructure Reference Architecture. Available at: https://www.emc.com/collateral/TechnicalDocument/h13378-evp-sddc-ra.pdf, accessed 20.10.2018

[6]. Arutyun Avetisyan, Sergey Gaysaryan, Viktor Ivannikov, Oleg Samovarov. Training center «University Cluster»: Blending Education, Science and Industry. Open Systems. DBMS, № 05, 2010 (in Russian)

[7]. Arutyun Avetisyan, Oleg Samovarov, Sergey Gaysaryan, Eshsou Hashba. Opencirrus, Russian Segment. Open Systems. DBMS, № 05, 2011 (in Russian)

[8]. M.V. Kraposhin, O.I. Samovarov, S.V. Strijhak. Web Laboratory UniHUB in the Scope of Program «University Cluster». In Proc. of the Workshop «Multiphysical Modelling in OpenFOAM», Riga, October 20-21, 2011

[9]. M.V. Kraposhin, O.I. Samovarov, S.V. Strijhak. Features of the implementation of the Web-laboratory of continuum mechanics based on the technological platform of the program «University Cluster». In Proc. of the International supercomputer conference with elements of the scientific school for young people «Scientific

service on the Internet: an exaflop future», 2011, pp. 473-475 (in Russian)

[10]. M.V. Kraposhin, O.I. Samovarov, S.V. Strijhak. Experience of using open source software for calculating spatial hydrodynamics of industrial scale. In Proc. of the All-Russian Conference «Free Software 2010», 2010, pp. 44-46 (in Russian).

Combining dynamic symbolic execution, code static analysis and fuzzing[1]

[1]*A.Yu. Gerasimov <agerasimov@ispras.ru>*
[2] *S.S. Sargsyan <sevaksargsyan@ispras.ru>*
[1] *S.F. Kurmangaleev <kursh@ispras.ru>*
[2]*J.A. Hakobyan <jivan@ispras.ru>*
[2] *S.A. Asryan <asryan@ispras.ru>*
[1] *M.K. Ermakov <mermakov@ispras.ru>*

[1] *Ivannikov Institute for System Programming, 109004, Russia, Moscow, Alexandra Solzhenitsyna, 25*
[2] *Yerevan State University, System Programming Laboratory, 0025, Armenia, Yerevan, Alex Manuugian, 1*

Abstract. This paper describes a new approach for dynamic code analysis. It combines dynamic symbolic execution and static code analysis with fuzzing to increase efficiency of each component. During fuzzing we recover indirect function calls and pass that information to the static analysis engine. This improves static path detection in the control flow graph of a program. Detected paths are used in dynamic symbolic execution to construct inputs which will cover new paths during execution. These inputs are used by the fuzzing tool to improve test-case generation and increase code coverage. The proposed approach can be used for classic fuzzing when the main goal is achieving high code coverage. As well it can be used for targeted analysis of paths and code fragments in the program. In this case the fuzzing tool accepts a set of programs addresses with potential defects and passes them to the static analysis engine. The engine constructs all paths connecting program entry point to the given addresses. Finally, dynamic symbolic execution is used to construct the set of inputs, which will cover these paths. Experimental results have shown that the proposed method can effectively detect different program defects.

Keywords: fuzzing; directed fuzzing; static analysis;, path detection; dynamic symbolic execution

DOI: 10.15514/ISPRAS-2018-30(6)-2

Для цитирования: Gerasimov A.Yu., Sargsyan S.S., Kurmangaleev S.F., Hakobyan J.A., Asryan S.A., Ermakov M.K. Combining dynamic symbolic execution and fuzzing. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 25-38. DOI: 10.15514/ISPRAS-2018-30(6)-2

[1] Research is funded within the scope of RFBR grant 17-07-00702

1. Introduction

Dynamic program analysis has proven to be one of the most effective bugs finding techniques. It has a low false positive rate and most of the detected defects can be reproduced. There are several approaches for dynamic analysis. Fuzzing [1] is one of the most effective and widely used techniques, which detects defects and provides inputs to reproduce them. But it has some limitations. For example, fuzzing itself is not usable for analysis of the specific program fragments. The main reason is that inputs are randomly generated in an attempt to increase the code coverage. Dynamic symbolic execution [2] is used for systematic generation of program inputs to cover all possible execution paths. It is significantly slower than fuzzing and cannot be applied to analysis of large programs.

One of the most widely used fuzzing tools is AFL (American Fuzzy Lop) [3, 4, 5, 6]. It is a coverage guided fuzzing tool, which uses genetic algorithms for test case selection and mutation adoption. AFL can perform static instrumentation of the target program or dynamic binary code instrumentation based on QEMU [7] for coverage gathering. *LibFuzzer* [8] is an embedded fuzzing library in *LLVM* [9] compiler infrastructure, which provides the means to fuzz individual program function. *Syzkaller* [10] performs fuzzing of system functions calls for operating systems (OS) based on their descriptions. It generates and runs small programs containing system functions calls and monitors the OS state. If a crash is detected the corresponding input and generated program are stored for debugging purposes. *Peach* [11] is used for network protocol fuzzing. It introduces the concept of pit files, which describe target protocols. Grammar-based fuzzing [12] is used for fuzzing of programs (compilers, interpreters, parsers, translators etc.) accepting BNF structured inputs. It has predefined specifications for more than 120 programming languages and data formats.

Symbolic execution of a program typically refers to the process of traversing its execution tree while evaluating internal and external program data as abstract symbolic variables instead of concrete values. Program instructions applied to these variables form *path constraints* (typically represented as SMT – Satisfiability Modulo Theory – formulas). Working with these path constraints allows one to identify valuable information about multiple potential concrete execution paths at once. Dynamic symbolic execution (*DSE*) tools incorporate various techniques and improvements of basic symbolic execution to allow one to solve various practical program analysis tasks. They are widely used to perform automatic execution tree traversal by generating concrete input data. In turn, these data sets are used as test suites for defect detection and various coverage-related analyses for the target program. *Avalanche* [13, 14], *DySy* [15], *BINSEC/SE* [16] are well known *DSE* tools.

There are advantages and limitations for both fuzzing and dynamic symbolic execution. Black-box and grey-box fuzzing tools can generate a lot of inputs in a limited time, but suffer from random nature of data generation algorithm and the

only feedback which is used to support genetic algorithms is coverage data and crash/hang information for program under analysis. On the other hand, dynamic symbolic execution tools perform aggressive instrumentation of program under analysis to gather execution traces in terms of SMT formulas which drastically influences performance of program under analysis. Also, dynamic symbolic execution suffers from the *path explosion problem* [17]. Recent research focuses on combining different analysis methods to overcome limitations of methods applied separately. Amongst known solutions we want to mention jFuzz [18], Driller [19], a hybrid symbolic execution assisted fuzzing method [20] which combine fuzzing and symbolic execution to overcome known limitations of methods.

In this paper we propose an approach for combining fuzzing, dynamic symbolic execution and static code analysis for program defects detection.

2. Proposed fuzzing tool

2.1 The Architecture of the tool

The tool consists of four basic components (fig. 1). The first component is a fuzzing tool, which provides a set of mutations and basic infrastructure. The second component is a *DynamoRIO* [21] based client library for code coverage gathering. The third component is the dynamic symbolic execution tool *Anxiety* [22]. The fourth component is a program binary code static analysis engine. The proposed tool is able to perform classic fuzzing, where the main goal is to increase code coverage as much as possible. Additionally, it can perform directed analysis of the target program – instead of trying to increase code coverage the tool tries to generate input data to cover specified fragments of the target program.

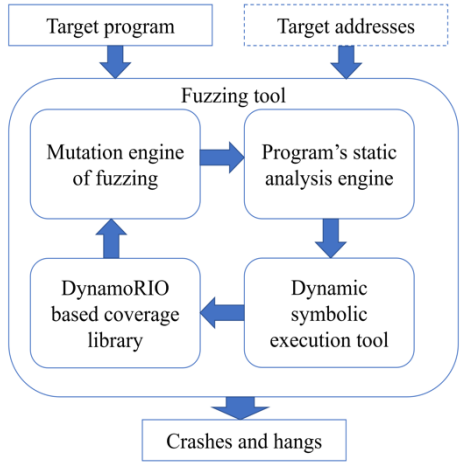


Fig. 1. The architecture of the tool

For directed fuzzing the tool accepts a set of addresses which should be executed during analysis. At first, classic fuzzing is performed until coverage stops to increase for some time (controlled by user). This typically means that there are certain fragments of code which are completely inaccessible during execution (i.e. dead code) or can only be reached with an input data set with internal dependencies that are too complex for the semi-random input mutation algorithms. In order to generate these input data sets we employ dynamic symbolic execution guided by static analysis.

2.2 Guided dynamic symbolic execution

Anxiety, the dynamic symbolic execution tool used within the system, implements «offline» concolic execution:

- it continuously performs concrete executions *along* with symbolic execution of the target program using initial input data sets and input data sets generated by the tool;
- thus, a concrete execution for an input data set produces a symbolic path constraint for this specific data set;
- this path constraint includes a number of branch points explicitly influenced by the input data set;
- for every branch point in the path constraint an attempt is made to invert the corresponding comparison and check whether the modified path constraint is *satisfiable*;
- the process of checking for satisfiability automatically produces a different input data set which is presumed to force the execution of the program onto a different path at the corresponding branch point;
- upper and lower depth limits are used to avoid processing the same branch points (producing input data sets processed previously) and creating path constraints too large to check in a limited time.

The number of branch points is a critical factor of the analysis complexity. During guided symbolic execution certain branch points are processed in a different manner based on fuzzing goals:

- «black» lists are used to skip certain branch points which were already covered during normal fuzzing (meaning that fuzzing produced at least two different input data sets which force the program execution differently for every branch point among given);
- «white» lists are used to augment path constraints with external information – which direction at the branch point must be taken for all generated paths.

Classic fuzzing, where code coverage increase is the main goal may also be improved via DSE integration. The only difference is in the list of basic blocks passed to DSE. Static analysis detects the list of basic blocks, whose both branches

were executed and pass them to DSE as a «black» list (since no new information we will be gained by inverting such blocks).

In both cases, traces of the target program execution are stored in order to perform indirect call recovery (function pointers, virtual functions). This information is used to improve static analysis which in its turn improves the results of other components.

Static analysis is periodically invoked during fuzzing to keep the data base of the target program updated using recovered indirect call addresses. This enables mutual improvement for static and dynamic analysis. Experimental results prove the effectiveness of this approach.

2.3 Static analysis engine

The static analysis engine has two basic functionalities: detecting paths in a control flow graph and program trace analysis. In the first case the tool identifies a number of paths between two program addresses. The number of limitations are applied for optimization: path's maximum length, maximum number of usages for each basic block or a function during path construction etc. These limitations are necessary to overcome the path explosion problem. Path construction consists of two basic stages (fig. 2). The first stage filters some functions based on call graph. It uses forward and backward *BFS* (Breadth-First Search) algorithm for entry and destination addresses of a target program to determine all functions which should be included in the path detection process. In the second stage we use modified DFS (Depth-First Search) for path detection. Then we construct a «white» list for DSE. It contains all basic blocks from detected paths which have branch instructions. The «white» list is used by DSE to generate data which will cover both branches of each basic block.

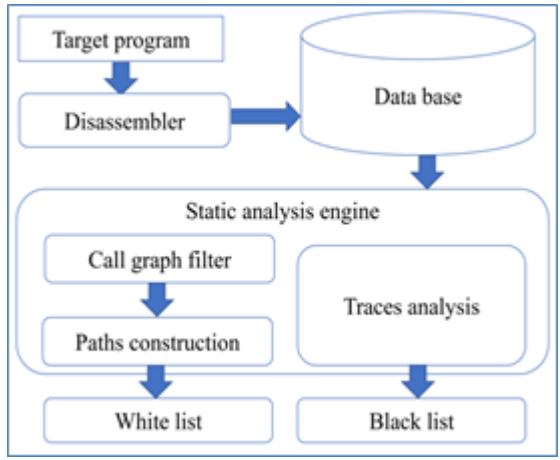


Fig. 2. Static path detection

In the second case, static analysis loads the set of traces generated by fuzzing tool and tries to find all basic blocks whose both branches were executed. Then it creates a «black» list based on these blocks to be used by DSE for optimization.

2.4 Switching metric

To switch between the fuzzing tool and DSE (static analysis included) we use a variable parameter ***N***. DSE is invoked if below formula is satisfied:

$$total\_execs - last\_effective\_exec > 10000 * N$$

where ***total_execs*** – number of executions in the moment when we try to invoke DSE, ***last_effective_exec*** – number of executions when the fuzzing last time was able to detect new execution path, ***N*** – is specified by user.
If the fuzzing tool was not able to open new execution traces for some time , then we invoke DSE.

2.5 DSE run time metric

We use special metric for calculating the maximum amount of time to allow for the DSE stage. This amount (in seconds) is calculated according to below formula:

$$runtime = 30 + total\_execs / 50000$$

where ***runtime*** – time limit for a DSE run, ***total_execs*** – number of executions at the moment when we try to invoke DSE.
The running time for DSE is at least 30 seconds (the number is determined according to experimental results). Our experiments show that less than 30 second for DSE is not enough to achieve valuable results for an average program. We increase DSE run time limit in one second after each 50.000 executions, which enables it to run longer during fuzzing.

2.6 Mutual improvement of static analysis results

While the target program is processed, static analysis engine precision is continuously improving due to indirect call address recovery. *DynamoRIO* [18] based coverage library has trace generation support, which allows us to recover actual addresses for indirect call instructions. During fuzzing process, unique traces are generated for the target program. Then they are analyzed for indirect call address recovery. The process is simple, for each executed block we store information about previously executed block. Then based on that information the actual address is recovered: if there is block in trace which belongs to some function *f* and previously executed block belongs to some function *g*, then there is an edge between *g* and *f* functions in the call graph. The newly detected edges are added to the target program data base.
Improved static analysis has positive impact on DSE results. It allows to construct more inputs which are covering different execution paths between program entry

point and destination addresses (a direct fuzzing case). These inputs improve the coverage of the fuzzing tool and improve its effectiveness. Proposed scheme of interaction between these three tools allows iteratively improve the results of each other and overall fuzzing results.

3. Results

3.1 Results of fuzzing integrated with DSE

In the table below (Tab. 1) you will find experimental results of classic fuzzing (with aim of code coverage increase) integrated with DSE. In this case we try to increase code coverage as much as possible. All detected crashes were verified manually.

Table 1. Classic fuzzing guided code coverage increase results

Operating system	Test name	Detected crashes	Running time (hours:minutes)
Debian-6.0.10	blast2	3	0:15
Debian-6.0.10	faad	1	0:20
Debian-6.0.10	efax	1	0:30
Debian-6.0.10	wavpack	5	0:30
Debian-6.0.10	tic	4	1:00
Debian-6.0.10	ul	7	1:00
Debian-6.0.10	Bsd-form	6	12:00

3.2 Results of directed fuzzing

Results of the directed fuzzing for programs from Linux distribution and DARPA [23] Cyber Grand Challenge are presented in Table 2. Static analysis has detected potential program addresses which may have defects. We run fuzzing in directed mode to generate data, which will cover specified addresses in an attempt to crash them. The last column shows the number of hits for detected address list. The fist value is the number of addresses for which the fuzzing tool was able to generate input data to cover them during execution. The second value is the number of potential buggy addresses detected by static analysis. For example, for the test ***FableReport*** static analysis has detected 15 potential defect addresses, but fuzzing tool managed to cover only 7 of them. The number of crashes is not synchronous with hit addresses due to several reasons:

- program can crash in the same address with different execution paths and fuzzing will consider it as different crashes

- if fuzzing managed to generate data which will cover specified address, it is not necessary that program should crash; the address may be false positive from static analysis or generated data do not crash it.

All results were verified manually.

Table 2. Directed fuzzing guided by static analysis results

Operating system	Test name	Crashes	Runing time (hours:minutes)	Hits
Debian-6.0.10	faad	2	21:00	1/1
Debian-6.0.10	passwd	2	0:20	1/1
Debian-6.0.10	uuenview	13	0:50	1/1
DARPA	Flash_File_System	35	2:00	1/1
DARPA	3D_Image_Toolkit	30	19:00	1/1
DARPA	Charter	9	20:00	1/1
DARPA	Diary_Parser	9	20:00	1/1
DARPA	PRU	2	1:00	1/1
DARPA	Recipe_Database	23	20:00	1/1
DARPA	SCUBA_Dive_Logging	10	20:00	1/1
DARPA	SFTSCBSISS	1	20:00	1/1
DARPA	Simple_Stack_Machine	15	20:00	1/1
DARPA	CML	10	20:00	1/1
DARPA	Eddy	9	4:00	1/1
DARPA	FablesReport	3	4:00	7/15
DARPA	Multipass3	7	4:00	1/3
DARPA	Online_job_application	4	4:00	1/1
DARPA	Overflow_Parking	2	4:00	1/1
DARPA	PTassS	5	4:00	1/2
DARPA	Sample_Shipgame	5	4:00	2/2
DARPA	SAuth	1	4:00	1/3

4. Discussion

A similar approach is used in *Badger* [24] tool. It combines fuzzing and dynamic symbolic execution in the following way: when the input is passed to symbolic execution it tries to update this input until it reaches new coverage or find a path with lower cost of analysis in terms of computational resources. This approach uses trie-based [25] symbolic execution to predict and reduce the complexity of dynamic

symbolic execution by saving a trie-like structure for path constraints gathered during path exploration until new part of path detected to execute it in symbolic manner. *Qsym* is another analysis tool [26] which combines symbolic and fuzzing. It uses optimistic solving of relaxed path constraints trying to find new paths with small cost of computations in solver and pruning conditions gathered from repetitive basic blocks from symbolic formulae to simplify constraints relying on fuzzing tool as an efficient validator of generated input.

Our approach differs from the proposed solutions. It uses static analysis to guide fuzzing and dynamic symbolic execution through continuously updated program call graph to reach destination address with the help of dynamic symbolic execution.

5. Conclusion and future work

Indirect call instructions addresses are not fully recovered based on program traces. There can be addresses, which will not be recovered because corresponding path is not executed during fuzzing. Future research directions are:

- add alias analysis on program's binary representation to improve indirect call addresses recovery;
- use available information/traces obtained from fuzzing for alias analysis improvement.

References

[1]. Fuzzing (online publication). Available at: https://en.wikipedia.org/wiki/Fuzzing, 11.12.2018

[2]. Ting Chen, Xiao-song Zhang, Shi-ze Guo, Hong-yuan Li, Yue Wu. State of the art: Dynamic symbolic execution for automated test generation. Future Generation Computer Systems, volume 29, issue 7, 2013, pp.1758-1773

[3]. American fuzzy lop (online publication). Available at: http://lcamtuf.coredump.cx/afl/, 11.12.2018

[4]. A fork of AFL for fuzzing Windows binaries (online publication). Available at: https://github.com/ivanfratric/winafl, 11.12.2018

[5]. American fuzzy lop for network fuzzing (unofficial) (online publication). Available at: https://github.com/jdbirdwell/afl, 11.12.2018

[6]. Technical «whitepaper» for afl-fuzz (online publication). Available at: http://lcamtuf.coredump.cx/afl/technical_details.txt, 11.12.2018

[7]. QEMU (online publication). Available at: https://www.qemu.org/, 11.12.2018

[8]. libFuzzer – a library for coverage-guided fuzz testing (online publication). Available at: https://llvm.org/docs/LibFuzzer.html, 11.12.2018

[9]. The LLVM Compiler Infrastructure (online publication). Available at: https://llvm.org, 11.12.2018

[10]. Syzkaller is an unsupervised, coverage-guided kernel fuzzer (online publication). Available at: https://github.com/google/syzkaller, 11.12.2018

[11]. Peach (online publication). Available at: https://www.peach.tech/products/peach-fuzzer/, 11.12.2018

[12]. Sevak Sargsyan, Shamil Kurmangaleev, Matevos Mehrabyan, Maksim Mishechkin, Tsolak Ghukasyan, Sergey Asryan. Grammar-Based Fuzzing. In Proc. of the Ivannikov Memorial Workshop. Yerevan, Armenia, 2018, pp. 32-36

[13]. Ildar Isaev, Denis Sidorov, Alexander Gerasimov, Mikhail Ermakov. Avalanche: Using dynamic analysis for automatic defect detection in programs based on network sockets. Trudy ISP RAN/Proc. ISP RAS, vol. 21, 2011, pp. 55-70

[14]. M.K. Ermakov, A.Y. Gerasimov. Avalanche: adaptation of parallel and distributed computing for dynamic analysis to improve performance of defect detection. Trudy ISP RAN/Proc. ISP RAS, vol. 25, 2013, pp. 29-38. doi:10.15514/ISPRAS-2013-25-2

[15]. Christoph Csallner, Nikolai Tillmann, Yannis Smaragdakis. DySy: Dynamic Symbolic Execution for Invariant Inference. In Proc. of the 30th international conference on Software, 2008, pp. 281-290

[16]. Robin David, Sebastien Bardin, Thanh Dinh Ta, Laurent Mounier, Josselin Feist, Marie-Laure Potet, Jean-Yves Marion. BINSEC/SE: A Dynamic Symbolic Execution Toolkit for Binary-level Analysis. In Proc. of the 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), 2016, pp. 653-656

[17]. R.S. Boyer, B. Elspas, K.N. Levitt. SELECT – F Formal System for Testing and Debugging Programs by Symbolic Execution. In Proc. of the Internations Conference on Reliable software. pp. 234-245, Los Angeles, Califirnia, USA, Aprul 21-23, 1975

[18]. Jayaraman K.k, Harvison D., Ganesh V., Kiezun A. jFuzz: A Concolic Whitebox Fuzzer for Java. In Proc. of the First NASA Formal Methods Symposium, 2009, pp. 121-125

[19]. N. Stephens, J. Grosen, C. Salls, A. Dutcher, R. Wang, J. Corbetta, Y. Shoshitaishvili, C. Kruegel, G. Vigna. Driller: Augmenting fuzzing through selective symbolic execution. In Proc. of the 2016 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, Feb. 2016

[20]. Li Zhang, Vrizlynn L. L. Thing. A hybrid symbolic execution assisted fuzzing method. In Proc. of the TENCON 2017 - 2017 IEEE Region 10 Conference, doi: 10.1109/TENCON.2017.8227972, 5-8 Nov. 2017

[21]. DynamoRIO (online publication). Available at: http://www.dynamorio.org/, 11.12.2018.

[22]. A. Gerasimov, S. Vartanov, M. Ermakov, L. Kruglov, D. Kutz, A. Novikov, S. Asryan. Anxiety: a dynamic symbolic execution framework. In Proc. of the 2017 Ivannikov ISPRAS Open Conference, 2017, pp. 16-21. doi:10.1109/ISPRAS.2017.00010

[23]. Cyber Grand Challenge (online publication). Available at: https://www.darpa.mil/program/cyber-grand-challenge, 11.12.2018

[24]. Yannic Nolle, Rody Kersten, Corina S. Păsăreanu. Badger: complexity analysis with fuzzing and symbolic execution. In Proc. of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 322-332, Amsterdam, Netherlands — July 16 - 21, 2018

[25]. Guowei Yang, Corina S. Păsăreanu, Sarfraz Khurshid. Memoized symbolic execution. In Proc. of the 2012 International Symposium on Software Testing and Analysis, pp. 144-154 Minneapolis, MN, USA — July 15 - 20, 2012

[26]. Insu Yun, Sangho Lee, Meng Xu, Yeongjin Jang, Taesoo Kim. QSYM: a practical concolic execution engine tailored for hybrid fuzzing. In Proc. of the 27th USENIX Conference on Security Symposium, pp. 745-761, Baltimore, MD, USA — August 15 - 17, 2018

Комбинирование динамического символьного исполнения, статического анализа кода и фаззинга

[1] *А.Ю. Герасимов <agerasimov@ispras.ru>*
[2] *С.С. Саргсян <sevaksargsyan@ispras.ru>*
[1] *Ш.Ф. Курмангалеев <kursh@ispras.ru>*
[2] *Д.А. Акопян <jivan@ispras.ru>*
[2] *С.А. Асрян <asryan@ispras.ru>*
[1] *М.К. Ермаков <mermakov@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, Москва, Александра Солженицына, 25*
[2] *Ереванский государственный университет, Лаборатория системного программирования, 0025, Армения, Ереван, ул. Алека Манукяна, 1*

Аннотация. В этой статье описывается новый подход для динамического анализа программ. Он совмещает динамическое символьное исполнение программ и статический анализ кода программ с фаззингом для повышения эффективности каждого из методов. В процессе фаззинга восстанавливаются вызовы по вычисляемым адресам и расширенный граф вызовов передается модулю статического анализа. Это позволяет улучшить вычисление путей исполнения программы в процессе статического анализа. Открытые новые пути исполнения в программе передаются модулю динамического символьного исполнения для генерации новых наборов внешних данных программы с целью исполнения и анализа программы по открытым путям исполнения. Новые наборы входных данных передаются модулю фаззинга для увеличения покрытия программы с их использованием в качестве затравки. Предложенный подход может быть использован в рамках классического алгоритма работы фаззинга с целью достижения высокого покрытия кода программы тестовыми наборами. Также предложенный метод может использоваться для направленного анализа путей и фрагментов кода программы. В этом случает фаззер формирует набор адресов и передает их модулю статического анализа. Статический анализ формирует набор путей, которые приводят к исполнению инструкций по этим адресам от точки входа в программу. Далее модуль динамическое символьного исполнения используется для построения наборов входных данных для прохождения по этим путям. Результаты экспериментов показывают высокую эффективность обнаружения программных ошибок при применении предложенного метода.

Ключевые слова: фаззинг; направленный фаззинг; статический анализ; обнаружение путей исполнения; динамическое символьное исполнение

DOI: 10.15514/ISPRAS-2018-30(6)-2

Для цитирования: Герасимов А.Ю., Саргсян С.С., Курмангалеев Ш.Ф., Акопян Д.А., Асрян С.А., Ермаков М.К. Комбинирование динамического символьного исполнения, статического анализа кода и фаззинга. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 25-38. DOI: 10.15514/ISPRAS-2018-30(6)-2

Список литературы

[1]. Fuzzing (online publication). Режим доступа: https://en.wikipedia.org/wiki/Fuzzing, 11.12.2018
[2]. Ting Chen, Xiao-song Zhang, Shi-ze Guo, Hong-yuan Li, Yue Wu. State of the art: Dynamic symbolic execution for automated test generation. Future Generation Computer Systems, volume 29, issue 7, 2013, pp.1758-1773
[3]. American fuzzy lop (online publication). Режим доступа: http://lcamtuf.coredump.cx/afl/, 11.12.2018
[4]. A fork of AFL for fuzzing Windows binaries (online publication). Режим доступа: https://github.com/ivanfratric/winafl, 11.12.2018
[5]. American fuzzy lop for network fuzzing (unofficial) (online publication). Режим доступа: https://github.com/jdbirdwell/afl, 11.12.2018
[6]. Technical «whitepaper» for afl-fuzz (online publication). Режим доступа: http://lcamtuf.coredump.cx/afl/technical_details.txt, 11.12.2018
[7]. QEMU (online publication). Режим доступа: https://www.qemu.org/, 11.12.2018
[8]. libFuzzer – a library for coverage-guided fuzz testing (online publication). Режим доступа: https://llvm.org/docs/LibFuzzer.html, 11.12.2018
[9]. The LLVM Compiler Infrastructure (online publication). Режим доступа: https://llvm.org, 11.12.2018
[10]. Syzkaller is an unsupervised, coverage-guided kernel fuzzer (online publication). Режим доступа: https://github.com/google/syzkaller, 11.12.2018
[11]. Peach (online publication). Режим доступа: https://www.peach.tech/products/peach-fuzzer/, 11.12.2018
[12]. Sevak Sargsyan, Shamil Kurmangaleev, Matevos Mehrabyan, Maksim Mishechkin, Tsolak Ghukasyan, Sergey Asryan. Grammar-Based Fuzzing. In Proc. of the Ivannikov Memorial Workshop. Yerevan, Armenia, 2018, pp. 32-36
[13]. Исаев И.К., Сидоров Д.В., Герасимов А.Ю., Ермаков М.К. Avalanche: Применение динамического анализа для автоматического обнаружения ошибок в программах, использующих сетевые сокеты. Труды ИСП РАН, том 21, 2011, стр. 55-70
[14]. Ермаков М.К., Герасимов А.Ю. Avalanche: применение параллельного и распределенного динамического анализа программ для ускорения поиска дефектов и уязвимостей. Труды ИСП РАН, том 25, 2013, стр. 29-38. doi:10.15514/ISPRAS-2013-25-2
[15]. Christoph Csallner, Nikolai Tillmann, Yannis Smaragdakis. DySy: Dynamic Symbolic Execution for Invariant Inference. In Proc. of the 30th international conference on Software, 2008, pp. 281-290
[16]. Robin David, Sebastien Bardin, Thanh Dinh Ta, Laurent Mounier, Josselin Feist, Marie-Laure Potet, Jean-Yves Marion. BINSEC/SE: A Dynamic Symbolic Execution Toolkit for Binary-level Analysis. In Proc. of the 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), 2016, pp. 653-656
[17]. R.S. Boyer, B. Elspas, K.N. Levitt. SELECT – F Formal System for Testing and Debugging Programs by Symbolic Execution. In Proc. of the Internations Conference on Reliable software. pp. 234-245, Los Angeles, Califirnia, USA, Aprul 21-23, 1975
[18]. Jayaraman K.k, Harvison D., Ganesh V., Kiezun A. jFuzz: A Concolic Whitebox Fuzzer for Java. In Proc. of the First NASA Formal Methods Symposium, 2009, pp. 121-125
[19]. N. Stephens, J. Grosen, C. Salls, A. Dutcher, R. Wang, J. Corbetta, Y. Shoshitaishvili, C. Kruegel, G. Vigna. Driller: Augmenting fuzzing through selective symbolic execution.

In Proc. of the 2016 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, Feb. 2016

[20]. Li Zhang, Vrizlynn L. L. Thing. A hybrid symbolic execution assisted fuzzing method. In Proc. of the TENCON 2017 - 2017 IEEE Region 10 Conference, doi: 10.1109/TENCON.2017.8227972, 5-8 Nov. 2017

[21]. DynamoRIO (online publication). Режим доступа: http://www.dynamorio.org/, 11.12.2018.

[22]. A. Gerasimov, S. Vartanov, M. Ermakov, L. Kruglov, D. Kutz, A. Novikov, S. Asryan. Anxiety: a dynamic symbolic execution framework. In Proc. of the 2017 Ivannikov ISPRAS Open Conference, 2017, pp. 16-21. doi:10.1109/ISPRAS.2017.00010

[23]. Cyber Grand Challenge (online publication). Режим доступа: https://www.darpa.mil/program/cyber-grand-challenge, 11.12.2018

[24]. Yannic Nolle, Rody Kersten, Corina S. Păsăreanu. Badger: complexity analysis with fuzzing and symbolic execution. In Proc. of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 322-332, Amsterdam, Netherlands — July 16 - 21, 2018

[25]. Guowei Yang, Corina S. Păsăreanu, Sarfraz Khurshid. Memoized symbolic execution. In Proc. of the 2012 International Symposium on Software Testing and Analysis, pp. 144-154 Minneapolis, MN, USA — July 15 - 20, 2012

[26]. Insu Yun, Sangho Lee, Meng Xu, Yeongjin Jang, Taesoo Kim. QSYM: a practical concolic execution engine tailored for hybrid fuzzing. In Proc. of the 27th USENIX Conference on Security Symposium, pp. 745-761, Baltimore, MD, USA — August 15 - 17, 2018

О новом поколении промежуточных представлений, применяемых для анализа бинарного кода[1]

[1,2] *М.А. Соловьев <icee@ispras.ru>*
[1] *М.Г. Бакулин <bakulinm@ispras.ru>*
[1] *М.С. Горбачев <sadbear@ispras.ru>*
[1,2] *Д.В. Манушин <dman95@ispras.ru>*
[1,2] *В.А. Падарян <vartan@ispras.ru>*
[1] *С.С. Панасенко <spanasenko@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский государственный университет им. М.В. Ломоносова, 119991, Россия, г. Москва, Ленинские горы, д. 1*

Аннотация. Многие программные инструменты анализа бинарного кода работают не напрямую с машинными командами, а с промежуточными представлениями, в которые этот код транслируется. В статье рассмотрены различные задачи анализа бинарного кода, сформулированы требования к промежуточному представлению, которое могло бы использоваться сразу для многих задач. Базовые требования дополнены требованиями, вытекающими из особенностей целевых процессорных архитектур. Проведен обзор существующих подходов к декодированию машинных команд и описанию их семантики и предложены новые подходы к решению этих задач: унифицированная схема декодирования и промежуточное представление для задания семантики команд, позволяющее учитывать особенности выборки команд и обработки исключений. Показан способ применения предложенных подходов к моделированию работы процессора при конкретной и абстрактной интерпретации и символьном выполнении.

Ключевые слова: абстрактная интерпретация; анализ бинарного кода; динамический анализ; компиляторные технологии; обратная инженерия ПО; символьное выполнение; статический анализ.

DOI: 10.15514/ISPRAS-2018-30(6)-3

Для цитирования: Соловьев М.А., Бакулин М.Г., Горбачев М.С., Манушин Д.В., Падарян В.А., Панасенко С.С. О новом поколении промежуточных представлений, применяемом для анализа бинарного кода. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 39-68. DOI: 10.15514/ISPRAS-2018-30(6)-3

[1] Работа поддержана грантом РФФИ 18-07-01256 А.

1. Введение

С задачами анализа исполняемого бинарного кода сталкиваются как разработчики ПО, так и специалисты по кибербезопасности. Необходимость проводить анализ на уровне бинарного кода обусловлена не только ситуациями, когда исходные тексты недоступны или частично утрачены. Для высокоуровневого языка программирования затруднительно либо невозможно оценить последствия срабатывания программного дефекта. Оптимизирующие преобразования кода, проводимые современными компиляторами, способны внести серьезные и потенциально эксплуатируемые дефекты, когда сталкиваются с определенными аспектами поведения программы, выходящими за рамки спецификации языка программирования [1]. Разработчики вынуждены работать с исполняемыми файлами при поиске дефектов методами динамического анализа, отладке и тестировании, в некоторых задачах повторного использования кода. Специалисты по аудиту безопасности оказываются в схожей ситуации при поиске недекларированных возможностей и оценке безопасности ПО.

В настоящий момент уже существует множество программных инструментов, в различной степени автоматизирующих анализ бинарного кода при решении практических задач. Общим местом у значительной части таких инструментов оказывается промежуточное представление, в которое переводится анализируемый код, подобно тому, как это происходит в компиляторе. Исторически такой способ работы с кодом применялся в бинарной трансляции, а затем – в бинарном инструментировании [2] и задачах анализа.

Распространенные процессорные архитектуры имеют достаточно обширные наборы команд, поэтому целесообразно реализовать сложный анализ на уровне промежуточного представления, упрощающего «крупные» команды целевой архитектуры путем их дробления на более мелкие элементы. Такой подход хорош также тем, что позволяет посредством трансляции в промежуточное представление кода различных целевых архитектур добиться более простой процедуры адаптации инструмента анализа к новой целевой архитектуре: достаточно разработать модуль трансляции.

Помимо положительных аспектов, практическое применение промежуточных представлений сталкивается с рядом проблем.

В первых работах [3] применялись промежуточные представления, непосредственно заимствованные из компилятора, а именно, LLVM [4]. К сожалению, такой подход оказался малоэффективен, т.к. существенная составляющая LLVM – типы данных – в бинарном коде отсутствует и не может быть восстановлена во время трансляции.

Следующая «волна» работ была посвящена разработке специализированных промежуточных представлений, отвечающих условиям анализа бинарного кода: Vine [5], Pivot [6], BAP [7] и REIL [8]. Эти представления изначально были рассчитаны на то, что будут получены из бинарного кода с целью

анализа свойств. Проблемы проявились в том, что разные программные инструменты применяют различные по своей форме и степени детальности промежуточные представления, а сам функционал трансляции каждый раз реализуется заново. Помимо очевидного увеличения трудозатрат на разработку новых инструментов, сложившаяся ситуация также выражается в возможности рассогласования результатов при использовании нескольких инструментов, так как разные инструменты могут иметь разные неточности в обработке отдельных команд и их характеристик.

Специализация представлений позволила относительно успешно применять их к коду процессорных архитектур общего назначения (x86, ARM, MIPS), но некоторые аспекты поведения этих машин и других, менее привычных (но не менее распространенных) процессорных архитектур эти представления описать не могут. В первую очередь это затрагивает особенности обработки исключений, прохождение команд по конвейеру и обращение в память. Вопросы безопасности устройств интернета вещей (IoT) делают все более острой необходимость анализировать код микроконтроллеров, тогда как возможностей имеющихся программных инструментов оказывается недостаточно.

В данной статье предлагается новое промежуточное представление, с одной стороны являющееся специализированным для решения задач анализа бинарного кода, а с другой – универсальным в своих возможностях менять уровень детализации при описании работы широкого класса процессорных архитектур, как общего назначения, так и применяемых в микроконтроллерах.

Дальнейший материал организован следующим образом. Во втором разделе рассматриваются наиболее распространенные задачи анализа бинарного кода с целью сбора требований к промежуточному представлению. В третьем разделе описаны ключевые особенности различных процессорных архитектур, которые также необходимо учесть при проектировании промежуточного представления. В четвертом разделе описаны существующие решения по декодированию машинного кода и предложен способ декодирования, который является составной частью разработанного подхода. Пятый раздел содержит обзор существующих подходов к моделированию операционной семантики бинарного кода и описывает предлагаемый в рамках данной работы подход. В шестом разделе приводится общая схема предлагаемого решения. Седьмой раздел содержит заключение и перечисление направлений дальнейших работ.

2. Типовые сценарии анализа бинарного кода

Для того чтобы определить требования к универсальному промежуточному представлению для анализа бинарного кода, необходимо рассмотреть распространенные задачи. На Рис. 1 такие задачи разнесены на разные уровни, исходя из того, решают они целевую задачу или играют вспомогательную роль. Условно можно сгруппировать их в три категории, согласно тому, какой

характер имеют действия над промежуточным представлением: абстрактная интерпретация, конкретное и символьное выполнение.

Рис. 1 – Задачи анализа бинарного кода, предполагающие использование промежуточного представления
Fig. 1 – Binary code analysis problems benefitting from use of an intermediate representation

2.1. Конкретное выполнение

Эмуляция на основе динамической двоичной трансляции (DBT) – традиционное и распространенное применение промежуточного представления, полученного из бинарного кода. Эмулируемый код линейно декодируется до передачи управления, полученный фрагмент кода (блок трансляции) переводится во внутреннее представление, над ним проводятся оптимизации, после чего представление транслируется в код хостовой машины [9]. Чтобы общая производительность эмулятора не страдала, необходимо добиться компромисса между качеством оптимизации и накладными расходами. В силу этого работа с промежуточным представлением ограничивается «легкими» оптимизациями: удалением мертвого кода, продвижением констант и копирований, анализом живых переменных.

В задачах автоматизации отладки, таких как отладка обращений к памяти, поиск гонок, и также при профилировании, трансляция дополняется инструментированием. Наиболее известны системы Valgrind [2], Pin [10] и DynamoRIO [11].

В целом, существуют два подхода к инструментированию [2]. При применении подхода “copy and annotate” проводится декодирование команд программы при ее выполнении и некоторые из них, в зависимости от решаемой задачи, заменяются вызовами вспомогательных функций, которые осуществляют дополнительную обработку, либо производится замена операндов и т.п. Так реализованы системы Pin и DynamoRIO. Второй подход, “disassemble and resynthesize”, схож с тем, как работает ранее описанная DBT с углубленным разбором операционной семантики каждой команды: инструментирование выполняется над промежуточным представлением, которое оптимизируется и используется затем для генерации кода хостовой

машины. Этот подход, реализованный в Valgrind, существенно более гибкий, но в то же время связан с повышенными накладными расходами.

От промежуточного представления в этих сценариях требуется быть удобным для проведения базовых оптимизаций: по возможности не иметь побочных эффектов и соответствовать SSA-форме; а также предоставлять простое, но емкое API к результатам декодирования команды: коду операции, операндам, режиму адресации и т.п.

2.2. Символьное выполнение

Символьное выполнение – более общая модель выполнения программы, по-прежнему ограниченная одним путем, но заменяющая конкретные значения переменных на абстрактные (символьные) значения. Как правило, формулы, задающие взаимосвязи между такими значениями, – это бесквантворные предикаты первого порядка над массивами битовых векторов (QF_ABV).

Символьное выполнение используется в задачах автоматического поиска программных дефектов и оценки их критичности. Архитектура соответствующих программных средств (S2E [12], Mayhem [13], работы А. Федотова и В. Каушана [14]) уже устоялась, она представляет собой цепочку нескольких видов анализа и преобразований кода (рис. 2).

Рис. 2 – Типовая архитектура инструмента символьного анализа
Fig. 2 – Generic architecture of a symbolic analysis tool

Выполнение программы, сопровождающееся поддержкой символьного состояния, обеспечивается средствами бинарного инструментирования (Valgrind [2], Pin [10], QEMU [9]). Динамический анализ помеченных данных отбирает команды, обрабатывающие пользовательский (символьный) ввод. Затем отобранные команды транслируются в промежуточное представление. Для промежуточного представления определяются правила символьной интерпретации, выполнение которых вырабатывает выражения над символьными переменными и константами. Для описания текущего пути выполнения все пройденные условные переходы транслируются в ограничения над символьными выражениями, что позволяет сформировать предикат пути. Предикат пути, дополненный предикатом безопасности – формальным описанием некоторого типа ошибки, предается в SMT-решатель.

Если операции промежуточного преставления оказываются «близки» к логике QF_ABV, то затраты на вторую трансляцию удастся уменьшить, а сами правила окажутся тривиальными.

Следует отметить, что возможна архитектура символьного анализа и с другим порядком действий, когда весь код транслируется в промежуточное представление и уже на нем ведется отслеживание символьных значений, без отдельного этапа отслеживания помеченных данных.

2.3. Абстрактная интерпретация

При проведении ручного анализа бинарного кода основным инструментом аналитика является дизассемблер. Дизассемблер осуществляет декодирование команд из сегмента кода одним из двух способов: либо последовательным просмотром от начала сегмента, либо методом рекурсивного спуска от точек входа [15]. В основном применяется второй способ, т.к. он сохраняет свою работоспособность в случае, когда между подпрограммами есть пропуски или данные (например, таблицы переходов). Также применяется подход спекулятивного дизассемблирования [16], когда каждое смещение в сегменте кода рассматривается как возможное начало кода функции, а затем на этапе разрешения конфликтов отбрасываются неверно выбранные.

Таким образом, для решения задачи дизассемблирования требуется, как минимум, декодер машинных команд с представлением результата в ассемблерном виде, а для подходов, основанных на рекурсивном спуске, необходима также классификация команд по признаку передачи управления и возможность вычисления целевого адреса перехода. Один из способов, как можно вычислить возможные исполнительные адреса – применение алгоритма VSA [17], способного аппроксимировать значения адресов и целых чисел. Результат его работы позволяет не только обнаружить фрагменты кода (по адресам перехода), но и улучшить результаты при поиске дефектов и уязвимостей путем статического анализа.

Для поиска дефектов и уязвимостей в бинарном коде применяются подходы, основанные на статическом анализе и символьном выполнении. Например, система BINSIDE [18] использует промежуточное представление REIL [8] для последующей трансляции в бит-код LLVM [4], поверх которого при помощи разработанных чекеров ведется поиск определенных видов дефектов.

По своей сути и статический анализ, и символьное выполнение представляют собой разновидности абстрактной интерпретации [19]. Разница заключается в том, что в первом случае одновременно исследуются все пути в программе (и, как правило, при помощи итеративного применения передаточных функций находится некоторое решение, соответствующее неподвижной точке – MFP-решение), а во втором случае исследуется некоторое подмножество путей, причем каждый путь анализируется независимо. В случае, когда удается рассмотреть все возможные пути, полученное решение (MOP-решение)

оказывается более точным, чем MFP-решение. Однако на практике просмотр всех путей возможен только для самых примитивных программ.

Вне зависимости от того, каким образом применяется абстрактная интерпретация, от инфраструктуры анализа требуется декодирование команд и представление их семантики в упрощенном виде для того, чтобы решетки и передаточные функции, которые задают абстрактную интерпретацию, были менее сложными в описании. Именно поэтому в задачах поиска дефектов и уязвимостей в бинарном коде чаще всего можно встретить использование достаточно мощных промежуточных представлений. Кроме системы BINSIDE, схожие задачи решаются в системах BitBlaze [5] и BAP [7], каждая из которых имеет свое промежуточное представление для описания семантики машинных команд.

3. Особенности целевых процессорных архитектур

Как показывает практика авторов при работе над задачами динамического анализа бинарного кода [20], все процессорные архитектуры общего назначения достаточно близки друг к другу как с точки зрения наборов команд и их кодировки, так и с точки зрения функционирования конвейера и подсистемы памяти. Существенные отличия от процессоров общего назначения проявляют микроконтроллеры и DSP-процессоры. Среди таких отличий – принадлежность к «не-фон-Неймановским» архитектурам, разрывное размещение команд в памяти, отсутствие останова конвейера при конфликте по данным или управлению и т.п. В данном разделе приведен обзор наиболее важных особенностей некоторых целевых процессорных архитектур, которые должны быть учтены при разработке промежуточного представления для анализа бинарного кода.

3.1. Семейство ARM

Семейство процессорных архитектур ARM в настоящее время представлено архитектурами ARMv7 и ARMv8, которые в свою очередь имеют деление на профили “A”, “R” и “M”. В зависимости от модели, процессор ARM может поддерживать до трех наборов команд: наборы A32 и A64 с фиксированной длиной команд и набор T32 с переменной длиной команд. В каждый момент времени текущий используемый набор команд определяется управляющими регистрами машины и может быть изменен во время работы при помощи специальных команд.

Среди особенностей можно выделить то, каким образом реализовано условное выполнение команд. В наборе A32 практически все команды имеют поле cond, определяющее, при каких условиях команда будет выполняться. В наборе A64 только небольшая часть команд имеет такое поле. В наборе T32, помимо нескольких команд, которые имеют явное поле условия, имеется команда IT,

которая в зависимости от вычисленного значения условия определяет, какие из команд следующего условного блока (4 команды) будут выполнены при истинном значении, а какие – при ложном.

Другой особенностью является наличие сопроцессоров, работа с которыми ведется при помощи команд MCR и MRC. Специализированные решения на базе архитектуры ARM могут иметь помимо стандартных сопроцессоров произвольные расширения.

3.2. Семейство AVR

Семейство микроконтроллеров AVR имеет очень небольшой набор команд. Команды могут иметь длину в 16 или 32 бита, длина команды может быть однозначно определена по битовому префиксу. Микроконтроллеры AVR имеют отдельное пространство памяти кода и памяти данных, причем часть последнего используется для доступа к периферии. Таким образом, основной особенностью является принадлежность этих микроконтроллеров к «не-фон-Неймановским» архитектурам.

3.3. Семейство MIPS

Семейство процессорных архитектур MIPS является характерным примером RISC-архитектуры: команды имеют фиксированную длину с простой структурой и имеют достаточно простую семантику, благодаря чему легко могут быть декодированы и проанализированы с точки зрения их поведения. Однако в силу того, что существует большое число версий архитектуры MIPS и множество различных расширений, без точного знания версии и набора реализованных расширений некоторые команды не могут быть декодированы однозначно. Номер версии архитектуры может быть считан из системных регистров (при этом некоторые эмуляторы не вполне точно реализуют этот функционал), но набор расширений, поддерживаемый данной машиной, необходимо получать извне.

Конвейер MIPS работает со слотами задержки, обработка команд передачи управления должна осуществляться с учетом этой особенности.

3.4. Семейство PowerPC

Процессорные архитектуры семейства PowerPC также являются яркими представителями RISC-архитектур: все команды имеют фиксированную длину с небольшим количеством форматов кодировки (т.е. расположения полей). Семантика команд несложна, и единственной отличительной особенностью является возможность явного указания при арифметических командах флага, включающего или выключающего обновление слова состояния в регистре *XER*.

3.5. Семейство RISC-V

Новое семейство процессорных архитектур RISC-V было спроектировано таким образом, чтобы максимально упростить реализацию «в железе» процессоров этой архитектуры и инструментов, которые работают с бинарным кодом RISC-V. Команды имеют равную длину, имеется четкое разбиение на подмножества команд (каждая реализация может поддерживать только часть из них), причем принадлежность команды к определенному подмножеству может быть определена по битовой маске. Большая часть подмножеств работает всего с четырьмя форматами кодировок команд, что делает декодирование несложным. Однако имеется одно исключение: расширение “C” набора команд дополняет «обычные» 32-разрядные кодировки команд 16-разрядными кодировками, которые позволяют кодировать некоторые команды базового набора более кратко. Длина команды всегда определяется двумя битами в кодировке.

3.6. Семейство Texas Instruments

Семейство процессорных архитектур Texas Instruments (TMS) представлено обширным набором микроконтроллеров. Наиболее яркие особенности имеют микроконтроллеры с явным параллелизмом на уровне команд, например TMS320C67x. Здесь единицей выборки является не одна команда, а пакет, состоящий из восьми команд. Кодировка пакета указывает, какие из этих команд могут быть выполнены параллельно. Интересным моментом является то, что процессор допускает передачу управления в середину пакета, что соответствует частичному его выполнению.

Все процессорные архитектуры TMS имеют слоты задержки, соответственно, команды передачи управления требуют специализированной обработки.

3.7. Семейство x86

Процессорные архитектуры семейства x86 (в том числе x86-64) в настоящее время наиболее распространены среди процессорных архитектур общего назначения и, тем самым, бинарный код для этих архитектур чаще всего становится объектом анализа. Семейство x86 относится к CISC-архитектурам, имеет огромное число команд, множество режимов адресации, несколько вариантов кодировки одних и тех же команд (например, с использованием VEX-префикса или без него), переменную длину команды, которая не может быть вычислена по какому-либо префиксу фиксированного размера, и т.п. Кроме того, многие команды (особенно системные) весьма сложны, что требует их упрощения при анализе путем разбиения на более мелкие составные части. Таким образом, полноценная поддержка целевой архитектуры x86 является непростой задачей с точки зрения реализации декодера команд и описания их операционной семантики.

С другой стороны, конвейер x86-процессоров с точки зрения разработчика достаточно прост и не имеют каких-либо существенных особенностей, влияющих на анализ кода. То же можно сказать и про систему памяти: ее сложность заключается лишь в количестве последовательных трансляций адресов, но каждый отдельный этап не является сложным. Определенным исключением из сказанного являются новые наборы команд, связанные с организацией транзакционной модели памяти (TSX) и защищенных областей памяти (SGX), однако в зависимости от решаемой задачи их обработка часто может быть существенно упрощена без потери точности результатов.

3.8. Семейство Xtensa

Семейство микроконтроллеров Xtensa достаточно популярно при реализации устройств интернета вещей, поскольку решения на базе Xtensa предоставляют доступ к 802.11-сетям (WiFi) и Bluetooth «из коробки». Базовый набор команд очень невелик (менее 100 простых команд), команды имеют фиксированную длину в 24 бита и простую кодировку. Подобно тому, как устроено расширение “C” в RISC-V, некоторые микроконтроллеры Xtensa поддерживают короткую запись длиной 16 бит для части команд. В некоторых моделях поддерживаются аппаратные циклы: регистр *LBEG* определяет адрес начала цикла, регистр *LEND* – адрес конца, а регистр *LCOUNT* содержит число оставшихся операций. При выборке команд процессор автоматически осуществляет декремент значения *LCOUNT* при достижении адреса *LEND* и при необходимости передает управление на адрес *LBEG*. Это поведение необходимо учесть при анализе потока управления кода для Xtensa.

Процессоры семейства Xtensa также могут работать в связке с внешними сопроцессорами, поэтому для полноценного анализа бинарного кода этой целевой архитектуры необходимо знать, какие команды добавляют к базовому набору эти сопроцессоры.

3.9. Требования к промежуточному представлению

Подводя итоги перечисленных выше особенностей, можно сформулировать набор требований, которым должно удовлетворять представление для анализа бинарного кода, чтобы оно было пригодно для разностороннего анализа всех перечисленных процессорных архитектур.

I. Требуется поддержка «не-фон-Неймановских» архитектур, имеющих несколько адресных пространств памяти.
II. Некоторые архитектуры поддерживают аппаратные циклы и другие механизмы, влияющие на выборку команд. Как следствие, наличие таких механизмов необходимо учитывать при интерпретации и анализе потока управления.

III. Многие RISC-архитектуры имеют слоты задержки, причем их число и поведение команд передачи управления с точки зрения выполнения или отбрасывания команд в слотах задержки в разных архитектурах может отличаться. Такая информация тоже влияет как на интерпретацию, так и на анализ потока управления.

IV. Необходимо поддержать возможность работы с кодировками команд переменной длины.

V. Во многих процессорных архитектурах декодирование команды зависит от значений управляющих регистров (например, текущий набор команд в ARM, признак разрядности кода в x86 и т.п.). Декодер должен иметь доступ к контексту, откуда могут быть получены эти сведения.

VI. Некоторые процессорные архитектуры имеют различные версии и наборы расширений. Эта информация также должна использоваться при декодировании, т.к. иначе однозначное декодирование команд невозможно.

VII. При декодировании необходима поддержка процессорных архитектур с явным параллелизмом на уровне команд.

4. Декодирование машинных команд

Любое промежуточное представление для анализа бинарного кода стоится в два этапа, первым из которых является декодирование команд анализируемой программы или системы, а вторым собственно трансляция их в некоторую единообразную форму, которая в дальнейшем и анализируется. В данном разделе обсуждается первый из этих этапов с учетом тех требований, которые были сформулированы выше в подразделе 3.9.

4.1. Обзор существующих решений

Среди существующих программных средств, которые могут использоваться для декодирования бинарного кода, можно выделить две основные группы. К первой группе относятся декодеры, которые обеспечивают поддержку какой-либо одной целевой архитектуры (или семейства). Такие средства имеют, как правило, достаточно детальное представление разобранной команды: не только текстовый вид команды, но и ее структура с точки зрения мнемоники, префиксов, флагов, режимов адресации операндов и т.п. В то же время тот факт, что для различных целевых архитектур нужно использовать различные библиотеки декодирования с различными программными интерфейсами означает, что в инструментах, настраиваемых по целевой процессорной архитектуре, потребуется реализация дополнительного уровня унификации этих интерфейсов, что может нивелировать выгоду от использования готовой библиотеки.

Вторая группа представлена средствами, поддерживающими множество целевых процессорных архитектур и формирующими результат в виде универсальных структур, не зависящих от текущей архитектуры. На данный момент наиболее часто встречается использование библиотек из состава binutils [21], библиотеки Capstone [22] и декодирование команд средствами интерактивного дизассемблера IDA Pro [23]. Рассмотрим возможности, предоставляемые этими средствами, более подробно.

Библиотека декодирования из состава binutils в основном используется для получения текстового представления декодированной команды. Логика декодирования описана в виде функций на языке Си без использования внешних спецификаций и чаще всего представлена в виде каскада операторов *switch*, что делает отладку декодеров и внесение изменений затруднительным. Основным достоинством binutils является обширный набор поддерживаемых целевых архитектур, однако качество такой поддержки плавает: для основных архитектур общего назначения декодеры хорошо отлажены, но качество кода для декодирования команд микроконтроллеров существенно ниже.

Библиотека Capstone позволяет получать не только текстовый вид команды, но и структуру, где единообразно (т.е. в независимом от целевой архитектуры виде) описаны ее основные свойства. Среди таких свойств принадлежность команды к определенной группе (например, к командам передачи управления) и список читаемых и записываемых командой регистров. Поддерживается несколько целевых архитектур, в т.ч. ARM, M68K, MIPS, PowerPC, TMS320C64x, x86 и др. В то же время библиотека имеет ряд архитектурных ограничений, которые не позволяет с ее помощью решить все задачи, связанные с декодированием машинных команд. Некоторые наиболее существенные ограничения перечислены далее.

I. Детальная информация о команде доступна для каждой целевой архитектуры в виде своей структуры данных, что существенно уменьшает степень универсальности библиотеки с точки зрения интерпретации результатов декодирования.
II. Даже в рамках структур с детальной информацией отсутствуют сведения о количество и способе обработки слотов задержки, а такая информация существенна для всех потоково-чувствительных видов анализа бинарного кода.
III. Отсутствует поддержка архитектур с явным параллелизмом на уровне команд.
IV. Нет возможности получить список участков памяти, с которыми работает команда. Таким образом, анализ потока данных при использовании этой библиотеки ограничен регистрами.

Реализация декодеров в Capstone выполнена в виде кода на языке Си с обширным применением макросов, что делает поддержку кода непростой задачей.

Таким образом, библиотека Capstone хорошо подходит для поверхностного анализа бинарного кода (в частности, задач дизассемблирования и простых случаев анализа потоков управления и данных), но не пригодна в текущем виде для углубленного анализа.

Интерактивный дизассемблер IDA Pro является в настоящий момент инструментом выбора при автоматизированном исследовании бинарного кода, когда пользователь работает с восстановленным по бинарному кода ассемблерным листингом. Дизассемблер поддерживает множество целевых архитектур и имеет SDK для реализации такой поддержки в виде подключаемых модулей. Однако качество существующих декодеров сильно отличается от модуля к модулю. В частности, информация о читаемых и записываемых командой регистрах доступна лишь для части целевых архитектур, в другой части она не реализована или работает некорректно. Что касается предлагаемого SDK, его длинная история и тот факт, что изначально интерфейсы IDA Pro проектировались только для кода архитектуры x86, привели к тому, что программные интерфейсы являются достаточно запутанными, имеют множество неявных контрактов и т.п.

Каждый модуль декодирования реализуется на языке Си++ или Python и, так же, как и в перечисленных выше средствах, логика декодирования описана прямо в коде. Единого способа декларативного описания набора команд не предусмотрено.

Таким образом, перечисленные средства декодирования, являясь отличными решениями для определенных узких задач, не являются универсальными. В следующем подразделе предлагается подход к декодированию, основанный на декларативном задании набора команд, который является более гибким и, по мнению авторов, легче поддерживаемым.

4.2. Предлагаемый подход к декодированию

При разработке предлагаемого подхода к декодированию помимо тех требований, которые вытекают из особенностей процессорных архитектур, в качестве основополагающего также выдвигалось требование получения унифицированного декодера, т.е. такого который обрабатывает декларативно заданную спецификацию конкретной архитектуры набора команд (ISA) и предоставляет для всех архитектур одинаковый программный интерфейс.

Команда	**Кодировка (младшие биты справа)**	**Предикат**
lw x8, 0(x9)	000000000000_01001_010_01000_0000011	
lw x9, 1(x8)	000000000001_01000_010_01001_0000011	
ld x8, 0(x9)	000000000000_01001_011_01000_0000011	поддержка команд RV64I
or x6, x5, x4	0000000_00100_00101_110_00110_0110011	
ori x6, x5, 4	000000000100_00101_110_00110_0010011	

Рис. 3. Фрагмент таблицы декодирования для RISC-V
Fig. 3. RISC-V decode table fragment

Вообще говоря, с логической точки зрения спецификация кодировок команд некоторого набора может быть описана в виде таблицы, которая ставит в соответствие каждой конкретной команде полную кодировку (т.е. такую, где все биты имеют конкретные значения) и предикат, накладываемый на текущее состояние машины. Фрагмент такой таблицы для процессоров RISC-V приведен на рис. 3. Предикат необходим, чтобы учесть переключаемые во время выполнения наборы команд (например, в процессорах ARM), размеры операндов по умолчанию (например, в процессорах x86), состав расширений архитектуры и другие подобные характеристики, не присутствующие явно в кодировке команды. Понятно, что подобная таблица будет иметь огромный размер, поэтому при проектировании логики декодирования основной целью является структуризация этой таблицы путем группировки ее строк. Если временно отбросить необходимость работы с предикатом, то остается набор битовых строк, которым соответствуют различные команды. Структуры данных для быстрого поиска в таких массивах данных хорошо известны, одной из них является сжатый бор, который группирует битовые строки по общим префиксам (рис. 4). Однако в реальности весьма часто встречаются ситуации, когда определить, например, мнемонику команды по нетривиальному префиксу невозможно. В частности, в архитектуре RISC-V многие команды имеют разбитое на две части поле кода операции: часть битов располагается в начале, а часть – в самом конце кодировки. Таким образом, требуется модификация структуры бора для адаптации к решаемой задаче.

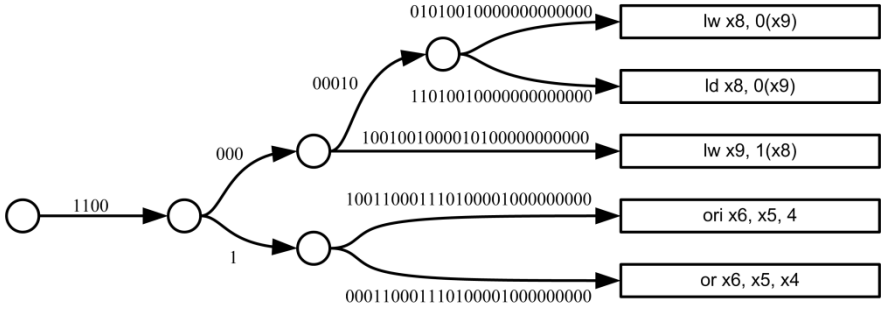


Рис. 4. Фрагмент сжатого бора для RISC-V
Fig. 4. RISC-V compressed trie fragment

Заметим, что можно рассматривать отдельные битовые поля из кодировок команд не обязательно последовательно. Если разбить кодировку команды на отдельные поля, то, вообще говоря, можно проводить сравнения битов в этих полях с записанными в таблице в произвольном порядке. Такой подход может быть формализован в виде автомата и соответствующего ему дерева, где на ребрах указаны значения битовых полей, совпадение с которыми разрешает переход в соответствующую вершину. Еще раз отметим, что порядок ребер

вдоль пути от корня дерева к листу не имеет значения, т.е. такие деревья можно преобразовывать путем перестановки вершин, не меняя задаваемые ими кодировки.

Структура кодировок команд в процессорах, как правило, такова, что сначала может быть декодирована некоторая общая форма команды (код операции вместе с модификаторами, набор режимов адресации и размеры операндов), а далее определены и сами операнды (т.е. конкретные регистры, адреса памяти и т.п.). Такая структура объясняется тем, как обычно «в железе» реализуются начальные этапы конвейера. Исходя из этих соображений, разобьем дерево на два слоя. В верхнем (ближе к корню) слое сосредоточим проверку битовых полей, определяющих код операции и режимы адресации операндов, а в нижнем (ближе к листьям) – связанные с окончательным определением операндов. Если построить такое двухслойное дерево, то можно будет заметить, что нижний уровень будет содержать большое количество повторяющихся шаблонов. Перейдя от дерева к ациклическому графу, эти повторы можно ликвидировать. Однако при этом только часть результата декодирования может храниться в листьях, а часть должна накапливаться в процессе прохода по верхнему слою. Пример такого графа показан на Рис. 5, для того, чтобы не перегружать иллюстрацию, ребра между слоями не показаны. В левой части рисунка находится слой декодирования кода операции и режимов адресации операндов, в правой части – слой декодирования самих операндов. При этом набор операндов, которые будут декодироваться, определяется листовой вершиной первого слоя.

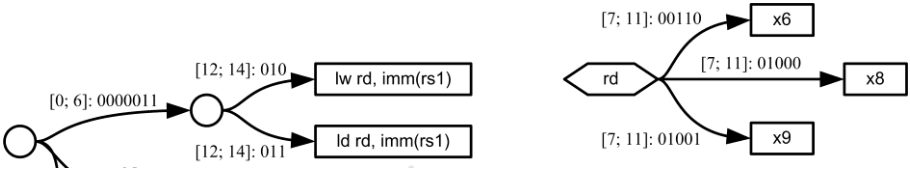


Рис. 5. Фрагмент ациклического графа декодера RISC-V
Fig. 5. RISC-V decode acyclic graph fragment

Вернемся теперь к влиянию состояния машины на декодирование команд. Заметим, что, по сути, соответствующий предикат фигурирует как конъюнкт в общем виде кодировки наряду с конъюнктами, отвечающими за проверку значений битовых полей. Тогда такие предикаты могут быть добавлены в рассматриваемый граф в виде еще одного слоя, наиболее близкого к корню. Правило перехода по ребрам в этом слое следующее: переход возможен, если указанный на ребре предикат на значения частей системных регистров и характеристики конкретной модели процессора принимает истинное значение.

Если описанный граф для какого-либо процессора построен, то результатом декодирования команды является совокупность информации в вершинах, отвечающих предикату и битовой кодировке команды или, говоря иначе, путь

на этом графе. С частями графа, описывающими декодирование операндов, можно связать соответствующие режимы адресации. Для каждого режима адресации можно определить набор характеристик операнда, которые в рамках этого режима не являются фиксированными. В примере для RISC-V для режима адресации «регистр» такой характеристикой будет номер регистра, для режима «константа» – номер константы, а для режима «косвенная адресация» – номер регистра и смещение. Тогда часть пути в графе, проходящая по слою кода операции, задает мнемонику и режимы адресации операндов, а часть, проходящая по слою операндов, окончательно декодирует характеристики операндов. Например, результатом декодирования битового вектора 000000000000_01001_010_01000_0000011 будет структура, содержащая следующие сведения:

- мнемоника команды: LW;
- первый операнд имеет режим адресации «регистр», характеристика «номер регистра» имеет значение “x8”;
- второй операнд имеет режим адресации «косвенная адресация»;
- характеристика «номер регистра» имеет значение “x9”, а «смещение» равно 0.

Такое представление результатов декодирования позволяет:

- построить текстовое представление команды;
- при наличии сохраненного пути по графу восстановить битовую кодировку;
- иметь возможность определять в унифицированном виде формулы, отбирающие какие-либо команды по мнемонике и набору операндов, что будет необходимо в дальнейшем при трансляции отдельных машинных команд в промежуточное представление, описывающее операционную семантику.

5. Описание операционной семантики машинных команд

В предыдущем разделе были рассмотрены вопросы декодирования. Перейдем теперь к обсуждению вида той части промежуточного представления, которая отвечает за описание операционной семантики машинных команд.

5.1. Обзор существующих решений

Как было сказано во введении, появление первых промежуточных представлений, достаточно точно описывающих операционную семантику бинарного кода, обязано в первую очередь задачам динамической бинарной трансляции и инструментирования. В частности, полносистемный эмулятор с открытым исходным кодом QEMU [9] основан на бинарной трансляции гостевого кода в код машины, на которой производится эмуляция. Трансляция работает на базе промежуточного представления TCG. Гостевой код

переводится в это представление, а затем по нему осуществляется генерация кода инструментальной машины. Аналогичный подход реализован в системе динамического бинарного инструментирования Valgrind [2], где применяется промежуточное представление VEX. Представления TCG и VEX достаточно близки по своим основным характеристикам и без существенных изменений много лет успешно применяются в своих задачах.

Однако для решения задач углубленного анализа потоков данных и управления в бинарном коде эти представления малопригодны: и TCG, и VEX имеют поддержку вызова вспомогательных функций на языке Си (helpers), которые могут произвольным образом менять состояние анализируемой системы. Эти функции встречаются достаточно часто, поскольку используются для эмуляции сложных команд целевых процессоров.

Еще одной проблемой данных представлений является отложенная обработка флагов, которая оправдана при динамическом анализе (т.к. позволяет не высчитывать значения флагов, которые в дальнейшем не будут использоваться), но существенно затрудняет статический анализ и символьное выполнение на базе такого представления, поскольку такое поведение разрывает цепочки зависимостей по данным. Кроме того, в обеих системах этапы построения промежуточного представления и кодогенерации описаны в виде модулей на языке Си, что затрудняет их модификацию и отладку.

Специализированные промежуточные представления для анализа бинарного кода, лежащие в основе таких систем, как Vine [5], BAP [7], а также представление REIL [8] не имеют неявных действий, поэтому более пригодны для различных сценариев анализа, хотя и более «многословны». Трансляторы здесь так же, как и в QEMU и Valgrind, реализованы в виде программных модулей и не поддерживают декларативное задание набора команд целевой машины. Кроме того, в этих представлениях фиксировано множество элементарных операций, на которые требуется разбивать поведение машинной команды, причем это множество ограничено: в своем текущем виде эти представления пригодны, в основном, для команд целочисленной арифметики, побитовой логики и передачи управления. В этих представлениях также нет моделей памяти и конвейера, обработки исключений, что делает их использование в задачах полносистемного анализа бинарного кода затруднительным.

Представление Pivot [6] было предложено в первую очередь для решения задач динамического оффлайн-анализа бинарного кода процессорных архитектур x86 (и x86-64), PowerPC, MIPS и ARMv6, однако впоследствии было также успешно применено для некоторых сценариев статического анализа бинарного кода этих архитектур. Ключевыми отличиями этого представления являются:

- произвольный набор адресных пространств (в том числе пространств регистров), что позволяет поддерживать «не-фон-Неймановские»

архитектуры и отделить регистры целевой машины от временных переменных (виртуальных регистров);

- временные переменные находятся в форме статического единичного присваивания, что упрощает дальнейший статический анализ;
- набор базовых операций, через которые выражается операционная семантика, не фиксирован: в зависимости от задачи пользователь может либо описать новые операции, либо выразить семантику команды через существующие;
- операции могут иметь неявные побочные эффекты, выражающиеся в чтении или записи отдельных битов слова состояния, но для каждой операции эти множества специфицированы, что является некоторым компромиссом между полностью явным описанием всех эффектов и отложенным вычислением флагов;
- реализован язык задания внешних спецификаций семантики команд, трансляторы строятся автоматически по этим спецификациям.

Несмотря на успешный опыт использования данного представления в ряде проектов ИСП РАН, со временем проявились некоторые его принципиальные ограничения. Во-первых, при его разработке не была учтена потребность в описании поведения машины «между командами», т.е. при выборе очередной команды, обработке прерываний и т.п.

Во-вторых, простая модель памяти различных адресных пространств (байтовые массивы), используемая в Pivot, в полносистемных задачах анализа оказывается недостаточной: необходимо учитывать такие особенности функционирования аппаратуры, как трансляция адресов (в том числе многоуровневая), отображенные на память устройства и прямой доступ к памяти (DMA).

В-третьих, набор библиотек, которые используются при работе с эти представлением, включает только средства трансляции и конкретной интерпретации, а всю логику последующего анализа (например, итеративный алгоритм для поиска MFP-решения задачи потока данных или логику продвижения символьного состояния) приходится реализовывать в каждом случае заново.

Наконец, среди академических работ, посвященных вопросам описания операционной семантики машинных команд, можно выделить различные диалекты языка nML [24], языки ISDL [25], L3 [26] и SAIL [27]. Наиболее интересен подход, применяемый в L3 и SAIL: здесь для описания и декодирования, и поведения команд используется диалект чисто функционального языка. Вся работа машины, по сути, описана одной чистой функцией, которая принимает на вход текущее состояние и кодировку очередной команды и возвращает новое состояние машины. Таким образом, работа машины по выполнению некоторой последовательности команд может быть описана как композиция вызовов этой функции. Код чисто

функциональной парадигмы может быть проанализирован гораздо легче, чем машинный код, поэтому применяемый функциональный язык и используется в качестве промежуточного представления.

К недостаткам здесь можно отнести ограниченную выразительную мощность перечисленных языков (в частности, имеются сложности с описанием команд, работающих с числами с плавающей точкой, векторных команд, системных команд), отсутствие возможности задания особенностей выборки команд (слоты задержки, аппаратная поддержка циклов) и обработки прерываний и исключений, т.е. действий по аппаратному переключению контекста. Кроме того, отсутствует поддержка «не-фон-Неймановских» архитектур.

5.2. Предлагаемое промежуточное представление

В настоящем подразделе описано предлагаемое промежуточное представление Pivot 2, являющееся развитием разработанного авторами ранее представления, описанного в работе [6].

5.2.1. Адресные пространства

Представление Pivot 2 единообразно описывает все данные, с которыми могут работать команды целевого процессора, в виде набора *адресных пространств*. Каждое адресное пространство считается изолированным и имеет свой фиксированный размер адреса. Одним из адресных пространств является пространство регистров: регистрам целевой машины назначаются некоторые адреса с учетом их вложенности, и в дальнейшем работа с этим пространством ведется так же, как и с пространством памяти. Для «не-фон-Неймановских» архитектур такая модель позволяет описать все используемые пространства памяти, не прибегая к трансляции адресов и другим подобным приемам.

Адресные пространства делятся на два типа: *локальные* и *удаленные*. Под локальными адресными пространствами понимаются такие, которые могут быть рассмотрены как простой массив байтов, локальный для анализируемого потока выполнения. В частности, это означает, что значение, записанное по какому-либо адресу, при последующем чтении будет получено без изменений. Кроме того, доступ к локальным адресным пространствам всегда должен заканчиваться успешно, он не может приводить к исключению. Примером локального адресного пространства является набор регистров общего назначения.

Удаленные адресные пространства, во-первых, не гарантируют какого-либо конкретного поведения при доступе (например, пространство портов ввода-вывода или пространство памяти с отображенными на память устройствами); и, во-вторых, могут заканчиваться ошибкой. Для удаленных адресных пространств помимо размера адреса задается размер битового вектора, описывающего возникшую ошибку. Такое разделение позволяет моделировать поведение команд процессора, которые могут вызывать исключения.

При работе с промежуточным представлением все адресные пространства, с которыми ведется работа, объединяются в *таблицу адресных пространств*, которая сопоставляет с каждым пространством некоторый индекс.

5.2.2. Операции

Наиболее мелкой единицей задания операционной семантики в представлении Pivot 2 является *операция*. Операция принимает на вход набор битовых векторов и формирует другой набор битовых векторов на выходе, т.е. может иметь произвольное число аргументов и результатов. Набор операций не фиксирован, что предоставляет возможность относительно несложного расширения промежуточного представления. Вместе с тем, к операциям предъявляются два следующих важных требования:

- операция не имеет каких-либо входов и выходов, кроме своих формальных аргументов и результатов, т.е. является «чистой» функцией;
- в рамках одной операции зависимости по данным представляют собой полный двудольный граф: все входы влияют на все выходы.

Таким образом, некоторая функция *add*, которая принимает два 32-разрядных числа и возвращает их сумму и признак переполнения, является операцией, т.к. на каждый из вырабатываемых результатов влияют все аргументы. Функция *xchg*, которая принимает два битовых вектора некоторой длины и возвращает их в измененном порядке, не является операцией, т.к. нет влияния каждого входа на каждый выход.

Сформулированные свойства операций позволяют проводить некоторые виды анализа потока данных даже без знания семантики каждой операции, например построение срезов или анализ помеченных данных.

Все используемые при анализе операции объединяются в *таблицу операций*, где за каждой операцией закрепляется некоторый индекс.

5.2.3. Временные переменные

Так как Pivot 2 является представлением бинарного кода, оно не имеет типов, и работает только с битовыми векторами различной длины. Все временные переменные находятся в форме статического единичного присваивания, тем самым размер переменной определяется один раз в точке ее определения. Временные переменные имеют нумерацию, начиная с 1. Эти номера локальны в пределах фрагмента, структурного элемента представления, который описан далее в п. 5.2.6. Временная переменная с номером 0 воспринимается особым образом: она считается всегда определенной как битовый вектор из единственного нулевого бита.

Набор временных переменных с последовательными номерами образует *группу*, которая может быть задана как полуинтервал номеров входящих в нее переменных. Понятие группы будет использоваться в дальнейшем при обсуждении операторов и передачи данных между базовыми блоками.

5.2.4. Операторы

Оператор представляет собой абстракцию одного неделимого действия в рамках промежуточного представления. Набор операторов в предлагаемом представлении фиксирован и не зависит от специфики набора команд целевой машины. Операторы могут принимать на вход набор номеров временных переменных (возможно пустой) и формировать на выходе группу временных переменных (также возможно пустую). В отличие от операций, некоторые операторы имеют побочные эффекты, но их суть фиксирована для каждого типа оператора. Операторы либо осуществляют пересылку данных (**MIX**, **EXTRACT**, **CONCAT**, **INIT**), либо выполняют некоторые действия (**INVOKE** и **CALL**), либо описывают обращения к запоминающим устройствам целевой машины (**LOAD.L**, **LOAD.R**, **STORE.L** и **STORE.R**).

Оператор MIX используется для группировки переменных: принимаемые на вход временные переменные последовательно копируются в переменные выходной группы. Например, оператор **MIX**, имеющий на входе набор { 1, 7, 2 }, а на выходе формирующий группу, начиная с номера 10, осуществит копирование переменной 1 и переменную 10, переменной 7 в переменную 11 и переменной 2 в переменную 12. В граничном случае, когда обрабатываемый набор переменных пуст, оператор **MIX** вырождается в пустой оператор **NOP**.

Оператор EXTRACT используется для выделения части битового вектора в своей единственной входной переменной в отдельную выходную переменную. При этом полуинтервал индексов битов, ограничивающий выделяемое поле, является константным параметром оператора. Индексы не могут зависеть от временных переменных. Данный оператор позволяет проводить более точный анализ помеченных данных даже в тех случаях, когда не учитывается семантика отдельных операций.

Оператор CONCAT конкатенирует несколько битовых векторов в один результирующий. Размер формируемой переменной определяется как сумма размеров входных переменных.

Оператор INIT заносит в переменную константный битовый вектор.

Оператор INVOKE применяет указанную (заданную индексом в таблице операций) операцию к набору переменных и формирует на выходе группу, соответствующую результату операции.

Оператор CALL вызывает фрагмент промежуточного представления (заданный индексом в таблице фрагментов) как подпрограмму. Подобно оператору **INVOKE** он принимает набор входных переменных и формирует на выходе группу. При интерпретации поддерживается стек вызовов, элементами которого являются пары (*индекс вызывающего базового блока*, *номер оператора в блоке*). Таким образом, при достижении выходной вершины вызванного фрагмента управление будет возвращено на следующий оператор после оператора **CALL.**

Оператор LOAD.L описывает чтение из локального адресного пространства. Для таких операторов указывает индекс адресного пространства из таблицы, номер переменной, содержащей адрес и размер считываемого значения. Также указывается порядок байтов: little endian или big endian. Это необходимо из-за того, что даже в рамках одного набора команд могут встречаться команды, которые по-разному воспринимают порядок байтов в памяти. Результат чтения заносится в указанную временную переменную.

Оператор STORE.L описывает запись в локальное адресное пространство: указывается индекс адресного пространства, номер переменной с адресом и номер переменной с записываемым значением, а также порядок байтов.

Оператор LOAD.R используется для осуществления чтения из удаленного адресного пространства. Напомним, что чтение из удаленного адресного пространства может закончиться неуспешно, поэтому помимо тех атрибутов, которые задаются для оператора **LOAD.L**, в данном случае дается также ссылка на фрагмент-обработчик ошибки. Этому фрагменту в качестве входного параметра будет передаваться битовый вектор с описателем ошибки, размер которого задан для каждого удаленного адресного пространства.

Оператор STORE.R осуществляет запись в удаленное адресное пространство и отличается от **STORE.L** тем же, чем **LOAD.R** от **LOAD.L**.

Операторы **LOAD.R** и **STORE.R** могут быть аннотированы флагом **PROBE**, который меняет их поведение следующим образом: само обращение не производится, а лишь проверяется его допустимость. Если обращение может быть проведено успешно, то выполнение переходит к следующему оператора, а иначе передается на обработчик ошибки.

Таким образом, всего зафиксировано 10 операторов, каждый из которых имеет понятное поведение и достаточно легко может быть проанализирован, что являлось одной из целей при проектировании промежуточного представления.

5.2.5. Базовые блоки

Неразрывная с точки зрения потока управления последовательность операторов объединяется в *базовый блок*. Помимо этой последовательности, базовый блок указывает также *входную группу* локальных переменных и *выходную группу*. Когда управление передается из одного базового блока в другой, значения переменных из выходной группы первого из них копируются в переменные входной группы второго. Таким образом, если два блока соединены ребром, они должны быть согласованы по входу и выходу. Такой подход позволяет не вводить в явном виде φ-функцию, заменив ее простыми действиями, производимыми на ребрах.

Для упрощения работы с промежуточным представлением любой базовый блок имеет формально два исходящих ребра: «ложное» и «истинное», а также выделенную временную переменную, которая управляет ветвлением. Эта управляющая переменная всегда должна иметь размер в 1 бит и определяться в текущем блоке или каком-либо его доминаторе. В случае, когда в реальности

из базового блока исходит только одно ребро (безусловный переход), в качестве управляющей переменной выбирается специальная переменная с номером 0, которая всегда имеет «ложное» значение. Если базовый блок должен был бы иметь более чем 2 перехода, такой блок заменяется каскадом ветвлений.

Все анализируемые в рамках какой-либо задачи базовые блоки объединяются в *таблицу базовых блоков*, где за ними закрепляются индексы. Таким образом, ссылки на последующие блоки хранятся в виде таких индексов.

5.2.6. Фрагменты

Наиболее крупной структурной единицей промежуточного представления является *фрагмент*. Логически фрагмент представляет собой направленный граф с единственным входом и единственным выходом, т.е. гамак. Вершинами в этом графе являются базовые блоки, а ребра соответствуют возможным передачам управления. Таким образом, фрагмент может рассматриваться как подпрограмма.

Входной базовый блок может иметь непустую входную группу. В этом случае соответствующие переменные должны быть определены до начала работы с фрагментом и являются его входными параметрами. Аналогично, значения переменных из выходной группы выходного базового блока представляют собой результат работы фрагмента. Благодаря этому одни фрагменты могут вызывать другие при помощи оператора **CALL** или использоваться как обработчики исключительных ситуаций в операторах **LOAD.R** и **STORE.R**.

Фрагменты, так же, как и другие составные элементы промежуточного представления, объединяются в *таблицу фрагментов* и получают индексы для перекрестных ссылок.

5.2.7. Контексты

Все перечисленные таблицы: адресных пространств, операций, базовых блоков и фрагментов, в совокупности образуют *контекст*. Последующий анализ всегда ведется в рамках какого-либо контекста.

6. Анализ кода на базе декларативной модели процессора

Общее решение, позволяющее задать декларативную модель процессора и являющееся основной для проведения широкого спектра разновидностей анализа бинарного кода, может быть получено в результате комбинации предложенных выше в подразделах 4.2 и 5.2 подходов и идеи абстрактной интерпретации [19].

Рассмотрим сначала блоки функциональности, которые должна предоставлять модель процессора.

I. Декодирование бинарного кода целевой процессорной архитектуры. В зависимости от задачи источником бинарного кода может выступать статический образ скомпилированной программы, снимок памяти или

последовательность выполняемых команд при динамическом анализе. Результатом работы этого блока является структура, описывающая декодированную команду, в частности ее длину, мнемонику, набор операндов (в т.ч. режимы адресации и конкретные номера регистров, адреса, смещения и т.п.).

II. Трансляция одной декодированной команды или блока из нескольких команд в промежуточное представление, описанное в подразделе 5.2. Для того чтобы осуществить трансляцию, следует использовать внешние спецификации процессоров, которые с командами целевых машин сопоставляют Pivot-фрагменты. В результате может быть получен Pivot-код, описывающий поведение одной или нескольких машинных команд.

III. Оптимизация полученного Pivot-кода. Так как единицей трансляции изначально является одна машинная команда, при компоновке фрагментов неизбежны избыточные вычисления, в частности вычисление адресов операндов. Для того чтобы уменьшить объем промежуточного представления, который будет в дальнейшем анализироваться, необходимо провести следующие оптимизации в пределах фрагмента:

   i. провести свертку и продвижение констант;
   ii. исключить общие подвыражения;
   iii. исключить избыточные операторы *LOAD.L* и *STORE.L* (при этом операторы *LOAD.R* и *STORE.R*, логика которых может менять в разных сценариях анализа, вообще говоря, не подлежат исключению).

IV. Для некоторых сценариев анализа требуется также моделировать поведение машины «на стыке» команд, а именно логику выборки очередной команды, обработку исключений и т.п. Это необходимо для полносистемного анализа, а также для более точного построения графа потока управления в случаях, когда целевая архитектура имеет аппаратную поддержку циклов и т.п.

Таким образом, декларативная модель процессора состоит из следующих частей:

- декодер команд, представленный в виде, описанном выше в разделе 4.2;
- набор Pivot-фрагментов, которые задают операционную семантику отдельных машинных команд, а также таблица сопоставления этих фрагментов с результатами декодирования;
- фрагмент Pivot-кода, описывающего логику процессора по обработке исключений и выборке очередной команды, т.е. действий, которые выполняются между командами.

При наличии базы таких декларативных моделей процессоров появляется возможность единообразной работы с бинарным кодом различных целевых процессорных архитектур посредством предоставления возможности абстрактной интерпретации поверх промежуточного представления.

Зафиксируем некоторую решетку абстрактных состояний L и зададим следующие две передаточные функции:

- передаточная функция для базовых блоков ТВ отображает абстрактное состояние на входе в блок в состояние на выходе;
- передаточная функция для ребер ТЕ отображает абстрактное состояние на входе в ребро в состояние на выходе.

Обратим внимание, что эти определения предполагают задачу, в которой распространение состояния ведется в прямом направлении. В задаче с обратным направлением прохода по ребрам графа потока управления будут использоваться симметричные определения. Отметим также, что вместо функции T_B может быть задана функция T_S, работающая над отдельным оператором промежуточного представления. Функций T_B в этом случае будет строиться как композиция T_S для составляющих базовый блок операторов.

В совокупности решетка, заданные передаточные функции и начальное состояние во входной вершине дают некоторую постановку задачи анализа потока данных (и соответствующую абстрактную интерпретацию) для Pivot-фрагмента. Алгоритм, который строит MFP-решение этой задачи, не зависит от постановки, и будет являться универсальным. Более того, при указанном способе задания передаточных функций алгоритм может быть реализован двумя способами: с хранением состояний как для базовых блоков (классический вариант, описанный, например, в [28]), так и для ребер.

Указанный алгоритм, по сути, описывает подход к статическому анализу в универсальном виде. В зависимости от выбора решетки и передаточных функций он может использоваться для вычисления возможных значений переменных, статического анализа помеченных данных, поиска дефектов и т.д. Кроме того, на базе того же самого алгоритма могут быть реализованы перечисленные выше оптимизационные преобразования, которые устраняют артефакты трансляции бинарного кода в промежуточное представление.

Символьное выполнение может быть реализовано похожим способом. Так как символьное выполнение предполагает анализ каждого пути в отдельности (и, в идеале, построение MOP-решения задачи анализа потока данных), но при этом каждый выполняемый оператор также рассматривается в рамках некоторой абстрактной интерпретации, поставленная в указанном виде задача анализа потока данных может использоваться и в этом случае. Основным отличием будет необходимость реализации диспетчера, который выбирает очередной просматриваемый путь при достижении ветвления в программе.

Порядок: статика, динамика, частный случай динамики – символьное выполнение, частный случай символьного выполнения – конкретное выполнение.

Наконец, динамический анализ может рассматриваться как упрощенный вид символьного выполнения, когда для каждого ветвления известна ветвь, по которой в дальнейшем пойдет управление. Таким образом, и динамический анализ является частным случаем абстрактной интерпретации и может быть

проведен на базе описанных выше конструкций. Отдельно отметим задачу конкретной интерпретации кода: нетрудно видеть, что путем отождествления конкретных и абстрактных состояний и выбором соответствующего вида решетки L можно проводить конкретную интерпретацию на базе абстрактной.
Наконец, добавим, что в силу того, что декартово произведение решеток является решеткой, в рамках одного запуска может проводиться анализ сразу нескольких аспектов поведения программы. Это, в частности, интересно при динамическом анализе (например, одновременно с конкретным выполнением программы может проводиться анализ доступа к неинициализированным ячейкам памяти, анализ примитивов синхронизации и т.п.) и при смешанном выполнении, когда часть программы или программной системы выполняется символьно, а часть – конкретно.

7. Заключение

В данной работе систематизированы задачи и подходы в анализе бинарного кода. С учетом особенностей современных процессорных архитектур и опыта существующих средств декодирования машинных команд и описания их операционной семантики, предложено новое промежуточное представление для анализа бинарного кода. Показано, что данное представление может быть использовано единообразно в различных задачах статического, динамического анализа бинарного кода и при символьном выполнении.
К дальнейшей работе авторы относят реализацию предложенных идей в виде набора библиотек с открытым исходным кодом и формирование базы декларативных описаний распространенных процессоров.

Список литературы

[1]. Wang X., Zeldovich N., Kaashoek M. F., Solar-Lezama A. A Differential Approach to Undefined Behavior Detection. ACM Transactions on Computer Systems, vol. 33, no. 1, art. 1, 2015, 29 p. DOI: 10.1145/2699678.
[2]. Nethercote N., Seward J. Valgrind: a framework for heavyweight dynamic binary instrumentation. ACM SIGPLAN Notices, 2007, vol. 42, no. 6, pp. 89-100.
[3]. Chipounov V., Candea G. Enabling sophisticated analyses of x86 binaries with RevGen. In Proc. of the IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), 2011, pp. 211-216.
[4]. Lattner C., Adve V. LLVM: A compilation framework for lifelong program analysis & transformation. In Proc. of the international symposium on Code generation and optimization: feedback-directed and runtime optimization, 2004, pp. 75-86.
[5]. Song D., Brumley D., Yin H., Caballero J., Jager I., Kang M.G., Liang Z., Newsome J., Poosankam P., Saxena P. BitBlaze: A new approach to computer security via binary analysis. Information systems security, 2008, pp. 1-25.
[6]. Падарян В.А., Соловьев М.А., Кононов А.И. Моделирование операционной семантики машинных инструкций. Программирование, том 37, № 3, 2011, стр. 50-64.

[7]. Brumley D., Jager I., Avgerinos T., Schwartz E.J. BAP: a binary analysis platform. Computer Aided Verification, 2011, pp. 463-469.

[8]. Dullien T., Porst S. REIL: A platform-independent intermediate representation of disassembled code for static code analysis. In Proc. of the CanSecWest Conference, 2009.

[9]. Bellard F. QEMU, a fast and portable dynamic translator. In Proc. of the USENIX Annual Technical Conference, 2005.

[10]. Luk C.K., Cohn R., Muth R., Patil H., Klauser A., Lowney G., Wallace S., Reddi V.J., Hazelwood K. Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation. ACM SIGPLAN Notices, vol. 40, no. 6, 2005, pp. 190-200.

[11]. Bruening D., Amarasinghe S. Efficient, transparent, and comprehensive runtime code manipulation. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2004.

[12]. Chipounov V., Kuznetsov V. S2E: A Platform for In Vivo Multi-Path Analysis of Software Systems. In Proc. of the 16th Intl. Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS. 2011.

[13]. Cha S. K., Avgerinos T., Rebert A., Brumley D. Unleashing mayhem on binary code. IEEE Symposium on Security and Privacy (SP), 2012, pp. 380-394.

[14]. Падарян В.А., Каушан В.В., Федотов А.Н. Автоматизированный метод построения эксплойтов для уязвимости переполнения буфера на стеке. Труды ИСП РАН, том 26, вып. 3, 2014, стр. 127-144. DOI: 10.15514/ISPRAS-2014-26(3)-7.

[15]. Kruegel C., Valeur F., Robertson W., Vigna G. Static Analysis of Obfuscated Binaries. In Proc. of the 13th USENIX Security Symposium, 2004, pp. 255-270.

[16]. Ben Khadra M. A., Stoffel D., Kunz W. Speculative disassembly of binary code. In Proc. of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems, 2016.

[17]. Balakrishnan G., Reps T. Analyzing Memory Accesses in x86 Executables. In Proc. of the 13th International Conference on Compiler Construction, 2004, pp. 5-23.

[18]. Aslanyan H., Asryan S., Hakobyan J., Vardanyan V., Sargsyan S., Kurmangaleev S. Multiplatform Static Analysis Framework for Program Defects Detection. In Proc. of the CSIT Conference, 2017.

[19]. Cousot P., Cousot R. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Proc. of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, 1977, pp. 238-252.

[20]. Padaryan V.A., Getman A.I., Solovyev M.A., Bakulin M.G., Borzilov A.I., Kaushan V.V., Ledovskikh I.N., Markin Yu.V., Panasensko S.S. Methods and software tools to support combined binary code analysis. Programming and Computer Software, vol. 40, no. 5, 2014, pp. 276-287.

[21]. GNU Binutils. URL: http://sourceware.org/binutils/, дата обращения: 03.12.2018.

[22]. Capstone. URL: http://www.capstone-engine.org/, дата обращения: 03.12.2018.

[23]. IDA Pro. URL: https://www.hex-rays.com/products/ida/index.shtml, дата обращения: 03.12.2018.

[24]. Fauth A., Van Praet J., Freericks M. Describing instruction set processors using nML. European Design and Test Conference, 1995, pp. 503-507.

[25]. Hadjiyiannis G., Hanono S., Devadas S. ISDL: An instruction set description language for retargetability. In Proc. of the 34th annual Design Automation Conference, 1997, pp. 299-302.

[26]. Fox A. Improved tool support for machine-code decompilation in HOL4. In Proc. of the International Conference on Interactive Theorem Proving, 2015, pp. 187-202.
[27]. Gray K.E., Kerneis G., Mulligan D., Pulte C., Sarkar S., Sewell, P. An integrated concurrency and core-ISA architectural envelope definition, and test oracle, for IBM POWER multiprocessors. In Proc. of the 48th International Symposium on Microarchitecture, 2015, pp. 635-646.
[28]. Muchnick S.S. Advanced Compiler Design & Implementation. Morgan Kaufmann Publishers, 1997.

Next generation intermediate representations for binary code analysis

[1,2] *M.A. Solovev <icee@ispras.ru>*
[1] *M.G. Bakulin <bakulinm@ispras.ru>*
[1] *M.S. Gorbachev <sadbear@ispras.ru>*
[1,2] *D.V. Manushin <dman95@ispras.ru>*
[1,2] *V.A. Padaryan <vartan@ispras.ru>*
[1] *S.S. Panasenko <spanasenko@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia*

Abstract. A lot of binary code analysis tools do not work directly with machine instructions, instead relying on an intermediate representation from the binary code. In this paper, we first analyze problems in binary code analysis that benefit from such an IR and construct a list of requirements that an IR suitable for solving these problems must meet. Generally speaking, a universal binary analysis platform requires two principal components. The first component is a retargetable instruction decoder that utilizes external specifications for describing target instruction sets. External specifications facilitate maintainability and allow for quickly adding support for new instruction sets. We analyze some of the more common ISAs, including those used in microcontrollers, and from that produce a list of requirements for a retargetable decoder. We then survey existing multi-ISA decoders and propose our vision of a more generic approach, based on a multi-layered directed acyclic graph describing the decoding process in universal terms. The second component of an analysis platform is the actual architecture-neutral IR. In this paper we describe such existing IRs, and propose Pivot 2, an IR that is low-level enough to be easily constructed from decoded machine instructions, and at the same time is also easy to analyze. The main features of Pivot 2 are explicit side effects, SSA variables, a simpler alternative to phi-functions, and an extensible elementary operation set at the core. The IR also supports machines that have multiple memory address spaces. Finally, we propose a way to tie the decoder and the IR together to fit them to most binary code analysis tasks through abstract interpretation on top of the IR. The proposed scheme takes into account various aspects of target architectures that are overlooked in many other works, including pipeline specifics (handling of delay slots, hardware loop support, etc.), exception and interrupt management, and a generic address space model where accesses may

have arbitrary side effects due to memory-mapped devices or other non-trivial behavior of the memory system.

Keywords: abstract interpretation; binary code analysis; compiler techniques; dynamic analysis; software reverse engineering; static analysis; symbolic execution.

DOI: 10.15514/ISPRAS-2018-30(6)-3

For citation: Solovev M.A., Bakulin M.G., Gorbachev M.S., Manushin D.V., Padaryan V.A., Panasenko S.S. Next generation intermediate representations for binary code analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 39-68 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-3

References

[1]. Wang X., Zeldovich N., Kaashoek M. F., Solar-Lezama A. A Differential Approach to Undefined Behavior Detection. ACM Transactions on Computer Systems, vol. 33, no. 1, art. 1, 2015, 29 p. DOI: 10.1145/2699678.

[2]. Nethercote N., Seward J. Valgrind: a framework for heavyweight dynamic binary instrumentation. ACM SIGPLAN Notices, 2007, vol. 42, no. 6, pp. 89-100.

[3]. Chipounov V., Candea G. Enabling sophisticated analyses of x86 binaries with RevGen. In Proc. of the IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), 2011, pp. 211-216.

[4]. Lattner C., Adve V. LLVM: A compilation framework for lifelong program analysis & transformation. In Proc. of the international symposium on Code generation and optimization: feedback-directed and runtime optimization, 2004, pp. 75-86.

[5]. Song D., Brumley D., Yin H., Caballero J., Jager I., Kang M.G., Liang Z., Newsome J., Poosankam P., Saxena P. BitBlaze: A new approach to computer security via binary analysis. Information systems security, 2008, pp. 1-25.

[6]. Padaryan V.A., Solov'ev M.A., Kononov A.I. Simulation of operational semantics of machine instructions. Programming and Computer Software, vol. 37, no. 3, 2011, pp. 161-170. DOI: 10.1134/S0361768811030030.

[7]. Brumley D., Jager I., Avgerinos T., Schwartz E.J. BAP: a binary analysis platform. Computer Aided Verification, 2011, pp. 463-469.

[8]. Dullien T., Porst S. REIL: A platform-independent intermediate representation of disassembled code for static code analysis. In Proc. of the CanSecWest Conference, 2009.

[9]. Bellard F. QEMU, a fast and portable dynamic translator. In Proc. of the USENIX Annual Technical Conference, 2005.

[10]. Luk C.K., Cohn R., Muth R., Patil H., Klauser A., Lowney G., Wallace S., Reddi V.J., Hazelwood K. Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation. ACM SIGPLAN Notices, vol. 40, no. 6, 2005, pp. 190-200.

[11]. Bruening D., Amarasinghe S. Efficient, transparent, and comprehensive runtime code manipulation. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2004.

[12]. Chipounov V., Kuznetsov V. S2E: A Platform for In Vivo Multi-Path Analysis of Software Systems. In Proc. of the 16th Intl. Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS. 2011.

[13]. Cha S. K., Avgerinos T., Rebert A., Brumley D. Unleashing mayhem on binary code. IEEE Symposium on Security and Privacy (SP), 2012, pp. 380-394.
[14]. Padaryan V.A., Kaushan V.V., Fedotov A.N. Automated exploit generaton method for stack buffer overflow vulnerabilities. Trudy ISP RAN/Proc. ISP RAN, vol. 26, no. 3, 2014, pp. 127-144 (in Russian). DOI: 10.15514/ISPRAS-2014-26(3)-7.
[15]. Kruegel C., Valeur F., Robertson W., Vigna G. Static Analysis of Obfuscated Binaries. In Proc. of the 13th USENIX Security Symposium, 2004, pp. 255-270.
[16]. Ben Khadra M. A., Stoffel D., Kunz W. Speculative disassembly of binary code. In Proc. of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems, 2016.
[17]. Balakrishnan G., Reps T. Analyzing Memory Accesses in x86 Executables. In Proc. of the 13th International Conference on Compiler Construction, 2004, pp. 5-23.
[18]. Aslanyan H., Asryan S., Hakobyan J., Vardanyan V., Sargsyan S., Kurmangaleev S. Multiplatform Static Analysis Framework for Program Defects Detection. In Proc. of the CSIT Conference, 2017.
[19]. Cousot P., Cousot R. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Proc. of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, 1977, pp. 238-252.
[20]. Padaryan V.A., Getman A.I., Solovyev M.A., Bakulin M.G., Borzilov A.I., Kaushan V.V., Ledovskikh I.N., Markin Yu.V., Panasensko S.S. Methods and software tools to support combined binary code analysis. Programming and Computer Software, vol. 40, no. 5, 2014, pp. 276-287.
[21]. GNU Binutils. URL: http://sourceware.org/binutils/, дата обращения: 03.12.2018.
[22]. Capstone. URL: http://www.capstone-engine.org/, дата обращения: 03.12.2018.
[23]. IDA Pro. URL: https://www.hex-rays.com/products/ida/index.shtml, дата обращения: 03.12.2018.
[24]. Fauth A., Van Praet J., Freericks M. Describing instruction set processors using nML. European Design and Test Conference, 1995, pp. 503-507.
[25]. Hadjiyiannis G., Hanono S., Devadas S. ISDL: An instruction set description language for retargetability. In Proc. of the 34th annual Design Automation Conference, 1997, pp. 299-302.
[26]. Fox A. Improved tool support for machine-code decompilation in HOL4. In Proc. of the International Conference on Interactive Theorem Proving, 2015, pp. 187-202.
[27]. Gray K.E., Kerneis G., Mulligan D., Pulte C., Sarkar S., Sewell, P. An integrated concurrency and core-ISA architectural envelope definition, and test oracle, for IBM POWER multiprocessors. In Proc. of the 48th International Symposium on Microarchitecture, 2015, pp. 635-646.
[28]. Muchnick S.S. Advanced Compiler Design & Implementation. Morgan Kaufmann Publishers, 1997.

Тестирование правил настройки сетевого коммутатора программно конфигурируемой сети

И.Б. Бурдонов <igor@ispras.ru>[1]
Н.В. Евтушенко <yevtushenko@ispras.ru>
А.С. Косачев <kos@ispras.ru>[2]
Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. В настоящее время SDN-технология активно используется в виртуальных сетях для реализации различных служебных сервисных функций. В основе сети лежит связный неориентированных граф физических связей (англ. Resource Network Connectivity Topology, RNCT), вершинами которого являются сетевые коммутаторы (switches) и хосты (hosts). В настоящей работе рассматривается топология, когда каждый хост соединен ровно с одним коммутатором. Коммутаторы работают по таблицам правил, которые настраиваются централизованно с помощью контроллера, работающего независимо от сетевого оборудования. Настройка коммутаторов сети предназначена для обеспечения передачи пакетов из начальных хостов в конечные хосты в зависимости от значений параметров пакетов. В статье обсуждается связь настроек коммутаторов и множества реализуемых ими путей передачи пакетов в зависимости от свойств графа физических связей. Исследуется задача тестирования настройки коммутаторов. Под целью тестирования понимается подача пакетов, позволяющих «пройти» по каждому правилу каждого коммутатора хотя бы один раз, подав пакет с необходимыми параметрами. Показывается, что в общем случае не любая настройка любого коммутатора проверяема. Возможности тестирования зависят от принимаемых гипотез о работе коммутатора. В статье рассматриваются две гипотезы: гипотеза о коммутаторе предполагает, что работа коммутатора не зависит от настроек других коммутаторов; более сильная гипотеза о правиле, кроме этого, предполагает, что работы коммутатора по данному правилу не зависит от других правил в настройке этого коммутатора. После введения, в разд. 2 вводятся необходимые определения и обозначения. Раздел 3 посвящен взаимосвязи путей в сети и правил в коммутаторах. В разд. 4 рассматривается тестирование на основе гипотезы о правиле, доказываются необходимые и достаточные условия возможности проверки заданного правила заданного коммутатора. В разд. 5 рассматривается тестирование на основе гипотезы о коммутаторе и доказывается необходимое (но не достаточное) условие и достаточное (но не необходимое) условие проверяемости любой настройки коммутатора. В заключении обсуждаются проблемы, возникающие при установлении

[1] Работа частично поддержана проектом РФФИ № 17-07-00682 А.
[2] Работа частично поддержана проектом РФФИ № 16-07-01106 А.

необходимого и достаточного условия проверяемости любой настройки коммутатора, а также ставится задача определения таких условий для заданной настройки коммутатора.

Ключевые слова: виртуальная сеть; SDN-технология; хосты и коммутаторы; настройка сетевых коммутаторов; передача пакетов; реберно-простые пути; тестирование сетевых коммутаторов.

DOI: 10.15514/ISPRAS-2018-30(6)-4

Для цитирования: Бурдонов И.Б., Евтушенко Н.В., Косачев А.С. Тестирование правил настройки сетевого коммутатора программно конфигурируемой сети. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 69-88. DOI: 10.15514/ISPRAS-2018-30(6)-4

1. Введение

В настоящее время большое внимание уделяется развитию виртуальных сетей, одной из технологий предля которых является SDN-технология (Software-Defined Networking), позволяющая эффективно выполнять реализацию различных служебных сервисных функций с использованием общих ресурсов и принципов управления ими. В SDN-технологии логическое централизованное управление сетевыми устройствами осуществляется через контроллер, работающий независимо от сетевого оборудования, который может рассматриваться как определенная операционная система [1]. Как результат, SDN обеспечивает гибкую управляемость и программируемость путем разделения уровней управления и данных (уровень пересылки) через введение открытого интерфейса между этими уровнями. Несмотря на большое количество публикаций по верификации SDN-сетей, по-прежнему актуальны исследования по верификации и тестированию сетевых устройств на различных уровнях, чтобы устранить ошибки в неправильных конфигурациях или программном обеспечении, которые могут привести к сбоям сети [2]. Кроме того, даже если каждый компонент хорошо отлажен в изоляции, их композиция может столкнуться с проблемами взаимодействия, и одним из решений в данном случае является разработка методов для тестирования системы в целом.

Существующие подходы к тестированию для программируемых инфраструктур можно разделить на три группы. Подходы первой группы направлены на создание «критических» ситуаций для сети или ее компонента. Вводимые данные являются «неожидаемыми» запросами; результаты тестирования оцениваются в зависимости от способности контроллера (или коммутатора) правильно обработать запрос или отклонить его [3, 4]. Вторая группа методов относится к тестированию конформности или соответствия, когда имеется формальная спецификация тестируемого компонента и набор тестов генерируется на основе этой спецификации [5, 6]. Такие методы на основе формальных моделей в настоящее время активно используются для проверки правильности, например, критических модулей платформ

виртуализации, таких как, например, модуль размещения [7]. Подходы последней третьей группы представляют собой комбинацию формальных методов верификации и тестирования, а именно, контрпримеры, полученные при верификации, используются в качестве тестов [8, 9].

В данной работе исследуется задача тестирования работы коммутатора по его настройке. Под целью тестирования понимается подача пакетов, позволяющих «пройти» по данному правилу настройки данного коммутатора хотя бы один раз, подав пакет с необходимыми параметрами. Это возможно не для всякой настройки коммутаторов. В частности, может возникать «зацикливание», когда пакет бесконечно двигается по циклу графа связей. Достаточно очевидно, что возможности тестирования зависят от принимаемой гипотезы о работе коммутаторов. В данной работе рассматриваются две такие гипотезы и, соответственно, два способа тестирования.

Сначала исследуется тестирование отдельного правила в настройке отдельного коммутатора. Для полноты тестирования принимается (сильная) гипотеза о правиле: каковы бы ни были таблицы настройки данного коммутатора, содержащие данное правило, программа коммутатора, обрабатывающая эти таблицы, при пересылке пакетов выполняет данное правило так, как если бы других правил в таблицах не было, т.е. независимо от других правил. Иными словами, предполагается, что в программе коммутатора не может быть ошибок, сводящихся к такой зависимости и означающих, что коммутатор неправильно срабатывает по данному правилу только при наличии некоторых других правил в таблицах коммутатора, в то время как при наличии/отсутствии других правил в своих таблицах работает правильно. Если гипотеза о правиле не верна, то тестирование отдельного правила может не выявить такого рода ошибки.

Поэтому необходимо более «тонкое» тестирование, не опирающееся на гипотезу о правиле. Гораздо более реалистичной представляется более слабая гипотеза о коммутаторе: работа данного коммутатора по его правилам не зависит от правил других коммутаторов. Соответственно, тестировать нужно не отдельное правило данного коммутатора, а всю его настройку целиком как заданную совокупность его правил. Однако не любая настройка любого коммутатора проверяема, т.е. не всегда возможно при фиксированной настройке данного коммутатора так настроить остальные коммутаторы, чтобы настройка сети оказалась правильной (пакеты пересылаются, откуда надо и куда надо без зацикливаний) и давала возможность проверить каждое правило фиксированной настройки данного коммутатора. В данной статье исследует вопрос о том, при каких условиях любая настройка данного коммутатора может быть проверяема. Эти условия, естественно, налагаются на граф физических связей сети, разбиение узлов сети на хосты и коммутаторы, выделение начальных и конечных хостов, а также место в сети данного коммутатора.

Структура статьи следующая. В разд. 2 вводятся необходимые определения и обозначения. Разд. 3 посвящен взаимосвязи путей в сети и правил в коммутаторах. В разд. 4 рассматривается тестирование отдельного правила настроки коммутатора, здесь доказывается ряд утверждений, в частности, необходимые и достаточные условия возможности проверки заданного правила заданного коммутатора. В разд. 5 рассматривается тестирование всей сосокупности правил настройки коммутатора, доказывается необходимое (но не достаточное) и достаточное (но не необходимое) условия проверяемости любой настройки данного коммутатора. В заключении обсуждаются проблемы, возникающие при установлении необходимого и достаточного условия проверяемости любой настройки коммутатора, а также ставится задача определения таких условий для заданной настройки коммутатора.

2. Определения и обозначения

В работе рассматривается сеть, состоящая из коммутаторов, хостов и физических связей между ними, которая задается с помощью конечного неориентированного связного графа без кратных ребер и петель $G = (V, E)$ где $E \subseteq \{ \{a, b\} \mid a \in V \,\&\, b \in V \,\&\, a \neq b \}$, называемого далее *графом физических связей* (*Resource Network Connectivity Topology, RNCT*). Вершины в графе G разделяются на хосты и коммутаторы: $V = H \cup S$, $H \cap S = \varnothing$. Каждый хост соединен только с одним коммутатором: $\forall\, h \in H\ \boldsymbol{deg}(h) = 1 \,\&\, \exists\, s \in S\ \{h, s\} \in E$, где $\boldsymbol{deg}(x)$ – степень вершины x. Предполагается, что каждое ребро может передавать пакеты в двух направлениях. Заметим, что если граф G не связен, то это означает только, что каждую компоненту связности можно рассматривать как самостоятельную сеть, которая тестируется отдельно. В рамках данной статьи будем считать, что G, H и S фиксированы и не будем указывать их в используемых ниже обозначениях функций, зависящих от них. Для вершины $a \in V$ обозначим её открытое множество соседей (*open neighborhood*) через $N(a) = \{\ b \in V \mid \{a, b\} \in E\ \}$.

В настоящей работе мы будем исходить из следующего основного предположения: выходные порты, по которым коммутатор посылает принятый им пакет, зависят от многих параметров, и в частности, могут зависеть от порта коммутатора, по которому принимается пакет.

Мы предполагаем, что тестер может:

1) настраивать коммутаторы через контроллер,
2) посылать пакеты из хостов,
3) наблюдать передачу пакетов между любыми двумя соседними узлами сети.

Таким образом, примером тестируемой системы является система, обведенная пунктирной линией на рис. 1 [10]. Входными символами для системы являются множества путей, которые необходимо реализовать, а выходными

символами – пути, реализованные на графе физических связей, которые наблюдаются при мониторинге.

Хосты и коммутаторы. Вершины в графе разделяются на хосты и коммутаторы. Хост может генерировать передаваемые пакеты и передавать их в единственный коммутатор, который связан с этим хостом [10]. Коммутаторы только передают принятые пакеты; более того, мы предполагаем, что коммутатор не изменяет заголовок принятого пакета. Иными словами, заголовок пакета не изменяется при передаче по сети.

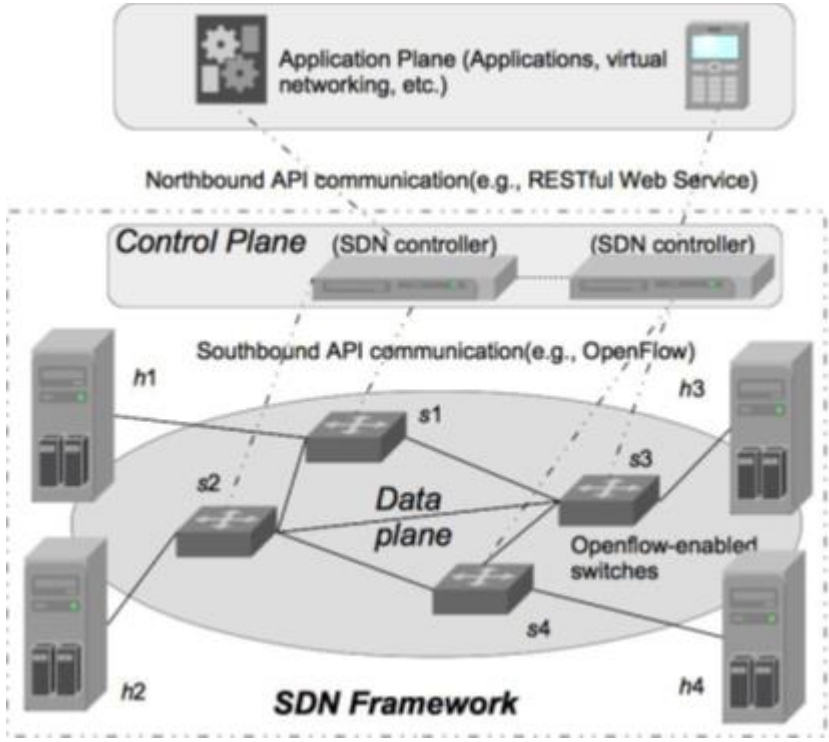


Рис.1. Пример SDN-структуры [10]
Fig. 1. SDN structure example [10]

Пути. Поскольку в графе G нет кратных ребер, путь p будет пониматься как последовательность соседних вершин графа G. Формально: путь – это последовательность вершин такая, что вершины, соседние в последовательности, являются соседними в графе (соединены ребром). Будем говорить, что путь $p = x_1, ..., x_n$ начинается в вершине x_1, заканчивается в вершине x_n, ведет от x_1 до x_n, имеет длину n-1, и проходит по дуге (a, b), если $a = x_i$ и $b = x_{i+1}$ для некоторого $i = 1 .. n$-1. Если путь проходит по дуге (a, b) или

дуге (b, a), то будем говорить, что путь проходит по ребру $\{a, b\}$. Если путь $p = x_1, ..., x_n$ заканчивается в той вершине, в которой начинается путь $q = y_1, ..., y_k$, то есть $x_n = y_1$, то конкатенацией этих путей будем называть путь $p▪q = x_1, ..., x_n, y_2, ..., y_k$ для $k > 1$ или путь $p▪q = p$ для $k = 1$. Путь называется *реберно-простым*, если он проходит по каждой дуге (т.е. по каждому ребру в каждом направлении) не более одного раза: $(x_i, x_{i+1}) = (x_j, x_{j+1}) \Rightarrow i = j$. Путь называется вершинно-простым, если все его вершины разные: $x_i = x_j \Rightarrow i = j$. Путь будем называть *правильным*, если вершина-хост является только началом и/или концом пути. Правильный путь будем называть *полным*, если он начинается и заканчивается в хосте.

На произвольном множестве P путей вводится отношение частичного порядка «является отрезком»: путь q является отрезком пути $p▪q▪r$. Обозначим через $P^{\#}$ множество отрезков длиной не меньше 2 путей из множества P: $P^{\#} = \{ q \mid \exists p, r (p▪q▪r \in P \,\&\, |q| > 2) \}$. Множество максимальных путей во множестве путей P обозначим $P^{max} = \{ q \mid \forall p, r (p▪q▪r \in P \Rightarrow |p| = |q| = 1) \}$. Если множество P состоит из путей длиной не менее 2, то $P^{\# max} = P^{max}$.

Вектор значений параметров. По каждому пути передаются пакеты с заголовками, согласно вектору значений параметров [11]; поэтому можно считать, что каждому пути поставлен в соответствие некоторый вектор значений параметров. Для заданного вектора значений параметров определены два множества $H_0 \subseteq H$ и $H_1 \subseteq H$ *начальных* и *конечных хостов*, из которых можно передавать и в которые должен прийти пакет, имеющий заголовок с данным вектором значений параметров: множество H_0 содержит все хосты, в которых могут генерироваться пакеты; множество H_1 содержит все хосты, в которые должны приходить пакеты.

В настоящей работе мы предполагаем, что реализация двух множеств путей, помеченных различными векторами значений параметров, осуществляется независимо. Поэтому в дальнейших рассуждениях мы рассматриваем свойства множества путей и правил для заданного вектора значений параметров, который не будем указывать в используемых ниже обозначениях функций, зависящих от него.

Порты. Будем считать, что с каждой вершиной a графа ассоциированы множество входных портов и множество выходных портов, причём каждый входной (выходной) порт соответствует некоторому одному ребру, инцидентному вершине a, и, наоборот, каждому ребру, инцидентному вершине a, в вершине a соответствуют её входной и её выходной порты. Тем самым, между множеством инцидентных вершине рёбер, множеством её входных портов и множеством её выходных портов существует взаимно-однозначное соответствие. Поскольку в графе G нет кратных ребер и петель, каждому ребру, инцидентному вершине a, взаимно-однозначно соответствует соседняя с a вершина. Поэтому без ограничения общности мы можем вместо идентификатора порта вершины a использовать идентификатор соседа вершины a.

Таким образом, в настоящей работе мы предполагаем, что через контроллер таблицы коммутаторов настраиваются так, чтобы (для данного вектора значений параметров) каждое правило в таблице определяло выходные порты, по которым посылается пакет, только в зависимости от порта, по которому принят пакет. Поскольку в графе G нет кратных ребер, то это равносильно тому, что в коммутаторе определяется соседний коммутатор, которому нужно передать пакет, принятый от предыдущего коммутатора. Мы также предполагаем, что коммутаторы *«чистые»*, т.е. реализуют в своих таблицах только то, что мы через контроллер попросили сделать.

Правила. Мы предполагаем, что коммутатор работает по правилам, отражаемым в его таблицах. Опираясь на основное предположение модели, *примитивным правилом* (для данного вектора значений параметров) будем считать тройку $(a, s, b) \in V \times S \times V$, где a и b – соседи коммутатора s. Такое правило говорит, что коммутатор s, получая пакет (с данным вектором значений параметров) от соседа a, должен отправить его соседу b. Если есть несколько правил, отличающихся только соседом b, то это значит, что коммутатор s выполняет *клонирование*, т.е. принятый пакет посылается сразу нескольким соседям. Правило в таблице может быть не примитивным, т.е. описывать множество примитивных правил. Например, правило (all, s, b) означает, что пакет (с данным вектором значений параметров), принятых коммутатором s от любого соседа, посылается соседу b. При выполнении правил в коммутаторах в общем случае присутствуют приоритеты. Однако приоритетами можно пренебречь, если использовать достаточно простую редукцию. Пусть есть два правила, выражаемые двумя множествами примитивных правил: A и B, и первое правило приоритетнее второго. При таких условиях остаются примитивные правила $A \cup \{ (a, s, b) \in B \mid \forall\, b' \, (a, s, b') \notin A \}$, т.е. в настройке коммутатора остается только множество равно приоритетных примитивных правил.

Множество правил коммутатора s называется *настройкой коммутатора* s и обозначается $T_s \subseteq N(s) \times \{s\} \times N(s)$. Объединение $T = \cup \{ T_s \mid s \in S \}$ настроек всех коммутаторов сети называется *настройкой сети*, при этом через $T(s) = \{ (a, s, b) \in T \}$ обозначим настройку коммутатора s как часть настройки сети T.

3. Взаимосвязь путей и правил

В этом разделе мы обсуждаем (для пакетов с данным вектором значений параметров) два вопроса: 1) какие пути будут проходить пакеты при заданной настройке сети, 2) при какой настройке сети пакеты будут проходить заданное множество полных путей.

Правила порождают пути. Правило (a, b, c) порождает путь a, b, c длиной 2. Если правилами порождены два пути вида $p{\cdot}(x, y){\cdot}q$ и $p`{\cdot}(x, y){\cdot}q`$, то эти правила порождают также пути $p{\cdot}(x, y){\cdot}q`$ и $p`{\cdot}(x, y){\cdot}q$, отличные от пути (x, y),

по которым также пройдут пакеты. Таким образом, настройка сети T порождает следующее множество путей, которое будем обозначать $T\uparrow$: $\forall\, a, b, c, x, y, p, q, p`, q`$

(1) $(a, b, c) \in T \quad \vdash (a, b, c) \in T\uparrow,$

(2) $(|p| > 1 \vee |q`| > 1)\ \&\ p{\cdot}(x, y){\cdot}q \in T\uparrow\ \&\ p`{\cdot}(x, y){\cdot}q` \in T\uparrow \quad \vdash p{\cdot}(x, y){\cdot}q` \in T\uparrow.$

Правила порождают только правильные пути длиной не менее 2, и вместе с каждым порожденным путем порождают все его отрезки длиной не менее 2: $T\uparrow = T\uparrow^{\#}$. Если правила порождают полный путь, то он, очевидно, является максимальным во множестве порожденных путей. Обратное, однако, не всегда верно. Если есть правило (a, b, c), но в коммутаторе c нет правила вида (b, c, e), то путь a, b, c может быть только конечным отрезком порождаемого пути, а такой путь не является полным, поскольку c коммутатор. В этом случае даже если пакет пройдет путь a, b, c, после этого он будет потерян. Если есть правило (a, b, c), но в коммутаторе a нет правила вида (e, a, b), то путь a, b, c может быть только началом порождаемого пути, а такой путь не является полным, поскольку a коммутатор. В этом случае для прохождения пути a, b, c пакет должен был бы генерироваться в коммутаторе a, что противоречит тому, что пакеты генерируются только в хостах. Правила могут порождать «зацикливание» пакетов, т.е. бесконечные пути: например, три правила (a, b, c), (b, c, a) и (c, a, b) порождают цикл a, b, c, a, b с повтором дуги (a, b) и, следовательно, порождают бесконечный путь $a, b, c, a, b, c, \ldots$

Правильные пути порождают правила. Для того чтобы пакет мог пройти правильный путь $p{\cdot}(a, b, c){\cdot}q$, необходимо правило (a, b, c). Формально множество P правильных путей порождает следующее множество правил, которое будем обозначать $P\downarrow$: $\forall\, a, b, c, p, q$

(3) $p{\cdot}(a, b, c){\cdot}q \in P \quad \vdash (a, b, c) \in P\downarrow.$

Согласно правилам вывода 1-3, заданное множество P правильных путей порождает множество путей $P\downarrow\uparrow$ следующим образом:

$\forall\, a, b, c, x, y, p, q, p`, q`$

(4) $p{\cdot}(a, b, c){\cdot}q \in P \quad \vdash (a, b, c) \in P\downarrow\uparrow,$

(5) $(|p| > 1 \vee |q`| > 1)\ \&\ p{\cdot}(x, y){\cdot}q \in P\downarrow\uparrow\ \&\ p`{\cdot}(x, y){\cdot}q` \in P\downarrow\uparrow \quad \vdash p{\cdot}(x, y){\cdot}q` \in P\downarrow\uparrow.$

Как отмечено выше, мы предполагаем, что пакеты должны генерироваться и приниматься без дальнейшей пересылки только в хостах [10]; поэтому нас будут интересовать полные пути, т.е. правильные пути, которые начинаются и заканчиваются в хостах. Определим понятие правильной настройки, при которой, во-первых, все максимальные пути в $T\uparrow$ полные (полный путь всегда максимален среди правильных путей) и ведут из начальных хостов во все конечные хосты, а, во-вторых, любой порождаемый путь является отрезком максимального пути, т.е. не происходит «зацикливания». Такое «зацикливание» возникает тогда, когда пакет дважды проходит дугу. Поэтому

для того, чтобы не было «зацикливаний», необходимо и достаточно, чтобы все пути в $T\uparrow$ были реберно-простые. Для заданных множеств H_0 начальных и H_1 конечных хостов настройка сети T называется *правильной* (относительно данного вектора значений параметров), если выполняются следующие условия.

1) Множество $T\uparrow$ содержит только реберно-простые (правильные) пути (*условие отсутствия циклов*).
2) Каждый максимальный путь из $T\uparrow$ начинается в начальном хосте (*условие начала путей*).
3) Каждый максимальный путь из $T\uparrow$ заканчивается в конечном хосте, и для любого начального хоста h_0: если хотя бы один путь из $T\uparrow$ начинается в h_0, то для любого конечного хоста h_1, в $T\uparrow$ существует путь из h_0 в h_1 (*условие конца путей*).

Заметим, что условия правильности настройки требует, чтобы пакеты генерировались в начальных хостах, но не требуют, чтобы это можно было делать в каждом начальном хосте. Если в $T\uparrow$ все пути реберно-простые, то в $T\uparrow$ существуют пути, максимальные по длине среди всех путей, и любой путь является отрезком некоторого максимального пути из $T\uparrow^{max}$. Очевидно, что для правильной настройки T во множестве $T\uparrow$ полные пути совпадают с максимальными путями. Заметим также, что T не обязано содержать настройки всех коммутаторов. Если в T нет ни одного правила для коммутатора s, т.е. правила вида (a, s, b), то такой коммутатор «глотает» все поступающие к нему пакеты. Однако это не создает никаких проблем: если порождаемое множество $T\uparrow^{max}$ не содержит путей, проходящих через коммутатор s, то таких «глотаемых» пакетов не будет.

Множество P полных путей будем называть *реализуемым*, если настройка $P\downarrow$ правильная. Очевидно, что для множества P полных путей $P \subseteq P\downarrow\uparrow^{max}$. Реализуемое множество P полных путей будем называть *строго реализуемым*, если выполнено дополнительное условие:

4) $P = P\downarrow\uparrow^{max}$.

4. Тестирование правила

Для правильной настройки сети T для каждого правила каждого коммутатора существует, по крайней мере, один путь, при передаче пакетов по которому используется (проверяется) это правило, т.е. имеет место следующее утверждение.

Утверждение 4.1. Пусть T правильная настройка. Тогда для любого правила (a, s, b) любого коммутатора s во множестве $T\uparrow$ существует путь $p{\cdot}(a, s, b){\cdot}q$, который начинается в некотором хосте из H_0 и заканчивается в хосте из H_1.

Доказательство. По правилу вывода (1) каждое правило настройки (a, s, b) порождает, по крайней мере, путь a, s, b. Если все пути, порождаемые настройкой сети, реберно-простые (условие отсутствия «зацикливания»), то путь a, s, b является отрезком некоторого максимального порождаемого пути p▪(a, s, b)▪q. Но если настройка T правильная, то этот путь начинается в начальном хосте (условие начала путей) и заканчивается в конечном хосте (условие конца путей). □

Заметим, что для некоторых правил таких полных путей, «проверяющих» это правило, может быть несколько.

Из утверждения 4.1 следует, что тестирование правильной настройки T сводится к посылке пакета (с данным вектором значений параметров) из каждого хоста h, для которого $T^{\uparrow max}$ содержит путь, начинающийся в h. Наблюдая пересылки пакетов между соседними узлами, можно обнаружить любую ошибку или в настройке коммутатора, и/или в работе коммутатора по правильной настройке.

Будем говорить, что правило (a, s, b) *проверяемо*, если существует правильная настройка сети T такая, что правило $(a, s, b) \in T$. Если верна гипотеза о правиле, то работа коммутатора s по правилу (a, s, b) не зависит от других правил в настройке коммутатора s и от настроек других коммутаторов сети. Эта гипотеза дает возможность проверять правила настройки «по одному»: если при некоторой настройке сети коммутатор правильно работает по данному правилу, то он будет правильно работать по этому правилу при любой настройке сети, в которой есть это правило. Поэтому для полной проверки работы коммутаторов сети достаточно отдельно проверить работу каждого коммутатора по каждому правилу его настройки, которое проверяемо. Для каждого проверяемого правила выполняется настройка сети так, чтобы по ней можно было проверить это правило: послать «пакет», проходящий по этому правилу хотя бы один раз. Такое тестирование можно также понимать как первоначальное «грубое» тестирование настроек, если гипотеза о правиле не верна. В этом разделе мы сформулируем и докажем необходимое и достаточное условие проверяемости правила.

Пусть s – коммутатор, и граф G содержит ребра $\{s, a\}$ и $\{b, s\}$. Рассмотрим граф, получающийся из графа G удалением обоих ребер $\{s, a\}$ и $\{b, s\}$. Обозначим через V_a, V_s, V_b множества вершин тех компонент связности этого графа, которым принадлежат вершины a, s, b, соответственно.

Утверждение 4.2. Необходимым и достаточным условием существования реберно-простого пути, начинающегося в хосте из H_0, заканчивающегося в хосте из H_1 и содержащего отрезок (a, s, b), где $a \neq b$, является условие $H_0 \cap V_a \neq \varnothing \vee H_1 \cap V_b \neq \varnothing \vee H_0 \cap V_s \neq \varnothing$ & $H_1 \cap V_s \neq \varnothing$.

Доказательство. Необходимость (см. рис. 2 сверху). Поскольку $H_0 \subseteq V$, $H_1 \subseteq V$, $H_0 \neq \varnothing$, $H_1 \neq \varnothing$, и $V = V_a \cup V_b \cup V_s$, из равенства $V_a = V_b$ следует $H_0 \cap V_a \neq \varnothing \vee H_0 \cap V_s \neq \varnothing$ и $H_1 \cap V_b \neq \varnothing \vee H_1 \cap V_s \neq \varnothing$, что влечет истинность

условия. Поэтому, если условие нарушено, то $V_a \neq V_b$, т.е. ребро $\{s, a\}$ или ребро $\{b, s\}$ является мостом. Рассмотрим произвольный путь p_{0a} от $h_0 \in H_0$ до a, и произвольный путь q_{b1} от b до некоторого хоста $h_1 \in H_1$. Такие пути существуют, если существует путь, начинающийся в хосте из H_0, заканчивающийся в хосте из H_1 и содержащий отрезок (a, s, b).

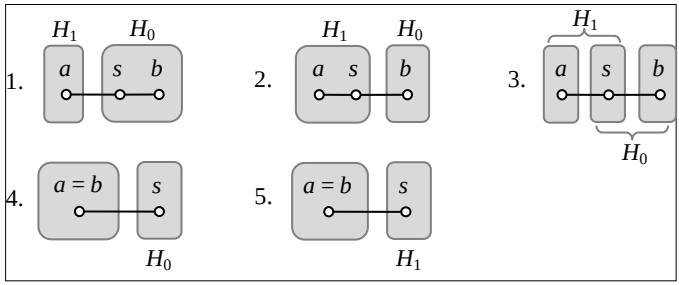


Рис. 2. Случаи, когда правило (a, s, b) проверить нельзя
Fig. 2. Cases where the rule (a, s, b) cannot be checked

Если ребро $\{s, a\}$ является мостом, а ребро $\{b, s\}$ не является мостом (см. рис. 2, случай 1), то из нарушения условия следует, что $H_0 \cap V_a = \varnothing$ (и, следовательно, $H_0 \cap V_s \neq \varnothing$), $H_1 \cap V_b = \varnothing$ и $H_1 \cap V_s = \varnothing$. А тогда пути p_{0a} и q_{b1} проходят по дуге (s, a). Если ребро $\{b, s\}$ является мостом, а ребро $\{s, a\}$ не является мостом (см. рис. 2, случай 2), то из нарушения условия следует, что $H_1 \cap V_b = \varnothing$ (и, следовательно, $H_1 \cap V_s \neq \varnothing$), $H_0 \cap V_a = \varnothing$ и $H_0 \cap V_s = \varnothing$. А тогда пути p_{0a} и q_{b1} проходят по дуге (b, s). Если оба ребра $\{s, a\}$ и $\{b, s\}$ являются мостами (см. рис. 2, случай 3), то из нарушения условия следует, что $H_0 \cap V_a = \varnothing$, $H_1 \cap V_b = \varnothing$ и $H_0 \cap V_s = \varnothing \vee H_1 \cap V_s = \varnothing$. А тогда пути p_{0a} и q_{b1} проходят либо по дуге (b, s), если $H_0 \cap V_s = \varnothing$, либо по дуге (s, a), если $H_1 \cap V_s = \varnothing$. Поэтому любой путь вида $p_{0a}{\cdot}(a, s, b){\cdot}q_{b1}$ не является реберно-простым, поскольку дважды проходит по дуге (s, a) или по дуге (s, b).

Достаточность. Условие утверждения представляет собой дизъюнкцию трех выражений. Рассмотрим три случая, соответствующие этим трем выражениям. Если $H_0 \cap V_a \neq \varnothing$ (см. рис. 3, случай 1), то существует вершинно-простой путь p_{0a} от некоторой вершины $h_0 \in H_0$ до a, не проходящий по ребрам $\{s, a\}$ и $\{b, s\}$. Поскольку граф G связен, существует вершинно-простой путь q_{b1} от b до некоторого хоста $h_1 \in H_1$. Пути $p_{0a}{\cdot}(a, s, b)$ и q_{b1} имеют общую вершину b, поэтому их можно представить в виде $p_{0a}{\cdot}(a, s, b) = p_{0c}{\cdot}p_{cb}$ и $q_{b1} = q_{bc}{\cdot}q_{c1}$, где p_{0c} и q_{bc} заканчиваются в одной вершине c – последней на пути q_{b1} общей вершине путей $p_{0a}{\cdot}(a, s, b)$ и q_{b1}. Обозначим через p_{bc} путь p_{cb}, проходимый в обратном порядке от b к c. Тогда путь $p_{0a}{\cdot}(a, s, b){\cdot}p_{bc}{\cdot}q_{c1}$ является искомым реберно-простым путем.

Если $H_1 \cap V_b \neq \varnothing$ (см. рис. 3, случай 2), то существует вершинно-простой путь q_{b1} от b до некоторого хоста $h_1 \in H_1$, не проходящий по ребрам $\{s, a\}$ и $\{b, s\}$.

Поскольку граф G связен, существует вершинно-простой путь p_{0a} от h_0 до a. Пути (a, s, b)▪q_{b1} и p_{0a} имеют общую вершину a, поэтому их можно представить в виде (a, s, b)▪$q_{b1} = q_{ac}$▪q_{c1} и $p_{0a} = p_{0c}$▪p_{ca}, где q_{ac} и p_{0c} заканчиваются в одной вершине c – первой на пути p_{0a} общей вершине путей (a, s, b)▪q_{b1} и p_{0a}. Обозначим через q_{ca} путь q_{ac}, проходимый в обратном порядке от c к a. Тогда путь p_{0c}▪q_{ca}▪(a, s, b)▪q_{b1} является искомым реберно-простым путем.

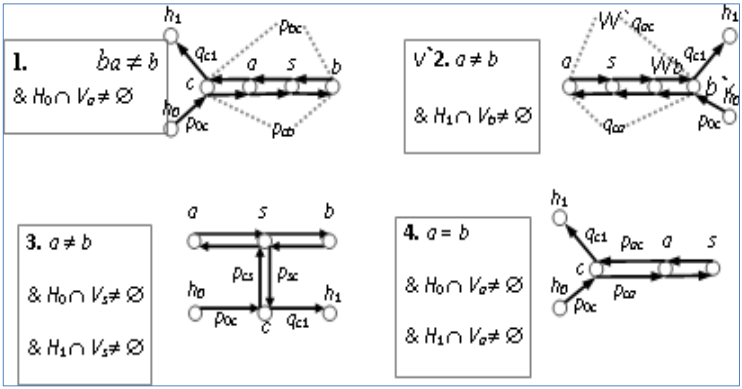


Рис. 3. Случаи, когда правило (a, s, b) проверить можно
Fig. 3. Cases where the rule (a, s, b) can be checked

Если $H_0 \cap V_s \neq \varnothing$ & $H_1 \cap V_s \neq \varnothing$ (см. рис. 3, случай 3), то существуют вершинно-простые пути p_{0s} от h_0 до s и q_{s1} от s до некоторого хоста $h_1 \in H_1$, не проходящие по ребрам $\{s, a\}$ и $\{b, s\}$. Эти пути имеют общую вершину s, поэтому их можно представить в виде $p_{0s} = p_{0c}$▪p_{cs} и $q_{s1} = q_{sc}$▪q_{c1}, где p_{0c} и q_{sc} заканчиваются в одной вершине c – первой на пути p_{0s} общей вершине путей p_{0s} и q_{s1}. Обозначим через p_{sc} путь p_{cs}, проходимый в обратном порядке от s к c. Тогда, поскольку $a \neq b$, путь p_{0s}▪(s, a, s, b, s)▪p_{sc}▪q_{c1} является искомым реберно-простым путем. □

Аналогично доказывается подобное утверждение, когда $a = b$.

Утверждение 4.3. Необходимым и достаточным условием существования реберно-простого пути, начинающегося в хосте из H_0, заканчивающегося в хосте из H_1 и содержащего отрезок (a, s, a), является следующее условие: $H_0 \cap V_a \neq \varnothing$ & $H_1 \cap V_a \neq \varnothing$.

Доказательство. Необходимость (см. рис. 2 снизу). Рассмотрим произвольный путь p_{0a} от некоторой вершины $h_0 \in H_0$ до a, и произвольный путь q_{a1} от a до некоторого хоста $h_1 \in H_1$. Если $H_0 \cap V_a = \varnothing$ (см. рис. 2, случай 4), то путь p_{0a} проходит по дуге (s, a). Если $H_1 \cap V_a = \varnothing$ (см. рис. 2, случай 5), то путь q_{a1} проходит по дуге (s, a). В обоих случаях любой путь вида p_{0a}▪(a, s, a)▪q_{a1} не является реберно-простым, поскольку дважды проходит по дуге (s, a).

Достаточность (см. рис. 3, случай 4). Если $H_0 \cap V_a \neq \varnothing$, то существует вершинно-простой путь p_{0a} от h_0 до a, не проходящий по ребру $\{s, a\}$. Если $H_1 \cap V_a \neq \varnothing$, то существует вершинно-простой путь q_{a1} от a до некоторого хоста $h_1 \in H_1$, не проходящий по ребру $\{s, a\}$. Пути p_{0a} и q_{a1} имеют общую вершину a, поэтому их можно представить в виде $p_{0a} = p_{0c}{\cdot}p_{ca}$ и $q_{a1} = q_{ac}{\cdot}q_{c1}$, где p_{0c} и q_{ac} заканчиваются в одной вершине c – последней на пути q_{a1} общей вершине путей p_{0a} и q_{a1}. Обозначим через p_{ac} путь p_{ca}, проходимый в обратном порядке от a к c. Тогда путь $p_{0a}{\cdot}(a, s, a){\cdot}p_{bc}{\cdot}q_{c1}$ является искомым реберно-простым путем. □

Утверждение 4.4. Пусть P – множество путей, начинающихся в одном и том же хосте $h_0 \in H_0$, заканчивающихся в некоторых хостах из H_1, такое, что все пути из $P{\downarrow}{\uparrow}$ реберно-простые, и $P{\downarrow}{\uparrow}^{max} = P$. Тогда множество P можно расширить до множества путей, обладающего теми же свойствами, но содержащего пути до всех хостов из H_1.

Доказательство. Пусть в хосте $h \in H_1$ не заканчивается ни один путь из P. Тогда выберем в графе G вершинно-простой путь q из h_0 в h. Если нет ребра, которое путь q проходит в том же направлении, в каком его проходит хотя бы один путь из P, добавим путь q ко множеству P. В противном случае путь q можно представить в виде $q = q_1{\cdot}(x, y){\cdot}q_2$, где ребро $\{x, y\}$ – последнее на пути q, которое какой-нибудь путь $p \in P$ проходит в том же направлении (x, y). Такой путь p можно представить в виде $p = p_1{\cdot}(x, y){\cdot}p_2$. Тогда добавим путь $p_1{\cdot}(x, y){\cdot}q_2$ ко множеству P. Сделаем это для каждого пути из P, который проходит по тому же ребру $\{x, y\}$ в том же направлении (x, y). Очевидно, полученное множество путей обладает теми же свойствами, что исходное множество P, но теперь в нём есть путь от h_0 до h. Будем повторять эту операцию добавления путей до тех пор, пока в H_1 есть хосты, в которых не заканчиваются пути из P. В конечном итоге получим искомое множество путей. □

Из утверждений 4.2, 4.3 и 4.4 непосредственно следует следующее утверждение.

Утверждение 4.5. Необходимым и достаточным условием того, что правило (a, s, b) проверяемо, является следующие условия: при $a \neq b$ – условие $H_0 \cap V_a \neq \varnothing \vee H_1 \cap V_b \neq \varnothing \vee H_0 \cap V_s \neq \varnothing \;\&\; H_1 \cap V_s \neq \varnothing$, а при $a = b$ – условие $H_0 \cap V_a \neq \varnothing \;\&\; H_1 \cap V_a$.

Из этого утверждения вытекают следующие достаточно простые следствия. Пусть s – коммутатор, и граф G содержит ребра $\{s, a\}$ и $\{b, s\}$.

Следствие 4.1. Правило (a, s, b) не проверяемо, если a – хост, не являющийся начальным хостом, т.е. $a \in H \setminus H_0$, или b – хост, не являющийся конечным хостом, т.е. $b \in H \setminus H_1$.

Следствие 4.2. Если ребра $\{s, a\}$ и $\{b, s\}$ не являются мостами в графе G, то правило (a, s, b) проверяемое.

Следствие 4.3. Если $H_0 = H_1$, то правило (a, s, b) проверяемое.

Следствие 4.4. Если $H_0 \cap H_1 \neq \varnothing$ и $a \neq b$, то правило (a, s, b) проверяемое.

5. Проверяемость любой настройки коммутатора

Будем говорить, что данная настройка T_s коммутатора s *проверяема*, если существует правильная настройка сети T такая, что $T(s) = T_s$. Если верна гипотеза о коммутаторе, то работа коммутатора s по правилам настройки T_s не зависит от настроек других коммутаторов сети. Эта гипотеза дает возможность проверять настройки коммутаторов «по одному»: если коммутатор правильно работает по правилам настройки T_s при данной настройке сети T такой, что $T(s) = T_s$, то он будет правильно работать по правилам настройки T_s при любой настройке сети T, для которой $T(s) = T_s$. Поэтому для полной проверки работы сети достаточно отдельно проверить работу каждого коммутатора по каждой его настройке, которая проверяема. В этом разделе мы исследуем условия, при которых любая настройка заданного коммутатора проверяема. Мы сформулируем и докажем необходимое (но не достаточное) и достаточное (но не необходимое) условия проверяемости любой настройки коммутатора.

Утверждение 5.1. Пусть P – множество правильных путей длины не меньше 2. Тогда $P^{\#} \subseteq P{\downarrow}{\uparrow}$.

Доказательство. Пусть путь $x_1, ..., x_n$ принадлежит $P^{\#}$, и $n > 2$. Тогда этот путь является отрезком некоторого правильного пути $q{\cdot}(x_1, ..., x_n){\cdot}r \in P$. Этот путь по правилу вывода (4) порождает в $P{\downarrow}{\uparrow}$ пути, среди которых есть следующие: x_i, x_{i+1}, x_{i+2}, где $i = 1..n\text{-}2$. Тогда по правилу вывода (5) в $P{\downarrow}{\uparrow}$ порождается путь $x_1, ..., x_n$. □

Если для любых двух путей $p{\cdot}(x, y){\cdot}q$ и $p`{\cdot}(x, y){\cdot}q`$, принадлежащих множеству путей P и проходящих по одной дуге (x, y), пути $p{\cdot}(x, y){\cdot}q`$ и $p`{\cdot}(x, y){\cdot}q$, отличные от пути (x, y), также принадлежат P, то множество P будем называть *замкнутым по дуге* (x, y). Множество P будем называть *замкнутым по дугам*, если оно замкнуто по каждой дуге графа. Заметим, что правило вывода (5), по сути, говорит о замкнутости по дугам множества $P{\downarrow}{\uparrow}$. Из определения замкнутости по дугам множества путей непосредственно следует следующее утверждение.

Утверждение 5.2. Если все пути из множества путей P, проходящие через данную дугу, имеют одинаковые префиксы до этой дуги или одинаковые постфиксы после этой дуги, то множество P замкнуто по этой дуге.

Утверждение 5.3. Пусть множество P состоит из правильных путей длины не меньше 2 и замкнуто по дугам. Тогда $P{\downarrow}{\uparrow} = P^{\#}$.

Доказательство. Согласно утверждению 5.10 нам достаточно показать, что $P{\downarrow}{\uparrow} \subseteq P^{\#}$. Пусть путь $p = x_1, ..., x_n$ принадлежит $P{\downarrow}{\uparrow}$. Поскольку все пути в $P{\downarrow}{\uparrow}$ имеют длину не меньше 2, $n > 2$. Тогда для $i = 1..n\text{-}2$ найдутся пути $p_i{\cdot}(x_i, x_{i+1}, x_{i+2}){\cdot}q_i \in P$. Из замкнутости по дугам множества P следует, что пути

$p_1 \blacksquare (x_1, ..., x_{i+2}) \blacksquare q_i \in P$ для $i = 1..n\text{-}2$. В частности, $p_1 \blacksquare (x_1, ..., x_n) \blacksquare q_{n-2} \in P$, т.е. $p_1 \blacksquare p \blacksquare q_{n-2} \in P$. Следовательно, $p \in P^{\#}$. □

Утверждение 5.4. Пусть множество P реберно-простых полных путей замкнуто по дугам. Тогда $P{\downarrow}{\uparrow}$ состоит из реберно-простых путей, и $P{\downarrow}{\uparrow}^{max} = P$.

Доказательство. Поскольку каждый хост соединен ребром только с коммутатором, все полные пути имеют длину не меньше 2. Тогда по утверждению 5.3 $P{\downarrow}{\uparrow} = P^{\#}$. Поскольку отрезок реберно-простого пути является реберно-простым, а все пути в P реберно-простые, все пути в $P^{\#}$ и, следовательно, в $P{\downarrow}{\uparrow}$ тоже реберно-простые. Поскольку $P{\downarrow}{\uparrow} = P^{\#}$, имеем $P{\downarrow}{\uparrow}^{max} = P^{\# max} = P^{max}$. Но все пути в P полные, следовательно, они максимальные. Поэтому $P^{max} = P$, что влечет $P{\downarrow}{\uparrow}^{max} = P$. □

Обозначим через G_s подграф графа G, получающийся удалением коммутатора s вместе со всеми инцидентными ему ребрами.

Утверждение 5.5 (необходимое условие проверяемости любой настройки коммутатора). Если коммутатор s является *шарниром*[3] в графе G, то не любая настройка коммутатора s проверяема.

Доказательство. Пусть коммутатор s – шарнир в графе G. Тогда подграф G_s содержит, по крайней мере, две компоненты связности, и хотя бы одна из этих компонент содержит вершину из H_1. Обозначим множество вершин этой компоненты через V_1: $H_1 \cap V_1 \neq \varnothing$. Тогда найдётся вершина $a \in N(s) \setminus V_1$. Рассмотрим настройку коммутатора s, состоящую из одного правила (a, s, a). Необходимым условием проверяемости такой настройки является наличие в графе G_s хоста из H_0, из которого была бы достижима как вершина a, так и вершины из $H_1 \cap V_1$. Однако в графе G_s такого хоста быть не может. Поэтому такая настройка не проверяема. □

Следствие 5.1. Если у коммутатора s среди соседей есть хосты, то не любая настройка коммутатора s проверяема на этой физической сети.

Доказательство. Поскольку хост имеет степень 1, коммутатор s является шарниром. Тогда по утверждению 5.5, не любая настройка коммутатора s проверяема на заданном графе физических связей. □

Если ребро $\{u, v\}$ графа G является мостом, то будем называть его s-мостом, если вершины u и v коммутаторы. Компонента связности графа, получающегося из исходного графа удалением всех его мостов, называется *компонентой рёберной двусвязности* исходного графа. Если удаляются только s-мосты, то будем говорить о *компоненте рёберной s-двусвязности* графа.

Утверждение 5.6 (достаточное условие проверяемости любой настройки коммутатора). Если коммутатор s не является шарниром в графе G, и у графа

[3] Шарнир (*cut point, articulation point*) – вершина, удаление которой вместе с инцидентными ей ребрами увеличивает число компонентов связности графа.

G_s есть компонента рёберной s-двусвязности, в которой есть начальный и конечный хосты, то любая настройка коммутатора s проверяема.

Доказательство. Пусть коммутатор s не является шарниром в графе G, и у графа G_s есть компонента рёберной s-двусвязности $X = (V_X, E_X)$, в которой есть начальный и конечный хосты. Рассмотрим произвольную настройку T_s коммутатора s. Для ее проверяемости достаточно существования такого строго реализуемого множества P полных путей, которое для каждого правила $(a, s, b) \in T_s$ содержит путь, в котором есть отрезок (a, s, b). Доказательство конструктивное: мы опишем алгоритм построения множества P и докажем, что построенное множество удовлетворяет указанным требованиям. Шаги алгоритма иллюстрируются на рис. 4.

Алгоритм. Выберем какие-нибудь начальный хост $v \in V_X \cap H_0$ и конечный хост $w \in V_X \cap H_1$. Пусть $v`$ и $w`$ – коммутаторы, к которым подсоединены хосты v и w. Поскольку в компоненте X нет s-мостов, после удаления из этой компоненты хостов и инцидентных им ребер в ней нет мостов. Как известно, в графе без мостов для любых двух вершин существует реберно-простой цикл, проходящий через них. Поэтому для коммутаторов $v`$ и $w`$ в компоненте X существует реберно-простой цикл C, проходящий через них. Этот цикл порождает подграф $X_C = (V_C, E_C)$, состоящий из вершин и ребер цикла. Поскольку коммутатор s не является шарниром в графе G, граф G_s связный.

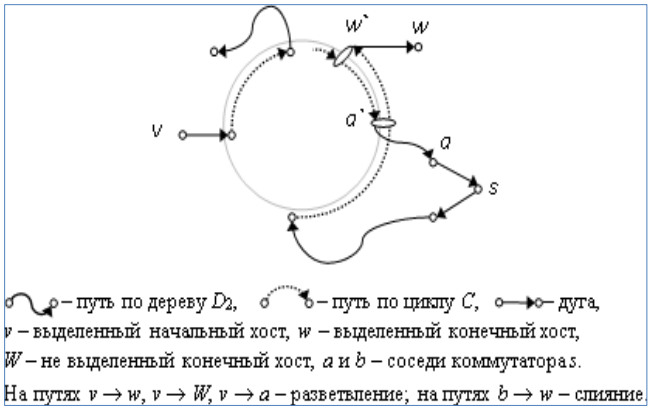


Рис. 4. Построение путей от одного начального хоста ко всем конечным хостам
Fig. 4. Building paths from one sender host to all destination hosts

Выберем в графе G_s произвольное остовное дерево $D = (V, E_D)$. Будем последовательно удалять ребра дерева D, не принадлежащие циклу C, т.е. ребра множества $E_D \setminus E_C$, если при этом не нарушается связность подграфа $(V, E_C \cup E_D)$. В результате получим лес D_1, в котором из каждой вершины x есть единственный вершинно-простой путь до цикла C – до вершины, которую мы обозначим через $x`$. Затем будем последовательно удалять ребра

леса D_1, инцидентные его листовым вершинам, не принадлежащим $H_1 \cup N(s) \cup V_C$. В результате получим лес D_2, в котором из каждой вершины $x \in H_1 \cup N(s)$ есть единственный вершинно-простой путь до цикла C – до вершины $x`$. Тогда для каждого правила $(a, s, b) \in T_s$ выбираем путь $p(a, s, b) = (v, v`)\cdot L_{v`a`}\cdot P_{a`a}\cdot(a, s, b)\cdot P_{bb`}\cdot R_{b`w`}\cdot(w`, w)$, где $L_{v`a`}$ – путь по циклу $C$ по часовой стрелке от $v`$ до $a`$, $P_{a`a}$ – путь по лесу D_2 от $a`$ до $a$, $P_{bb`}$ – путь по лесу D_2 от b до $b`$, $R_{b`w`}$ – путь по циклу $C$ против часовой стрелки от $b`$ до $w`$. Также для каждого конечного хоста $W \in H_1$ выбираем путь $p(W) = (v, v`)\cdot L_{v`W`}\cdot P_{W`W}$, где $L_{v`W`}$ – путь по циклу $C$ по часовой стрелке от $v`$ до $W`$, $P_{W`W}$ – путь по лесу D_2 от $W`$ до W. Множество выбранных путей обозначим P.

Докажем, что алгоритм строит требуемое множество путей P. По построению это множество для каждого правила $(a, s, b) \in T_s$ содержит путь $p(a, s, b)$, в котором есть отрезок (a, s, b). Каждый путь из P реберно-простой и полный: начинается в $v \in H_0$ и заканчивается в хосте из H_1, т.е. P – множество полных путей и выполнено условие 2 – условие начала путей в определении правильной настройки. Поскольку каждый путь из P заканчивается в хосте из H_1, а для каждого конечного хоста W множество P содержит путь $p(W)$, ведущий из $v \in H_0$ в W, т.е. выполнено условие 3 – условие конца путей в определении правильной настройки. Кроме того, множество P обладает тем свойством, что все пути, проходящие по одной дуге, имеют одинаковые префиксы до этой дуги (разветвление), если эта дуга лежит на пути вида $(v, v`)\cdot L_{v`a`}\cdot P_{a`a}\cdot(a, s)$ или $(v, v`)\cdot L_{v`W`}\cdot P_{W`W}$, или одинаковые постфиксы после этой дуги (слияние), если дуга лежит на пути вида $(s, b)\cdot P_{bb`}\cdot R_{b`w`}\cdot(w`, w)$. Следовательно, по утверждению 5.2 множество P замкнуто по дугам. А тогда по утверждению 5.4 множество $P{\downarrow}{\uparrow}$ состоит из реберно-простых путей, т.е. выполнено условие 1 – условие отсутствия циклов в определении правильности настройки, и $P{\downarrow}{\uparrow}^{max} = P$, т.е. выполнено условие 4 в определении строгой реализуемости. □

Следствие 5.2. Если в графе G_s нет s-мостов, то любая настройка коммутатора s проверяема.

6. Заключение

Утверждение 4.5 дает необходимое и достаточное условие проверямости заданного правила заданного коммутатора. Соответствующее тестирование всех проверяемых правил всех коммутаторов «по одному» будет полным, если верна (сильная) гипотеза о правиле: работа коммутатора по данному правилу не зависит от других правил настройки этого коммутатора и от настроек других коммутаторов.

Если гипотеза о правиле не верна, можно воспользоваться (слабой) гипотезой о коммутаторе, предполагающей независимость работы коммутатора только от настроек других коммутаторов. Утверждение 5.5 дает необходимое (но не

достаточное) условие проверяемости любой настройки коммутатора: коммутатор не должен быть шарниром в графе G. Утверждение 5.6 дает достаточное (но не необходимое) условие проверяемости любой настройки коммутатора: у графа G_s есть компонента рёберной s-двусвязности, в которой есть начальный и конечный хосты.

Остается неисследованным случай, когда коммутатор s не является шарниром, но на дереве компонент реберной s-двусвязности графа G_s есть компоненты, которые содержат начальные хосты, и есть компоненты, которые содержат конечные хосты, но нет компоненты, которая содержала бы как начальный, так и конечный хосты.

Кроме того, для коммутаторов соседних с хостами было бы естественно рассматривать не любые настройки, а только те, которые не являются заведомо не проверяемыми. Иными словами было бы правильно исключить из рассмотрения настройки, содержащие правило (a, s, b), если $a \in H \setminus H_0$, т.е. a – хост, но не начальный, и/или $b \in H \setminus H_1$, т.е. b – хост, но не конечный.

Все эти случаи, конечно, являются частными случаями более общей задачи: определить необходимые и достаточные условия проверяемости заданной настройки T_s данного коммутатора s. Условия, естественно, налагаются на совокупность $(G, S, H, H_0, H_1, s, T_s)$. Для проверяемой настройки коммутатора требуется также найти эффективный алгоритм построения проверяющей ее правильной настройки сети.

Благодарности

Авторы выражают благодарность исследователям Хорхе Лопэзу, Наталье Кушик, Джамалю Зеглашу университета Телеком Южный Париж (Еври, Франция) за многочисленные плодотворные дискуссии при подготовке данной работы.

Список литературы

[1]. Sezer S., Scott-Hayward S., Chouhan P. K. et al. Are we ready for sdn? Implementation challenges for software-defined networks. IEEE Communications Magazine, vol. 51, no. 7, 2013, pp. 36–43.

[2]. Gill P., Jain N., and Nagappan N. Understanding network failures in data centers: Measurement, analysis, and implications. In Proc. of the ACM SIGCOMM 2011 Conference, SIGCOMM ’11, 2011, pp. 350–361.

[3]. Scott C., Wundsam A., Raghavan B. et al. Troubleshooting blackbox sdn control software with minimal causal sequences. ACM SIGCOMM Computer Communication Review, vol. 44, no. 4, 2015, pp. 395–406.

[4]. Shalimov A., Zuikov D., Zimarina D., Pashkov V., and Smeliansky R. Advanced study of sdn/openflow controllers. In Proc. of the 9th Central & Eastern European Software Engineering Conference in Russia, 2013.

[5]. Yao J., Wang Z., Yin X., Shiyz X., and Wu J. Formal modeling and systematic black-box testing of sdn data plane. In The IEEE 22nd International Conference on Network Protocols, 2014, pp. 179–190.
[6]. Zhang Z., Yuan D., and Hu H. (2016). Multi-layer modeling of openflow based on efsm. In Proceedings of the 2016 4th International Conference on Machinery, Materials and Information Technology Applications, 2016, pp. 524-529.
[7]. J López, N. Kushik, D. Zeghlache. Quality Estimation of Virtual Machine Placement in Cloud Infrastructures. Lecture Notes in Computer Science, vol. 10533, 2017, pp. 213-229.
[8]. Canini M., Kostic D., Rexford J., and Venzano D. Automating the testing of openflow applications. In Proceedings of the 1st International Workshop on Rigorous Protocol Engineering, 2011.
[9]. Canini M., Venzano D., Peresini P. et al. A nice way to test openflow applications. In Proc. of the 9th USENIX conference on Networked Systems Design and Implementation, 2012, pages 127–140.
[10]. A. Berriri, J. López, N. Kushik, N. Yevtushenko, D. Zeghlache. Towards Model based Testing for Software Defined Networks. In Proc. of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, 2018, pp. 440-446.
[11]. ONOS Project. Key and field description. Режим доступа: https://wiki.onosproject.org/display/ONOS/Flow+Rules#FlowRules-Keyandfielddescription. Дата обращения 15.11.2018.

Testing switch rules in software defined networks

I.B. Burdonov <igor@ispras.ru>
N.V. Yevtushenko <yevtushenko@ispras.ru>
A.S. Kossatchev <kos@ispras.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. SDN-technology is efficiently used for implementing service function chains can be efficiently implemented utilizing common resources and their management principles in virtual networks. The network is based on a connected undirected graph of physical links called usually referred to as resource network connectivity topology (RNCT); graph nodes are network switches and hosts and each host is connected exactly with one switch. Switches operate based on rule tables that are configured by a controller that operates independently of network equipment. The configuration of network switches provides the transmission of packets from the initial to final hosts depending on the values of the packet parameters. The paper discusses the relationship between switch configurations and paths which are created for trasmitting packets depending on RNCT properties. It is shown that, in general, not any configuration of any switch is verifiable. Testing abilities depend on the accepted hypotheses about the switch operating. Two hypotheses are discussed in the paper: the switch hypothesis assumes that the switch operation does not depend on the settings of other switches; a stronger hypothesis about the rule, besides this, assumes that the switch operation according to this rule does not depend on other rules in the configuration of this switch. Section 2 contains preliminaries while Section 3 is devoted to the relationship between switch rules and sets of paths to be implemented. In Section 4, the problem of testing the switch configuration is considered based on the rule hypothesis; a number of statements are estableshed, in particular, the necessary and sufficient conditions of the ability of testing a given rule of a

given switch. Section 5 discusses and proves the necessary (but not sufficient) condition and sufficient (but not necessary) condition for checking any switch configuration based on the switch hypothesis. In conclusion, the problems of establishing the necessary and sufficient conditions for verifiability of any switch configuration are discussed.

Keywords: SDN-technology; hosts and switches; network switch configuration; packet transmission; edge simple paths; testing network switches.

DOI: -10.15514/ISPRAS-2018-30(6)-4

For citation: Burdonov I.B., Yevtushenko N.V., Kossatchev A.S. Testing switch rules in software defined networks. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 69-88 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-4

References

[1]. Sezer S., Scott-Hayward S., Chouhan P. K. et al. Are we ready for sdn? Implementation challenges for software-defined networks. IEEE Communications Magazine, vol. 51, no. 7, 2013, pp. 36–43.

[2]. Gill P., Jain N., and Nagappan N. Understanding network failures in data centers: Measurement, analysis, and implications. In Proc. of the ACM SIGCOMM 2011 Conference, SIGCOMM '11, 2011, pp. 350–361.

[3]. Scott C., Wundsam A., Raghavan B. et al. Troubleshooting blackbox sdn control software with minimal causal sequences. ACM SIGCOMM Computer Communication Review, vol. 44, no. 4, 2015, pp. 395–406.

[4]. Shalimov A., Zuikov D., Zimarina D., Pashkov V., and Smeliansky R. Advanced study of sdn/openflow controllers. In Proc. of the 9th Central & Eastern European Software Engineering Conference in Russia, 2013.

[5]. Yao J., Wang Z., Yin X., Shiyz X., and Wu J. Formal modeling and systematic black-box testing of sdn data plane. In The IEEE 22nd International Conference on Network Protocols, 2014, pp. 179–190.

[6]. Zhang Z., Yuan D., and Hu H. (2016). Multi-layer modeling of openflow based on efsm. In Proceedings of the 2016 4th International Conference on Machinery, Materials and Information Technology Applications, 2016, pp. 524-529.

[7]. J López, N. Kushik, D. Zeghlache. Quality Estimation of Virtual Machine Placement in Cloud Infrastructures. Lecture Notes in Computer Science, vol. 10533, 2017, pp. 213-229.

[8]. Canini M., Kostic D., Rexford J., and Venzano D. Automating the testing of openflow applications. In Proceedings of the 1st International Workshop on Rigorous Protocol Engineering, 2011.

[9]. Canini M., Venzano D., Peresini P. et al. A nice way to test openflow applications. In Proc. of the 9th USENIX conference on Networked Systems Design and Implementation, 2012, pages 127–140.

[10]. A. Berriri, J. López, N. Kushik, N. Yevtushenko, D. Zeghlache. Towards Model based Testing for Software Defined Networks. In Proc. of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, 2018, pp. 440-446.

[11]. ONOS Project. Key and field description. Available at: https://wiki.onosproject.org/display/ONOS/Flow+Rules#FlowRules-Keyandfielddescription. Accessed 15.11.2018.

Тестирование соответствия реализаций протокола EAP и его методов спецификациям Интернета

[1] *А.В. Никешин <alexn@ispras.ru>*
[1,2] *В.З. Шнитман <vzs@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский физико-технический институт, 141700, Московская область, г. Долгопрудный, Институтский пер., 9*

Аннотация. В данной статье представлены результаты проекта по созданию тестового набора для тестирования соответствия реализаций протокола EAP и его методов спецификациям Интернета. В основе проекта лежит использование технологии UniTESK, позволяющей автоматизировать процесс верификации сетевых протоколов на основе их формальных моделей и расширения JavaTesK, реализующего эту технологию на языке программирования. Совместное использование методов мутационного тестирования позволяет протестировать устойчивость реализации протокола к искаженным сообщениям. Данный подход доказал свою эффективность, позволив обнаружить несколько критических уязвимостей и другие отклонения от спецификации в выбранных реализациях протокола.

Ключевые слова: тестирование, верификация, формальные методы, формальные спецификации, тестирование с использованием моделей, безопасность, аутентификация, контроль доступа, EAP, методы EAP, UniTESK, мутационное тестирование

DOI: 10.15514/ISPRAS-2018-30(6)-5

Для цитирования: Никешин А.В., Шнитман В.З. Тестирование соответствия реализаций протокола EAP и его методов спецификациям Интернета. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 89-104. DOI: 10.15514/ISPRAS-2018-30(6)-5

1. Введение

Аутентификация в современных сетях является одним из основных механизмов безопасности. В корпоративных решениях довольно часто используется аутентификация по протоколу EAP на базе сервера RADIUS [1, 2]. Поскольку в данном процессе участвуют разнородные объекты, возникает задача обеспечения совместимости реализаций, одним из основных методов решения которой является тестирование на соответствие

стандарту. Технология UniTESK предоставляет средства автоматизации тестирования соответствия формальным спецификациям, и одна из областей, где эта технология используется, – тестирование протоколов [3]. Особенность тестирования протоколов состоит в том, что требуется также проводить тестирование на нестандартных или искаженных входных данных, что довольно актуально для систем безопасности. Такие ситуации постоянно возникают из-за ошибок пользователей или из-за целенаправленных действий злоумышленника, но при этом часто недостаточно полно определены в спецификации протокола. В наших работах мы совмещаем методы верификации на основе формальных спецификаций и методы мутации данных, что позволяет значительно улучшить качество тестирования реализаций протоколов.

Основные особенности протокола EAP и его методов достаточно подробно изложены в нашей статье «Обзор расширяемого протокола аутентификации и его методов» [4], поэтому здесь мы на них останавливаться не будем.

2. Формальная спецификация

Формальная спецификация протокола EAP состоит из следующих компонентов:

- модели протокола, которая содержит набор структур данных, моделирующих внутреннее состояние и управляющие данные протокола и его методов в соответствии со стандартами;
- спецификационных стимулов, которые формализуют требования к изменению состояния реализации EAP при внешнем воздействии на систему;
- спецификационных реакций, которые формализуют требования к реакциям реализации EAP на внешние воздействия.

Поскольку протокол EAP использует пошаговую схему обработки сообщений (новый запрос отправляется только после получения ответа на предыдущий), в спецификации каждое отправленное тестируемой системе сообщение рассматривается как отдельный стимул, соответствующий обработке исходящего модельного сообщения EAP.

Каждое сообщение от тестируемой системы (или его отсутствие) рассматривается как отдельная реакция на соответствующий стимул.

Параметрами спецификационных методов служат сообщения EAP и его методов в модельном представлении. Наборы полей соответствующих классов зависят от структур данных, описанных в стандартах используемых методов EAP.

3. Модельное состояние

Модельное состояние протокола состоит из структур данных, относящихся к текущей сессии EAP, и структур данных, относящихся к используемому методу.

Параметры сессии включают следующие блоки данных:

Current state	текущее состояние сессии
Current method	используемый метод EAP
Peer Identity	идентификатор партнера
Result	результат аутентификации
Keys	ключи безопасности, передаваемые другим приложениям
Request Messages	множество запросов сессии
Response Messages	множество ответов сессии

Модельное состояние метода EAP включает следующие данные:

Current state	текущее состояние метода
Identity	идентификатор партнера, согласованный текущим методом (может отличаться от указанного выше идентификатора Peer Identity, а также может заменяться другими структурами данных, если это требуется в конкретном методе EAP)
Peer parameters	параметры безопасности партнера, необходимые для данного метода аутентификации (такие как пароль, сертификат и др.)
Result	результат аутентификации данного метода
Transient Keys	временные ключи криптографических алгоритмов для защиты текущего обмена

Кроме этого, у каждого метода могут быть дополнительные параметры, сильно зависящие от его спецификации. Так, например, для метода EAP-SIM определены следующие параметры [5]:

version list	список версий метода
selected version	согласованная версия метода
nonceMT	случайное число
nonceS	случайное число
peerIDs	множество идентификаторов партнера (EAP-SIM позволяет использовать разные идентификаторы в зависимости от режима аутентификации)
security parameters	параметры безопасности для передачи зашифрованных данных, если такие используются
result indication	дополнительная защита результата аутентификации
counter	счетчик для защиты целостности сообщений

triplet list	список триплетов GSM – параметры, из которых создаются ключи безопасности

4. Модель сообщений EAP

Структура сообщений в основном определяется используемым методом EAP. Спецификация протокола EAP определяет четыре типа сообщений, задающих общий шаблон процесса аутентификации [1]:

Request (запрос);

Response (ответ);

Success (успех);

Failure (неудача).

Также можно выделить несколько вспомогательных подтипов в рамках сообщений Request/Response.

Identity Используется для запроса идентификатора клиента. Спецификация требует использовать данный тип запроса исключительно для целей маршрутизации и выбора подходящего метода EAP. Как правило, это первое сообщение от аутентификатора клиенту.

Notification Необязательный тип сообщений. Используется для передачи аутентификатором сообщений, выводимых на экран клиента. Может передаваться в любое время в процессе обмена EAP.

Nak Используется в ответных сообщениях партнера, когда предложенный аутентификатором метод EAP не поддерживается. Партнер может указать в ответном сообщении желаемые методы EAP.

Каждый метод EAP может определять дополнительные сообщения, свойственные исключительно этому методу. Так, например, метод EAP-SIM задает пять новых сообщений:

Start,

Challenge,

Notification,

Client-Error,

Re-authentication

и два десятка дополнительных атрибутов, инкапсулирующих необходимые данные.

Разработанная библиотека модельного представления сообщений EAP и некоторых его методов позволяет создавать различные варианты как сообщений, так и их последовательностей.

5. Тестирование реализации протокола

Инициализация тестового сценария зависит от используемого протокола нижележащего уровня. Мы используем Radius и 802.1x [2, 6]. Процесс обмена,

как правило, начинает аутентификатор сообщением EAP-Request/Identity. В случае взаимодействия тестового сценария напрямую с сервером (без использования аутентификатора) начальное сообщение предполагается полученным.

Стимулами являются сообщения от инструментального узла в качестве ответов реализации сервера. В тестовом сценарии создаются ответы партнера в модельном представлении, передаваемые затем функции отправки сообщений. В предусловии спецификационных функций стимулов проверяется правильность структуры тестового сообщения и его своевременность, и на основании этого делается вывод о том, должен ли на него быть ответ, сообщение об ошибке или реализация должна его проигнорировать. Из модельного представления тестового сообщения строится реализационное, которое отправляется в сеть. Сборщик реакций в течение заданного времени собирает ответные сообщения целевой системы. Из реализационных сообщений строятся их модельные представления. В постусловии реакций данные проверяются на соответствие требованиям спецификации: допустимость сообщения и его структура. После проверки всех требований, результат передается тестовому сценарию, где в зависимости от плана сценария, принимается решение о продолжении или завершении информационного обмена. В случае выявления нарушения требований принимается решение о критичности ошибки и возможности отправки следующих сообщений.

Модель применяемого метода EAP определяет множество состояний тестируемой системы, на его основе строится конечный автомат, переходы которого сопоставлены с соответствующими тестовыми воздействиями. При выполнении перехода определенное воздействие передается на тестируемую реализацию, после чего регистрируются реакции реализации и проверяется корректность наблюдаемого поведения системы. Обход автомата по всем достижимым состояниям модели протокола осуществляется инструментарием UniTESK. Методы мутации позволяют вносить искажения в сообщения на любом этапе обмена. При этом совместное использование корректных и измененных сообщений позволяет тестовому сценарию "преодолеть" необходимые проверки в реализации, пройти через все значимые состояния протокола и в каждом состоянии выполнить необходимый набор тестовых воздействий.

6. Инструментальные средства

Разработка тестового набора велась с использованием инструмента автоматизированного тестирования JavaTesK [7], который реализует технологию UniTESK на базе объектно-ориентированного языка с автоматической сборкой мусора и обеспечивает значительное упрощение модели протокола и тестовых сценариев.

7. Тестовый стенд

Основная схема применения протокола EAP, представленная в спецификации, предполагает наличие трех узлов:

- **Клиент (партнер):** Сетевой узел, которому требуется пройти аутентификацию.
- **Аутентификатор:** Сетевой узел, управляющий доступом к ресурсам, с которым соединяется клиент.
- **Сервер EAP:** Внутренний сервер, реализующий методы EAP и выполняющий аутентификацию партнера. Данный объект определяет, прошла ли аутентификация успешно или нет.

Клиент запрашивает доступ к некоторому ресурсу, подключаясь к аутентификатору. Аутентификатор передает запрос с данными клиента серверу EAP. Сервер EAP запрашивает дополнительные данные у клиента. Обмен сообщениями между клиентом и сервером EAP продолжается до тех пор, пока выбранный метод аутентификации не завершится успешно или с ошибкой. Сервер EAP осуществляет обмен данными с аутентификатором и информирует его о результатах. На основании этого аутентификатор, принимает решение о предоставлении клиенту доступа к ресурсу.

На клиентском узле исполняется основной поток управления тестовой системы под управлением UniTESK, обход тестового автомата и верификация наблюдаемых реакций. На сервере EAP функционирует тестируемая реализация протокола. Тестовые сообщения протокола, сформированные модельной реализацией, передаются тестируемой системе, после чего регистрируются реакции тестируемого узла.

Протокол EAP может одновременно использоваться в разных средах передачи данных. Как правило, аутентификатор и сервер EAP общаются через проводной канал по протоколу AAA RADIUS. Клиент и аутентификатор используют другой стек сетевых протоколов и, возможно, другую среду передачи данных.

Следует понимать, что аутентификатор применяется для управления доступом, для расширения топологии сети и из-за возможности использовать проводные и беспроводные каналы. С точки зрения самого процесса аутентификации он используется как ретранслятор, передавая пакеты EAP между клиентом и сервером. Фактически аутентификация выполняется между двумя последними. Поэтому для тестирования непосредственно сервера EAP наличие или отсутствие аутентификатора не принципиально. На разных этапах нашего проекта при тестировании реализаций протокола EAP мы использовали разные механизмы взаимодействия с сервером EAP: как непосредственно по протоколу RADIUS, так и через аутентификатор (преимущественно через коммутатор Dell Networking N2048 по протоколу 802.1x - EAP over LAN [6]).

8. Тестируемые реализации протокола EAP

Используемые в нашем проекте реализации протокола отбирались исключительно по их доступности: бесплатная версия FreeRADIUS, условно-бесплатные Clearbox и TekRADIUS, и имеющаяся в нашем распоряжении версия Windows Server 2012 [8-11]. Также отметим, что в процессе поиска нам попадались реализации, которые давно не обновлялись или совсем заброшены их авторами, такие реализации для нас интереса не представляли.

FreeRADIUS. По сути, единственная многофункциональная бесплатная реализация, пережившая многих конкурентов. Данная реализация позиционируется разработчиком как самый распространенный сервер RADIUS, к достоинствам которого относятся свободное распространение, открытый исходный код, развитая функциональность, поддержка многочисленных методов аутентификации EAP. Сервер установлен под ОС CentOS 7.

Windows Server 2012. Доступная для нашего проекта полнофункциональная коммерческая реализация.

Clearbox Enterprise Server 6.5.2. Условно-бесплатная версия, предлагающая в начальный период эксплуатации полную функциональность. Сервер установлен под ОС Windows 10.

TekRADIUS 5.4. Условно-бесплатная версия. Возможности бесплатной версии сильно ограничены. Сервер установлен под ОС Windows 10. Отметим, что данная реализация, хотя и использует SQL-сервер для хранения базы данных пользователей, тем не менее, позволяет использовать некоторую функциональность без добавления каких-либо пользователей, используя настройки по умолчанию. Именно такая конфигурация использовалась в наших тестах.

9. Тестируемые методы EAP

Несмотря на большое разнообразие существующих методов EAP, предлагаемый к использованию указанными выше реализациями набор методов гораздо скромнее (возможно платные программные продукты корректируют набор методов EAP в зависимости от установленного оборудования). Мы условно раздели их на группы, на которые и опирались в своей работе.

Базовая функциональность и базовые методы протокола EAP. Это функциональность и требования, представленные в основной спецификации протокола и обязательные для всех реализаций [1]. Сюда же относятся простейшие методы (такие как NAK, EAP-MD5 и др.), среди которых EAP-MD5 поддерживается всеми реализациями, хотя и не рекомендуется к самостоятельному использованию как небезопасный.

Методы, использующие данные смарт-карт. В основном предлагается метод EAP-SIM [5], иногда дополнительно метод EAP-AKA/AKA’ [12, 13].

Метод аутентификации EAP-SIM использует параметры и алгоритмы SIM-карты для аутентификации и создания криптографических ключей. Данный метод основан на механизмах аутентификации GSM (GSM – стандарт мобильных сетей второго поколения) и разрабатывался для аутентификации специализированных устройств, использующих SIM-карты [14]. В Windows Server заявлена поддержка данных методов, но вероятно, требуется наличие физического устройства. Реализация Clearbox данные методы не поддерживает. Реализация TekRADIUS предлагает метод EAP-SIM в платной версии и, вероятно, также требует наличия физического устройства. В наших экспериментах такие устройства не применяются, а ручная настройка необходимых параметров поддерживается только реализацией FreeRADIUS для метода EAP-SIM, которым мы и ограничились.

Туннельные методы EAP. Переход на данные механизмы аутентификации является общей тенденцией и поэтому они представлены во всех современных реализациях EAP. Появившиеся вначале несколько туннельных методов: PEAP (Protected EAP), EAP-TTLS (EAP tunneled TLS), EAP-FAST (EAP Flexible Authentication via Secure Tunneling), не имеют статуса «Стандарта Интернет» [15-17]. Поэтому разработчики выбирают поддержку какого-либо метода исходя из своих предпочтений. Созданный позднее метод TEAP [18] на данный момент не поддерживается реализациями. Все указанные методы EAP для создания туннеля используют протокол TLS [19]. Основным туннельным методом FreeRADIUS является TTLSv0. Windows Server использует исключительно PEAP собственной разработки – PEAPv0. Clearbox предлагает метод PEAP. TekRADIUS в бесплатной версии предлагает метод PEAP. Также отметим, что метод Microsoft PEAP не имеет единого стандарта, в него довольно часто по мере необходимости вносятся изменения. Поэтому в нашей работе тестировалась только общая функциональность PEAP, свойственная всем туннельным методам на базе протокола TLS.

10. Результаты тестирования

В данном разделе представлены результаты прогона разработанного нами тестового набора, позволившего обнаружить целый ряд уязвимостей и отклонений от спецификации в выбранных реализациях протокола.

10.1 Тестирование базовой функциональности и базовых методов протокола EAP

FreeRADIUS

- Если на запрос аутентификации от сервера отправить сообщение с типом NAK и пустым полем данных (данное поле должно содержать не менее одного байта), то сервер отвечает сообщением об ошибке (хотя, неправильно сформированные сообщения должны просто отбрасываться).

- Если на запрос аутентификации от сервера отправить сообщение с типом MD5-Challenge и пустым полем данных (данное поле должно содержать не менее одного байта), то сервер отвечает сообщением об ошибке (неправильно сформированные сообщения должны просто отбрасываться).
- Если на запрос аутентификации от сервера отправить сообщение с кодом Request (ответные сообщения всегда должны иметь код Response) и типом NAK, то сервер отвечает сообщением об ошибке (неправильно сформированные сообщения должны просто отбрасываться).
- Если на запрос аутентификации от сервера отправить сообщение с кодом Request (ответные сообщения всегда должны иметь код Response) и типом MD5-Challenge, то сервер отвечает сообщением об ошибке (неправильно сформированные сообщения должны просто отбрасываться).
- Если на запрос аутентификации от сервера отправить сообщение с типом Identity, то сервер отвечает повторным сообщением с запросом аутентификации (сообщение нарушает порядок обмена и должно быть отброшено).

Windows Server 2012

- Реализация полностью игнорирует содержание поля Identity (идентификатор клиента) в первом сообщении клиента.

Clearbox enterprise server

- При получении от клиента первого сообщения EAP типа REQUEST / RESPONSE с любым значением метода EAP и любыми дополнительными данными реализация начинает аутентификацию, используя настройки политики безопасности с методом EAP по умолчанию (первым сообщением от клиента предполагается RESPONSE/ Identity).
- При получении от клиента допустимого сообщения EAP (Request, Response, Success, Failure; тип метода EAP > 0 для Request/Response) со значением поля длины 0, 1 или 2 на любом этапе сессии реализация падает. Данная уязвимость не требует знания каких–либо данных пользователя. Поскольку данная уязвимость относится к общему заголовку EAP, умный аутентификатор может заблокировать подобные сообщения.

TekRADIUS

- Во входящих сообщениях EAP-Identity, если данные не пустые, реализация не проверяет поле длины.

10.2 Тестирование метода EAP-SIM

FreeRADIUS

- Почти всегда в ответ на неправильный первый ответ от клиента сервер отправляет повторный начальный запрос (начинается заново процесс аутентификации).
- Согласно спецификации метода EAP-SIM первое ответное сообщение клиента EAP-Response/SIM/Start должно содержать атрибуты IDENTITY, NONCE_MT, SELECTED_VERSION. Если в сообщении присутствуют атрибуты NONCE_MT и SELECTED_VERSION, на другие неуместные атрибуты сервер не обращает внимания (в таких случаях сервер должен прерывать аутентификацию и отправлять сообщение об ошибке),
    - в том числе сервер не обращает внимания на присутствие или отсутствие атрибута клиента IDENTITY.
- Поле длины атрибута IDENTITY клиента не проверяется.
- Содержание атрибута IDENTITY клиента полностью игнорируется.
- Если длина атрибута NONCE_MT больше положенного (спецификация определяет фиксированную длину данного атрибута 20 байт), сервер начинается процесс аутентификации заново.
- Спецификация определяет фиксированную длину атрибута SELECTED_VERSION 4 байта. Сервер принимает данный атрибут большей длины, обрабатывая только первые 4 байта.

10.3 Тестирование туннельных методов

FreeRADIUS, метод TTLSv0

- В заголовке TTLSv0 поле Длина присутствует во всех фрагментах (При использовании протокола TLS большие блоки данных разбиваются на фрагменты и отправляются последовательно несколькими сообщениями EAP, при этом полная длина исходного TLS-сообщения передается только с первым фрагментом и не должна присутствовать в остальных фрагментах).
- Если во входящем сообщении установлен бит M (указывает, что используется фрагментация и текущий фрагмент не последний), реализация ждет следующие фрагменты независимо от значений других полей.
- Если во входящем сообщении установлен бит M, реализация требует, чтобы в следующих входящих фрагментах в заголовке TTLSv0 присутствовало поле длины. Если поле длины отсутствует, реализация отправляет пустое сообщение и ждет следующий фрагмент (такой пустой цикл можно повторить до 50 раз, после чего реализация завершает сессию).

- Значение поля Version (версия метода EAP-TTLS) не проверяется.
- В различных случаях не проверяется согласование длины пакета EAP, длины данных TLS и длины в заголовке TTLSv0.
- Во входящем сообщении Acknowledgement Packet (пустой пакет, означающий ожидание от партнера дополнительных данных или следующего фрагмента) значение поля длины при установленном бите L игнорируется.
- Во входящем сообщении Acknowledgement Packet значение бита S не проверяется (Данный бит устанавливается только в самом первом сообщении сессии для начала обмена TLS).
- Также обнаружена критическая уязвимость реализации. После установления TLS-туннеля при получении внутри туннеля данных определенного формата реализация выдает ошибку "Segmentation fault" и "падает". Данная уязвимость проявляется до аутентификации клиента (т.е. до использования туннелируемых методов аутентификации) и позволяет любому клиенту (злоумышленнику достаточно знать имя клиента, для которого применяется метод EAP-TTLS) провести DoS атаку ("отказ в обслуживании"). Уязвимость наблюдается для FreeRADIUS 3.0.13 под ОС CentOS 7 (release 7.4.1708).

Windows Server 2012, метод PEAP

- Реализация игнорирует бит S ("Start TLS") во входящем сообщении от клиента (данный бит используется для инициации сервером TLS обмена и применяется только в первом пустом сообщении от сервера клиенту).
- В некоторых случаях игнорирует значение поля длины в заголовке PEAP (данное поле указывает полную длину TLS сообщения до фрагментации).
- В некоторых случаях не проверяется согласование длины пакета EAP, длины данных TLS и длины в заголовке PEAP.
- Реализация отбрасывает входящие сообщения с битом M (указывает, что используется фрагментация и текущий фрагмент не последний), если считает, что текущее сообщение не должно быть фрагментировано (например, сообщение EAP-TTLS-TLS_ClientHello).
- Если реализация ждет следующий фрагмент от клиента, а получает пустое сообщение, то отправляет в ответ также пустое сообщение (EAP-TTLS Acknowledgement packet). Такой цикл можно повторять очень долго, в реализации вероятно отсутствует ограничение количества таких сообщений (например, FreeRADIUS прерывает аутентификацию после 50 таких циклов).

Clearbox Enterprise Server, метод PEAP

- В некоторых случаях игнорирует значение поля длины в заголовке PEAP (данное поле указывает полную длину TLS-сообщения до фрагментации).

- Значение поля Version (версия метода PEAP) не проверяется.
- Реализация требует, чтобы в сообщении Acknowledgement Packet (пустой пакет) в поле флаг все биты были 0, включая зарезервированные (Reserved bits). Согласно спецификации, зарезервированные биты должны устанавливаться в 0 при отправке и игнорироваться при получении.
- При получении пустого сообщения EAP-Response/type=EAP-TLS (без каких-либо данных) реализация падает. Данная ошибка наблюдается в туннельных методах при отправлении такого сообщения после создания TLS-туннеля (в качестве внутреннего метода EAP). Также такое поведение часто наблюдается в ответ на сообщение EAP-Request/TLS-Start (т.е. до согласования туннеля), но почему-то не всегда, иногда требуется несколько раз повторить тест.
- После установления защищенного туннеля (после сообщения TLS-Finished) для продолжения обмена партнер должен отправить серверу специальный пустой пакет Acknowledgement Packet. Если в этом сообщении в поле флаг установлен бит L, то реализация отвечает пакетом RADIUS/Access-Accept без атрибута EAP-Message.

TekRADIUS, метод PEAP

- Во время создания TLS-туннеля при получении сообщения EAP/TLS-ClientHello:
  - если установлен бит L (поле флаг метода PEAP, указывает на присутствие поля длины TLS сообщения), реализация игнорирует значения полей флага (в.ч. биты S и Version) и длины,
  - если установлен бит L, реализация игнорирует добавление лишних байт к сообщению,
  - если бит L не установлен, а другие биты поля флаг не 0 (включая Reserved bits), реализация игнорирует сообщение.
- Во время создания TLS-туннеля при получении сообщения EAP/TLS-Finished:
  - если установлен бит L, реализация игнорирует значения полей флага (в.ч. биты S и Version) и длины,
  - если бит L не установлен, а другие биты поля флаг не 0 (включая Reserved bits), реализация сообщает об ошибке (TLS-Alert),
  - если установлен бит M (указывает наличие следующих фрагментов) и сообщение не пустое (присутствуют TLS данные), реализация отправляет EAP пакет из одних нулей, включая заголовок EAP (т.е. значение Radius атрибута EAP-Message заполнено нулями, при этом остальные поля сообщения Radius правильные).
- После создания TLS-туннеля при инициализации внутреннего метода и получении неправильного сообщения реализация обычно отвечает

сообщением EAP из единственного нулевого байта (внутри туннеля).

- Также обнаружено интересное поведение реализации. После создания TLS-туннеля и получения сообщения Acknowledgement Packet, начинается согласование внутреннего метода EAP под защитой созданного туннеля. Реализация отправляет сообщение EAP-Identity. Если от партнера в ответ приходит EAP-Response/type=MD5 с произвольным значением контрольной MD5-суммы, то реализация отвечает сообщением RADIUS-Access-Challenge/EAP-Success. Такое поведение представляется нам очень странным и, несомненно, нарушает безопасность системы аутентификации.

11. Заключение

В данной статье представлены результаты работы по созданию тестового набора для тестирования соответствия реализаций протокола EAP и его методов спецификациям Интернет. Наша методика тестирования основана на использовании комбинации двух подходов к верификации реализаций сетевых протоколов. Первый подход базируется на создании формальных моделей протоколов и опирается на технологию UniTESK, обеспечивающую автоматизацию процесса верификации. В нашем проекте мы использовали расширение JavaTesK, реализующего эту технологию на языке программирования Java. Второй подход связан с использованием методов мутационного тестирования, которые позволяют протестировать устойчивость реализации протокола к искаженным сообщениям и значительно улучшить качество тестирования реализаций. Такая методика доказала свою эффективность, позволив обнаружить несколько критических уязвимостей и другие отклонения от спецификаций в выбранных реализациях протокола EAP и его методов.

Благодарности

Работа выполнялась при поддержке РФФИ (проект № 16-07-00603 «Верификация функций безопасности и оценка устойчивости к атакам реализаций протокола аутентификации EAP»).

Список литературы

[1]. IETF RFC 3748. B. Aboba, et al. Extensible Authentication Protocol (EAP). June 2004. Доступно по ссылке: https://tools.ietf.org/html/rfc3748, дата обращения 01.12.2018.

[2]. IETF RFC 3579. B. Aboba and P. Calhoun. RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP). September 2003. Доступно по ссылке: https://tools.ietf.org/html/rfc3579, дата обращения 01.12.2018.

[3]. Bourdonov I., Kossatchev A., Kuliamin V., and Petrenko A. UniTesK Test Suite Architecture. Proceedings of FME 2002. LNCS 2391, 2002, pp. 77–88

[4]. Никешин А.В., Шнитман В.З. Обзор расширяемого протокола аутентификации и его методов. Труды ИСП РАН, том 30, вып. 2, 2018 г., стр. 113-148. DOI: 10.15514/ISPRAS-2018-30(2)-7

[5]. IETF RFC 4186. Haverinen & Salowey. Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). January 2006. Доступно по ссылке: https://tools.ietf.org/html/rfc4186, дата обращения 01.12.2018.

[6]. IEEE Standard 802.1X-2010 - IEEE Standard for Local and metropolitan area networks – Port-Based Network Access Control, 2010.

[7]. JavaTESK. Доступно по ссылке: http://www.unitesk.ru/content/category/5/25/60/, дата обращения 01.12.2018.

[8]. FreeRADIUS. Доступно по ссылке: http://freeradius.org, дата обращения 01.12.2018.

[9]. Clearbox Enterprise Server. Доступно по ссылке: http://xperiencetech.com/, дата обращения 01.12.2018.

[10]. TekRADIUS. Доступно по ссылке: https://www.kaplansoft.com/tekradius/, дата обращения 01.12.2018.

[11]. Windows Server 2012 R2. Доступно по ссылке: https://www.microsoft.com, дата обращения 01.12.2018.

[12]. IETF RFC 4187. Arkko & Haverinen. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). January 2006. Доступно по ссылке: https://tools.ietf.org/html/rfc4187, дата обращения 01.12.2018.

[13]. IETF RFC 5448. Arkko, et al. Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA'). May 2009. Доступно по ссылке: https://tools.ietf.org/html/rfc5448, дата обращения 01.12.2018.

[14]. European Telecommunications Standards Institute. GSM Technical Specification GSM 03.20 (ETS 300 534): Digital cellular telecommunication system (Phase 2); Security related network functions, August 1997.

[15]. Microsoft Corporation. [MS-PEAP]: Protected Extensible Authentication Protocol (PEAP). December 2017. Доступно по ссылке: https://msdn.microsoft.com/en-us/library/cc238354.aspx, 06.12.2018, дата обращения 01.12.2018.

[16]. IETF RFC 5281. Funk & Blake-Wilson. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0). August 2008. Доступно по ссылке: https://tools.ietf.org/html/rfc5281, дата обращения 01.12.2018.

[17]. IETF RFC 4851. Cam-Winget, et al. The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST). May 2007. Доступно по ссылке: https://tools.ietf.org/html/rfc4851, дата обращения 01.12.2018.

[18]. IETF RFC 7170. Zhou, et al. Tunnel Extensible Authentication Protocol (TEAP) Version 1. May 2014. Доступно по ссылке: https://tools.ietf.org/html/rfc7170, дата обращения 01.12.2018.

[19]. IETF RFC 5246. Dierks, T. and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. August 2008. Доступно по ссылке: https://tools.ietf.org/html/rfc5246, дата обращения 01.12.2018.

Conformance testing of Extensible Authentication Protocol implementations

[1] *A.V. Nikeshin <alexn@ispras.ru>*
[1,2] *V.Z. Shnitman <vzs@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Moscow Institute of Physics and Technology (State University), 9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia*

Abstract. The paper presents a model-based approach to conformance testing of Extensible Authentication Protocol (EAP) implementations. Conformance testing is the basic tool to ensure interoperability between implementations of a protocol. Using UniTESK technology allows automating the verification of network protocols based on their formal models. Additional applying of mutation testing allows evaluating the robustness of the implementations to receive incorrect packets. We applied the test suite to several implementations of EAP and present brief results. This approach has proved to be effective in finding several critical vulnerabilities and other specification deviations in the EAP implementations.

Keywords: testing, verification, formal methods, formal specifications, model-based testing, security, authentication, access control, EAP, EAP methods, UniTESK, mutation testing.

DOI: 10.15514/ISPRAS-2018-30(6)-5

For citation: Nikeshin A.V., Shnitman V.Z. Conformance testing of Extensible Authentication Protocol implementations. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 89-104 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-5

References

[1]. IETF RFC 3748. B. Aboba, et al. Extensible Authentication Protocol (EAP). June 2004. Available at: https://tools.ietf.org/html/rfc3748, accessed 01.12.2018.
[2]. IETF RFC 3579. B. Aboba and P. Calhoun. RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP). September 2003. Available at: https://tools.ietf.org/html/rfc3579, accessed 01.12.2018.
[3]. Bourdonov I., Kossatchev A., Kuliamin V., and Petrenko A. UniTesK Test Suite Architecture. Proceedings of FME 2002. LNCS 2391, 2002, pp. 77–88.
[4]. Nikeshin A.V., Shnitman V.Z. The review of Extensible Authentication Protocol and its methods. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue. 2, 2018, pp. 113-148 (in Russian). DOI: 10.15514/ISPRAS-2018-30(2)-7.
[5]. IETF RFC 4186. Haverinen & Salowey. Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). January 2006. Available at: https://tools.ietf.org/html/rfc4186, accessed 01.12.2018.
[6]. IEEE Standard 802.1X-2010 - IEEE Standard for Local and metropolitan area networks – Port-Based Network Access Control, 2010.

[7]. JavaTESK. Available at: http://www.unitesk.ru/content/category/5/25/60/, accessed 01.12.2018.
[8]. FreeRADIUS. Available at: http://freeradius.org, accessed 01.12.2018.
[9]. Clearbox Enterprise Server. Available at: http://xperiencetech.com/, accessed 01.12.2018.
[10]. TekRADIUS. Available at: https://www.kaplansoft.com/tekradius/, accessed 01.12.2018.
[11]. Windows Server 2012 R2. Available at: https://www.microsoft.com/, accessed 01.12.2018.
[12]. IETF RFC 4187. Arkko & Haverinen. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). January 2006. Available at: https://tools.ietf.org/html/rfc4187/, accessed 01.12.2018.
[13]. IETF RFC 5448. Arkko, et al. Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA'). May 2009. Available at: https://tools.ietf.org/html/rfc5448/, accessed 01.12.2018.
[14]. European Telecommunications Standards Institute, GSM Technical Specification GSM 03.20 (ETS 300 534): Digital cellular telecommunication system (Phase 2); Security related network functions, August 1997.
[15]. Microsoft Corporation. [MS-PEAP]: Protected Extensible Authentication Protocol (PEAP). December 2017. Available at: https://msdn.microsoft.com/en-us/library/cc238354.aspx, 06.12.2018/, accessed 01.12.2018.
[16]. IETF RFC 5281. Funk & Blake-Wilson. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0). August 2008. Available at: https://tools.ietf.org/html/rfc5281/, accessed 01.12.2018.
[17]. IETF RFC 4851. Cam-Winget, et al. The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST). May 2007. Available at: https://tools.ietf.org/html/rfc4851/, accessed 01.12.2018.
[18]. IETF RFC 7170. Zhou, et al. Tunnel Extensible Authentication Protocol (TEAP) Version 1. May 2014. Available at: https://tools.ietf.org/html/rfc7170.
[19]. IETF RFC 5246. Dierks, T. and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. August 2008. Available at: https://tools.ietf.org/html/rfc5246/, accessed 01.12.2018.

Алгоритм построения расписаний выполнения параллельных задач на группах кластеров с процессорами различной производительности и его анализ в среднем[1]

[1] *Д.О. Лазарев <dennis810@mail.ru>*
[1,2] *Н.Н. Кузюрин <nnkuz@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский физико-технический институт, 141700, Московская область, г. Долгопрудный, Институтский пер., 9*

Аннотация. В работе рассмотрена задача построения расписаний выполнения параллельных вычислительных задач на группах кластеров с одинаковым числом $\boldsymbol{w}$ одинаковых процессоров, производительность которых для разных кластеров различная. Проведён вероятностный анализ задачи. Получены нижние оценки. Показано, что если число процессоров, необходимых для решения любой задачи имеет равномерное распределение на отрезке $[\boldsymbol{o}, \boldsymbol{w}]$ для любого алгоритма составления расписаний величина математического ожидания свободного объёма вычислений равна $\boldsymbol{\Omega}(\boldsymbol{w}\sqrt{\boldsymbol{N}})$. Получены верхние оценки. Был предложен онлайновый алгоритм построения расписаний с распределением задач в ограниченные области Limited Hash Scheduling для задачи построения расписаний, работающий в режиме closed-end, с математическим ожиданием свободного объёма вычислений, равным $\boldsymbol{O}\left(\boldsymbol{w}\sqrt{\boldsymbol{N}\ \boldsymbol{ln}\ \boldsymbol{N}}\right)$.

Ключевые слова: построение расписаний; онлайновый алгоритм; режим closed-end; вероятностный анализ; процессоры различной производительности; алгоритм размещения задач в ограниченные области; Limited Hash Scheduling.

DOI: 10.15514/ISPRAS-2018-30(6)-6

Для цитирования: Лазарев Д.О., Кузюрин Н.Н. Алгоритм построения расписаний выполнения параллельных задач на группах кластеров с процессорами различной производительности и его анализ в среднем. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр.105-122. DOI: 10.15514/ISPRAS-2018-30(6)-6

[1] Работа выполнена при финансовой поддержке РФФИ, проект 17-07-01006

1. Введение

Задача построения расписаний выполнения параллельных задач на группах кластеров с процессорами различной производительности (далее – задача построения расписаний), объединённых, например, сетями Grid[15] с одинаковым числом процессоров на кластерах ранее изучалась лишь в худшем случае. В случае однопроцессорных машин, в работе [1] был предложен алгоритм A, абсолютная точность R_A которого была не более 8. В работе [2] оценка абсолютной точности была улучшена до $3 + \sqrt{8} \approx 5.828$, а также был предложен рандомизированный алгоритм с абсолютной точностью $R_A \leq 4.311$.

Если представить каждую задачу в виде прямоугольника, ширине которого соответствует число процессоров, необходимых для решения задачи, а высоте прямоугольника без учёта сжатия- нормализованное время исполнения задачи(на кластере единичной производительности), то задача построения расписаний выполнения параллельных задач может рассматриваться, аналогично работе [10], как задача упаковки прямоугольников в несколько полубесконечных полос, в которой, попадая в полосу с номером j, прямоугольник сжимается по высоте в v_j раз, где v_j-скорость j-ой полосы. Только, ввиду того, что задача не обязательно должна занимать последовательные процессоры на кластере, прямоугольник может быть разбит по ширине на несколько меньших прямоугольников прямыми, параллельными боковым сторонам полос.

Таким образом, алгоритмы для задач упаковки в несколько полос могут оказаться эффективными для задачи построения расписаний. Так, например, любой алгоритм для задачи Multiple Strip Packing с полосами одинаковой ширины [6], [11] может использоваться для задачи построения расписаний для процессоров одинаковой производительности с одинаковым числом процессоров на кластере, а любой алгоритм для задачи упаковки прямоугольников в полосы различной ширины [12], [13], [14] может быть использован для задачи построения расписаний для процессоров одинаковой производительности с различными числами процессоров на кластерах. Обратное, однако, неверно.

В [3] С.Н. Жуку удалось получить оценку $R_A \leq 2e$ для задачи построения расписаний на группах кластеров с процессорами различной производительности с различным числом процессоров на кластерах, обобщив алгоритм из работы [4] для задачи упаковки прямоугольников в полосы разной ширины на случай задачи построения расписания на кластере.

В настоящей работе был построен онлайновый алгоритм для задачи построения расписаний Limited Hash Scheduling для N задач, с мат. ожиданием свободного объёма вычислений $E(V_{sp}) = O(w\sqrt{N \ln N})$. При построении алгоритма использовались идеи разбиения на области из работы [5] и упаковки не поместившихся в области прямоугольников из работы [6], а также

методы анализа в среднем алгоритмов из работы [7]. Так как предложенный алгоритм упаковывает каждую задачу на последовательные процессоры, то, положив все ускорения равными, можно свести классическую задачу Strip Packing к задаче построения расписаний, улучшив лучшую из известных оценок для задачи Strip Packing при анализе в среднем [8], [6] до $E(S_{sp}) = O(\sqrt{N \ln N})$.

Также было доказано, что при числе процессоров на каждом кластере, равном w и при числе процессоров, необходимых для решения каждой задачи, имеющем равномерное рапределение на $[o, w]$ любой алгоритм составления расписаний имеет математическое ожидание свободного объёма вычислений $E(V_{sp}) = \Omega(w\sqrt{N})$, где N- число задач, V_{sp} - свободный объём вычислений.

2. Постановка задачи

Имеется группа из k вычислительных кластеров с одинаковым числом процессоров w у каждого кластера и различной производительностью процессоров. Будем называть v_i ускорением кластера с номером i. Так, выполнение любой задачи T_j на кластере с номером i занимает в v_i раз меньше времени, чем нормализованное время выполнения этой задачи на кластере единичной производительности h_i. Требуется составить расписание выполнения задач без прерываний на k вычислительных кластерах, минимизирующее время L завершения выполнения последней задачи.

Будем рассматривать онлайновые алгоритмы, получающие задачи одну за другой и при распределении каждой задачи не знающие ни времени исполнения, ни числа процессоров следующих задач. Рассматриваем алгоритмы, работающие в режиме closed-end, т.е. знающие число задач N, для которых нужно составить расписание до получения первой задачи. Будем проводить вероятностный анализ в среднем случае в предположении, что числа процессоров, необходимых для решения каждой из задач w_i, $i \in \{1,2,\dots,N\}$- независимые в совокупности случайные величины, имеющие равномерное распределение на $[o, w]$, где w- число процессоров на каждом кластере. Предположим также, что нормализированное время выполнения каждой задачи на кластере единичной производительности h_i, $i \in \{1,2,\dots,N\}$- - также независимая случайная величина, имеющая равномерное на $[0,1]$ распределение.

Обозначим за L время завершения выполнения последней задачи, также L будем называть временем работы алгоритма (на данном наборе задач).

Для задачи i с числом процессоров w_i и нормализованным временем h_i, будем называть величину $w_i h_i$ объёмом вычислений.

За V_i обозначим суммарный объём вычислений по задачам, выполненным на вычислительном кластере с номером i.

За время не большее, чем L с ускорением w_i, на -ом кластере сумеем выполнить объём вычислений $V_i \leq v_i w L$. Свободным объём вычислений на кластере с номером i назовём $U_i = v_i w L - V_i \geq 0$.

За свободный объём вычислений примем сумму свободных объёмов вычислений по всем кластерам: $V_{sp} = \sum_{i=1}^{k} U_i$.

За W обозначим сумму объёмов вычислений для всех задач, которые надо распределить: $W = \sum_{i=1}^{N} h_i w_i$.

$$E(W) = \sum_{i=1}^{N} E(h_i)E(w_i) = \frac{Nw}{4}$$

За L' обозначим минимальное время, за которые задачи вычислительным объёмом W могут быть посчитаны на группе данных вычислительных устройств: $L' = \frac{W}{w\sum_{i=1}^{k} v_i}$.

За $\bar{L}$ обозначим минимальное время, за которые задачи вычислительным объёмом EW могут быть посчитаны на группе данных вычислительных устройств: $\bar{L} = \frac{E(W)}{w\sum_{i=1}^{k} v_i} = \frac{N}{4\sum_{i=1}^{k} v_i}$.

Будем оценивать качество работы алгоритма путём анализа величины $E(V_{sp})$.

Так как $L \geq L'$, то $L \geq \frac{W}{w\sum_{i=1}^{k} v_i}$, следовательно, $E(L) \geq \frac{E(W)}{w\sum_{i=1}^{k} v_i} = \frac{N}{4\sum_{i=1}^{k} v_i} = L$, поскольку $E(W) = \frac{Nw}{4}$.

Пусть время выполнения задач есть L. Тогда $E(V_{sp}) = E(L)w\sum_{i=1}^{k} v_i - E(W) = E(L - \bar{L})w\sum_{i=1}^{k} v_i$. Так как $EW = \bar{L}w\sum_{i=1}^{k} v_i$, то

$$\frac{E(V_{sp})}{E(W)} = \frac{E(L-\bar{L})}{\bar{L}} \quad (1)$$

Отсюда, доказательство того, что $E(V_{sp}) = O(w\sqrt{N \ln N})$, равносильно доказательству того, что $E(L) = \bar{L}(1 + O(\sqrt{\frac{\ln N}{N}}))$:

$$E(V_{sp}) = O(w\sqrt{N \ln N}) \Leftrightarrow E(L) = \bar{L}(1 + O(\sqrt{\frac{\ln N}{N}}))$$

3. Нижние оценки $E(V_{sp})$

Для получения нижнних оценок докажем следующую лемму:

Лемма 1. Пусть 2 случайные величины X и Y имеют непрерывные плотности распределений f_X и f_Y , причём X и Y - симметричны относительно своих мат. ожиданий и принимают лишь неотрицательные значения. Тогда $X + Y$ - тоже имеет непрерывную функцию плотности распределения и симметрична относительно своего мат. ожидания.

Доказательство. $f_{X+Y}(t) = \int_o^t f_X(\tau) f_Y(t - \tau) d\tau \Rightarrow f_{X+Y}(t)$ – непрерывна.
Без ограничения общности рассуждений предположим, что $E(X) \leq E(Y), t \leq \leq E(X) + E(Y)$.
Покажем, что $f_{X+Y}(t) = \int_o^t f_X(\tau) f_Y(t - \tau) d\tau = \int_o^t f_{X+Y}(2E(X) + 2E(Y) - t)(\tau) d\tau = \int_o^{2E(X)+2E(Y)-t} f_X(\tau) f_Y(2E(X) + 2E(Y) - t - \tau) d\tau$.

Посмотрим, при каких τ $f_X(\tau) f_Y(2E(X) + 2E(Y) - t - \tau)$ не равно нулю тождественно.

$$0 \leq \tau \leq 2E(X) \cap 0 \leq 2E(X) + 2E(Y) - t - \tau \leq 2E(Y) \Leftrightarrow 2E(X) - t \leq \leq \tau \leq 2E(X), \text{значит } f_{X+Y}(2E(X) + 2E(Y) - t) =$$

$$= \int_{2EX-t}^{2EX} f_X(\tau) f_Y(2E(X) + 2E(Y) - t - \tau) d\tau =$$

$$= (X \text{ и } Y - \text{симметричны относительно своих мат. ожиданий}) =$$

$$= \int_{2EX-t}^{2EX} f_X(2EX - \tau) f_Y(-2E(X) + t + \tau) d\tau = (\alpha = 2EX - \tau) =$$

$$= \int_o^t f_X(\alpha) f_Y(t - \alpha) d\alpha = f_{X+Y}(t) \blacksquare$$

Теорема 1.(О нижних оценках для задачи о составлении расписаний). При любом способе распределения N задач на группе кластеров с w процессорами различных скоростей, нормализованные времена исполнения которых - независимые в совокупности случайные величины, равномерно распределённые на $[0,1]$ и числа процессоров, занимаемые задачами- независимые в совокупности случайные величины, равномерно распределённые на $[o, w]$,

$$E(V_{sp}) = \Omega(\sqrt{N})$$

Обозначения.
За $W_{>w/2}$ обозначим суммарный объём вычислений для всех задач, занимающих больше, чем $\frac{w}{2}$ процессоров, а за $W_{\leq w/2}$ обозначим суммарный объём вычислений для всех задач, занимающих не более, чем $\frac{w}{2}$ процессоров.
За $h_{>w/2}$ обозначим суммарное нормализованное время исполнения для всех задач, занимающих больше, чем $\frac{w}{2}$ процессоров, а за $h_{\leq w/2}$ обозначим суммарное нормализованное время исполнения для всех задач, занимающих не более, чем $\frac{w}{2}$ процессоров.
За M обозначим число задач, число процессоров, необходимых для решения которых $w_i \geq \frac{w}{2}$.

Пусть $\bar{M} = \frac{N}{2} + \frac{\sqrt{N}}{2}$, где N- число задач.

Доказательство.

Если удастся показать, что

$$\exists \epsilon_0 : \mathbb{P}\left\{ wh_{>\frac{w}{2}} - W \geq \frac{w\sqrt{N}}{8} \right\} \geq \epsilon_0, \qquad (2)$$

то с вероятностью ϵ_0, $V_{sp} \geq \frac{w\sqrt{N}}{8}$, значит, $E(W) \geq \epsilon_0 w\sqrt{N}/8 = \Omega(w\sqrt{N})$.

Рассмотрим случайное событие Z, получающееся в результате двух испытаний:

1. Одно испытание Z_0 в схеме Бернулли, возвращающееся 0 или 1 с одинаковой вероятностью.
2. Если после первого испытания $Z_0 = 0$, то $Z = U(0, w/2]$, иначе, $Z = U(w/2, w]$, где за $U(a,b]$ обозначена случайная величина, имеющая равномерное распределение на полуинтервале $(a,b]$.

Плотность вероятности случайной величины Z поточечно равна плотности распределения равномерно распределённой на $(0,w]$ случайной величины $U(0,w]$, следовательно, $Z = U(0,w]$.

Поэтому будем рассматривать следующую последовательность испытаний I, в результате которой нам будут известны все параметры задач:

1. Серия испытаний Бернулли из N испытаний с вероятностью $\frac{1}{2}$ того, что число процессоров $w_i \leq \frac{w}{2}$ и с вероятностью $\frac{1}{2}$ того, что число процессоров $w_i > \frac{w}{2}$, $i \in \{1, \dots, N\}$.
2. Выбираем нормализованное время исполнения каждой из задач h_i, имеющее равномерное распределение на $[0,1], h_1, \dots, h_N$ - независимы в совокупности.
3. Для всех $1 \leq i \leq N$: из первого испытания $w_i \leq \frac{w}{2}$, выбираем w_i, как случайную велиину, равномерно распределённую на полуинтервале $(o, w/2]$, а для всех осталных i выбираем w_i, как случайную велиину, равномерно распределённую на полуинтервале $(w/2, w]$.

По интегральной теореме Лапласа-Муавра,

$$\mathbb{P}\left\{ M \geq \frac{N}{2} + \frac{\sqrt{N}}{2} \right\} \geq \mathbb{P}\left\{ 1 \leq \frac{M - \frac{N}{2}}{\frac{\sqrt{N}}{2}} \leq 2 \right\} \geq (N > N_0) \geq$$

$$\geq \frac{1}{\sqrt{2\pi}} \int_1^2 e^{-\frac{x^2}{2}} dx \geq 1/30$$

Пусть $M \geq \bar{M} = \frac{N}{2} + \frac{\sqrt{N}}{2}$. Рассмотрим $h_{w/2}$. Эта случайная величина равна сумме по всем M задачам, занимающим более $\frac{w}{2}$ процессоров, нормализованных времён исполнения этих задач,- случайных симметричных

относительно мат. ожидания случайных величин $U(0,1]$, имеющих равномерное распределение на $(0,1]$. Значит, по Лемме 1, $h_{>w/2}$ - имеет симметричное относительно своего математического ожидания, равного $\frac{M}{2}$, распределение, следовательно,

$$\mathbb{P}\left\{h_{>\frac{w}{2}} \geq \frac{M}{2}\right\} = \frac{1}{2}$$

Аналогично,

$$\mathbb{P}\left\{h_{\leq\frac{w}{2}} \leq \frac{N-M}{2}\right\} = \frac{1}{2}$$

Предположим, что после первых 2 шагов в последовательности испытаний I верно событие A: $\left(M \geq \frac{N}{2} + \frac{\sqrt{N}}{2}\right) \cap \left(h_{>\frac{w}{2}} \geq \frac{M}{2}\right) \cap \left(h_{\leq\frac{w}{2}} \leq \frac{N-M}{2}\right)$. $\mathbb{P}(A) \geq \frac{1}{120}$.

Тогда $W_{\frac{w}{2}} = \sum_{i:w_i>\frac{w}{2}} h_i w_i$. При выбранном h_i и равномерно распределённом на $(w/2, w]w_i$, случайная величина, равная $h_i w_i$- симметрична относительно своего мат. ожидания, равного $\frac{3wh_i}{4}$. Тогда, по Лемме 1, $W_{>w/2}$- симметрична относительно своего мат. ожидания, более или равного $\frac{3wh_i}{4}$ и $\mathbb{P}\left(W_{>\frac{w}{2}} \leq \frac{3wh_{\geq w/2}}{4}\right) \geq \frac{1}{2}$. Аналогично, $\mathbb{P}\left(W_{\leq\frac{w}{2}} \leq \frac{w(N-M)}{8}\right) \geq \frac{1}{2}$. Тогда обозначим за B следующее случайное событие:

$$B = \left(M \geq \frac{N}{2} + \frac{\sqrt{N}}{2}\right) \cap \left(h_{>\frac{w}{2}} \geq \frac{M}{2}\right) \cap \left(h_{\leq\frac{w}{2}} \leq \frac{N-M}{2}\right) \cap \left(W_{>\frac{w}{2}} \leq \frac{3wh_{>w/2}}{4}\right) \cap \left(W_{\leq\frac{w}{2}} \leq \frac{w(N-M)}{8}\right)$$

$$\mathbb{P}(B) \geq \frac{1}{480} = \epsilon_0$$

Если B - верно, то

$$\left(W = W_{\leq\frac{w}{2}} + W_{>\frac{w}{2}} \leq \frac{w\left(N + 6h_{>w/2} - M\right)}{8}\right) \cap \left(h_{>\frac{w}{2}} \geq \frac{M}{2}\right) \Rightarrow wh_{>\frac{w}{2}} - W \geq \frac{w\left((8-6)h_{>w/2} - N + M\right)}{8} \geq \frac{w\sqrt{N}}{8}$$

Значит, верна формула 2, и, стало быть, верно, что

$$E(V_{sp}) \geq \frac{1}{3840} w\sqrt{N} = \Omega\left(w\sqrt{N}\right) \blacksquare$$

4. Алгоритм Limited Hash Scheduling для задачи построения расписаний

Обозначения.

$$v_i = \frac{v_i}{2\sum_{i=1}^{k} v_i}\ , i \in \{1, \dots, k\}$$

$$\bar{L} = \frac{EW}{w\ \sum_{i=1}^{k} v_i} = \frac{N}{4\sum_{i=1}^{k} v_i}$$

$$s = \left\lfloor \frac{\sqrt{N}}{k} \right\rfloor$$

4.1 Разбиение на области

Определение 1. Назовём вычислительной областью, или просто областью некоторое подмножество пространства $\prod$ = время × процессоры. Будем говорить, что задача кладётся на верх текущей упаковки в вычислительной области, если задача начинает исполняться на процессорах, принадлежащих области во всё время исполнения задачи, в самый ранний момент времени из возможных после завершения всех предыдущих задач, исполняемых внутри данной области.

Предположим без ограничения общности рассуждений, что $v_1 \geq v_2 \geq \cdots \geq v_k$. Тогда рассмотрим вычислительную область $\prod_1$, равную произведению временного отрезка $[0, \bar{L}]$ на все процессоры всех кластеров. Разделим $\prod_1$ по времени на s меньших непересекающихся вычислительных областей с числом процессоров wk и временем исполнения $\frac{\bar{L}}{s}$. В итоге получим sk вычислительных областей шириной w и временем $[(i-1)\frac{\bar{L}}{s}, i\frac{\bar{L}}{s}]$, $\forall i \in \{1, \dots, k\}$, по s областей для каждого вычислительного кластера.

Области кластера с ускорением v_1, в свою очередь, разделим по ширине на 2 подобласти, состоящие из двух областей шириной

$$\frac{w\ i\ v_1}{s} \text{ и } w - \frac{w\ i\ v_1}{s}, \qquad \forall i \in \{1, \dots, s\}$$

Области кластера с ускорением $v_l, \forall l = \{1, \dots, k\}$, разделим по ширине на 2 подобласти, состоящие из двух областей шириной

$$\frac{w\ i\ v_l}{s} + w\sum_{i=1}^{l-1} v_i \text{ и } w - \left(\frac{w\ i\ v_l}{s} + w\sum_{i=1}^{l} v_i\right), \qquad \forall i \in \{1, \dots, s\}$$

Всего получим $2sk$ областей.

Ниже разбиение на области проиллюстрировано на рисунке 1, на котором полосам соответствуют кластеры с соответствующим ускорением(speeding), координате x- число процессоров, координате y- время(работы вычислительных кластеров).

4.2 Алгоритм упаковки в области Limited Hash Scheduling

Определение 2. Если после завеhшения выполнения предыдущих задач исполняемых в области, задачу можно исполнить на процессорах, в каждый момент времени исполнения задачи принадлежащих области, то говорим, что задача исполнима в области.

Определение 3. Скажем, что задача исполняется на верху текущей упаковки некоторой вычислительного кластера, если она начинает исполняться в самый ранний момент времени, но не ранее завершения выполнения всех областей(см. разбиение из предыдущего подраздела, т.е. не ранее, чем $\bar{L}$) и не ранее завершения выполнения всех задач, исполняемых на данном вычислительном кластере.

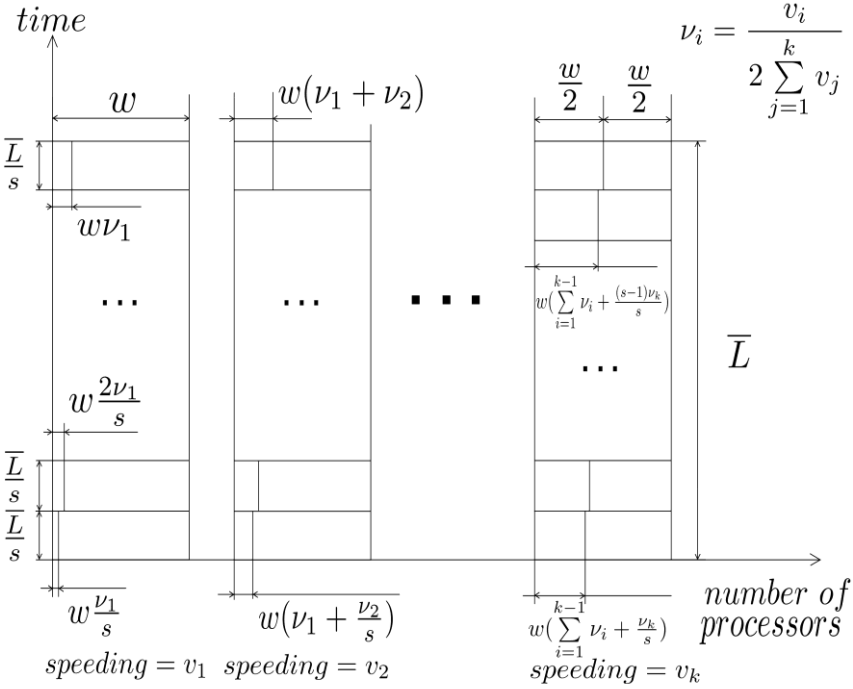


Рис. 1. Разбиение вычислительной области $\prod_1$ на $2sk$ меньших областей
Fig. 1. Hashing $\prod_1$ into $2sk$ areas with smaller volume of computations.

Алгоритм составления расписаний с использованием ограниченных областей Limited Hash Scheduling:

1. Если очередная задача исполнима в одной из $2sk$ областей, полученных в предыдущем подразделе(при условии исполнения предыдщих задач согласно алгоритму), то кладём задачу на верх текущей упаковки в области с минимальным числом процессором, в которой данная задача исполнима.

2. Иначе задача называется **выпавшей**, либо **попавшей в переполнение** и исполняется на верху текущей упаковки одного из тех вычислительных кластеров, при исполнении на которых время завершения задачи - самое меньшее из возможных.

5. Верхние оценки $E(V_{sp})$ для алгоритма Limited Hash Scheduling

В работе [7] была показана следующая Лемма, являющаяся усилением неравенства Азумы в случае определённым образом распределённых случайных величин.

Лемма. (Трушников, 2013.) Пусть случайная величина $X = X_1 + X_2 + \cdots + X_N$, где $X_i = \xi_i \eta_i$, ξ_i принимает значение 1 с вероятностью p и 0 с вероятностью $1 - p$, η_i — равномерно распределенная на отрезке (0,1] случайная величина, причем все случайные величины $\xi_i, \eta_i, i \in \{1, \dots, N\}$ независимы в совокупности. Тогда для любого α из интервала (0,1] выполняется неравенство

$$\mathbb{P}\{X > (1+\alpha)EX\} \le e^{-\frac{5}{9}\alpha^2 E(X)}$$

Докажем следующую лемму, покрывающую случаи $\alpha > 1$:

Лемма 2. В обозначениях предыдущей леммы, для любого α из интервала (0, 1] выполняется неравенство

$$\mathbb{P}\{X > (1+\alpha)EX\} \le e^{-\frac{5}{9}\alpha^2 E(X)}$$

,а при α > 1 верно, что

$$\mathbb{P}\{X > (1+\alpha)EX\} \le e^{-\frac{1}{3}\alpha E(X)}$$

Доказательство. Нужно доказать утверждение в случае $\alpha > 1$. В работе [7] было доказано, что

$$\mathbb{P}\{X > (1+\alpha)EX\} \le \exp\left(pN\left(\frac{(e^t - 1)}{t} - 1 - \frac{t}{2}(1+\alpha)\right)\right) \quad \forall t > 0$$

Выберем $t = 1$. Тогда

$$\mathbb{P}\{X > (1+\alpha)EX\} \le \exp\left(pN\left(e - 1 - 1 - \frac{1}{2}(1+\alpha)\right)\right)$$

$$\le \exp\left(pN\left(e - \frac{5}{2} - \frac{1}{2}\alpha\right)\right) \le \exp\left(pN\left(-\frac{1}{6}\alpha\right)\right) = e^{-\frac{1}{3}\alpha EX} \quad \blacksquare$$

Отсюда выведем следующую лемму:

Лемма 3. Пусть случайная величина $X = X_1 + X_2 + \cdots + X_N$, где $X_i = \xi_i \eta_i$, ξ_i принимает значение 1 с вероятностью p и 0 с вероятностью $1 - p$, η_i — равномерно распределенная на отрезке (0,1] случайная величина, причем все случайные величины $\xi_i, \eta_i, i \in \{1, \dots, N\}$ независимы в совокупности. Тогда при всех $N \ge N_0$

$$\mathbb{P}\{X > EX + 2\sqrt{N \ln N}\} \le N^{-3}$$

Доказательство.

1. $EX > 2\sqrt{N \ln N}$. Тогда по первой части Леммы 2,

$$\mathbb{P}\{X > EX + 2\sqrt{N \ln N}\} = \mathbb{P}\left\{X > EX\left(1 + \frac{2\sqrt{N \ln N}}{EX}\right)\right\}$$
$$\le \exp\left(-4\frac{5}{9}\frac{N \ln N}{E^2 X}EX\right) \le \left(\text{т. к. } EX < \frac{N}{2}\right) \le \exp\left(\frac{-40 \ln N}{9}\right) \le N^{-3}$$

2. $EX \le 2\sqrt{N \ln N}$. Тогда по второй части Леммы 2,

$$\mathbb{P}\{X > EX + 2\sqrt{N \ln N}\} = \mathbb{P}\left\{X > EX\left(1 + \frac{2\sqrt{N \ln N}}{EX}\right)\right\}$$
$$\le \exp\left(-\frac{2\sqrt{N\, ln\, N}}{3}\right) \le (\forall N \ge N_0) \le N^{-3} \quad \blacksquare$$

Упорядочим области по числу процессоров в порядке убывания: пусть первая область содержит $u_1 = w\left(1 - \frac{v_1}{s}\right)$ процессоров, …, область с номером $2sk$ содержит $u_{2sk} = w\frac{v_1}{s}$ процессоров. Скажем, что область с номером i имеет ускорение v_{n_i}.

Определние 4. За $H_i, i = \{1, \dots, 2sk\}$ обозначим сумму времён исполнения всех задач, число процессоров, необходимых для решения которых не превосходит ширину самой широкой области u_1, но строго больше, чем ширина $i-1$-ой области u_{i-1}: $H_i = \sum_{(1 \le j \le N) \cap (u_{i-1} < w_j \le u_1)} h_j$, где w_j и h_j- число процессоров и нормализованное время, необходимые для решения задачи номер j;

За H_0 обозначим сумму времён исполнения всех задач, число процессоров, необходимых для решения которых строго больше, чем ширина 1-ой области u_1: $H_i = \sum_{(1 \le j \le N) \cap (w_j > u_1)} h_j$.

Определние 5. За l_i обозначим сумму времён i самых широких областей, умноженную на ускорение того кластера, в вычислительном пространстве которой данная область лежит: $l_i = \sum_{j=1}^{i} v_{n_j}$.

Напомним, что задача, не исполнимая ни в одной из областей при условии составления расписания для предыдущих задач согласно с алгоритмом Limited Hash Scheduling, называется выпавшей задачей.

Определние 6. За H обозначим суммарное время исполнения всех выпавших задач.

Лемма 4.

$$H \le H_0 + \max\{0, \max_i\{H_i - l_i + i\}\}$$

Доказательство. Назовём область числом процессоров u_i не переполненной, если все задачи, для решения которых нужно не больше, чем u_i процессоров, упакованы в области числом процессоров $\leq u_i$.
Остальные области назовём переполненными.
Если область i ускорением v_{n_i}- переполнена, то некая задача, требующая не более, чем u_i процессоров, не исполнима ни в одной из областей шириной $\leq u_i$. Значит, суммарная нормализованная высота задач, исполняемых в -ой области не менее, чем $v_i \frac{\bar{L}}{s} - 1$.
Пусть i_0- номер не переполненной области с самым большим числом процессоров. Если все области - переполнены, то определим $i_0 = 2sk + 1$ и $u_i = 0$. Заметим, что все задачи с $w_i < u_{i_0}$ - не выпавшие. Если $i_0 = 1$, то $H = H_0$. Если $i_0 > 1$, в каждую из областей с номером $i < i_0$, попали задачи суммарным временем выполнения не менее, чем $v_i \frac{\bar{L}}{s} - 1$. Значит, $H \leq H_0 + (H_{i_0-1} - l_{i_0-1} + i_0 - 1) \leq H_0 + \max\{0, \max_i\{H_i - l_i + i\}\}$ ■

Предложение 1. После завершения работы алгоритма Limited Hash Scheduling при $H > 0$,

$$L \leq \bar{L} + \frac{H}{\sum_{i=1}^{k} v_i} + \frac{k}{\sum_{i=1}^{k} v_i},$$

где L- самое позднее время завершения задачи на одном из вычислительных кластеров, $\bar{L} = \frac{N}{4\sum_{i=1}^{k} v_i}$- время начала выполнения выпавших задач, H- суммарное время исполнения всех выпавших задач.

Обозначение.
За α_i, $i \in \{1, \dots, k\}$ обозначим $\max\{\bar{L}$, время завершения последней задачи на вычислителном кластере с номером $i\}$.

Доказательство. $H = \sum_{i=1}^{k} (\alpha_i - \bar{L}) v_i$. Заметим, что

$$L \leq \bar{L} + \frac{H}{\sum_{i=1}^{k} v_i} + \frac{k}{\sum_{i=1}^{k} v_i} \Leftrightarrow \sum_{i=1}^{k} (L - \alpha_i) v_i \leq k$$

Докажем последнее неравенство индукцией по числу попавших в переполнение задач, что $\sum_{i=1}^{k} (L^j - \alpha_i^j) v_i \leq k$, где L^j и α_i^j- промежуточные значения L и α_i после попадания в переполнение j задач.
База индукции при $j = 0$ очевидна: $0 \leq k$.
Переход. Пусть $\sum_{i=1}^{k} (L^{j-1} - \alpha_i^{j-1}) v_i \leq k$. Рассмотрим 2 случая:
1. $L_j = L_{j-1}$. Тогда $\sum_{i=1}^{k} (L^j - \alpha_i^j) v_i \leq \sum_{i=1}^{k} \left(L^{j-1} - \alpha_i^{j-1}\right) v_i \leq k$
2. $L_j > L_{j-1}$. Тогда из того, что, в соответствии с алгоритмом Limited Hash Scheduling, при вычислении задачи с номером j на верху текущей упаковки в том кластере, в который алгоритм размещает задачу, время завершения задачи минимальное и из того, $h_j \leq 1$, следует, что

$$L^j - \alpha_i^j \le \frac{1}{v_i} \ \forall 1 \le i \le k \Rightarrow \sum_{i=1}^{k} \left(L^{j-1} - \alpha_i^{j-1}\right) v_i \le k \quad \blacksquare$$

Предложение 2. $E(H_i) \le l_i \ \forall i \in \{1, \dots, k\}$ (см. определения 4 и 5.)

Доказательство. Пусть $\left[\frac{i}{s}\right] = m, i - ms = p.$

$$l_i = s\frac{\bar{L}}{s}\sum_{i=1}^{m} v_i + \frac{\bar{L}}{s}\sum_{i=1}^{p} v_{m+1} = \frac{\frac{N}{4}\left(\sum_{i=1}^{m} v_i + \frac{p}{s} v_{m+1}\right)}{\sum_{i=1}^{k} v_i}, \qquad i \le sk$$

$E(H_i) = \frac{N}{2}\,\mathbb{P}(u_{i-1} \le w_1 \le u_1)$, где w_1 - имеет равномерное распределение на $[0, w]$.

$$E(H_i) = \frac{N}{2}\frac{v_1\,\frac{s-1}{s} + \sum_{i=1}^{m} v_i + \frac{p+1}{s} v_{m+1}}{2\sum_{i=1}^{k} v_i} = l_i - \frac{N}{4\sum_{i=1}^{k} v_i}\frac{v_1 - v_{m+1}}{s}$$

$$\le l_i, \text{т. к. } v_1 \ge v_{m+1}, \text{при всех } i \le sk$$

Аналогично доказывается при $i > sk$, что $E(H_i) \le l_i \quad \blacksquare$

Теорема 2 (Верхние оценки $E(V_{sp})$ для алгоритма Limited Hash Scheduling). При составлении расписаний согласно алгоритму Limited Hash Scheduling, для числа вычислительных кластеров $k \le \sqrt{N}$ ускорением процессоров самого быстрого кластера $v_1 \le \sqrt{\ln N}\frac{\sum_{i=1}^{k} v_i}{k}$, верно, что для всех $N > N_0 \in \mathbb{N}$ математическое ожидание свободного объёма вычислений

$$E(V_{sp}) \le 4w\sqrt{N \ln N} = O(w\sqrt{N \ln N}),$$

где w- число процессоров на каждом из вычислительных кластеров.

Доказательство.

$$E\left(V_{sp}\right) = E\left(Lw\sum_{i=1}^{k} v_i - W\right) = E\left(Lw\sum_{i=1}^{k} v_i\right) - E(W) = E(L - \bar{L})w\sum_{i=1}^{k} v_i \quad (3),$$

где V_{sp}- свободный объём вычислений, L- время завершения исполнения последней задачи, W- суммарный объём вычислений по всем задачам, $\bar{L} = \frac{N}{4\sum_{i=1}^{k} v_i}$.

По предложению 1,

$$(L - \bar{L})w\sum_{i=1}^{k} v_i \le wH + wk \Rightarrow E(L - \bar{L})w\sum_{i=1}^{k} v_i \le E(wH) + E(kH), \qquad (4)$$

где H - суммарное время выполнения выпавших задач.

По Лемме 4, так как в силу выбора $s, \frac{\sqrt{N}}{k} \le s < 2\frac{\sqrt{N}}{k}$

$H \le H_0 + \max\{0, \max_i\{H_i - l_i + i\}\} \Rightarrow EH \le EH_0 +$

$$+E(\max\{0, \max_i\{H_i - l_i\}) + i \le (i \le 2sk)$$
$$\le E(H_0) + E(\max\{0, \max_i\{H_i - l_i\}\}) + 4\sqrt{N} \quad (5)$$

(см. определения 4 и 5)

В последней формуле заметим, что $E(H_0) = \frac{N}{2}\mathbb{P}\left(w_i \ge w\left(1 - \frac{v_1}{2s\sum_{i=1}^{k} v_i}\right)\right) = \frac{N}{2}\frac{v_1}{2s\sum_{i=1}^{k} v_i}$. По условию теоремы, $v_1 \le \sqrt{\ln N}\frac{\sum_{i=1}^{k} v_i}{k} \Rightarrow E(H_0) \le \frac{N}{2}\frac{\sqrt{\ln N}}{2sk} < \frac{\sqrt{N \ln N}}{2}$.

Также по предложению 2, выполнено, что $E(H_i) \le l_i$, следовательно, по лемме 3, $\forall i$, $\mathbb{P}(H_i \ge l_i + 2\sqrt{N \ln N}) \le \mathbb{P}(H_i \ge E(H_i) + 2\sqrt{N \ln N}) \le N^{-3}$. Значит, $\mathbb{P}\{\max_i\{H_i - l_i\} \ge 2\sqrt{N \ln N}\} \le N^{-2}$

Стало быть, $E(max\{0, \max_i\{H_i - l_i\}\}) \le 2\sqrt{N \ln N} + O(N^{-1})$. Подставим полученные результаты в формулу 5:

$$E(H) = E(H_0) + E(\max\{0, \max_i\{H_i - l_i\}\}) + 4\sqrt{N}$$
$$\le \sqrt{N \ln N}/2 + 2\sqrt{N \ln N} + O(N^{-1}) + 4\sqrt{N}$$

Подставляя полученный результат в формулы 3 и 4, имеем:

$$E(V_{sp}) \le E(wH) + kw \le w\left(\frac{\sqrt{N \ln N}}{2} + 2\sqrt{N \ln N} + O(N^{-1}) + 4\sqrt{N}\right) + kw \le$$
$$w\left(\frac{\sqrt{N \ln N}}{2} + 2\sqrt{N \ln N} + O(N^{-1}) + 5\sqrt{N}\right) \le (N > N_0) \le 4w\sqrt{N \ln N}$$
$$= O(w\sqrt{N \ln N}) \quad \blacksquare$$

6. Заключение и направление дальнейших исследований

Была рассмотрена задача построения расписаний выполнения вычислительных задач на группах вычислительных кластеров с одинаковым числом w одинаковых процессоров, производительность которых для разных кластеров различная. Число процессоров, необходимых для решения задач является случайной величиной, имеющих равномерное на $[0, w]$ распределение. Был проведён вероятностный анализ. Для любого алгоритма построения расписаний была получена оценка снизу на математическое ожидание свободного объёма вычислений $E(V_{sp}) = \Omega(w\sqrt{N})$.

Был предложен алгоритм Limited Hash Scheduling размещения задач в ограниченные области с $E(V_{sp}) = \Omega(w\sqrt{Nln\,N})$, работающий в режиме closed-end, то есть знающий число задач N до начала построения расписания. Алгоритм с меньшей шириной областей также может быть применён для числа процессоров , необходимых для решения каждой задачи, равномерно распределённом на $[o, u]$, при $w \vdots u$. В общем случае при $u < v$, даже для

задачи Bin Packing не известно онлайнового алгоритма с математическим ожиданием незаполненного пространства контейнерои $E(L_{sp}) = o(N)$ [9].

Большой интерес представляет исследование задачи в случае алгоритмов, работающих в режиме open-end, т.е. не имеющих никаких данных о числе задач, для которых требуется построить расписание. В работе [9] для задачи Bin Packing упаковки объектов размером, имеющим равномерное распределение на $[0, u]$, $u \leq 1$ в контейнеры единичного размера при анализе в среднем в режиме open-end $E(L_{sp}) = \Omega(\sqrt{N})$. Интересен вопрос о возможности получении оценки $E(V_{sp}) = \Omega(w\sqrt{N})$ для любого онлайнового алгоритма, работающего в режиме open-end для задачи построения расписаний.

Список литературы

[1]. Aspens J., Azar Y., Fiat A., Plotkin S., Waarts O. On - line load balancing with applications to machine scheduling and virtual circuit routing. In Proc. of the 25th ACM STOC. 1993. pp. 623 - 631, DOI: 10.1145/167088.167248

[2]. Berman P., Charikar M., Karpinski M. On-line load balancing for related machines. LNCS, v. 1272, 1997, pp. 116-125, DOI: https://doi.org/10.1007/3-540-63307-3_52

[3]. С.Н. Жук. О построении расписаний выполнения параллельных задач на гуппах кластеров с различной производительностью. Труды ИСП РАН, том 23, 2012, стр. 447-454, DOI: 10.15514/ISPRAS-2012-23-27

[4]. S.N. Zhuk. Approximate algorithms for packing rectangles into several strips. Discrete Mathematics and Applications, vol 18, issue 1, 2006, pp. 91-105, DOI: https://doi.org/10.1515/156939206776241264

[5]. М.А. Трушников. Об одной задаче Коффмана-Шора, связанной с упаковкой прямоугольников в полосу. Труды ИСП РАН, том 22, 2012, стр. 456-462, DOI: 10.15514/ISPRAS-2012-22-24

[6]. Лазарев Д. О., Кузюрин Н. Н. Алгоритм упаковки прямоугольников в несколько полос и анализ его точности в среднем. Труды ИСП РАН, том 29, вып. 6, 2017 г., стр. 221-228, DOI: 10.15514/ ISPRAS-2017-29(6)-13

[7]. Трушников М. А. Вероятностный анализ нового алгоритма упаковки прямоугольников в полосу., Труды ИСП РАН, том 24, 2013, стр. 457-468, DOI: 10.15514/ISPRAS-2013-24-21

[8]. Лазарев Д.О., Кузюрин Н.Н. Об онлайновых алгоритмах для задач упаковки в контейнеры и полосы, их анализе в худшем случае и в среднем. Труды ИСП РАН, том 30, вып. 4, 2018 г., стр. 209-230. DOI:10.15514/ISPRAS-2018-30(4)-14

[9]. E. G. Coffman, C. Courcoubetis, M. R. Garey, D. S. Johnson, L. A. McGeoch, P. W. Shor, R. R. Weber, M. Yannakakis. Fundamental discrepancies between average-case analyses under discrete and continuous distributions: a bin packing case study. In Proc. of the twenty-third annual ACM Symposium on Theory of computing, 1991, pp. 230-240, doi:10.1145/103418.103446

[10]. Кузюрин Н., Грушин Д., Фомин С. Проблемы двумерной упаковки и задачи оптимизации в распределенных вычислительных системах. Труды ИСП РАН, том 26, вып. 1, 2014 г., стр. 483–502, DOI: 10.15514/ISPRAS-2014-26(1)-21

[11]. Ye D., Han X., Zhang G. Online multiple-strip packing. Theoretical Computer Science, Volume 412, Issue 3, 2011, pp. 233-239. DOI: https://doi.org/10.1016/j.tcs.2009.09.29

[12]. Жук С.Н. Онлайновый алгоритм упаковки прямоугольников в несколько полос с гарантированными оценками точности. Труды ИСП РАН, том 12, 2007 г., стр. 7-16, ISSN 2079-8056

[13]. Zhuk S.N. On-line algorithms for packing rectangles into several strips. Discrete Mathematics and Applications, vol. 17, issue 5, 2007, pp. 517-531. DOI: https://doi.org/10.1515/dma.2007.040

[14]. Жук С.Н. О построении расписаний выполнения параллельных задач на группах кластеров с различной производительностью. Труды ИСП РАН, том 23, 2012 г., стр. 447-454. DOI: https://doi.org/10.15514/ISPRAS-2012-23-27

[15]. Foster I., Kesselman C. The grid: Blueprint for a new computing infrastructure. Morgan Kaufmann Publishers Inc., 1999, 748 p.

On-line algorithm for scheduling parallel tasks on related computational clusters with processors of different capacities and its average-case analysis.

[1] *D.O. Lazarev <dennis810@mail.ru>*
[1,2] *N.N. Kuzyurin <nnkuz@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *Moscow Institute of Physics and Technology (State University), 9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia*

Abstract. In this article the on-line problem of scheduling parallel tasks on related computational clusters with processors of different capacities was studied in average case. In the problem the objective is to make a schedule on $\boldsymbol{k}$ clusters with $\boldsymbol{w}$ processors each of $\boldsymbol{N}$ tasks, where the task $\boldsymbol{i}, \boldsymbol{i} \leq \boldsymbol{N}$ requires time $\boldsymbol{h_i}$ on cluster with nominal capacity of processors and $\boldsymbol{w_i} \leq \boldsymbol{w}$ processors. We presume for all $\boldsymbol{1} \leq \boldsymbol{i} \leq \boldsymbol{N}$ that $\boldsymbol{w_i}$ has uniform distribution on $(\boldsymbol{0}, \boldsymbol{w}]$ and that $\boldsymbol{h_i}$ has uniform distribution on $(\boldsymbol{0}, \boldsymbol{1}]$. The processors on different clusters have different capacities $\boldsymbol{v_1}, \dots, \boldsymbol{v_k}$. The task with nominal time $\boldsymbol{h_i}$ will require $\boldsymbol{w_i}$ processors be computed in time $\frac{\boldsymbol{h_i}}{\boldsymbol{v_j}}$ on cluster number $\boldsymbol{j}$. Let sum volume of computations $\boldsymbol{W}$ be the sum of volumes of computations for each task: $\boldsymbol{W} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{w_i h_i}$. Let $\boldsymbol{L}$ be the minimal time at which all clusters will compute all the tasks, assigned to them, where each task is assigned to one cluster. The expected value of free volume of computations $\boldsymbol{E}(\boldsymbol{V_{sp}})$is used to analyze the quality of an algorithm, where $\boldsymbol{V_{sp}} = \sum_{\boldsymbol{1 \leq i \leq k}} \boldsymbol{v_i}\ \boldsymbol{L} - \boldsymbol{W}$. It was shown that for every algorithm for scheduling parallel tasks on related clusters $\boldsymbol{E}(\boldsymbol{V_{sp}}) = \boldsymbol{\Omega}(\boldsymbol{w}\sqrt{\boldsymbol{N}})$. An on-line algorithm Limited Hash Scheduling was proposed, which has $\boldsymbol{E}(\boldsymbol{V_{sp}}) \leq \boldsymbol{4}\left(\boldsymbol{w}\sqrt{\boldsymbol{N}\ln \boldsymbol{N}}\right) = \boldsymbol{O}\left(\boldsymbol{w}\sqrt{\boldsymbol{N}\ln \boldsymbol{N}}\right), \boldsymbol{for}\ \boldsymbol{N} > \boldsymbol{N_0} \in \mathbb{N}$ if $\boldsymbol{k} \leq \sqrt{\boldsymbol{N}}$ and $\boldsymbol{v_j} \leq \sqrt{\ln \boldsymbol{N}} \frac{\sum_{i=1}^{k} v_i}{\boldsymbol{k}}\ \forall\ \boldsymbol{1} \leq \boldsymbol{j} \leq \boldsymbol{k}$. The idea of the algorithm is to schedule tasks of close required number of required processors into different limited in time and the number of allowed to use processors areas on clusters.

Keywords: on-line algorithm; closed-end; probabilistic analysis; processors of different capacities; scheduling using limited computational areas; Limited Hash Scheduling.

DOI: 10.15514/ISPRAS-2018-30(6)-6

For citation: Lazarev D.O., Kuzyurin N.N. On-line algorithm for scheduling parallel tasks on related computational clusters with processors of different capacities and it's average-case analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 105-122 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-6

References

[1]. M. Aspens J., Azar Y., Fiat A., Plotkin S., Waarts O. On - line load balancing with applications to machine scheduling and virtual circuit routing. In Proc. of the 25th ACM STOC. 1993. pp. 623 - 631, DOI: 10.1145/167088.167248

[2]. Berman P., Charikar M., Karpinski M. On-line load balancing for related machines. LNCS, v. 1272, 1997, pp. 116-125, DOI: https://doi.org/10.1007/3-540-63307-3_52

[3]. Zhuk S.N. On-line algorithm for scheduling parallel tasks on a group of related clusters. Trudy ISP RAN/Proc. ISP RAS, vol. 23, 2012, pp. 447-454 (in Russian). DOI: 10.15514/ISPRAS-2012-23-27

[4]. S.N. Zhuk. Approximate algorithms for packing rectangles into several strips. Discrete Mathematics and Applications, 2006, vol 18, issue 1, pp. 91-105, DOI: https://doi.org/10.1515/156939206776241264

[5]. M. A. Trushnikov. On one problem of Koffman-Shor connected to strip packing problem. Trudy ISP RAN/Proc. ISP RAS, vol. 22, 2012, pp. 456-462 (in Russian). DOI: 10.15514/ISPRAS-2012-22-24 p.

[6]. Lazarev D.O., Kuzyrin N.N. An algorithm for Multiple Strip Package and its average case evaluation. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 6, 2017. pp. 221-228 (in Russian). DOI: 10.15514/ISPRAS-2017-29(6)-1

[7]. M. A. Trushnikov. Probabilistic analysis of a new strip packing algorithm. Trudy ISP RAN/Proc. ISP RAS, vol. 24, 2013, pp. 457-468 (in Russian). DOI: 10.15514/ISPRAS-2013-24-21

[8]. Lazarev D.O., Kuzjurin N.N. On on-line algorithms for Bin, Strip and Box Packing, and their worst- and average-case analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 4, 2018. pp. 209-230 (in Russian). DOI: 10.15514/ISPRAS-2018-30(4)-14

[9]. E. G. Coffman, C. Courcoubetis, M. R. Garey, D. S. Johnson, L. A. McGeoch, P. W. Shor, R. R. Weber, M. Yannakakis. Fundamental discrepancies between average-case analyses under discrete and continuous distributions: a bin packing case study. In Proc. of the twenty-third annual ACM Symposium on Theory of computing, 1991, pp. 230-240, doi:10.1145/103418.103446

[10]. N.N. Kuzyurin, D.A. Grushin, S.A. Fomin. Two-dimensional packing problems and optimization in distributed computing systems. Trudy ISP RAS/Proc. ISP RAS. 2014, vol. 26, issue 1, 2015, pp. 483–502 (in Russian). DOI: 10.15514/ISPRAS-2014-26(1)-21

[11]. Ye D., Han X., Zhang G. Online multiple-strip packing. Theoretical Computer Science, Volume 412, Issue 3, 2011, pp. 233-239. DOI: https://doi.org/10.1016/j.tcs.2009.09.29

[12]. Zhuk S.N. Online algorithm for packing rectangles into several strips with guaranteed accuracy estimates. Trudy ISP RAN/Proc. ISP RAS, vol. 12, 2007, pp. 7-16 (in Russian). ISSN 2079-8056

[13]. Zhuk S.N. On-line algorithms for packing rectangles into several strips. Discrete Mathematics and Applications, vol. 17, issue 5, 2007, pp. 517-531. DOI: https://doi.org/10.1515/dma.2007.040

[14]. Zhuk S.N. On-line algorithm for scheduling parallel tasks on a group of related clusters. Trudy ISP RAN/Proc. ISP RAS, vol. 23, 2012, pp. 447-454 (in Russian). DOI: https://doi.org/10.15514/ISPRAS-2012-23-27

[15]. Foster I., Kesselman C. The grid: Blueprint for a new computing infrastructure. Morgan Kaufmann Publishers Inc., 1999, 748 p.

О задаче эффективного управления вычислительной инфраструктурой[1]

[1] *Д.А. Грушин <grushin@ispras.ru>*
[1,2] *Н.Н. Кузюрин <nnkuz@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский физико-технический институт, 141700, Московская область, г. Долгопрудный, Институтский пер., 9*

Аннотация. В настоящее время такие компании как Amazon, Google, Facebook, Microsoft, Yahoo! и другие управляют огромными дата-центрами из десятков тысяч узлов. Эти кластеры используются одновременно многими организациями и пользователями (облачная модель вычислений). Пользователи запускают задания, каждое из которых может состоять из одной или нескольких параллельных задач. Поток задач обычно представляет собой смесь: коротких, долгих, интерактивных, пакетных, и задач с различным приоритетом. Планировщик кластера решает, как разместить эти задачи на узлах, где они запускаются в виде процессов, контейнеров или виртуальных машин. Оптимизация размещения задач планировщиком позволяет улучшить степень использования узлов (machine utilization), сократить время отклика, сбалансировать нагрузку на части кластера, получить предсказуемую производительность приложений, повысить отказоустойчивость. Получить оптимальное размещение сложно. Для этого требуется решить задачу многокритериальной оптимизации, что требует времени. Это приводит к задержке при размещении очередной задачи, негативно сказывается на времени отклика и пропускной способности. С ростом размеров кластера и потока задач удовлетворить оба данных критерия становится сложным, и необходимо отдавать приоритет только одному. В данной статье мы рассмотрим архитектуру современных планировщиков и математические постановки задачи оптимизации.

Ключевые слова: оптимизация; планирование; виртуализация; облачные вычисления

DOI: 10.15514/ISPRAS-2018-30(6)-7

Для цитирования: Грушин Д.А., Кузюрин Н.Н. О задаче эффективного управления вычислительной инфраструктурой. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 123-142. DOI: 10.15514/ISPRAS-2018-30(6)-7

[1] Работа выполнена при финансовой поддержке РФФИ, проект 17-07-01006.

1. Введение

С начала 2000 годов началось активное внедрение вычислительных кластеров[2] за счет использования стандартных компонент: обычных двух- или четырехпроцессорных рабочих станций (или персональных компьютеров) и коммуникационного оборудования (Myrinet, SCI, Fast Ethernet и др.) [4].

Основной частью спектра решаемых задач были параллельные MPI-программы [5]. Для решения таких задач кластер мог обходиться без системы управления (программа запускалась напрямую на всех узлах). Однако такой режим с трудом применим даже для одного пользователя. Для гибкого совместного использования дорогого вычислительного ресурса, система управления должна разделять кластер между поступающими от пользователей заданиями. Для этого предусмотрена очередь или набор очередей; необходимое количество узлов для запуска каждого задания выделяется таким образом, чтобы максимально использовать вычислительные возможности кластера. Более подробный список основных функций, обеспечиваемых системами управления, был впервые сформулирован в работе [6] и, позднее, в [7].

С развитием сетевых технологий стала набирать популярность архитектура Grid. Grid позволяет совместно использовать вычислительные ресурсы, которые принадлежат различным организациям и которые могут быть расположены в различных, географически разделенных административных областях. В Grid могут объединяться разнородные вычислительные ресурсы: персональные компьютеры, рабочие станции, кластеры и супер-компьютеры [8].

С появлением технологии виртуализации в процессорах и реализацией поддержки в операционных системах вычислительная инфраструктура начала стремительно изменяться. Между аппаратным и прикладным уровнями появился уровень виртуализации. Начался бум облачных вычислений. Теперь выделить для запуска программы набор “узлов” с одинаковым окружением стало значительно проще. Однородные кластеры стали превращаться в наборы разнородных серверов (в большей степени по причине обновления оборудования), сильно увеличиваясь в размерах, при этом спектр решаемых задач значительно расширился.

К сожалению, в настоящее время только небольшое число ученых имеет доступ к большим кластерам, и большинство работ на тему оптимизации вычислений сфокусировано на получении оптимального размещения в рамках ограниченных ресурсов. В реальности, задача оптимизации для конкретного кластера имеет множество возможных решений, относительная полезность каждого из которых зависит от характеристик потока задач и целей оператора кластера.

[2] Кластер определяется как множество рабочих станций (узлов), связанных коммуникационной средой и способных работать как единое целое за счет дополнительного программного обеспечения, называемого системой управления кластером.

Поток задач в большинстве случаев разнородный и на практике алгоритмы планировщика требуют ручной настройки для достижения хорошего качества размещения. С ростом размеров кластера данная задача становится трудновыполнимой.

Для принятия решения о размещении очередной задачи планировщику требуется время. Чем сложнее решение, тем это время больше. Для современных планировщиков оно находится в интервале от нескольких сотен миллисекунд до нескольких секунд. Очевидно, с ростом потока задач данная задержка будет приводить к увеличению времени отклика и времени ожидания, простою вычислительных ресурсов, создавая обратный эффект.

Конечно, проблема масштабирования систем управления и планировщиков больших кластеров не касается большинства пользователей. Наиболее простой способ масштабирования - это покупка необходимого количества ресурсов у облачного провайдера (Amazon Web Services, Google или Microsoft). Однако, масштабирование может быть важно и для небольших кластеров, если поток состоит из большого количества коротких интерактивных задач [9-10].

Также важно отметить, что качественное размещение экономит реальные деньги. Это важно как для владельцев больших систем (в Google отмечали, что их система планирования позволила избежать траты миллиардов долларов на строительство нескольких дата-центров [11]), так и для небольших организаций, где потеря нескольких сотен долларов в месяц из-за недогруженных виртуальных машин имеет значение.

Следует подчеркнуть, что оптимизация размещения задач на группах кластеров в различных моделях очень тесно связана с проблемами упаковки.

Эти вопросы будут рассмотрены далее для различных моделей.

2. Задача оптимизации размещения для однородного MPI кластера

Для параллельной программы необходимо выделить заданное количество процессорных ядер кластера в эксклюзивное пользование. В противном случае (две задачи используют одно ядро – overbooking) производительность сильно снижается, что приводит к непредсказуемому увеличению времени работы задач [12].

Узлы однородного кластера одинаковы, таким образом, кластер можно охарактеризовать количеством узлов (размер кластера).

Для этого нового класса задач в теории расписаний была предложена интересная геометрическая интерпретация в виде задачи упаковки последовательности прямоугольников в полубесконечную полосу. При этом ширина полосы соответствовала размеру кластера, а размеры прямоугольников соответствовали требованиям числа процессоров для задачи и времени исполнения. Эти модели очень тесно связаны [13-17].

Задача упаковки прямоугольников в полосы ставится следующим образом. Имеется набор различных прямоугольников и вертикальных (полубесконечных) полос. Стороны прямоугольников параллельны сторонам полос. Требуется упаковать прямоугольники в полосы без наложений и вращений (ортогональная ориентированная упаковка), оптимизируя заданные критерии (максимальную высоту заполнения, плотность заполнения, число прямоугольников в заполнении и другие). Эта модель очень тесно связана с оптимизацией выполнения параллельных задач на группах кластеров. Нам удалось получить целый ряд интересных и не имеющих аналогов результатов [18-22].

На практике наиболее распространенным способом распределения задач на кластере является алгоритм Backfill, когда для заполнения “пустых мест” в расписании используются задачи не с начала очереди [23]. При этом время работы задачи неизвестно, либо задается пользователем. В последнем случае оказывается, что время работы пользователи задают с большой положительной погрешностью, так как система принудительно завершает задачи по истечении выделенного времени [24]. В [25] было предложено не использовать время задаваемое пользователем, а предсказывать его на основе истории запуска (журнала) задач.

Задача упаковки прямоугольников в применении к вычислительному кластеру оптимизирует плотность размещения задач. В общем случае критериев оптимизации размещения задач на кластере может быть несколько. Наиболее распространены варианты:

- минимизация среднего времени ожидания задачи в очереди;
- минимизация максимального времени выполнения группы задач (makespan);
- максимизация пропускной способности - числа завершенных задач в единицу времени;
- минимизация простоев процессоров.

Для однородного кластера также интересной является задача снижения энергопотребления за счет гибких стратегий управления состоянием узлов (включения/выключения) и порядком выполнения заданий в очереди.

Основные способы, основаны на использовании программных средств, позволяющих повысить энергоэффективность кластера [26].

- Упаковка задач на минимальном количестве узлов и последующее выключение простаивающих узлов кластера.
- Перераспределение вычислительных заданий по времени при условии наличия многотарифной схемы оплаты электроэнергии (например, день-ночь). Тогда за счет повышения загрузки системы ночью днем вычислительная нагрузка будет снижена, и простаивающие компоненты вычислительной системы отключены. При этом общее количество потребленной электроэнергии не снижается, однако уменьшается ее

стоимость, что также рассматривается как повышение энергоэффективности.

- Программное управление производительностью компонентов вычислительной системы. Современные процессоры и оперативная память имеют возможность динамически изменять свою частоту и рабочее напряжение. Такой механизм носит название DVS (dynamic voltage and frequency scaling). Основной принцип данного механизма заключается в том, что при понижении напряжения процессора время вычислений увеличивается, однако общее количество энергии, потраченной на вычисления, уменьшается.

Поступающие задачи обычно требуют различных уровней “качества обслуживания” (Quality of Service – QoS). В применении к размещению задач качество обслуживания определяется как время от постановки задачи в очередь до получения результатов пользователем.

На практике данный критерий является основным. В некоторых случаях, возможно незначительно снижать качество обслуживания получая при этом значительную оптимизацию по другому критерию. Однако, ни один планировщик не может иметь дело с широким диапазоном ограничений и требований задаваемых как пользователями, так и провайдерами ресурса (лицензионные и экономические ограничения, отказоустойчивость, перегрев, пул свободных адресов и др.) [27].

Данная проблема решается возможностью ручной настройки (администратор ресурса определяет степень важности критериев) и вариациями стратегий распределения, разработанных с учетом конкретных потребностей.

3. Grid, расширение задачи на несколько полос

С развитием Grid-вычислений появилась возможность управлять потоком задач распределяя их на различные группы кластеров. При этом кластеры могут иметь различное число процессоров.

В связи с различными размерами кластеров задача оптимизации усложнилась [28] и стала по существу двух-уровневой: сначала для задачи выбирается кластер, а потом оптимизация размещения на кластере выполняется одной из известных эвристик. Геометрическая модель упаковки в одну полосу была обобщена нами на случай нескольких полос различной ширины [18], [22], [29-32], [20], [15].

Большинство работ, связанных с темой оптимизации вычислений в Grid, исследуют влияние различных мета-алгоритмов как на первом уровне, так и на втором уровне (распределение на кластере) [33].

Одним из критериев оптимизации также может быть минимизация потребляемой энергии. Согласно статистике, большинство кластеров испытывает периодическую нагрузку – когда интенсивность потока задач различается в несколько раз в разное время суток. Это означает, что даже при

относительно плотной загрузке заданиями в среднем, существуют периоды, когда большая часть узлов кластера не выполняет заданий и простаивает [26].

Для одиночного кластера, загруженного неравномерным потоком задач, достаточным является отключение простаивающих узлов. При этом от планировщика требуется применять различные стратегии сжатия - размещения максимального количества задач на минимальном количестве узлов.

В системе из нескольких кластеров, находящихся под управлением одного менеджера ресурсов – брокера существует несколько возможностей для экономии электроэнергии (как в количественном смысле, так и в денежном - снижая стоимость энергии) [34].

- Различная энергоэффективность (отношение производительности к энергопотреблению) кластеров;
- Географическое положение кластеров. Стоимость энергии в разных регионах может существенно различаться. Отправляя задачи на кластер с более низкой ценой электроэнергии можно уменьшить общую стоимость. Стоимость энергии различается также в разное время суток, что даёт дополнительные возможности выбора если кластеры находятся в разных часовых поясах.

4. Виртуализация

Виртуализация позволяет нескольким виртуальным машинам, часто с различными операционными системами, работать изолированно на одной физической машине. Каждая виртуальная машина имеет свой собственный набор виртуального оборудования (процессор, память, сетевые интерфейсы, дисковое хранилище), на котором загружаются операционная система и приложения. Операционная система использует набор аппаратных средств и не знает о совместном использовании с другими гостевыми операционными системами, работающими на одной и той же физической аппаратной платформе. **Гипервизор** (основной программный компонент технологии виртуализации) управляет взаимодействием между вызовами гостевой операционной системы, виртуальным оборудованием и фактическим выполнением, выполняемым на базовом физическом оборудовании.

Технологии виртуализации появились еще в 1960-х годах прошлого века. General Electric, Bell Labs, IBM пытались решить задачи расширения размеров оперативной памяти, возможности исполнения нескольких программ и др. Результатом стали такие разработки как суперкомпьютер Atlas, IBM 7044 (M44), CP/CMS, Livermore Time-Sharing System, Cray Time-Sharing System и др. [35]

Новейшая история виртуализации делится условно на два периода. Первый начался примерно в 1998 году и продолжался несколько лет. В эти годы происходило активное распространение идей виртуализации параллельно с

разработкой практических технологий. Когда технологии достигли “критической массы”, примерно в 2004 году, начался второй период – активное внедрение.

Сегодня виртуализация - это ключевая технология облачных вычислений и управления современным центром обработки данных. *Облачные вычисления* помогают решать проблемы с масштабируемостью, безопасностью и управлением глобальной ИТ-инфраструктурой, в то же время эффективно снижая затраты [36].

В отличие от классической виртуализации, контейнеризация в Linux – это технология виртуализации на уровне операционной системы, которая предлагает “легкую” виртуализацию. **Контейнеры** часто называют легковесными виртуальными машинами, однако, контейнер не является виртуальной машиной. Группа процессов, которая работает как контейнер, имеет свою собственную корневую файловую систему, но разделяет ядро с операционной системой хоста.

Система LXC (Linux Containers) - была добавлена в ядро Linux в 2008 году. LXC использует комбинацию таких функций ядра, как cgroups (позволяет изолировать и отслеживать использование ресурсов) и пространства имен (позволяют разделять группы так, чтобы они не могли “видеть” друг друга) [37].

Docker, появившийся несколько позже, позиционировался, как инструмент для упрощения работы по созданию и управлению контейнерами. Изначально Docker использовал LXC, затем была разработана библиотека под названием libcontainer. Docker сделал контейнеры доступными для обычного разработчика и системного администратора путем упрощения процесса и стандартизации интерфейса. Контейнеры Docker упаковывают программное обеспечение в полную файловую систему, в которой содержится все необходимое для запуска: код, среда выполнения, системные инструменты и системные библиотеки [38].

4. Различные архитектуры современных планировщиков.

Управление кластером и распределение задач являются ключевым компонентом современного дата-центра. Статья разработчиков Google [2] и книга [39], описывают архитектуру и задачи, решаемые современным планировщиком.

Планировщик Borg обеспечивает эффективное использование ресурсов за счет управления доступом, упаковки задач, наложению (over-commitment) и использованию контейнеров.[3]

[3]В оригинале “изоляция на уровне процесса”. Borg начал разрабатываться до активного распространения контейнеризации и виртуализации.

Он поддерживает приложения с высокой степенью доступности (high-availability applications) и использует эвристики распределения, уменьшающие время восстановления приложения и снижающие вероятность одновременного отказа копий приложения.

Borg предоставляет пользователям декларативный язык спецификации заданий, интеграцию сервисов имен, мониторинг работы в режиме реального времени и инструменты для анализа и моделирования поведения системы.

Borg – пример планировщика с *монолитной архитектурой*. Единственный процесс планировщика работает на одной машине и назначает задачи на остальные машины в кластере. Весь поток задач обрабатывается одним планировщиком (также планировщики с данной архитектурой можно называть **централизованными**). Это просто, универсально и позволяет применять сложные стратегии распределения.

Например, планировщики Paragon [40], Quasar [41], Mage [42] используют технологии машинного обучения для исключения негативного взаимного влияния потоков задач друг на друга.

Если решение о размещении очередной задачи требует сложных вычислений, то возникает задержка. Чем больше размер кластера и интенсивнее поток задач, тем больше задержка. Это накладывает ограничения на возможности использования таких планировщиков. Частичное решение проблемы было предложено авторами системы Firmament [43].

Одной из стратегий оптимизации, доступной централизованным планировщикам является консолидация задач без ухудшения качества обслуживания [44-45]. Это очень эффективная стратегия. В определенных случаях оптимизация от контролируемой консолидации может достигать сотен процентов [46].

Большинство кластеров выполняют задачи разного типа, требующих различных стратегий распределения. Уместить и поддерживать реализацию для каждого типа задач в одном планировщике, который обрабатывает смешанные (гетерогенные) рабочие нагрузки, может быть сложной задачей.

Авторы планировщика Mesos [47] поняли, что динамическое разделение кластера между различными типами потоков задач (Hadoop, Spark, TensorFlow) является ключевым фактором обеспечения эффективного использования вычислительных реурсов.

Планировщики Mesos и Yarn [48] имеют **двухуровневневую архитектуру**. На первом уровне ресурсы разделяются между планировщиками среды выполнения приложений, на втором – данные планировщики распределяют задачи.

Такая схема дает возможность использовать более эффективные стратегии управления вычислительными ресурсами в зависимости от специфики запускаемых приложений.

Однако, разделение процесса принятия решения о размещении задачи в двухуровневой архитектуре имеет также некоторые недостатки, так как планировщик лишается преимущества “всевидения”. В качестве последствий можно указать следующее.

- **Прерывание задач** (preemption) становится труднореализуемым. Ресурсы, занимаемые задачами, не видны планировщику первого уровня и он не может принять решение о замене задачи. Для реализации этого механизма необходимо передавать информацию или правила политики прерывания в планировщик среды выполнения приложений.
- **Интерференция** различных заданий (workloads), приводящая к потере производительности, так как планировщик может не знать “соседей” по серверу.
- **Сложность протокола** взаимодействия планировщиков двух уровней. При попытке решить вышеперечисленные проблемы возникает увеличение объема информации для обмена между планировщиками, что приводит к усложнению протокола их взаимодействия.

Планировщики **с разделяемым состоянием** призваны решить эти проблемы с помощью полу-распределенной модели. Состояние кластера независимо меняется планировщиками второго уровня. Если происходит конфликт, то изменение отменяется. Примерами являются планировщики Omega [49], Apollo [9], Nomad [50].

В системах с **распределенным состоянием** планировщики полностью независимы и не взаимодействуют между собой. Каждый планировщик работает со своей локальной копией состояния кластера. Задачи могут поступать на вход любому планировщику, который может размещать их на любых узлах кластера.

Таким образом, разделение ресурсов кластера между задачами является результатом случайности в решениях независимых планировщиков без какого-либо централизованного управления.

Данный принцип вероятностной балансировки нагрузки был описан в работе [51], а наиболее известным планировщиком с распределенным состоянием является Sparrow [52].

Планировщики с распределенным состоянием используют простую логику принятия решения о размещении задачи и, тем самым, могут обрабатывать поток заданий с большой скоростью.

Недостатком планировщиков с распределенным состоянием можно назвать невозможность реализовать строгий приоритет задач и справедливое распределение (fairness policies) [53]. Также, трудно исключить возможное взаимное влияние потоков задач друг на друга (интерференция).

Планировщики с **гибридной архитектурой** призваны решить проблемы и тех, и других разновидностей планировщиков за счет совмещения монолитной и распределенной архитектур. Мелкие задачи распределяет часть планировщика

с распределенной архитектурой, остальные - часть с монолитной. Гибридные планировщики описаны в работах Tarcil [54], Mercury [55], Hawk [56].

5. Размещение задач на примере планировщика Borg

Задание (возможно состоящее из нескольких задач) отправляется планировщику. Задание сохраняется в базе данных (распределенное хранилище), и составляющие задание задачи помещаются в очередь.

Очередь периодически сканируется планировщиком, который должен разместить задачу на подходящей машине. Сканирование очереди происходит в цикле по убыванию приоритета задач.

Алгоритм размещения состоит из двух этапов:

- поиск подходящих машин (на которых задача может быть выполнена),
- подсчет очков (scoring – процесс выбора машины из набора подходящих).

При поиске подходящих машин планировщик выбирает машины, которые удовлетворяют ограничениям задачи и имеют достаточно ресурсов (учитываются также ресурсы, занимаемые низкоприоритетными задачами, которые могут быть освобождены при необходимости).

При подсчете очков планировщик определяет значение функции $score(i,j)$ задачи j на машине i для всех машин. Выбирается машина с наименьшим значением.

Для вычисления значения $score$ планировщик учитывает требования задаваемые пользователем, однако, в большей степени, использует внутренние критерии оценки:

- минимизация количества и приоритета прерываемых (preempted) задач;
- выбор машин, на которых уже имеются необходимые для выполнения задачи данные;
- распределение/распыление (spreading) задач между энергетическими доменами и доменами безопасности; имеется ввиду попытка снижения пиковых энергетических нагрузок и снижения вероятности завершиния задачи из-за отказа оборудования;
- упаковка высокоприоритетных задач вместе с низкоприоритетными на одной машине, в случае всплеска нагрузки задачи с низким приоритетом могут быть прерваны, и освободившиеся ресурсы могут быть отданы оставшимся задачам с более высоким приоритетом.

Если выбранная машина не имеет достаточно ресурсов для запуска задачи, то планировщик начинает завершать (прерывать) задачи начиная с наименее приоритетных. Задачи отправляются обратно в очередь. Миграция и спящий режим не используются.

Наиболее важным критерием авторы указывают время запуска задачи - время от отправки задачи в очередь планировщика до запуска. Значения сильно разбросаны, медиана составляет 25 секунд. Из этого времени 80% занимает

установка необходимых пакетов ОС для работы задачи. Наиболее узким местом является запись на жесткий диск. Поэтому, для ускорения запуска задач планировщик старается размещать задачи на машины где уже присутсвуют необходимые пакеты.

6. Другие задачи планировщика.

Помимо эффективного распределения, любой планировщик должен уметь решать три основные задачи.

Подготовка (packaging). Программы, запускаемые планировщиком Borg, являются статически собранными бинарными файлами. Таким образом исключается проблема недостающих зависимостей в процессе выполнения. По сравнению с этим подходом, решение, предлагаемое Docker, гораздо универсальнее. Образы Docker позволяют собрать все необходимое для запуска программы в одном месте и использовать полученный образ на любом Docker сервере. В настоящее время стандартизацией формата образа контейнера занимается консорциум OCI [57].

Внедрение (deployment). Проблема, до сих пор не имеющая хорошего решения. Различные огранизации предлагают методы безопасного и надежного внедрения кода. В крупных компаниях существуют отдельные команды, занимающиеся разработкой инструментов автоматизации сборки и запуска новых версий ПО. Однако основные принципы внедрения ПО одинаковы. Необходимо остановить старую версию сервиса (программы) и запустить новую. При этом, желательно не допускать остановки обслуживания (обрыва соединений). В общем случае это делается путем направления всех поступающих запросов на новую версию сервиса, ожидая завершения обработки запросов старой версией.

Управления жизненным циклом (life-cycle). Планировщик должен контролировать выполнение сервиса, не допуская перебоев в обслуживании. Если программа завершается из-за внутренней ошибки, то планировщик может это определить и перезапустить процесс. В других случаях сервис может перестать отвечать на запросы из-за проблем с сетевым соединением. При отказе сервера необходимо перезапустить сервисы на других доступных серверах.

7. Заключение

Оптимизация управления является важнейшей задачей для современного вычислительного кластера и актуальна как для больших, так и для небольших систем. Наиболее распространена монолитная архитектура планировщика, которая позволяет реализовать сложные стратегии оптимизации и достичь хороших результатов. С другой стороны, при усложнении логики принятия решения о размещении задачи планировщик перестает масштабироваться. Решить проблему масштабирования призваны планировщики с

распределенной архитектурой, которые, в свою очередь, обладают некоторыми ограничениями.

Универсального решения в этом вопросе нет, и на практике зачастую приходится вручную настраивать различные параметры планировщика.

В ИСП РАН разработана система Fanlight – программная платформа для организации единой Web-среды для исследований, разработок и образования. Система может быть быстро развернута на имеющихся вычислительных ресурсах организации с последующей интеграцией в нее приложений, требующих поддержки аппаратного ускорения 3D графики. Вся последующая работа пользователя проводится через стандартный Web-браузер [58-59].

Мы разработали и интегрировали в систему Fanlight различные стратегии размещения контейнеров, позволяющие увеличить эффективность использования как CPU так и GPU устройств [60-61].

Список литературы

[1] Sinnen O. Task scheduling for parallel systems. John Wiley & Sons, 2007, 30 p.
[2] Verma A., Pedrosa L., Korupolu M. et al. Large-scale cluster management at google with borg. In Proc. of the Tenth European conference on computer systems, 2015, pp. 18:1–18:17.
[3] Ehrgott M. Multicriteria optimization. Springer, 2005, 320 p.
[4] Аветисян А., Грушин Д., Рыжов А. Системы управления кластерами. Труды ИСП РАН, том 3, 2002, стр. 39–62.
[5] Message P Forum. MPI: A message-passing interface standard. Technical Report. University of Tennessee, Knoxville, 1994, 228 p.
[6] Kaplan Joseph A., Nelson Michael L. A comparison of queueing, cluster and distributed computing systems. NASA Langley Technical Report, 1994, 49 p.
[7] Baker M., Fox G., Yau H. A review of commercial and research cluster management software. Northeast Parallel Architectures Center, 1996, 63 p.
[8] Foster I., Kesselman C. The grid: Blueprint for a new computing infrastructure. Morgan Kaufmann Publishers Inc., 1999, 675 p.
[9] Boutin E., Ekanayake J., Lin W. et al. Apollo: Scalable and coordinated scheduling for cloud-scale computing. In Proc. of the 11th USENIX conference on Operating Systems Design and Implementation, 2014. pp. 285–300.
[10] Delgado P., Dinu F., Kermarrec A.-M. et al. Hawk: Hybrid datacenter scheduling. In Proc. of the 2015 USENIX annual technical conference, 2015.
[11] Schwarzkopf M. Cluster scheduling for data centers. Queue, vol. 15, no. 5, 2017.
[12] Herbein S., Dusia A., Landwehr A. et al. Resource management for running hpc applications in container clouds. Lecture Notes in Computer Science, vol. 9697, 2016. pp. 261–278.
[13] Ye D., Han X., Zhang G. Online multiple-strip packing. Theoretical Computer Science, vol. 412, no. 3, 2011, pp. 233–239.
[14] Hurink J. L., Paulus J. J. Online algorithm for parallel job scheduling and strip packing. Lecture Notes in Computer Science, vol. 4927, 2007. pp. 67–74.
[15] Zhuk S. On-line algorithms for packing rectangles into several strips. Discrete Mathematics and Applications, vol. 17, no. 5, 2007, pp. 517–531.

[16] Johnson D. S., Demers A., Ullman J. D. et al. Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM Journal on computing, vol. 3, no. 4, 1974, pp. 299–325.

[17] Garey M. R., Graham R. L., Johnson D. S. et al. Resource constrained scheduling as generalized bin packing. Journal of Combinatorial Theory, Series A, vol. 21, no. 3, 1976, pp. 257–298.

[18] Zhuk S., Chernykh A., Avetisyan A. et al. Comparison of scheduling heuristics for grid resource broker. In Proc. of the Fifth Mexican International Conference, 2004, pp. 388–392.

[19] Tchernykh A., Ramírez-Alcaraz J. M., Avetisyan A. et al. Two level job-scheduling strategies for a computational grid. Lecture Notes in Computer Science, vol. 3911, 2005. pp. 774–781.

[20] Tchernykh A., Schwiegelshohn U., Yahyapour R. et al. On-line hierarchical job scheduling on grids with admissible allocation. Journal of Scheduling, vol. 13, no. 5, 2010, pp. 545–52.

[21] Tchernykh A., Schwiegelshohn U., Yahyapour R. et al. Online hierarchical job scheduling on grids. In From grids to service and pervasive computing, Springer, 2008, pp. 77–91.

[22] Аветисян А.И., Гайсарян С.С., Грушин Д.А., Кузюрин Н.Н., Шокуров А.В. Эвристики распределения задач для брокера ресурсов Grid. Труды ИСП РАН, том 5, 2004, стр. 41–62.

[23] Baraglia R., Capannini G., Pasquali M. et al. Backfilling strategies for scheduling streams of jobs on computational farms. In Making Grids Work, Springer, 2008, pp. 103–115.

[24] Mu'alem A.W., Feitelson D.G. Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 6, 2001, pp. 529–543.

[25] Nissimov A., Feitelson D.G. Probabilistic backfilling. Lecture Notes in Computer Science, vol. 4942, 2008. pp. 102–115.

[26] Иванников В.П., Грушин Д.А., Кузюрин Н.Н. и др. Программная система увеличения энергоэффективности вычислительного кластера. Программирование, том 36, no. 6, 2010, pp. 28–40.

[27] Baraglia R., Capannini G., Dazzi P. et al. A multi-criteria job scheduling framework for large computing farms. Journal of Computer and System Sciences, vol. 79, no. 2, 2013, pp. 230–244.

[28] Csirik J., Van Vliet A. An on-line algorithm for multidimensional bin packing. Operations Research Letters, vol. 13, no. 3, 1993, pp. 149–158.

[29] Кузюрин Н.Н., Поспелов А.И. Вероятностный анализ нового класса алгоритмов упаковки прямоугольников в полосу. Журнал вычислительной математики и математической физики, том 51, No. 10, 2011, стр. 1931–1936.

[30] Трушников М.А. Вероятностный анализ нового алгоритма упаковки прямоугольников в полосу. Труды ИСП РАН, том 24, 2013, стр. 457-468. DOI: 10.15514/ISPRAS-2013-24-21.

[31] Лазарев Д.О., Кузюрин Н.Н. Алгоритм упаковки прямоугольников в несколько полос и анализ его точности в среднем. Труды ИСП РАН, том 29, вып. 6, 2017, стр. 221–228. DOI: 10.15514/ISPRAS-2017-29(6)-13.

[32] Жук С.Н. О построении расписаний выполнения параллельных задач на группах кластеров с различной производительностью. Труды ИСП РАН, том 23, 2012, стр. 447-454. DOI: 10.15514/ISPRAS-2012-23-27.
[33] Tsai C.-W., Rodrigues J. J. Metaheuristic scheduling for cloud: A survey. IEEE Systems Journal, vol. 8, no. 1, 2014, pp. 279–291.
[34] Грушин Д.А., Кузюрин Н.Н. Энергоэффективные вычисления для группы кластеров. Труды ИСП РАН, том 23, 2012, стр. 433–46. DOI: 10.15514/ISPRAS-2012-23-26.
[35] Goldberg R.P. Survey of virtual machine research. Computer, vol. 7, no. 9, 1974, pp. 34–45.
[36] Magic Quadrant for Cloud Infrastructure as a Service, Worldwide. Available at: https://www.gartner.com/doc/3875999/magic-quadrant-cloud-infrastructure-service. Accessed 01.12.2018.
[37] Helsley M. LXC: Linux container tools. IBM devloperWorks Technical Library, 2009.
[38] Merkel D. Docker: Lightweight linux containers for consistent development and deployment. Linux Journal, no. 239, 2014.
[39] Barroso L. A., Clidaras J., Hoelzle U. The datacenter as a computer: An introduction to the design of warehouse-scale machines. Morgan & Claypool, 2013, 154 p.
[40] Delimitrou C., Kozyrakis C. Paragon: QoS-aware scheduling for heterogeneous datacenters. ACM SIGPLAN Notices, vol. 48, no. 4, 2013, pp. 77–88.
[41] Delimitrou C., Kozyrakis C. Quasar: Resource-efficient and qos-aware cluster management, ACM SIGPLAN Notices, vol. 49, no. 4, 2014, pp. 127–144.
[42] Romero F., Delimitrou C. Mage: Online and interference-aware scheduling for multi-scale heterogeneous systems. In Proc. of the 27th international conference on parallel architectures and compilation techniques, 2018, pp. 19:1–19:13.
[43] Gog I., Schwarzkopf M., Gleave A. et al. Firmament: Fast, centralized cluster scheduling at scale. In Proc. of the 12th usenix conference on operating systems design and implementation, 2016, pp. 99–115.
[44] Breitgand D., Epstein A. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds. In Proc. of the IEEE INFOCOM, 2012. pp. 2861–2865.
[45] Wang M., Meng X., Zhang L. Consolidating virtual machines with dynamic bandwidth demand in data centers. In Proc. of the IEEE INFOCOM, 2011. pp. 71–75.
[46] Urgaonkar B., Shenoy P., Roscoe T. Resource overbooking and application profiling in shared hosting platforms. In Proc. of the 5th symposium on operating systems design and implementation, 2002. pp. 239–254.
[47] Hindman B., Konwinski A., Zaharia M. et al. Mesos: A platform for fine-grained resource sharing in the data center. In Proc. of the 8th USENIX conference on Networked systems design and implementation, 2011. pp. 295-308.
[48] Vavilapalli V. K., Murthy A. C., Douglas C. et al. Apache hadoop yarn: Yet another resource negotiator. In Proc. of the 4th annual symposium on cloud computing, 2013, pp. 5:1–5:16.
[49] Schwarzkopf M., Konwinski A., Abd-El-Malek M. et al. Omega: Flexible, scalable schedulers for large compute clusters. In Proc. of the 8th ACM European conference on computer systems, 2013. pp. 351–364.
[50] Scheduling in Nomad. Available at: https://www.nomadproject.io/docs/internals/scheduling.html. Accessed 01.12.2018.

[51] Mitzenmacher M. The power of two choices in randomized load balancing. IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 10, 2001, pp. 1094–1104.
[52] Ousterhout K., Wendell P., Zaharia M. et al. Sparrow: Distributed, low latency scheduling. In Proc. of the twenty-fourth ACM Symposium on operating systems principles, 2013, pp. 69–84.
[53] Fagin R., Williams J. H. A fair carpool scheduling algorithm. IBM Journal of Research and development, vol. 27, no. 2, 1983, pp. 133–139.
[54] Delimitrou C., Sanchez D., Kozyrakis C. Tarcil: Reconciling scheduling speed and quality in large shared clusters. In Proc. of the sixth ACM Symposium on cloud computing, 2015, pp. 97–110.
[55] Karanasos K., Rao S., Curino C. et al. Mercury: Hybrid centralized and distributed scheduling in large shared clusters. In Proc. of the USENIX annual technical, 2015. pp. 485–497.
[56] Delgado P., Dinu F., Kermarrec A.-M. et al. Hawk: Hybrid datacenter scheduling In Proc. of the USENIX annual technical conference, 2015. pp. 499–510.
[57] The Open Container Initiative. Available at: https://www.opencontainers.org/, accessed 01.12.2018.
[58] Аветисян А.А., Гайсарян С.С., Самоваров О.И. и др. Организация предметно-ориентированных научно-исследовательский центров в рамках программы университетский кластер. Труды конференции «Научный сервис в сети интернет: Суперкомпьютерные центры и задачи», 2010, стр. 213–5.
[59] Самоваров О.И., Гайсарян С.С. Архитектура и особенности реализации платформы unihub в модели облачных вычислений на базе открытого пакета openstack. Труды ИСП РАН, том 26, вып. 1, 2014, стр. 403-420. DOI: 10.15514/ISPRAS-2014-26(1)-17.
[60] Грушин Д.А., Кузюрин Н.Н. Балансировка нагрузки в системе Unihub на основе предсказания поведения пользователей. Труды ИСП РАН, том 27, вып. 5, 2015, стр. 23-34. DOI: 10.15514/ISPRAS-2015-27(5)-2.
[61] Грушин Д.А., Кузюрин Н.Н. Задачи оптимизации размещения контейнеров MPI-приложений на вычислительных кластерах. Труды ИСП РАН, том 29, вып. 6, 2017, стр. 229–244. DOI: 10.15514/ISPRAS-2017-29(6)-14.

On an effective scheduling problem in computation clusters

[1]*D.A. Grushin <grushin@ispras.ru>*
[1,2]*N.N. Kuzyurin <nnkuz@ispras.ru>*

[1]*Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2]*Moscow Institute of Physics and Technology, Dolgoprudnyj, Institutskij alley, Moscow region, 141700, Russia*

Abstract. At present, big companies such as Amazon, Google, Facebook, Microsoft, Yahoo! own huge datacenters with thousands of nodes. These clusters are used simultaneously by many users. The users submit jobs containing one or more tasks. Task flow is usually a mix of short, long, interactive, batch, and tasks with different priorities. Cluster scheduler decides on which server to run the task, where the task is then run as a process, container or a virtual

machine. Scheduler optimizations are important as they provide higher server utilization, lower latency, improved load balancing, and fault tolerance. Achieving good task placement is hard. The problem has multiple dimensions and requires algorithmically complex optimizations. This increases placement latency and limits cluster scalability. In this paper we consider different cluster scheduler architectures and optimization problems.

Keywords: optimization; scheduling; virtualization; cloud computing

DOI: 10.15514/ISPRAS-2018-30(6)-7

For citation: Grushin D.A., Kuzyurin N.N. On an effective scheduling problem in computation clusters. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 123-142 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-7

References

[1] Sinnen O. Task scheduling for parallel systems. John Wiley & Sons, 2007, 30 p.
[2] Verma A., Pedrosa L., Korupolu M. et al. Large-scale cluster management at google with borg. In Proc. of the Tenth European conference on computer systems, 2015, pp. 18:1–18:17.
[3] Ehrgott M. Multicriteria optimization. Springer, 2005, 320 p.
[4] Аветисян А., Грушин Д., Рыжов А. Системы управления кластерами. Труды ИСП РАН, том 3, 2002, стр. 39–62.
[5] Message P Forum. MPI: A message-passing interface standard. Technical Report. University of Tennessee, Knoxville, 1994, 228 p.
[6] Kaplan Joseph A., Nelson Michael L. A comparison of queueing, cluster and distributed computing systems. NASA Langley Technical Report, 1994, 49 p.
[7] Baker M., Fox G., Yau H. A review of commercial and research cluster management software. Northeast Parallel Architectures Center, 1996, 63 p.
[8] Foster I., Kesselman C. The grid: Blueprint for a new computing infrastructure. Morgan Kaufmann Publishers Inc., 1999, 675 p.
[9] Boutin E., Ekanayake J., Lin W. et al. Apollo: Scalable and coordinated scheduling for cloud-scale computing. In Proc. of the 11th USENIX conference on Operating Systems Design and Implementation, 2014. pp. 285–300.
[10] Delgado P., Dinu F., Kermarrec A.-M. et al. Hawk: Hybrid datacenter scheduling. In Proc. of the 2015 USENIX annual technical conference, 2015.
[11] Schwarzkopf M. Cluster scheduling for data centers. Queue, vol. 15, no. 5, 2017.
[12] Herbein S., Dusia A., Landwehr A. et al. Resource management for running hpc applications in container clouds. Lecture Notes in Computer Science, vol. 9697, 2016. pp. 261–278.
[13] Ye D., Han X., Zhang G. Online multiple-strip packing. Theoretical Computer Science, vol. 412, no. 3, 2011, pp. 233–239.
[14] Hurink J. L., Paulus J. J. Online algorithm for parallel job scheduling and strip packing. Lecture Notes in Computer Science, vol. 4927, 2007. pp. 67–74.
[15] Zhuk S. On-line algorithms for packing rectangles into several strips. Discrete Mathematics and Applications, vol. 17, no. 5, 2007, pp. 517–531.
[16] Johnson D. S., Demers A., Ullman J. D. et al. Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM Journal on computing, vol. 3, no. 4, 1974, pp. 299–325.

[17] Garey M. R., Graham R. L., Johnson D. S. et al. Resource constrained scheduling as generalized bin packing. Journal of Combinatorial Theory, Series A, vol. 21, no. 3, 1976, pp. 257–298.

[18] Zhuk S., Chernykh A., Avetisyan A. et al. Comparison of scheduling heuristics for grid resource broker. In Proc. of the Fifth Mexican International Conference, 2004, pp. 388–392.

[19] Tchernykh A., Ramírez-Alcaraz J. M., Avetisyan A. et al. Two level job-scheduling strategies for a computational grid. Lecture Notes in Computer Science, vol. 3911, 2005. pp. 774–781.

[20] Tchernykh A., Schwiegelshohn U., Yahyapour R. et al. On-line hierarchical job scheduling on grids with admissible allocation. Journal of Scheduling, vol. 13, no. 5, 2010, pp. 545–52.

[21] Tchernykh A., Schwiegelshohn U., Yahyapour R. et al. Online hierarchical job scheduling on grids. In From grids to service and pervasive computing, Springer, 2008, pp. 77–91.

[22] Avetisyan A.I., Gaissaryan S.S., Grushin D.A. et al. Scheduling heuristics for grid resource broker. Trudy ISP RAN/Proc. ISP RAS, vol. 5, 2004, pp. 41–62 (in Russian).

[23] Baraglia R., Capannini G., Pasquali M. et al. Backfilling strategies for scheduling streams of jobs on computational farms. In Making Grids Work, Springer, 2008, pp. 103–115.

[24] Mu'alem A.W., Feitelson D.G. Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 6, 2001, pp. 529–543.

[25] Nissimov A., Feitelson D.G. Probabilistic backfilling. Lecture Notes in Computer Science, vol. 4942, 2008. pp. 102–115.

[26] Ivannikov V.P., Grushin D.A., Kuzyurin N.N., Pospelov A.I., Shokurov A.V. Software for improving the energy efficiency of a computer cluster. Programming and Computer Software, vol. 36, no. 6, 2010, pp. 327-336.

[27] Baraglia R., Capannini G., Dazzi P. et al. A multi-criteria job scheduling framework for large computing farms. Journal of Computer and System Sciences, vol. 79, no. 2, 2013, pp. 230–244.

[28] Csirik J., Van Vliet A. An on-line algorithm for multidimensional bin packing. Operations Research Letters, vol. 13, no. 3, 1993, pp. 149–158.

[29] Kuzjurin N.N., Pospelov A.I. Probabilistic analysis of a new class of rectangle strip packing algorithms. Computational Mathematics and Mathematical Physics, vol. 51, no. 10, 2011, pp. 1931–1936.

[30] Trushnikov M.A. Probabilistic analysis of a new rectangle strip packing algorithm. Trudy ISP RAN/Proc. ISP RAS, vol. 24, 2013, pp. 457-468 (in Russian). DOI: 10.15514/ISPRAS-2013-24-21.

[31] Lazarev D.O., Kuzjurin N.N. Rectangle packing algorithm into several strips and average case analisys. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 6, 2017, pp. 221–228 (in Russian). DOI: 10.15514/ISPRAS-2017-29(6)-13.

[32] Zhuk S. On parallel task scheduling on a group of clusters with different speeds. Trudy ISP RAN/Proc. ISP RAS, vol. 23, 2012, pp. 447-454. DOI: 10.15514/ISPRAS-2012-23-27.

[33] Tsai C.-W., Rodrigues J. J. Metaheuristic scheduling for cloud: A survey. IEEE Systems Journal, vol. 8, no. 1, 2014, pp. 279–291.

[34] Grushin D.A., Kuzjurin N.N. Energy effective computations on a group of clusters. Trudy ISP RAN/Proc. ISP RAS, vol. 23, 2012, pp. 433–46 (in Russian). DOI: 10.15514/ISPRAS-2012-23-26.
[35] Goldberg R.P. Distributed, low latency scheduling. In Proc. of the twenty-fourth ACM Symposium on operating systems principles, 2013, pp. 69–84.
[36] Magic Quadrant for Cloud Infrastructure as a Service, Worldwide. Available at: https://www.gartner.com/doc/3875999/magic-quadrant-cloud-infrastructure-service. Accessed 01.12.2018.
[37] Helsley M. LXC: Linux container tools. IBM devloperWorks Technical Library, 2009.
[38] Merkel D. Docker: Lightweight linux containers for consistent development and deployment. Linux Journal, no. 239, 2014.
[39] Barroso L. A., Clidaras J., Hoelzle U. The datacenter as a computer: An introduction to the design of warehouse-scale machines. Morgan & Claypool, 2013, 154 p.
[40] Delimitrou C., Kozyrakis C. Paragon: QoS-aware scheduling for heterogeneous datacenters. ACM SIGPLAN Notices, vol. 48, no. 4, 2013, pp. 77–88.
[41] Delimitrou C., Kozyrakis C. Quasar: Resource-efficient and qos-aware cluster management, ACM SIGPLAN Notices, vol. 49, no. 4, 2014, pp. 127–144.
[42] Romero F., Delimitrou C. Mage: Online and interference-aware scheduling for multi-scale heterogeneous systems. In Proc. of the 27th international conference on parallel architectures and compilation techniques, 2018, pp. 19:1–19:13.
[43] Gog I., Schwarzkopf M., Gleave A. et al. Firmament: Fast, centralized cluster scheduling at scale. In Proc. of the 12th usenix conference on operating systems design and implementation, 2016, pp. 99–115.
[44] Breitgand D., Epstein A. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds. In Proc. of the IEEE INFOCOM, 2012. pp. 2861–2865.
[45] Wang M., Meng X., Zhang L. Consolidating virtual machines with dynamic bandwidth demand in data centers. In Proc. of the IEEE INFOCOM, 2011. pp. 71–75.
[46] Urgaonkar B., Shenoy P., Roscoe T. Resource overbooking and application profiling in shared hosting platforms. In Proc. of the 5th symposium on operating systems design and implementation, 2002. pp. 239–254.
[47] Hindman B., Konwinski A., Zaharia M. et al. Mesos: A platform for fine-grained resource sharing in the data center. In Proc. of the 8th USENIX conference on Networked systems design and implementation, 2011. pp. 295-308.
[48] Vavilapalli V. K., Murthy A. C., Douglas C. et al. Apache hadoop yarn: Yet another resource negotiator. In Proc. of the 4th annual symposium on cloud computing, 2013, pp. 5:1–5:16.
[49] Schwarzkopf M., Konwinski A., Abd-El-Malek M. et al. Omega: Flexible, scalable schedulers for large compute clusters. In Proc. of the 8th ACM European conference on computer systems, 2013. pp. 351–364.
[50] Scheduling in Nomad. Available at: https://www.nomadproject.io/docs/internals/scheduling.html. Accessed 01.12.2018.
[51] Mitzenmacher M. The power of two choices in randomized load balancing. IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 10, 2001, pp. 1094–1104.
[52] Ousterhout K., Wendell P., Zaharia M. et al. Sparrow: Distributed, low latency scheduling. In Proc. of the twenty-fourth ACM Symposium on operating systems principles, 2013, pp. 69–84.

[53] Fagin R., Williams J. H. A fair carpool scheduling algorithm. IBM Journal of Research and development, vol. 27, no. 2, 1983, pp. 133–139.

[54] Delimitrou C., Sanchez D., Kozyrakis C. Tarcil: Reconciling scheduling speed and quality in large shared clusters. In Proc. of the sixth ACM Symposium on cloud computing, 2015, pp. 97–110.

[55] Karanasos K., Rao S., Curino C. et al. Mercury: Hybrid centralized and distributed scheduling in large shared clusters. In Proc. of the USENIX annual technical, 2015. pp. 485–497.

[56] Delgado P., Dinu F., Kermarrec A.-M. et al. Hawk: Hybrid datacenter scheduling In Proc. of the USENIX annual technical conference, 2015. pp. 499–510.

[57] The Open Container Initiative. Available at: https://www.opencontainers.org/, accessed 01.12.2018.

[58] Avetisyan A.A., Gaissaryan S.S., Samovarov O.I. et al. Organization of scientific centers in univercity cluster program. In Proc. of the All-Russian Conference on Scientific service in Internet: Supercomputer centers and problems, 2010, pp. 213–215 (in Russian).

[59] Samovarov O.I., Gaissaryan S.S. Architecture and implementation details of Unihub platform in cloud computing architecture based on Openstack package. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 1, 2014, pp. 403-420 (in Russian). DOI: 10.15514/ISPRAS-2014-26(1)-17.

[60] Grushin D.A., Kuzyurin N.N. Load balancing in unihub saas system based on user behavior prediction. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 5, 2015, pp. 23–34 (in Russian). DOI: 10.15514/ISPRAS-2015-27(5)-2.

[61] Grushin D.A., Kuzyurin N.N. Optimization problems running mpi-based hpc applications. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 6, 2017, pp. 229–244 (in Russian). DOI: 10.15514/ISPRAS-2017-29(6)-14.

Static verification for memory safety of Linux kernel drivers*

A.A. Vasilyev <vasilyev@ispras.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. Memory errors in Linux kernel drivers are a kind of serious bugs that can lead to dangerous consequences but such errors are hard to detect. This article describes static verification that aims at finding all errors under certain assumptions. Static verification of industrial projects such as the Linux kernel requires additional effort. Limitations of current tools for static verification disallow to analyze the Linux kernel as a whole, so we use a simplified automatically generated environment model. This model introduces inaccuracy, but provides ability for verification. In addition, we allow absent definitions for some functions which results in incomplete ANSI C programs. The current work proposes an approach to reveal issues with memory usage in such incomplete programs. Our static verification technique is based on Symbolic Memory Graphs (SMG) with extensions aiming to reduce a false alarm rate. We introduced an on-demand memory conception for simplification of kernel API models and implemented this conception in static verification tool CPAchecker. Also, we changed precision of a CPAchecker memory model from bytes to bits and supported structure alignment similar to the GCC compiler. We implemented the predicate extension for SMG to improve accuracy of the analysis. We verified of Linux kernel 4.11.6 and 4.16.10 with help of the Klever verification framework with CPAchecker as a verification engine. Manual analysis of warnings produced by Klever revealed 78 real bugs in drivers. We have made patches to fix 33 of them.

Keywords: shape analysis; static verification; symbolic memory graphs; memory model.

DOI: 10.15514/ISPRAS-2018-30(6)-8

For citation: Vasilyev A.A. Static verification for memory safety of Linux kernel drivers. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018. pp. 143-160. DOI: 10.15514/ISPRAS-2018-30(6)-8

1. Introduction

Operating system kernels are often written in the C programming language. This language is portable and effective, but unfortunately it is not memory safe. Memory issues can lead to vulnerabilities or unpredictable failures. Common methods such as testing are unable to find all problems. A probable solution to get an evidence of

* The research was supported by RFBR grant 18-01-00426

satisfiability of safety properties is formal methods and there are results of comprehensive formal verification of the seL4 microkernel [1]. However formal methods generally require a whole program and a complete model of its environment to produce an appropriate verdict. For example, Microsoft developed Static Driver Verifier (SDV) [2] to improve Microsoft Windows stability. SDV contains models of the kernel and drivers' environment, and over 60 API usage rules.

The Linux kernel is important open source software. There are many research and industrial projects for improving kernel quality by verification, testing, bug hunting, fuzzing and error reports. Coverity [3], Saturn [4], DDVerify [5], Coccinelle [6], Linux Driver Verification [7] are projects which work on improving Linux stability.

This article considers operating system kernel drivers with automatically generated environment models as a target for approbation of a memory verification technology. Main contributions of the paper are connected with extensions of an existed static memory verification approach to be able to perform Linux kernel drivers verification, which are described in Section 4.

2. Linux driver verification

The Linux kernel represents an industrial code base with more than 10 million lines of drivers' code. A distinctive feature of Linux is instability of internal interfaces. A high speed of changes with a distributed development process requires an efficient bug finding strategy.

The research of faults in Linux operating system drivers divides errors into typical and specific [8]. Specific faults in drivers are described as connected with hardware and not applicable to other drivers. Typical faults can be specified by some rule which is true for all or some group of drivers. Typical faults are further divided into:

- Linux specific faults, which correspond to rules of correct usage of the Linux kernel API;
- races and deadlocks, which are related with parallel execution;
- generic problems, which are common for C programs such as null pointer dereference, integer overflow, etc.

Authors show that 29.2% of typical errors fixed in stable branches of the Linux kernel are generic problems. Statistics of memory problems corresponding to all generic faults is shown in Table. 1.

Table. 1. Ratio of memory problems corresponding to all generic faults

Type	**Percentage**
NULL pointer dereference	30.4%
Resource: memory leak, double free, use after free	23.5%

Buffer overflow	7.8%
Uninitialized: uninitialized pointer free, write to unallocated memory	5.9%
Total	67.6%

This information shows that the main part of generic faults match memory errors. We suggest to improve situation with memory safety of the Linux kernel with help of static verification.

The Linux Driver Verification project (LDV) [7, 9, 10] aims at performing automatic static driver verification and reporting detected problems. It provides a static verification framework called Klever [11] for Linux kernel verification including automated environment model generation [12, 13], rules of correct kernel API usage, interfaces for storing and visualization of verification results [14]. As a verification engine *Klever* includes the CPAchecker [15] verification tool.

In this work, we added several extensions into the CPAchecker verification tool for memory safety verification and improved *Klever* environment models to check memory safety for drivers of the Linux kernel. We have made experimental evaluation on drivers of Linux kernel 4.11.6 and 4.16.10, analyzed all memory safety problems reported by the verification tool and classified them into bugs and false alarms. We prepared bug reports and fixes to the newest kernel versions. Regarding false alarms, we conclude that automatic environment generation heavily affects verification results and requires further improvement.

3. Symbolic memory graphs

The symbolic memory graph (SMG) algorithm [16] is a kind of shape analysis. It works with directional graph representation of a memory state. Nodes are used for symbolic values, memory regions and abstracted structures representation. Edges show references between nodes and are divided into *point-to edges* for pointers and *has-value edges*. Each edge and node in SMG has a set of *labels* representing size, offset and allocation status. One symbolic memory graph with abstractions can represent several memory states called concrete memory images. Set of all concrete memory images for SMG *G* is denoted as *MI(G)*.

Our SMG implementation in CPAchecker keeps mapping between global, stack variables and memory regions. Also, it tracks mapping between symbolic and concrete values. A memory graph is modified in correspondence with analyzed source code.

Detailed description of operations on SMG can be found at [16]. Here we provide a brief overview.

3.1. Read/write data reinterpretation

This operation emulates memory modification with validity checks.

Modifications: A level of details for a memory model allows to take into account such low level interpretation as unions and provide facility for reinterpretation values even on the same offset with different types.

Algorithm supports partial values overwrite if memory for corresponding field intersects. For example:

```
1   union {
2       int i;
3       char c;
4   } u;
5   u.i = 10;
6   u.c = 'A';
```

After line 5 union *u* will contain integer value 10 with size 4 byte, but after line 6 from this union we are able to read 1 byte char 'A' or an undefined 4 byte integer value.

Checks: For these operations, the algorithm performs checks against null pointer dereference and read/write within object bounds.

3.2. Join of SMGs

This operation is central one for abstraction and decision whether a current memory state is covered by another one and vice versa, so the algorithm can drop one of the states. It takes as input 2 SMGs G_1, G_2, compares their concrete memory images and produces join status with summarization SMG G. If $MI(G_1) \not\subseteq MI(G_2)$ and $MI(G_1) \not\supseteq MI(G_2)$ then SMGs are semantically incomparable and their join is undefined.

Algorithm travels through pair of SMGs and tries to join nodes. It is possible if nodes have same sizes, validity, and special conditions for join with abstract lists. Abstract lists are joinable if they have same head, previous and next fields offsets, a join result will have a number of elements equal to minimum from originals. Also, a result of a join region with an abstract list become an abstract list. It is possible to insert an empty list abstraction at any correct position in a graph to increase opportunity of correct join.

3.3. Summarizing sequences of objects to list abstraction

This operation comes from the shape analysis theory. Ideas for different abstractions could be found in Sagiv work [17]. SMG uses single and double linked lists as abstractions.

The algorithm discovers sequences of neighboring objects which could be considered as list entry candidates and then sequentially adds them into one abstract list and increases its size. An abstraction size is considered as number of elements necessarily present in the abstraction.

3.4. Abstract list materialization

Materialization is an operation for unfolding the abstraction to memory regions on write/read from abstracted regions.

3.5. Checking equality and inequality of values and pointers

The algorithm supports incomplete checking for equality and inequality of values and pointers. In some cases, it can fail with different point-to edges from one abstracted region.

The tool performs stack variables cleaning on function exit and checking for dangling pointers to allocated memory, which helps identify memory leak errors.

Let's consider analysis of a simple example:

```
   void main() {
1     void *array;
2     long b = 2;
3     long c = 3;
4     array = calloc(1, 16);
5     memcpy(&array[4], &b, 4);
6     memcpy(&array[5], &c, 4);
   }
```

Steps of the algorithm are shown in figs 1-6 below.

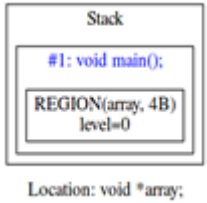


Fig. 1. Modification: allocate the 4 byte memory region on stack for pointer array

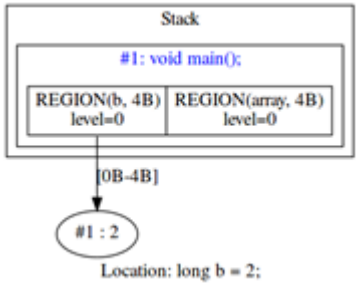


Fig. 2. Modification: allocate the 4 byte memory region on stack for variable b and assign it a new value #1 with explicit value 2

Check: a memory region size is sufficient for the assigned value.

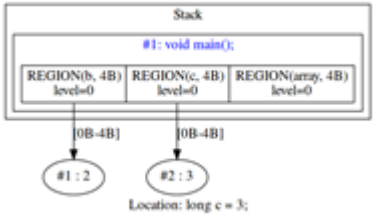


Fig. 3. Modification: allocate the 4 byte memory region on stack for variable c and assign it a new value #2 with explicit value 3

Check: a memory region size is sufficient for the assigned value

Fig. 4. Modification: allocate the 16 byte memory region on heap (mark it by tag calloc_ID3), fill it by NULL values, and assign to array a new point-to-value #4 which points to 0 offset of region calloc_ID3

Check: a region memory size is sufficient for the assigned value.

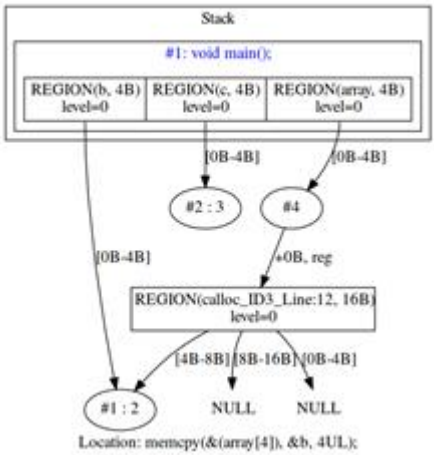


Fig. 5. Modification: assign 4 byte value #1 by offset 4 of region calloc_ID3

Check: dereference and assignment are done within allocated memory.

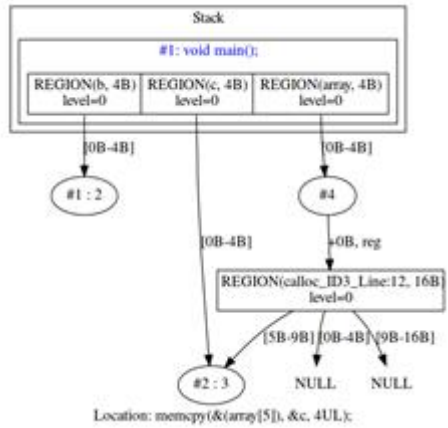


Fig. 6.Modification: assign 4 byte value #2 by offset 5 of region calloc_ID3, remove intersecting values, so value at offset 4 of region calloc_ID3 is not defined

Check: dereference and assignment are done within allocated memory.

4. Extensions for SMG

4.1. Bit precise model

The Linux kernel operates on structures with bit fields. We implemented bit fields in CPAchecker and switched SMG operations granularity from byte to bit precision. Also, we simulate structure alignment corresponding to GCC compiler memory usage.

4.2. Predicate extension

We implemented tracking of predicates over symbolic and concrete values stored in a memory graph. This feature allows filtering infeasible paths. On branching we perform a predicate satisfiability check to decide which branch is feasible. In addition, this method allows us to extend memory region over-read and overwrite checks for arrays using an error predicate check on a data reinterpretation operation.

4.3. On-demand memory

We consider the Linux kernel as trusted code and drivers as untrusted code in following sense: all structures provided to drivers by the kernel core are controlled by the kernel. We assume that the kernel recursively initializes all structure/union fields so drivers do not require to manage these structures. We supported the current point of view as the *on-demand memory (ODM)* concept within CPAchecker.

Allocation of *ODM* is made by special function *void* ext_allocation()*. A returned pointer allows any recursive dereference by any offset and distinguishes values by list of offsets and pointers from the original pointer. Additionally, any explicitly allocated memory which is reachable from on-demand memory is considered as automatically freed on program exit.

SMG implementation of ODM is done by special labels on memory regions and following behavior rules:

- any first dereference (read/write/free) of ODM pointers assumes that they are not NULL, ODM function pointers are an address to a pure function which returns nondeterministic value for non-pointer return value types or a pointer to ODM for pointer return value types;
- read memory:
  - read without previous read or write:
    - ✓ valid for any offset;
    - ✓ returns nondeterministic values for non-pointer types and a pointer to ODM for pointer types;
  - read after write:
    - ✓ valid for any offset;
    - ✓ returns values that were written by write;
  - read after read:
    - ✓ valid for any offset;
    - ✓ returns the same values that were read previously;
  - read after free is not valid.
- write memory:
  - write:
    - ✓ valid for any offset;
    - ✓ store new values in memory;
  - write after free is not valid.
  - free memory:
  - pointers to ODM are not subjected for memory leaks;
  - pointers to regular memory which are contained in ODM are not subjected for memory leaks;
  - free of any ODM offset is valid;
  - double free of ODM with the same offset is not valid;
  - read or write of freed ODM is not valid.

5. Configurable Program Analysis

The theory of SMG is implemented as Configurable Program Analysis (CPA) [18] within CPAchecker under the name SMGCPA.

Common CPA has *abstract domain*, *transfer*, *merge* and *stop* operators:

- *abstract domain* describes abstract states which represent sets of concrete states of the program;
- *transfer* gets one state and a control flow operation as input and returns all states which appears after applying the operation on the original state;
- *merge* takes 2 states as input and tries to combine them into one;
- *stop* identifies when one state is covered by others and decides whether it is required to continue analysis with a current state.

CPAchecker allows to combine different CPAs into one composite CPA. It works with a composite state which includes states of each involved CPAs. *Merge* produces a Cartesian product of separate analyses *merge* results.

SMGCPA fits into CPA conception with the following operators:

- abstract domain has SMG states as abstractions;
- transfer performs SMG transformations corresponding to a current control flow operation;
- merge tries to join SMGs from states and returns new SMG if join is successful;
- stop checks whether MI(G1) ⊆ MI(G2) or a state has memory issues.

6. Experimental results

Experiments were performed with the help of *Klever* static verification framework [11], that is a part of LDV project [7]. *Klever* automatically generates environment models for each separate driver.

We checked memory safety for drivers of Linux 4.11.6 and Linux 4.16.10.

Table 2 and 3 present results of experiments on 6224 and 5215 generated verification tasks for Linux 4.11.6 and 4.16.10 respectively. We used the 15 minutes CPU time limit for each verification task. We performed manual analysis of 561 Unsafe verdicts for Linux 4.11.6 and 266 Unsafe verdicts for Linux 4.16.10 and classified 49 Unsafes as real memory bugs and 512 as false alarms for Linux 4.11.6 and 29 real bugs and 237 false alarms for Linux 4.16.10.

Table 2. Evaluation on drivers of Linux 4.11.6

Safe	1560		
Unknown	4023	Timeouts	2594
		Others	1429

Unsafe	641	Bugs	49
		False alarms	512
		Without marks	80

Table 3. Evaluation on drivers of Linux 4.16.10

Safe	2093		
Unknown	2830	Timeouts	1293
		Others	1537
Unsafe	292	Bugs	29
		False alarms	237
		Without marks	26

Causes of false alarms (512 on 4.11.6 and 237 on 4.16.10) are the following.

- Imprecise environment models (258 + 96);

Automatically generated environment models could mistakenly provide wrong driver initialization and cleanup. Also, some emulated functions are imprecise for correct proof of memory safety.

- Absent function (139 + 58);

Current environment models do not contain functions imported from other drivers. This leads to false alarms if undefined functions are important for memory safety properties.

- Require predicate SMG (83 + 43);

These false alarms are connected mainly with arithmetic operations on unknown values. We expect that some common patterns used in software could be emulated by additional predicates description, e.g. bitwise AND on unsigned values provide result value less or equal to operands and this is common check for array dereference in the Linux kernel.

- SMG problems (13 + 32);

Problems with analysis such as missed values after merge and wrong assumptions about loop invariants.

- Verification task generator problems (10 + 5);

The verification task generator omits information about packed pragma for structures at final source files. Sometimes it provides less allocation sizes than unpacked structure sizes.

- Unknown allocation sizes (9 + 3);

If SMG can not derive explicit values for allocation sizes it uses a predefined value, which may be less than required.

The list of reported bugs is presented in Table 4. Not all bugs were reported, because some of them were detected in old unsupported drivers or were already fixed.

Table 4. Bugs in Linux 4.11.6 reported to Linux Kernel Mailing List (https://lkml.org/lkml)

Message ID	Subject
2017/8/1/615	Buffer overread in pv88090-regulator.ko
2017/8/10/693	hwmon:(stts751) buffer overread on wrong chip
2017/8/10/597	dmaengine: qcom_hidma: avoid freeing an uninitialized pointer
2017/8/15/322	ASoC: samsung: i2s: Null pointer dereference on samsung_i2s_remove
2017/8/10/535	i2c: use release_mem_region instead of release_resource
2017/8/16/493	mtd: plat-ram: Replace manual resource management by devm
2017/8/11/366	mISDN: Fix null pointer dereference at mISDN_FsmNew
2017/8/10/522	parport: use release_mem_region instead of release_resource
2017/8/11/368	video: fbdev: udlfb: Fix use after free on dlfb_usb_probe error path
2017/8/10/550	dvb-usb: Add memory free on error path in dw2102_probe()
2017/8/16/345	udc: Memory leak on error path and use after free

Table 5. Bugs in Linux 4.16.10 reported to Linux Kernel Mailing List (https://lkml.org/lkml)

Message ID	Subject
2018/7/6/412	uwb: hwa-rc: fix memory leak at probe
2018/7/18/551	media: dm1105: Limit number of cards to avoid buffer over read
2018/7/23/964	media: dw2102: Fix memleak on sequence of probes
2018/7/6/389	video: goldfishfb: fix memory leak on driver remove
2018/7/23/944	firmware: vpd: Fix section enabled flag on vpd_section_destroy
2018/7/27/764	misc: ti-st: Fix memory leak in the error path of probe()
2018/7/27/503	media: vimc: Remove redundant free
2018/7/23/949	gpio: ml-ioh: Fix buffer underwrite on probe error path
2018/7/27/769	can: ems_usb: Fix memory leak on ems_usb_disconnect
2018/7/27/661	regulator: tps65217: Fix NULL pointer dereference on probe
2018/7/27/655	scsi: 3ware: fix return 0 on the error path of probe
2018/7/27/772	net: mdio-mux: bcm-iproc: fix wrong getter and setter pair
2018/7/23/1020	HID: intel_ish-hid: tx_buf memory leak on probe/remove
2018/8/6/572	pinctrl: axp209: Fix NULL pointer dereference after allocation

2018/7/27/508	media: davinci: vpif_display: Mix memory leak on probe error path
2018/7/27/512	drm: qxl: Fix error handling at qxl_device_init
2018/7/27/727	fmc: Fix memory leak and NULL pointer dereference
2018/7/27/755	drm: qxl: Fix NULL pointer dereference at qxl_alloc_client_monitors_config
2018/6/9/253	staging: rts5208: add error handling into rtsx_probe
2018/7/27/644	tty: rocket: Fix possible buffer overwrite on register_PCI
2018/8/6/615	serial: mxs-auart: Fix potential infinite loop
2018/8/7/292	usb: gadget: fotg210-udc: Fix memory leak of fotg210->ep[i]

Let's consider the bug 2017/8/15/322 from Table 4 discovered in the Samsung I2S Controller driver within Linux 4.11.6 for which our patch was applied in 4.14-rc1.

```
1229 static int samsung_i2s_probe(struct platform_device *pdev)
1230 {
1231     struct i2s_dai *pri_dai, *sec_dai = NULL;
```

Fig. 7. (a) probe function

Klever provides a full error trace from an entry point to a error occurrence for the Unsafe verdict. The parts of the error trace for the Samsung I2S Controller driver are shown in fig. 7.

Fig. 7 (a) shows a part of the error trace with the declaration of the variable *struct i2s_dai *pri_dai* in function *samsung_i2s_probe()*. In the same function in fig. 7 (b) *pri_dai* is initialized by function *i2s_alloc_dai()* (line 1246), and field *sec_dai* becomes NULL (line 1095).

The third part of the error trace in fig. 7.(c) shows that *sec_dai* initialization is skipped by condition in line 1319 (*quirks & QUIRK_SEC_DAI*) triggered by device capabilities, so *pri_dai* is remained equal to NULL.

In the fig. 7, (d) we see that the structure *pri_dai* becomes stored at *driver_data* by *dev_set_drvdata()* in line 1363 and then extracted by *dev_get_drvdata()* in line 1382 of *samsung_i2s_remove()*. Next the driver assigns *sec_dai* in line 1383 and then perform dereference of *sec_dai* in line 1386 without check for NULL, which leads to NULL pointer dereference.

The bug can be reproduced on Samsung s3c6410-i2s and exynos7-i2s1 devices by inserting and removing driver module *sound/soc/samsung/i2s.ko*, because the condition in line 1319 is false for *i2sv3_dai_type* and *i2sv5_dai_type_i2s1* (see lines 1454 and 1477 in *sound/soc/samsung/i2s.c*).

```
1246            pri_dai = i2s_alloc_dai(pdev, 0);
1087              struct i2s_dai *i2s ;
1089              i2s = (struct i2s_dai *)tmp;
1090              assume(((unsigned long)i2s) != ((unsigned long)((struct i
1093              i2s->pdev = pdev;
1094              i2s->pri_dai = (struct i2s_dai *)0;
1095              i2s->sec_dai = (struct i2s_dai *)0;
1096              i2s->i2s_dai_drv.symmetric_rates = 1U;
1097              i2s->i2s_dai_drv.probe = &samsung_i2s_dai_probe;
1098              i2s->i2s_dai_drv.remove = &samsung_i2s_dai_remove;
1099              i2s->i2s_dai_drv.ops = &samsung_i2s_dai_ops;
1100              i2s->i2s_dai_drv.suspend = &i2s_suspend;
```

Fig. 7. (b) pri_dai initialization

```
1319    if (quirks & QUIRK_SEC_DAI) {
1320        sec_dai = i2s_alloc_dai(pdev, true);
1321        if (!sec_dai) {
1322            dev_err(&pdev->dev, "Unable to alloc I2S_sec\n");
1323            ret = -ENOMEM;
1324            goto err_disable_clk;
1325        }
1326
1327        sec_dai->lock = &pri_dai->spinlock;
1328        sec_dai->variant_regs = pri_dai->variant_regs;
1329        sec_dai->dma_playback.addr = regs_base + I2STXDS;
1330        sec_dai->dma_playback.chan_name = "tx-sec";
1331
1332        if (!np) {
1333            sec_dai->dma_playback.filter_data = i2s_pdata->dma_
1334            sec_dai->filter = i2s_pdata->dma_filter;
1335        }
1336
1337        sec_dai->dma_playback.addr_width = 4;
1338        sec_dai->addr = pri_dai->addr;
1339        sec_dai->clk = pri_dai->clk;
1340        sec_dai->quirks = quirks;
1341        sec_dai->idma_playback.addr = idma_addr;
1342        sec_dai->pri_dai = pri_dai;
1343        pri_dai->sec_dai = sec_dai;
1344
1345        ret = samsung_asoc_dma_platform_register(&pdev->dev,
1346                    sec_dai->filter, "tx-sec", NULL);
1347        if (ret < 0)
1348            goto err_disable_clk;
```

Fig. 7. (c) skipped pri_dai initialization

Fig. 7. (d) dev_set_drvdata/dev_get_drvdata and NULL pointer dereference

7. Conclusions and future work

We have presented the approach to find memory errors in Linux kernel drivers using static verification. Whereas the Linux kernel is widely tested, our experiments show that it is possible to find memory bugs in Linux kernel drivers with help of our static verification method.

We expect to reduce the false alarm rate by introducing a more precise predicate extension. Further efforts will be aimed at reducing the number of timeouts.

References

[1]. G. Klein, J. Andronick, K. Elphinstone, T. Murray, T. Sewell, R. Kolanski, and G. Heiser, Comprehensive formal verification of an os microkernel. ACM Transactions on Computer Systems, vol. 32, no. 1, 2014, pp. 2:1–2:70.

[2]. T. Ball, E. Bounimova, B. Cook, V. Levin, J. Lichtenberg, C. McGarvey, B. Ondrusek, S. K. Rajamani, and A. Ustuner, Thorough static analysis of device drivers. SIGOPS Operating Systems Review, vol. 40, no. 4, 2006, pp. 73–85.

[3]. D. Engler and M. Musuvathi. Static analysis versus software model checking for bug finding. Lecture Notes in Computer Science, vol. 2937, 2004, pp. 191–210.
[4]. Saturn. Precise and Scalable Software Analysis. Available at: http://saturn.stanford.edu/, accessed 01.12.2018.
[5]. T. Witkowski, N. Blanc, D. Kroening, and G. Weissenbacher. Model checking concurrent Linux device drivers. In Proceedings of the 22nd IEEE/ACM Int. Conference on Automated Software Engineering, 2007, pp. 501–504.
[6]. N. Palix, G. Thomas, S. Saha, C. Calvès, J. Lawall, and G. Muller. Faults in Linux: Ten years later. In Proceedings of the 16th Int. Conference on Architectural Support for Programming Languages and Operating Systems, 2011, pp. 305–318.
[7]. Linux driver verification project. Available at: http://linuxtesting.org/ldv, accessed 01.12.2018.
[8]. V. Mutilin, E. Novikov, and A. Khoroshilov. Analysis of typical faults in Linux operating system drivers. Trudy ISP RAN/Proc. ISP RAS, vol. 22, 2012, pp. 349–374 (in Russian). DOI: 10.15514/ISPRAS-2012-22-19.
[9]. A. Khoroshilov, V. Mutilin, A. Petrenko, and V. Zakharov. Establishing Linux driver verification process, Lecture Notes in Computer Science, vol. 5947, pp. 165–176, 2010.
[10]. I. Zakharov, M. Mandrykin, V. Mutilin, E. Novikov, A. Petrenko, and A. Khoroshilov. Configurable toolset for static verification of operating systems kernel modules. Programming and Computer Software, vol. 41, no. 1, 2015, pp. 49–64.
[11]. Klever verification framework. Available at: https://forge.ispras.ru/projects/klever, accessed 01.12.2018.
[12]. I.S. Zakharov, V.S. Mutilin, and A.V. Khoroshilov. Pattern-based environment modeling for static verification of linux kernel modules. Programming and Computer Software, vol. 41, no. 3, 2015, pp. 183–195.
[13]. A. Khoroshilov, V. Mutilin, E. Novikov, and I. Zakharov. Modeling environment for static verification of linux kernel modules. Lecture Notes in Computer Science, vol. 8974, 2015, pp. 400–414.
[14]. E. Novikov and I. Zakharov. Towards automated static verification of GNU C programs. Lecture Notes in Computer Science, vol. 10742, 2018, pp. 402–416.
[15]. D. Beyer and M. Keremoglu. CPAchecker: A tool for configurable software verification. Lecture Notes in Computer Science, vol. 6806, 2011, pp. 184–190.
[16]. K. Dudka, P. Peringer, and T. Vojnar. Byte-precise verification of low-level list manipulation. Lecture Notes in Computer Science, vol. 7935, 2013, pp. 215–237.
[17]. R. Wilhelm, S. Sagiv, and T. W. Reps. Shape analysis. Lecture Notes in Computer Science, vol. 1781, 2000, pp. 1–17.
[18]. D. Beyer, T. A. Henzinger, and G. Théoduloz. Configurable software verification: concretizing the convergence of model checking and program analysis. Lecture Notes in Computer Science, vol. 4590, 2007, pp. 504–518. [Online]. Available: http://portal.acm.org/citation.cfm?id=1770351.1770419

Статическая верификация ошибок использования памяти в модулях ядра ОС Linux

А.А. Васильев <vasilyev@ispras.ru>
Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Abstract. Ошибки использования памяти в модулях ядра операционной системы Linux сложно обнаружить, но они могут привести к серьезным последствиям. В данной статье мы описываем метод статической верификации, позволяющий обнаруживать все ошибки в рамках предположений метода. Статическая верификация крупных пректов таких, как ядро ОС Linux, требуют дополнительных усилий. Современные инструменты статической верификации не позволяют анализировать ядро как единое целое, поэтому мы используем упрощенную автоматически генерируемую модель окружения. Эта модель вносит некоторую неточность, но позволяет проводить статическую верификацию. Также мы допускаем отсутствие тела некоторых функций, что приводит к неполным программам, написанных на языке ANSI C. В данной работе предлагается подход к обнаружению ошибок использования памяти в таких неполных программах. Наша техника статической верификации основана на теории символических графов памяти и ее расширении для снижения количества ложных срабатываний. Мы ввели концепцию памяти по требованию для упрощения моделей интерфейсов ядра ОС и реализовали ее в фреймворке статической верификации CPAchecker. Также мы изменили точность модели памяти CPAchecker с байтов на поддержку отдельных битов и добавили поддержку выравнивания структур, аналогичное использованому в компиляторе. Для повышения точности анализа мы реализовали предикатное расширение состояния символического графа памяти. Мы провели проверку модулей ядра ОС Linux для версий 4.11.6 и 4.16.10 с помощью фреймворка статической верификации Klever с инструментом верификации CPAchecker, что позволило проанализировать 6224 и 5215 модулей соответствующих версий. Ручной анализ предупреждений от фреймворка Klever выявил 78 реальных ошибок в модулях ядра. Мы сделали патчи для исправления 33 из них.

Ключевые слова: анализ рекурсивных структур данных; статическая верификация; символические графы памяти; модели памяти.

DOI: 10.15514/ISPRAS-2018-30(6)-8

Для цитирования: Васильев А.А. Статическая верификация ошибок использования памяти в модулях ядра ОС Linux. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 143-160. DOI: 10.15514/ISPRAS-2018-30(6)-8

Список литературы

[1]. G. Klein, J. Andronick, K. Elphinstone, T. Murray, T. Sewell, R. Kolanski, and G. Heiser, Comprehensive formal verification of an os microkernel. ACM Transactions on Computer Systems, vol. 32, no. 1, 2014, pp. 2:1–2:70.

[2]. T. Ball, E. Bounimova, B. Cook, V. Levin, J. Lichtenberg, C. McGarvey, B. Ondrusek, S. K. Rajamani, and A. Ustuner, Thorough static analysis of device drivers. SIGOPS Operating Systems Review, vol. 40, no. 4, 2006, pp. 73–85.

[3]. D. Engler and M. Musuvathi. Static analysis versus software model checking for bug finding. Lecture Notes in Computer Science, vol. 2937, 2004, pp. 191–210.
[4]. Saturn. Precise and Scalable Software Analysis. Available at: http://saturn.stanford.edu/, accessed 01.12.2018.
[5]. T. Witkowski, N. Blanc, D. Kroening, and G. Weissenbacher. Model checking concurrent Linux device drivers. In Proceedings of the 22nd IEEE/ACM Int. Conference on Automated Software Engineering, 2007, pp. 501–504.
[6]. N. Palix, G. Thomas, S. Saha, C. Calvès, J. Lawall, and G. Muller. Faults in Linux: Ten years later. In Proceedings of the 16th Int. Conference on Architectural Support for Programming Languages and Operating Systems, 2011, pp. 305–318.
[7]. Linux driver verification project. Available at: http://linuxtesting.org/ldv, accessed 01.12.2018.
[8]. В.С. Мутилин, Е.М. Новиков, А.В. Хорошилов, Анализ типовых ошибок в драйверах операционной системы Linux. Труды ИСП РАН, том 22, 2012, стр. 349-374. DOI: 10.15514/ISPRAS-2012-22-19.
[9]. A. Khoroshilov, V. Mutilin, A. Petrenko, and V. Zakharov. Establishing Linux driver verification process, Lecture Notes in Computer Science, vol. 5947, pp. 165–176, 2010.
[10]. I. Zakharov, M. Mandrykin, V. Mutilin, E. Novikov, A. Petrenko, and A. Khoroshilov. Configurable toolset for static verification of operating systems kernel modules. Programming and Computer Software, vol. 41, no. 1, 2015, pp. 49–64.
[11]. Klever verification framework. Available at: https://forge.ispras.ru/projects/klever, accessed 01.12.2018.
[12]. I.S. Zakharov, V.S. Mutilin, and A.V. Khoroshilov. Pattern-based environment modeling for static verification of linux kernel modules. Programming and Computer Software, vol. 41, no. 3, 2015, pp. 183–195.
[13]. A. Khoroshilov, V. Mutilin, E. Novikov, and I. Zakharov. Modeling environment for static verification of linux kernel modules. Lecture Notes in Computer Science, vol. 8974, 2015, pp. 400–414.
[14]. E. Novikov and I. Zakharov. Towards automated static verification of GNU C programs. Lecture Notes in Computer Science, vol. 10742, 2018, pp. 402–416.
[15]. D. Beyer and M. Keremoglu. CPAchecker: A tool for configurable software verification. Lecture Notes in Computer Science, vol. 6806, 2011, pp. 184–190.
[16]. K. Dudka, P. Peringer, and T. Vojnar. Byte-precise verification of low-level list manipulation. Lecture Notes in Computer Science, vol. 7935, 2013, pp. 215–237.
[17]. R. Wilhelm, S. Sagiv, and T. W. Reps. Shape analysis. Lecture Notes in Computer Science, vol. 1781, 2000, pp. 1–17.
[18]. D. Beyer, T. A. Henzinger, and G. Théoduloz. Configurable software verification: concretizing the convergence of model checking and program analysis. Lecture Notes in Computer Science, vol. 4590, 2007, pp. 504–518. [Online]. Available: http://portal.acm.org/citation.cfm?id=1770351.1770419

Конфигурационная сборка варианта ядра Linux для прикладных систем[1]

С.В. Козин <kozyyy@yandex.ru>

Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
Национальный исследовательский университет Высшая школа экономики, 101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Операционная система Linux – это современная открытая операционная система, содержащая более 10 000 конфигурационных переменных и множество функциональных системных элементов. Ставится задача создания некоторого варианта ОС для класса прикладных систем (медицины, биологии и др.). Эта задача решается путем анализа базовых функций ядра ОС и выбора из множества элементов наиболее подходящих для оперативного управления прикладными функциями. На их основе создается модель вариабельности из базовых характеристик ОС и модель варианта ОС, включающая основные функциональные элементы ядра ОС. Эти модели тестируются на предмет правильности их идентификации и связей с другими элементами. Затем по этим моделям проводится конфигурирование варианта ОС в виде конфигурационного файла. Этот файл верифицируется, и проходит комплексное тестирование на наборе тестов, проверяющих правильность функционирования операционной среды и процессов обработки заданий прикладных систем. В данной работе рассматривается способ сборки готового варианта ядра операционной системы. Будут затронуты необходимые пакеты, патчи для них и способы их установки. Затем представляется способ конфигурирования собранного варианта системы и настройки ядра для запуска.

Ключевые слова: система Linux; модель характеристик; модель системы; верификация; тестирование; вариант ядра ОС; конфигурационная сборка; верификация исходного файла; тестирование выходного файла.

DOI:10.15514/ISPRAS-2018-30(6)-9

Для цитирования: Козин С.В. Конфигурационная сборка варианта ядра Linux для прикладных систем. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 161-170. DOI: 10.15514/ISPRAS-2018-30(6)-9

1. Введение

Операционная система Linux – это современная открытая операционная система, содержащая более 10 000 конфигурационных переменных и

[1] Работа поддержана грантом РФФИ №16-01-00352

множество функциональных системных элементов. Ставится задача создания некоторого варианта ОС для класса прикладных систем (медицины, биологии и др.) [1-3]. Эта задача решается путем анализа базовых функций ядра ОС и выбора из множества элементов наиболее подходящих для оперативного управления прикладными функциями.

На их основе создается модель вариабельности из базовых характеристик ОС и модель варианта ОС, включающая основные функциональные элементы ядра ОС [4-5]. Эти модели тестируются на предмет правильности их идентификации и связей с другими элементами.

Затем по этим моделям проводится конфигурирование варианта ОС в виде конфигурационного файла. Этот файл верифицируется, и проходит комплексное тестирование на наборе тестов, проверяющих правильность функционирования операционной среды и процессов обработки заданий прикладных систем [6-8].

В данной работе рассматривается способ сборки готового варианта ядра операционной системы.

К настоящему моменту сформировались стандарты, определяющие интерфейсы, облегчающие сборку вариантов разных систем, в том числе и ОС. К ним относятся стандарты:

- POSIX.1-2008;
- Filesystem Hierarchy Standard (FHS) Version 3.0;
- Linux Standard Base (LSB) Version 5.0 (2015).

Используемые пакеты, необходимые для удовлетворения требований LSB:

- **LSB Core**: Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib;
- **LSB Runtime Languages**: Perl.

Вариант ОС строится с использованием уже установленного дистрибутива Linux (такого как Debian, Open Mandriva, Fedora или openSUSE). Система Linux (хост) используется в качестве отправной точки для предоставления необходимых программ, включая компилятор, компоновщик и оболочку, для создания нового варианта ОС.

При этом в хост-систему необходимо установить пакеты:

Bash-3.2, Binutils-2.25, Bison-2.7, Bzip2-1.0.4, Coreutils-6.9, Diffutils-2.8.1, Findutils-4.2.31, Gawk-4.0.1, GCC-4.9, включая компилятор C++, g++ , Glibc-2.11, Grep-2.5.1a, Gzip-1.3.12, Linux Kernel-3.2, M4-1.4.10, Make-4.0, Patch-2.5.4, Perl-5.8.8, Sed-4.1.5, Tar-1.22, Texinfo-4.7, Xz-5.0.0.

Затем создается новый раздел на диске и файловая система, пригодная для Linux, выбираются и скачиваются необходимые пакеты и патчи для создания варианта системы и сохранения его в новой файловой системе и в новом разделе. Устанавливается ряд пакетов, которые будут формировать базовый пакет разработки (первичный вариант системы) для использования в реальной

системе. Некоторые из этих пакетов используются при сборке компилятора. После этого создается и конфигурируется финальный вариант системы.

2. Подготовка к созданию нового варианта ОС

Предлагается подход к созданию системы, который основан на использовании свободного пустого раздела для размещения на нем системы Linux (около 6 гигабайт), хранения всех исходных архивов и компиляции пакетов [1-4].

Кроме того, требуется свободное временное хранилище достаточно большого размера и дисковое пространство для компиляции пакетов. Например, рекомендуется использовать небольшой раздел диска для обеспечения пространства подкачки – swap.

Первоначально диск разбивается с помощью специальной утилиты, например, cfdisk или fdisk (жесткий диск) для создания нового раздела, например, /dev/sda для основного диска. После создания этого раздела создается файловая система, например, ext3 или ext4.

3. Подготовка инструментария (toolchain)

Пакет ОС Binutils устанавливается первым и как Glibc выполняет различные функциональные тесты на ассемблере и компоновщике, чтобы определить, какие программные функции включены или отключены. Неправильная конфигурация GCC или Glibc может привести к неочевидной поломке инструментальной цепочки, которая может не выявиться до конца сборки всего дистрибутива. Для предотвращения неправильных конфигураций выполняется тестовый набор, который позволяет обнаружить ошибки, прежде чем будет выполнена слишком большая работа по установке некоторых функций ОС.

Пакет Binutils устанавливает свой компилятор, компоновщик и заголовки APILinux. Это позволяет стандартной библиотеке C (Glibc) взаимодействовать с функциями, которые предоставляет ядро Linux.

Следующий пакет – Glibc. Для построения Glibc используются компилятор, бинарные инструменты и заголовки ядра. При этом в Glibc применяется компилятор, который был задан параметром -host, переданным скрипту configure (например, компилятор может быть i686-lfs-linux-gnu-gcc).

Бинарные инструменты и заголовки ядра более сложны в установке. После запуска configure проверяется содержимое файла config.make в каталоге glibc-build.

4. Создание версии ОС

Когда ядро загружает систему, требуется наличие нескольких узлов устройств, в частности консольных. Узлы устройств должны быть созданы на жестком диске, чтобы они были доступны до запуска udevd, а также при запуске Linux с init=/bin/bash. Рекомендуется подключение виртуальной

файловой системы (например, tmpfs) в каталоге /dev и возможность динамического создания устройств в этой виртуальной файловой системе по мере их обнаружения или доступа. Создание устройства обычно выполняется во время загрузки udev.

Linux поддерживает список смонтированных файловых систем в файле /etc/mtab. Современные ядра хранят этот список внутри себя и предоставляют его пользователю через файловую систему /proc. Чтобы работали утилиты, ожидающие наличия /etc/mtab, нужно создать соответствующую символическую ссылку.

В соответствии с документацией необходимо установить следующие пакеты:
Linux-4.18.5 API Headers, Man-pages-4.16, Glibc-2.28, Zlib-1.2.11, File-5.34, Readline-7.0, M4-1.4.18, Bc-1.07.1, Binutils-2.31.1, GMP-6.1.2, MPFR-4.0.1, MPC-1.1.0, Shadow-4.6, GCC-8.2.0, Bzip2-1.0.6, Pkg-config-0.29.2, Ncurses-6.1, Attr-2.4.48, Acl-2.2.53, Libcap-2.25, Sed-4.5, Psmisc-23.1, Iana-Etc-2.30, Bison-3.0.5, Flex-2.6.4, Grep-3.1, Bash-4.4.18, Libtool-2.4.6, GDBM-1.17, Gperf-3.1, Expat-2.2.6, netutils-1.9.4, Perl-5.28.0, XML::Parser-2.44, Intltool-0.51.0, Autoconf-2.69, Automake-1.16.1, Xz-5.2.4, Kmod-25, Gettext-0.19.8.1, Libelf 0.173, Libffi-3.2.1, OpenSSL-1.1.0i, Python-3.7.0, Ninja-1.8.2, Meson-0.47.1, Procps-ng-3.3.15, E2fsprogs-1.44.3, Coreutils-8.30, Check-0.12.0, Diffutils-3.6, Gawk-4.2.1, Findutils-4.6.0, Groff-1.22.3, GRUB-2.02, Less-530, Gzip-1.9, IPRoute2-4.18.0, Kbd-2.0.4, Libpipeline-1.5.0, Make-4.2.1, Patch-2.7.6, Sysklogd-1.5.1, Sysvinit-2.90, Eudev-3.2.5, Util-linux-2.32.1, Man-DB-2.8.4, Tar-1.30, Texinfo-6.5, Vim-8.1.

Чтобы начать сборку и установку окончательного варианта системы LFS, необходимо войти в среду chroot.

5. Конфигурирование варианта ОС

Для конфигурации варианта системы вы сможете выбрать параметры, включенные в ядро. В зависимости от архитектуры будут доступны разные наборы параметров, а также их содержимое. Однако некоторые параметры будут доступны независимо от варианта выбора встроенной архитектуры. Ниже приведен список основных параметров меню, доступных для всех встроенных архитектур Linux [4-8]:

- общая настройка;
- поддержка загружаемых модулей;
- блоки;
- сетевые настройки;
- драйверы устройств;
- файловые системы;
- параметры безопасности;
- криптографические параметры;
- библиотечные процедуры.

Ядро может включать в себя множество специальных опций безопасности в

составе стека SELinux, реализованного Национальным агентством безопасности США (NSA). Однако только для нескольмких встроенных устройств предпочтительно использование такой обширной функциональности.

Для начала проведения конфигурирования варианта системы ОС требуется проверить файл /etc/udev/rules.d/70-persistent-net.rules, чтобы определить имя, присвоенное сетевому устройству, и настройки именования через правила udev. Описание файла начинается с блока комментариев, за которым следуют две строки для каждого сетевого адаптера. Первая строка сетевой карты – это описание с комментариями, показывающее идентификаторы оборудования (например, поставщик PCI и идентификаторы устройств в случае карты PCI) и драйверы в круглых скобках.

Ни идентификатор аппаратного обеспечения, ни драйвер не используются для определения имени для предоставления интерфейса; эта информация предназначена только для справки. Вторая строка – это правило udev, которое соответствует сетевой карте и фактически присваивает имя.

Некоторое ПО, которое может быть установлено позже (например, различные медиаплееры), ожидает, что существуют символические ссылки /dev/cdrom и /dev/dvd, указывающие на устройство CD-ROM или DVD-ROM. Кроме того, может быть удобно помещать ссылки на эти символические ссылки в /etc/fstab.

udev поставляется со сценарием, который будет генерировать файлы правил для создания этих символических ссылок в зависимости от возможностей каждого устройства. Во-первых, сценарий может работать в режиме «by-path» (используется по умолчанию для устройств USB и FireWire), где создаваемые правила зависят от физического пути к устройству CD или DVD. Во-вторых, он может работать в режиме «по-id» (по умолчанию для устройств IDE и SCSI), где создаваемые ими правила зависят от идентификационных строк, хранящихся на самом устройстве CD или DVD. Путь определяется сценарием path_id в udev, а строки идентификации считываются из аппаратного обеспечения с помощью своих программ ata_id или scsi_id, в зависимости от того, какой тип устройства у вас есть.

Если ожидается, что физический путь к устройству (то есть порты и / или слоты, в которые он подключается) изменится, например, при потребности переместить диск на другой порт IDE или другой разъем USB, то необходимо использовать режим «by-id». С другой стороны, в случае изменения идентификации устройства или его уничтожения, его необходимо заменить другим устройством с теми же возможностями, подключая к одному и тому же разъему -path.

Для каждого устройства с проблемой дублирования находится соответствующий каталог в разделе /sys/class или /sys/block. Для видеоустройств это может быть /sys/class/video4linux/videoX. При этом анализируются атрибуты, которые идентифицируют устройство однозначно, а затем создаются символьные ссылки.

То, какие интерфейсы связаны с сетевым сценарием, обычно зависит от файлов в /etc/sysconfig/. Этот каталог должен содержать файл для каждого настраиваемого интерфейса, например, ifconfig.xyz, где «xyz» должно описывать сетевую карту. Обычно используется имя интерфейса (например, eth0). Внутри этого файла есть атрибуты этого интерфейса, такие как его IP-адрес, маски подсети и т. д.

Во время инициализации ядра первая запущенная программа указывается в командной строке, либо по умолчанию ей является init, которая считывает файл инициализации /etc/inittab.

Для ОС выполняется команда rc – для запуска всех скриптов, начинающихся с S в каталоге /etc/rc.d/rcS.d, а также скриптов, начинающихся с S в файле /etc/rc.d/rc?.d, где знак вопроса задается значением initdefault.

rc-скрипт читает библиотеку функций в /lib/lsb/init-functions. Эта библиотека также читает дополнительный файл конфигурации /etc/sysconfig/rc.site. Любые параметры файла конфигурации системы могут быть альтернативно размещены в этом файле, что позволяет консолидировать в нем все системные параметры.

Файл inputrc используется для конфигурирования библиотеки Readline, предназначенной для редактирования командных строк пользователя, вводимых с терминала. Readline используется Bash и большинством других оболочек, а также многими другими приложениями.

6. Подготовка к запуску варианта ОС

Файл /etc/fstab используется некоторыми программами для определения того, где должны быть установлены файловые системы по умолчанию, в каком порядке и какие из них необходимо проверить (для контроля целостности) перед установкой.

Файловым системам с MS-DOS или Windows-источником (т.е. Vfat, ntfs, smbfs, cifs, iso9660, udf) требуется специальная опция utf8, чтобы символы в именах файлов, не входящие в набор ASCII, были правильно интерпретированы. Для локалей, отличных от UTF-8, значение iocharset должно быть таким же, как и набор символов локали. Соответствующее определение набора символов (найденное в Файловых системах -> Поддержка родного языка при настройке ядра) должно быть скомпилировано в ядро или построено как модуль. Для файловых систем vfat и smbfs также необходима опция «codepage».

Построение ядра включает в себя несколько этапов – настройку, компиляцию и установку.

В большинстве случаев модули Linux загружаются автоматически, но иногда требуется определить другой порядок. Для этой цели нужно использовать modprobe или insmod, порядок для которых задается в файле /etc/modprobe.d/usb.conf. Этот файл необходимо создать так, чтобы, если USB-драйверы (ehci_hcd, ohci_hcd и uhci_hcd) были созданы как модули, они были

загружены в правильном порядке; ehci_hcd необходимо загрузить до ohci_hcd и uhci_hcd, чтобы избежать появления предупреждения во время загрузки.

Расположение загрузочного раздела GRUB – это выбор пользователя, который влияет на конфигурацию. Одна рекомендация состоит в том, чтобы иметь отдельный небольшой (рекомендуемый размер – 100 МБ) раздел только для загрузки информации. Таким образом, каждая сборка может обращаться к одним и тем же загрузочным файлам, и доступ может быть сделан из любой загруженной системы.

Используя приведенную выше информацию, можно определить соответствующий указатель для корневого раздела (или отдельного загрузочного раздела, если он используется), установить файлы GRUB и настроить загрузочный трек, сгенерировать /boot/grub/grub.cfg. После этого новая система готова к загрузке.

7. Заключение

В процессе разработки варианта ОС на базе ядра Linux был сконфигурирован некоторый вариант системы для конечного пользования. Данная система имеет все стандартные функции ядра Linux, функцию сжатия данных, файловую систему, интерфейс командной строки, функции арифметического вычисления для рациональных чисел (включая числа с плавающей точкой), функции чтения аудио-файлов, компилятор языка C и C++, систему защиты паролей, интерфейс установки пакетов pkg, список контроля доступа, текстовый редактор, возможность работы с сетью, поиск по системе, библиотеку функций баз данных, генератор хэш функций, XML-парсер, openSSL и некоторые другие вспомогательные функции.

Такая система может использоваться во многих областях. Она обладает необходимым набором инструментов и для домашнего пользования, если пользователь хорошо знаком с семейством операционных систем Linux.

Список литературы

[1] Лаврищева Е. М., Коваль Г.И., Слабоспицкая О.О., Колесник А.Л. Особенности процессов управления при создании семейств программных систем. Проблемы программирования, no. 3, 2009 г., стр. 40-49.

[2] Лаврищева Е. М., Коваль Г.И., Слабоспицкая О.О., Колесник А.Л. Теоретические аспекты управления вариабельностью в семействах программных систем. Вестник КНУ, серия физ.-мат. наук, no. 1, 2011 г., стр.151-158

[3] Лаврищева Е.М. Программная инженерия и технология программирования сложных систем. Учебник. 2-издание. Москва, Юрайт, 2018, 431 стр.

[4] Е.М. Лаврищева, В.С. Мутили, А.Г. Рыжов. Аспекты моделирования вариабельных программных и операционных систем. Сб. трудов XIX Всероссийский научной конференции «Научный сервис в сети Интернет», 2017, стр. 327-341.

[5] Лаврищева Е.М. Петренко А.К. Моделирование систем и их семейств. Труды ИСП РАН, 2016 г., том 28, вып. 6, с. 49-65. DOI: 10.15514/ISPRAS-2016-28(6)-4.

[6] И.С. Захаров, М.У. Мандрыкин, В.С. Мутилин, Е.М. Новиков, А.К. Петренко, А.В. Хорошилов. Конфигурируемая система статической верификации модулей ядра операционных систем. Труды ИСП РАН, том 26, вып. 2, 2014 г., стр. 5-42. DOI: 10.15514/ISPRAS-2014-26(2)-1.

[7] Kozin S.V., Mutilin V.S. Static Verification of Linux Kernel Configurations. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 4, 2017, pp. 217-230. DOI: 10.15514/ISPRAS-2017-29(4)-14.

[8] Кулямин В.В., Лаврищева Е.М., Мутилин В.С., Петренко А.К. Верификация и анализ вариабельных операционных систем. Труды ИСП РАН, 2016, том 28, вып. 3, стр. 189-208. DOI: 10.15514/ISPRAS-2016-28(3)-12.

Linux kernel configuration build for application systems

S.V. Kozin <kozyyy@yandex.ru>

Ivannikov Institute of System Programming, RAS,
25, A. Solzhenitsyn str., Moscow, 109004, Russia
National Research University Higher School of Economics,
20, Myasnitskaya str, Moscow., 101000, Russia

Abstract. The Linux operating system is a modern open operating system containing more than 10,000 configuration variables and a large variety of functional system elements that handle the processing of various kinds of tasks. The task is to create some version of the OS for a class of applied systems (medicine, biology, etc.). This task is solved by analyzing the basic functions of the OS kernel and choosing from a variety of elements the most suitable for the operational management of application functions. Based on them, a model of variability is created from the basic characteristics of the OS and the model of the OS variant, including the main functional elements of the OS kernel. These models are tested for the correctness of their identification and relationships with other elements. Then, using these models, the OS version is configured as a configuration file. This file is verified and undergoes comprehensive testing on a set of tests that verify the correct functioning of the operating environment and the processing of tasks of applied systems. This paper discusses how to build a ready-made version of the operating system kernel from source. The preparations, the necessary packages, the patches for them and the ways of their installation will be affected. Then it presents a method for configuring a system version assembled from source and configuring the kernel to run.

Keywords: Linux system; characteristics model; system model; verification; testing; OS kernel version; configuration building; source file verification; output file testing

DOI: 10.15514/ISPRAS-2018-30(6)-9

For citation: Kozin S.V. Linux kernel configuration build for application systems. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 161-170 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-9

References

[1] Lavrischeva E.M., Koval' G.I., Slabospitskaya O.O., Kolesnik A.L. Product Line Development Management Specifics. Problemy programmiovaniya [Problems of Software Developement], no. 3, 2009., pp. 40-49 (in Ukrainian).

[2] Lavrischeva E.M., Slabospitskaya O.O., Koval' G.I., Kolesnik A.L. Theoretical Aspects of Variability Management in Product Lines. Vesnik KNU seria fiz.-mat. nauk [Notes of KNU, series on maths and physics], no. 1, 2011, pp.151-158 (in Ukrainian).

[3] Lavrischeva E.M. Software engineering and programming technology for complex systems. Textbook. 2nd edition. Moscow, Yuright, 2018, 431 p. (in Russian).

[4] Lavrischeva K.M., Mutilin V.S., Ryzhov A.G. Aspects of Modeling of Variable Software and Operating Systems. In Proc. of the XIX All-Russian conference on Scientific Services in the Internet, 2017, pp. 327-341 (in Russian).

[5] Lavrischeva K.M., Petrenko A.K. Software Product Lines Modeling. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 49-64 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-4 (in Russian).

[6] Zakharov I.S., Mandrykin M.U., Mutilin V.S., Novikov E.M., Petrenko A.K., Khoroshilov A.V. Configurable Toolset for Static Verification of Operating Systems Kernel Modules. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 2, 2014, pp. 5-42. DOI: 10.15514/ISPRAS-2014-26(2)-1 (in Russian).

[7] Kozin S.V., Mutilin V.S. Static Verification of Linux Kernel Configurations. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 4, 2017, pp. 217-230. DOI: 10.15514/ISPRAS-2017-29(4)-14.

[8] Kuliamin V.V., Lavrischeva E.M., Mutilin V.S., Petrenko A.K. Verification and analysis of variable operating systems. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 3, 2016, pp. 189-208 (in Russian). DOI: 10.15514/ISPRAS-2016-1(2)-12/

Исследовательский поиск научных статей[1]

[1]*Я.Р. Недумов <yaroslav.nedumov@ispras.ru>*
[1,2,3,4] *С.Д. Кузнецов <kuzloc@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2]*Московский Государственный университет имени М. В. Ломоносова Москва, 119991, ГСП-1, Ленинские горы, д. 1*
[3] *Московский физико-технический институт, 141700, Московская область, г. Долгопрудный, Институтский пер., 9*
[4] *НИУ "Высшая школа экономики", 101000, Россия, г. Москва, ул. Мясницкая, д. 20*

Аннотация. В этой статье мы анализируем современные работы, посвященные поисковому поведению ученых, и работы, посвященные методам исследовательского поиска. Мы показываем, что четыре вида поиска из шести, характерных для ученых (в том числе самый субъективно сложный – исследование новых направлений и самый часто возникающий – поиск для поддержания осведомленности), можно с достаточной степенью уверенности считать исследовательским поиском. Таким образом, поисковые системы, предназначенные для поиска научных данных, должны реализовывать специфичные для исследовательского поиска инструменты. Чтобы проверить это, мы анализируем семнадцать специализированных поисковых систем: от встроенных в электронные библиотеки Scopus и WoS до Google Scholar и социальных сетей для ученых, таких как ResearchGate и Academia.edu. Мы приходим к выводу, что степень их адаптации к специфике исследовательского поиска оставляет простор для улучшений и обсуждаем возможные направления их развития.

Ключевые слова: исследовательский поиск; поисковые системы для ученых; поисковое поведение

DOI: 10.15514/ISPRAS-2018-30(6)-10

Для цитирования: Недумов Я.Р., Кузнецов С.Д. Исследовательский поиск научных статей. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 171-198. DOI: 10.15514/ISPRAS-2018-30(6)-10

[1] Эта работа поддержана грантом РФФИ 17-07-00978 А

1. Введение

Границы между традиционным поиском по ключевым словам и исследовательским поиском (exploratory search) довольно размыты. В своей классической работе 2006 года [1] Гари Марчионини определяет исследовательский (или разведочный) поиск от противного, сопоставляя его с традиционным поиском (lookup).

Задача традиционного поиска возникает, когда необходимо найти факты или ответы на четко сформулированные вопросы. Результаты традиционного поиска легко интерпретируемы и сравнительно компактны. Примеры запросов, характерных для традиционного поиска: сегодняшний курс валют, симптомы гриппа, в каком году была Куликовская битва и тому подобные.

Однако не все поисковые задачи можно свести к традиционному поиску. Часто пользователь поисковой системы до конца не понимает, что именно хочет найти, заранее не знает ключевых слов, изучает интересующую его тему уже в процессе поиска. Такого рода поиск может иметь сразу несколько целей, которые при этом могут изменяться по мере знакомства пользователя с предметной областью. Марчионини называет поиск такого рода *поиском для обучения* (searching to learn). Другая область исследовательского поиска связана с проведением исследований (investigation), когда пользователю требуется провести анализ, синтез, сопоставление данных из разных источников и т.д. Примером такого поиска может быть поиск с целью выбора товара для покупки или выбор места для отпуска.

В работе Уайта и Рота [2] делается попытка определить исследовательский поиск более формально. Для определения исследовательского поиска рассматриваются два аспекта: контекст проблемы (the problem context) и процесс поиска (the search process).

Для определения контекста проблемы, характерного для исследовательского поиска, авторы используют модель поисковой задачи, предложенную Марчионини в книге [3]. Марчионини рассматривает поисковую задачу в трех аспектах: специфичность цели (goal specifity), длительность (timeliness), объем результата (volume of the answer).

Специфичность цели по Марчионини характеризует степень ее определенности. Специфичность цели может быть высокой: например, если требуется найти актуальный курс валюты или прогноз погоды на завтра. А может быть низкой: например, если требуется разобраться в новой теории или выявить все точки зрения на некоторую проблему. В случае поиска фактов ищущий обычно достаточно уверен в том, нашел он уже искомый результат, или следует продолжать поиски. Напротив, когда специфичность цели низкая, трудно понять, следует ли уже закончить поиск или следует его продолжать, пытаясь получить более четкий и конкретный ответ.

Под длительностью Марчионини понимает ожидаемый необходимый объем времени для выполнения поиска. Для задач получения конкретных фактов

длительность обычно порядка нескольких секунд. Для задач изучения какой-то новой области длительность может варьироваться от месяцев и лет до бесконечности, в тех случаях, когда исследователь не рассчитывает полностью достигнуть цели. Наконец, объем характеризует сложность восприятия ответа поисковой системы. Объем может измеряться в битах информации или во времени, необходимом ищущему для его понимания. Объем может варьироваться от одиночных слов, чисел, изображений до коллекций документов или учебных курсов. Понять, что ответ поисковой системы содержит искомый курс валюты можно за несколько секунда, но если требовалось найти алгоритм решения некоторой задачи, то могут потребоваться часы, чтобы убедиться, что ответ поисковой системы релевантен запросу.

Если контекст проблемы является открытым, постоянным и многоаспектным, то есть если специфичность цели низкая, длительность поиска высокая, а объем результата большой, то такой контекст проблемы характерен для исследовательского поиска.

Второй аспект поиска, рассматриваемый Уайтом и Ротом, – процесс поиска. Для исследовательского поиска он должен быть оппортунистическим, итеративным и мультитактическим. К сожалению, они не дают более формального определения этим терминам, поэтому будем пользоваться ими в их словарных значениях. Таким образом, согласно Уайту и Роту, исследовательский поиск – это такой поиск, для которого контекст проблемы является открытым, постоянным и многоаспектным, а процесс поиска является оппортунистическим, итеративным и мультитактическим.

Интуитивно понятно, что поиск научных публикаций часто обладает многими характеристиками исследовательского поиска. Цель этой статьи – формализовать это интуитивное понимание, исследовать, какие поисковые задачи ученых можно отнести к исследовательскому поиску, какие существуют подходы к решению задачи исследовательского поиска вообще, и как они реализуются в специализированных поисковых системах для ученых.

Дальнейший текст статьи организован следующим образом. В следующем разделе мы более подробно рассмотрим, что и каким образом ищут ученые, а затем исследуем, насколько их поисковые задачи можно отнести к исследовательскому поиску. В четвертом разделе мы обсудим виды инструментов, предназначенных для ведения исследовательского поиска. В пятом разделе рассмотрим существующие поисковые системы для ученых и реализованные в них инструменты исследовательского поиска.

2. Поисковые задачи ученых

Начало исследований поискового поведения ученых прослеживают по крайней мере до сороковых годов прошлого века [4]. В классических работах Эллиса и др. [4, 5] авторы выделили восемь видов поискового поведения ученых: начало исследования (starting), анализ ссылок (chaining), просмотр

(browsing), дифференцирование (differentiating), мониторинг (monitoring), извлечение (extracting), верификация (verifying) и окончание исследования (ending).

- Начало исследования включает в себя активности по предварительному сбору информации, знакомству с предметной областью. Часто включает общение с коллегами или научным руководителем.
- Анализ ссылок – перемещение между статьями по библиографическим ссылкам, как назад, к статьям, процитированным в данной статье, так и вперед, к статьям, цитирующим данную.
- Просмотр – полунаправленное, полуструктурированное блуждание по коллекциям аннотаций с целью найти релевантные. Используется как при начальном знакомстве с областью, так и для поддержания осведомленности.
- Дифференцирование – неявное разделение публикаций на качественные и некачественные. Может базироваться как на метаданных – месте публикации, авторах, так и на анализе текста статей, использованных методах.
- Мониторинг – процесс поддержки осведомленности о происходящем в области научных интересов.
- Извлечение – систематическая проработка некоторого источника с целью извлечения релевантной информации.
- Верификация – проверка найденных фактов.
- Окончание исследования во многом повторяет начало. Хотя большая часть работы с литературой приходится на начало исследования, многие корреспонденты Эллиса возвращались к этой работе при написании отчета или статьи по завершающемуся проекту.

Авторы [4, 5] сопоставляли интервью физиков, химиков и социологов и пришли к выводу, что различия в поисковом поведении ученых разных специальностей несущественны. В более свежих работах [6, 7] тоже не обнаруживается существенная корреляция между областью научных интересов и поисковым поведением. Основные черты модели поиска [4] были подтверждены и дополнены в работе [8] десять лет спустя, после появления и широкого распространения универсальных поисковых движков, таких как Google.

В работе [9] модель поискового поведения Эллиса рассматривается в качестве формализации задачи исследовательского поиска наравне с моделью Марчионини.

Авторы более поздней статьи Атукорала и др. [10] исследовали поисковое поведение ученых в области компьютерных наук и то, как они используют те средства поиска, которые предоставляют им поисковые системы, как универсальные, так и специализированные. Компьютерные науки были выбраны в предположении, что исследователи в этой области раньше начали

пользоваться электронными поисковыми системами и обладают более развитыми навыками их использования. В ходе анализа авторы выделили четыре цели поиска при проведении научной работы: поддержка осведомленности, исследование новых направлений, обзор литературы и поиск коллег для совместной работы, а также две цели поиска при проведении образовательной работы: подготовка лекций и рекомендация статей студентам. Авторы не претендуют на то, что список целей является исчерпывающим или может быть обобщен на другие науки, но считают, что выделенные ими цели оказывают наибольшее влияние на поисковое поведение ученых.

В следующем разделе мы обсудим, насколько можно считать исследовательским каждый из видов поиска, выявленных Атукоралой с соавторами.

3. Относится ли поисковое поведение ученых к исследовательскому поиску?

В этом разделе для каждой из целей поиска ученых, выявленных в исследовании Атукоралы и др. [10], мы рассмотрим соответствующую поисковую задачу с точки зрения контекста проблемы и процесса поиска и таким образом определим, можно ли отнести ее к исследовательскому поиску или нет. Напомним, что всего были выделены шесть целей: поддержание осведомленности, исследование новых направлений, обзор литературы, подготовка лекций, рекомендация статей студентам и поиск потенциальных коллег для совместного проведения исследований. Далее мы рассмотрим каждую из них, опираясь на описания задачи и процесса поиска из исследования Атукоралы и др.

Участники исследования, осуществлявшие поиск с целью поддержки осведомленности, делали это, время от времени просматривая сайты профильных конференций, сайты издательств и личные страницы известных авторов и исследовательских групп в поисках новых статей. Оценим такой поиск с точки зрения контекста проблемы и процесса поиска.

Контекст проблемы в аспекте специфичности является открытым, потому что никогда нельзя быть уверенным, что охвачены все релевантные работы, всегда остается вероятность, что какая-то новая работа не попала в результаты поиска. Контекст является постоянным, потому что новые работы появляются все время и нельзя рассчитывать ознакомиться с ними со всеми и закончить поиск раз и навсегда. Контекст мультиаспектен, потому что результатом поиска является нечто априори неизвестное, мы не знаем, что содержится в новой статье: это может быть новый факт, лишь дополняющий существующую картину мира, а может быть новая теория, на осмысление которой уйдут годы.

Поиск с целью поддержки осведомленности по описанию авторов [10] выполняется время от времени, когда представится возможность, с

использованием как систем оповещений, так и ручного просмотра личных страниц избранных авторов. Процесс поиска можно считать оппортунистическим, итеративным и мультитактическим.

Таким образом, поиск с целью поддержания осведомленности в аспекте контекста проблемы является открытым, постоянным и мультиаспектным, а в аспекте процесса поиска – оппортунистическим, итеративным и мультитактическим.

Покажем, что поиск с целью исследования новых направлений также является исследовательским поиском. Рассмотрим характеристики контекста проблемы и процесса поиска. Контекст проблемы является открытым, так как трудно определить границу, при достижении которой мы можем быть уверены, что новая область полностью изучена, а поиск окончен. Ожидаемая длительность поиска с целью исследования новых направлений сравнительно велика и может достигать нескольких лет у начинающих ученых. Хотя и нельзя сказать, что такой поиск является постоянным, все же он не является традиционным с точки зрения ожидаемой длительности. После окончания поиска с целью исследования новых направлений ученый вероятнее всего продолжит исследования в области и будет осуществлять поиск с целью поддержки осведомленности. В этом смысле эти две цели поиска можно считать продолжением одной другой. Контекст проблемы является мультиаспектным, так же, как и для поиска с целью поддержания осведомленности. Априори мы не знаем границ и объема изучаемой области.

Процесс поиска с целью исследования новых направлений также является характерным для исследовательского поиска. Он итеративен, потому что мы постепенно уточняем свое понимание новой предметной области. Он оппортунистичен, потому что у ученого обычно нет готового учебного плана для изучения новой области. Он является мультитактическим, потому что на разных этапах используются разные источники и поисковые сервисы. Многие начинают с общения с коллегами, использования Википедии и Google, только затем переходя к более специализированным источникам [10].

Поиск с целью обзора литературы возникает в момент написания обзора существующих работ. В целом он похож на поиск с целью поддержки осведомленности, но более формализован. В случае же написания систематических обзоров последовательность действий задается совсем жестко. Специфичность задана выбранной методологией, поэтому можно точно сказать, когда поиск завершен, время поиска ограничено и объем результатов тоже, хотя и может быть достаточно велик. Таким образом, хотя поиск с целью составления обзора литературы в значительно меньшей степени является исследовательским, чем поиск с целью поддержки осведомленности, он все-таки далек от традиционного. Время поиска существенно больше нескольких секунд, объем результатов велик; кроме того, после анализа отобранных по методологии статей цель поиска может быть уточнена, и тогда

потребуется еще одна итерация. Таким образом, можно сделать вывод, что поиск с целью обзора литературы является исследовательским.

Поиск как при подготовке лекций, так и для рекомендации статей студентам во многом заключается в поиске уже известных ученому статей [10]. Специфичность цели здесь высока: пользователь уверен в том, что именно он хочет найти. Ожидаемое время поиска невелико (а в случае, если ищущий помнит название работы, задача сводится к традиционному поиску). Объем результата тоже невелик. Процесс поиска больше похож на характерный для исследовательского поиска. Некоторые респонденты просматривали результаты Google Scholar в поисках ссылок, по которым они уже переходили, некоторые использовали специальный BibTex файл, в котором сохраняли ссылки на все релевантные статьи. Процесс поиска не является итеративным. Таким образом, поиск с целью обучения как при подготовке лекций, так и для рекомендации статей для студентов можно считать исследовательским только с некоторой натяжкой.

Табл. 1. Сводная таблица целей использования электронных библиотек в контексте исследовательского поиска
Table 1. Summary table of the use of digital libraries in the context of exploratory search

Цель поиска	**Контекст проблемы**			**Процесс поиска**
	Специфич-ность	Длитель-ность	Объем результатов	
Поиск с целью исследования новых направлений	низкая	высокая	большой	итеративный, оппортунистический, мультитактический
Поиск с целью поддержания осведомленности	низкая	высокая	большой	итеративный, оппортунистический, мультитактический
Поиск с целью обзора литературы	высокая	высокая	большой	задан используемой методологией
Поиск с целью сотрудничества	высокая	средняя	большой	итеративный, оппортунистический, мультитактический
Поиск с целью обучения (подготовки лекций, рекомендации статей студентам)	высокая	низкая	небольшой	оппортунистический, мультитактический

Наконец, последний вид поиска: поиск рецензентов или потенциальных коллег для совместной работы. В этом случае специфичность цели высока: пользователь ожидает увидеть в результатах поиска профили ученых, и поисковая система вернет ему профили ученых. Оценка релевантности возвращенных результатов может оказаться сложнее. С одной стороны, респонденты Атокаралы упоминали, что одной из задач было нахождение профиля ученого, который приходил к ним на семинар или встретился на

конференции. В этом случае можно быстро оценить результат, и объем можно считать небольшим.

С другой стороны, поиск рецензента для статьи может оказаться куда более сложной задачей (и в некоторых журналах авторов статей просят самим предоставить список потенциальных рецензентов). Понять по доступной информации, сможет ли найденный кандидат справиться с написанием рецензии на статью, может оказаться вообще невозможно, придется отправлять текст нескольким кандидатам, чтобы они самостоятельно приняли решение. Процесс поиска также зависит от варианта задачи и для поиска рецензента вероятно будет итеративным, мультитактическим и оппортунистическим.

Таким образом, можно прийти к выводу, что этот последний вид поиска может быть как исследовательским, если мы не знаем, какого именно человека хотим найти (в случае поиска рецензента по не очень знакомой области), так и вполне традиционным (если мы ищем знакомого). Поэтому можно сделать вывод, что и наиболее трудоемкий вид поиска с целью исследования новых направлений, и чаще всего возникающий вид поиска с целью поддержки осведомленности являются по своей природе видами исследовательского поиска, и для их поддержки требуются специальные инструменты. Остальные результаты систематизированы в табл. 1. В следующем разделе мы рассмотрим, какие существуют инструменты для проведения исследовательского поиска.

4. Инструменты исследовательского поиска

По определению исследовательского поиска контекст проблемы характеризуется низкой специфичностью, высокой длительностью и большим объемом ответа, а процесс поиска является итеративным, оппортунистическим и мультитактическим. Такой поиск требует специальных инструментов, отличных от инструментов для традиционного поиска.

Авторы [2] предлагают восемь видов инструментов, характерных для систем исследовательского поиска:

- поддержка при составлении запроса и быстрое уточнение запроса;
- фасеты для фильтрации результатов по метаданным;
- использование контекста;
- интуитивно понятные визуализации, помогающие при принятии решений;
- поддержка обучения и понимания;
- поддержка совместной работы;
- истории поиска, рабочие места и отслеживание прогресса;
- поддержка управления задачами.

Многие из этих возможностей в том или ином виде уже внедрены в универсальные поисковые системы и не требуют представления, но многие не

получили распространения несмотря на прошедшие со времени написания статьи годы. Мы кратко опишем каждую из возможностей, но за более подробным и исчерпывающим описанием отсылаем к первоисточнику [2].

Существуют и другие классификации инструментов, предназначенных для исследовательского поиска. Автор одного из самых свежих обзоров по исследовательскому поиску [11] особенно отмечает важность инструментов обзора больших объемов результатов поиска и рассматривает четыре основных подхода: иерархическую классификацию, фасетную классификацию, динамическую кластеризацию и социальную классификацию (классификацию по социальным, экономическим, демографическим и другим критериям).

Сравнительно новым подходом к обработке больших коллекций результатов поиска является тематическое моделирование.

Кроме того, специально для анализа структуры научных областей разработан целый ряд специализированных инструментов, способных обрабатывать большие коллекции статей.

В ходе анализа всех этих инструментов мы разделили их на три группы, отражающие специфику контекста проблемы исследовательского поиска:

1. инструменты помощи при формулировании запроса.
2. инструменты помощи при анализе результатов.
3. инструменты поддержки длительного поиска.

Далее мы рассмотрим каждую из трех групп по очереди.

4.1. Инструменты помощи при формулировании запроса

Инструменты помощи при формулировании запроса помогают справиться с низкой специфичностью: так как пользователь не уверен в том, что именно он хочет найти, то он не уверен и в том, как сформулировать поисковый запрос. Системы исследовательского поиска должны помогать пользователю формулировать запрос. Суть быстрого уточнения запроса (rapid query refinement) [2] состоит в автоматическом расширении запроса за счет анализа близких запросов, сделанных ранее, возможно другими пользователями системы. Пользователь вводит (или начинает вводить) первое ключевое слово, а поисковая система предлагает ему несколько возможных продолжений запроса, из которых он может выбрать одно, после чего процесс может повторяться и для последующих слов запроса.

Интуиция, стоящая за этим методом, заключается в том, что разные люди могут искать одно и то же, и что, скорее всего, кто-то уже искал раньше то, что нужно пользователю сейчас. Таким образом, можно использовать историю близких поисковых запросов, чтобы вместо формулировки запроса с нуля, пользователь мог выбрать подходящий вариант из списка известных поисковой системе запросов.

Так как обычный запрос состоит всего из нескольких слов, зачастую многозначных, для повышения качества результатов необходимо дополнительно его расширять. Одним из способов такого расширения запроса является использование контекста. Контекст может быть получен или явно, за счет уточняющих вопросов пользователю, какие документы или фрагменты текста он считает релевантными, или неявно, за счет анализа пользовательского поведения.

Хотя большинство поисковых систем предполагают запрос в виде нескольких ключевых слов, возможны и более сложные запросы в виде фрагментов текста, картинок, аудио файлов и т.д. Например, авторы [12] использовали в качестве запросов фрагменты текста объемом около страницы и затем осуществляли поиск релевантных документов в коллекции с помощью тематического моделирования.

К этой же группе мы отнесем рекомендации, которые можно рассматривать в качестве неявных запросов, когда пользователю не нужно использовать ни ключевые слова, ни какой-то другой вид входных данных, а вместо этого система использует в качестве запроса уже накопленную в ней информацию о пользователе. Например, может использоваться информация о поисковых запросах, сделанных ранее, об отобранных пользователем статьях, о статьях, написанных самим пользователем, и так далее. В одном из последних обзоров методов рекомендации статей рассматривается более двухсот работ [13].

После того, как запрос сформулирован и отправлен, необходимо отобразить его результаты. Причем, поскольку для исследовательского поиска высокая точность формулировки запроса маловероятна, даже релевантные запросу результаты поиска могут оказаться нерелевантными действительной поисковой потребности и наоборот. По этой причине нельзя рассчитывать на то, что пользователю окажется достаточно одного или нескольких релевантных запросу результатов. Скорее всего, их потребуется значительно больше, причем на этом этапе необходимо предоставить пользователю средства для упрощения восприятия больших объемов результатов. Для этого служат инструменты из второй группы.

4.2. Инструменты помощи при анализе результатов

Количество результатов поискового запроса может быть очень велико. Классические поисковые системы взвешивают все результаты по релевантности запросу и возвращают их в виде списка, отсортированного от более релевантных результатов к менее релевантным. Однако для исследовательского поиска на первых итерациях большая часть даже релевантных введенному запросу результатов может оказаться нерелевантной поисковым потребностям пользователя из-за недостаточно хорошо сформулированного запроса. Из-за этого нельзя больше рассчитывать, что удовлетворяющий пользователя результат будет найден на одной из первых страниц. Пользователь должен будет так или иначе проанализировать

больший объем результатов, и простой сортированный список не очень для этого подходит.

Чтобы дать пользователю возможность приблизительно оценить структуру результатов поискового запроса и затем сфокусироваться на более релевантных частях, необходимо предоставлять пользователю средства обобщения, реферирования результатов поиска. Результат может быть рассеян по целой коллекции документов, и полнота поиска может оказаться значительно важнее точности. Хорошей иллюстрацией здесь может служить задача проведения систематического обзора: пропущенные релевантные результаты могут привести к тому, что из обзора будут сделаны неверные выводы. Для решения этой задачи разработан целый ряд методов. Первый, который мы рассмотрим, – это фасеты.

Фасеты позволяют ограничивать результаты поиска с помощью указания допустимых значений атрибутов каждого результата. Каждый фасет – это четко определенный аспект, характеристика или свойство объекта или класса объектов. Значения фасета разбивают множество объектов на непересекающиеся подмножества, объединение которых равно этому множеству объектов. Как правило, фасеты создаются вручную, и для каждого конкретного домена они свои. Например, для домена научных публикаций примерами фасетов могут служить год публикации, место публикации (название журнала или конференции), имя одного из авторов.

Для решения задачи обобщения результатов поиска фасеты применяются естественным образом. Все результаты поиска группируются в зависимости от значений каждого фасета, и затем пользователь может сужать пространство поиска, отбирая те или иные значения фасетов. Применительно к информационному поиску предполагается, что каждый фасет плоский, его значения не образуют иерархии. Некоторым обобщением фасетов можно считать иерархическую классификацию.

Иерархическая классификация предполагает разбиение результатов поиска по фиксированному множеству вложенных и непересекающихся в рамках одного уровня вложенности классов (фактически, дереву). Примерами подобных иерархических классификаторов являются каталоги веб-страниц, широко применявшиеся до появления поисковых систем, или, например, классификатор научных областей ACM[2]. Один раз разобравшись со структурой классификатора, пользователь сможет быстро оценивать множество результатов поиска и отбирать из него только нужные ему подмножества. Основным недостатком этого подхода является сложность создания и поддержания в актуальном состоянии подобных классификаторов. Для домена научных данных в бурно развивающихся научных областях классификаторы всегда оказываются на шаг позади.

[2] https://www.acm.org/about-acm/class

В тех случаях, когда применение классификатора по тем или иным причинам невозможно, применяют альтернативный инструмент: кластеризацию [11]. С помощью кластеризации можно автоматически выделить группы близких документов среди всех результатов поиска и позволить пользователю сэкономить усилия, просмотрев лишь по несколько документов из каждого кластера. Основной проблемой при реализации кластеризации является выбор хорошей функции расстояния между документами. Для перевода текстовых документов в векторное пространство могут применяться различные подходы [14-16].

Еще один способ неформальной классификации, применимой для исследовательского поиска, – это фолксономия, или народная классификация [11]. При использовании этого инструмента каждый класс задается соответствующим тегом, а каждый документ может быть отнесен к одному или нескольким классам путем ручного указания соответствующих тегов. Теги может проставлять каждый пользователь, формируя таким образом общую картину.

Кроме группировки результатов поиска самой по себе, в системах исследовательского поиска применяются различные методы визуализации результатов [2, 11]. Наконец, с точки зрения Уайта и Рота [2] системы исследовательского поиска должны не просто возвращать набор релевантных документов, но и помогать понять и изучить их. Во внимание должны приниматься сложность каждого документа, их взаимосвязи (в каком порядке их надо изучать), уровень подготовки пользователя. К сожалению, практических примеров реализации этой функциональности нам обнаружить не удалось.

В следующей группе собраны инструменты, помогающие при проведении длительного поиска.

4.3. Инструменты поддержки длительного поиска

Исследовательский поиск в отличие от традиционного поиска итеративен и протяжен во времени. Пользователь будет время от времени прерывать поиск, продолжать с того же места или возвращаться к уже просмотренным ранее результатам и так далее. Системы исследовательского поиска должны предоставлять соответствующие средства поддержки.

Среди инструментов исследовательского поиска, выделяемых авторами [2], к инструментам поддержки длительного поиска можно отнести следующие три категории:

- групповой поиск (collaboration);
- поддержка историй, рабочих мест и отслеживания прогресса (histories, workspaces, and progress updates);
- поддержка управления задачами (task management).

Системы исследовательского поиска должны сохранять историю поиска и позволять быстро возвращаться к полученным ранее результатам. Пользователи должны иметь возможность сохранять заметки по ходу поиска или необходимую информацию более сложной структуры. Кроме того, система должна отслеживать, какие пути уже исследованы, а какие нет, отслеживать скорость продвижения и предоставлять соответствующие данные пользователю. Сложный поиск должно быть можно выполнять не в одиночку, а группой. Это может оказаться полезным, потому что разные пользователи обладают разным опытом и могут дополнять друг друга. При наличии подходящего поискового интерфейса пользователи могут перенимать друг у друга как знания в изучаемой предметной области, так и знания о том, как искать.

Одна из классических универсальных систем для совместного поиска – SearchTogether [17]. SearchTogether предоставляла интегрированный интерфейс с функциями взаимного информирования о действиях всех членов группы, распределения работы и сохранения промежуточных результатов поиска.

Шесть лет спустя авторы [18] приходят к выводу, что большинство пользователей не использует никаких специальных средств для совместного поиска. Вместо этого они применяют универсальные инструменты: электронную почту, мессенджеры и социальные сети. Тем не менее, мы считаем, что для профессиональных пользователей поисковых систем необходимость изучения специализированного интерфейса не станет непреодолимым препятствием.

На этом мы заканчиваем краткий обзор инструментов исследовательского поиска. Мы описали три группы инструментов: инструменты помощи при формулировании запроса, инструменты помощи при анализе результатов и инструменты поддержки длительного поиска. В следующем разделе мы рассмотрим, какое применение они нашли в системах поиска научных публикаций.

5. Существующие подходы к решению задачи исследовательского поиска публикаций

Научные публикации представляют собой достаточно специфичный вид данных, что с одной стороны усложняет построение поисковых систем, а с другой стороны предоставляет дополнительные возможности повышения качества их работы по сравнению с универсальными поисковыми системами. Кратко рассмотрим основные специфические особенности научных публикаций.

Во-первых, специфичен текст статей: он написан в научном стиле, не окрашен эмоционально, включает большое количество узкоспециальных терминов, во многих областях науки – формул (например, химических, математических). Текст статей обладает достаточно устойчивой структурой: в большинстве

статей есть введение, обзор существующих работ, заключение. Во многих областях науки фактическим стандартом является наличие отдельного раздела с указанием методов, использованных при проведении исследования.

Во-вторых, текст статей существует не сам по себе, но вместе с метаданными. К метаданным относится информация об авторах статьи: их имена и аффилиации, адреса электронной почты; место публикации статьи – название журнала или конференции; аннотация – она обычно свободно доступна, даже если доступ к тексту статьи ограничен; год публикации и другие.

В-третьих, статьи со связями цитирования образуют ориентированный направленный граф без циклов – граф цитирования. Кроме собственно графа цитирования по статьям, также рассматриваются и производные от него: граф цитирования по авторам, графы социтирования по документам и по авторам [19].

Таким образом, хотя анализ непосредственно текстов научных статей и затруднен, за счет использования дополнительной информации – метаданных и графа цитирования, возможно повышение качества работы поисковых систем.

За прошедшие годы появилось немало исследовательских систем, реализующих те или иные средства исследовательского поиска для научных публикаций. Рассмотреть все их в рамках статьи не представляется возможным.

Авторы [20] в своем исследовании использования различных интерфейсов поиска рассмотрели две системы: ACM DL и Sowiport DL. С одной стороны, нам не известен исчерпывающий перечень таких систем, и даже известных систем слишком много, чтобы охватить их все в рамках одной статьи. С другой стороны, многие системы очень похожи между собой по отношению к предоставляемым функциям, и достаточно знать одну из них, чтобы понимать, чего ожидать от остальных.

Мы выделили несколько групп систем и выбрали наиболее известных, с нашей точки зрения, представителей в каждой из них:

- университетские разработки: CiteseerX [21], aMiner [22];
- общедоступные системы от крупных корпораций: Google Scholar и Microsoft Academic Search;
- SemanticScholar от института Аллена [23];
- социальные сети для ученых: Academia.edu, BibSonomy, Mendeley [24], ResearchGate;
- сайты электронных библиотек: ACM DL, arXiv.org, eLibrary, PubMed, ScienceDirect, Scopus, Springerlink, Web Of Science.

Далее в этом разделе мы покажем, какие средства поддержки исследовательского поиска используются в поисковых системах для научных публикаций и каким образом. Мы будем придерживаться разделения инструментов на три группы, введенного в предыдущем разделе. Сначала

рассмотрим инструменты помощи при формулировании запроса, затем инструменты анализа результатов и, наконец, инструменты поддержки длительного поиска.

5.1. Инструменты помощи при формулировании запроса в современных системах поиска результатов научных исследований

Специфическая терминология, используемая в научных статьях, сильно затрудняет формулировку поискового запроса для пользователя, не знакомого с предметной областью. Как отмечали респонденты [10], при исследовании новой области им часто приходится сначала использовать универсальные источники данных, такие как Google и Wikipedia, и только потом, после поверхностного ознакомления с предметной областью, они могут перейти к использованию специализированных поисковых систем. По этой причине особенно важно помогать пользователю на начальных этапах поиска, либо помогая ему сформулировать явный поисковый запрос, либо предполагая запрос неявно, за счет анализа истории поиска и уже отобранных пользователем релевантных статей. В этой группе инструментов мы рассмотрим средства быстрого уточнения запроса, использования контекста и рекомендаций. Сводные результаты можно увидеть в табл. 2.

С помощью быстрого уточнения запроса пользователь может выбирать из потенциально релевантных альтернатив, а не формулировать весь поисковый запрос целиком. Как ни странно, в большинстве рассмотренных систем никак не реализуется быстрое уточнение запроса, и в этом отношении они уступают универсальным поисковым движкам. В табл. 2 этот вариант обозначен словом *нет*. Среди оставшихся можно выделить две группы: в системах из первой группы реализуется классический вариант уточнения, основанный на агрегации запросов пользователей (обозначен в табл. 2 словом *классика*), системы из второй группы идут дальше и реализуют более доменоспецифичные подсказки (*адаптация к домену*). Например, MS Academic умеет распознавать в запросе названия журналов, а Academia.edu подсказывает так называемые исследовательские интересы (research interests).

Табл. 2. Поддержка инструментов помощи при формулировании запроса в современных системах поиска научных публикаций
Table 2. Support of tools to assist in the formulation of a query in modern systems of the search for scientific publications

Система	Быстрое уточнение запроса	Рекомендации
aMiner	адаптация к домену	для статьи
CiteseerX	нет	для статьи
Google Scholar	классика	для пользователя
MS Academic	адаптация к домену	для статьи

Semantic Scholar	классика, также есть возможность уточнить запрос со страницы с результатами	для статьи
Academia.edu	адаптация к домену	нет
BibSonomy	нет	нет
Mendeley	нет	для пользователя
ResearchGate	нет	для пользователя
ACM DL	нет	нет
arXiv.org	нет	нет
eLibrary	нет	нет
PubMed	классика	для статьи
ScienceDirect	нет	для пользователя*
Scopus	нет	нет
Springerlink	классика	нет
WOS	нет	нет

Современные универсальные поисковые движки автоматически отслеживают историю поисковых запросов пользователя и затем используют ее для ранжирования результатов (а также показа контекстной рекламы). Однако в специализированных поисковых системах для ученых аналогичную функциональности нам обнаружить не удалось. Мы делали по несколько запросов в режиме инкогнито, в обычном режиме и после авторизации на сайте (где это возможно), но не смогли обнаружить разницы в результатах, показанных на первых трех страницах поисковой выдачи. Таким образом, в этом аспекте все системы одинаковы, и мы не стали включать соответствующий столбец в сводную таблицу.

Если понимать под рекомендацией сценарий работы с системой, не предполагающий явного ввода запроса пользователем, то только в трех системах реализуется нечто подобное: Google Scholar, Mendeley и ResearchGate (значение *для пользователя* в сводной таблице). В ScienceDirect явным образом заявляется поддержка рекомендаций, но нам не удалось ими воспользоваться. Каждая из них так или иначе отслеживает активность пользователя и автоматически рекомендует статьи, релевантные его интересам. В некоторых системах реализована близкая функциональность – поиск связанных (related) статей (значение *для статьи* в сводной таблице). Остальные системы рекомендации не поддерживают. Следующий подраздел посвящен инструментам помощи при анализе результатов поиска.

5.2. Инструменты помощи при анализе результатов в современных системах поиска результатов научных исследований

В рассмотренных нами системах так или иначе реализуются четыре группы инструментов помощи при анализе результатов: фасеты, иерархическая классификация, фолксономия и визуализация результатов.

Табл. 3. Поддержка инструментов помощи при анализе результатов в современных системах поиска результатов научных исследований
Table 3. Support of assistance tools in analyzing the results in modern systems for search of research results

Система	Фасеты	Иерархическая классификация	Фолксономия	Визуализация
aMiner	расширенные	нет	есть	сложная
CiteseerX	нет	нет	нет	нет
Google Scholar	год	универсальный	нет*	простая*
MS Academic	расширенные	универсальный	нет	нет
Semantic Scholar	расширенные	нет	нет	простая
BibSonomy	нет	нет	есть	нет
Mendeley	нет	нет	нет	нет
ResearchGate	нет	нет	нет	нет
ACM DL	расширенные	доменный	нет	простая
arXiv.org	нет	универсальный	нет	нет
eLibrary	расширенные	универсальный	нет	нет
PubMed	расширенные	доменный	нет	нет
ScienceDirect	расширенные	универсальный	нет	нет
Scopus	расширенные	универсальный	нет	сложная
Springerlink	расширенные	универсальный	нет	сложная
WOS	расширенные	универсальный	нет	сложная

Academia.edu полноценный полнотекстовый поиск предоставляет только в платной версии, поэтому в этом подразделе мы ее не рассматривали.

Фасеты реализованы в большинстве систем, причем если базовые параметры, такие как год или место публикации, поддерживаются всеми, то более сложные параметры достаточно своеобразны. Например, Semantic Scholar умеет автоматически выявлять набор данных, использованный в статье для тестирования, MS Academic позволяет фильтровать результаты по аффилиациям авторов, PubMed поддерживает разделение по типу живых существ, исследованных в статье, и так далее. AMiner в своей подсистеме поиска экспертов предоставляет фасеты по индексу Хирша, полу и языку.

eLibrary поддерживает фасеты для просмотра сохраненных коллекций, но не для результатов поиска.

Необходимо также отметить, что в большинстве систем поддерживается расширенный поиск, позволяющий сформулировать сложный запрос, учитывающий разные параметры результатов. В сводной таблице мы использовали только три значения: *нет*, если фасеты не поддерживаются, *год*, если доступен только фасет по году публикации, *расширенные*, если в системе доступен более широкий набор фасетов.

Иерархические классификаторы в системах научного поиска используются только для структурирования направлений исследований. Картина здесь достаточно пестрая, и сравнивать разные системы между собой трудно. Можно разделить системы на адаптированные к конкретному домену, такие как PubMed или ACM DL, и универсальные, такие как Google Scholar или eLibrary. Адаптированные к домену системы, как правило, предоставляют более богатые классификаторы, а универсальные – менее богатые. С другой стороны, в некоторых системах используются классификаторы, сделанные вручную (как например MeSH в PubMed), а некоторые строят классификатор автоматически, как, например, MS Academic. Кроме того, польза от использования классификатора зависит от того, каким образом к нему привязываются статьи. К сожалению, ответы на эти вопросы как правило недокументированы, поэтому в этом случае мы ограничились классификацией на три класса: *нет* классификатора, *доменный* классификатор, *универсальный* классификатор.

Фолксономия существующими системами поддерживается слабо, поддержка есть только в aMiner и специальных системах, таких как Bibsonomy. В Google Scholar можно пользоваться приватными тегами (что, строго говоря, нельзя называть фолксономией). Как ни странно, поддержки тегов нет ни в одной из систем, которые можно считать социальными сетями. Ни в Academia.edu, ни в Mendeley, ни в ResearchGate поддержки фолксономии нам обнаружить не удалось.

В рассмотренных системах, как правило, применяются достаточно простые виды визуализаций или не применяется никаких. Наиболее частый вариант – столбчатая диаграмма по годам. Встречаются и более сложные варианты: столбчатые диаграммы по каждому значению фасета (Springerlink), диаграммы влияния авторов (Semantic Scholar). Microsoft Academic Search и CiteseerX полностью пренебрегают визуализацией (хотя Microsoft Academic Search на странице часто задаваемых вопросов обещает вернуть визуализацию графа цитирования из предыдущей версии системы). Semantic Scholar предоставляет гистограммы по годам для результатов поиска статей и для цитирований статей и авторов на соответствующих страницах. aMiner на странице автора предоставляет диаграмму изменения интересов автора по годам.

Куда более мощные средства визуализации больших объемов статей, которые могут быть получены в результате поиска по ключевым словам, предоставляют средства, разрабатываемые в области картографирования науки, такие как Citespace, Sci2Sci, SciMat. Эти системы берут на вход данные о статьях и их цитированиях и позволяют анализировать заданную таким образом предметную область. Один из основных сценариев работы с такими системами заключается в выполнении поиска, например, в Scopus, экспорта результатов и затем их анализа. К сожалению, автоматически получить существенное количество результатов поискового запроса в большинстве поисковых систем невозможно и далеко не все из них предоставляют информацию о цитированиях. Кроме того, при таком сценарии использования отсутствует возможность интерактивного взаимодействия с поисковой системой. После уточнения запроса необходимо снова вручную выполнять экспорт результатов поиска и заново начинать анализ. С другой стороны, средства анализа набора статей, реализованные в системах картографирования науки, существенно богаче всего, что реализовано в рассмотренных поисковых системах. Поддерживается и графовая кластеризация, и автоматическая генерация меток для них, и различные способы визуализации результатов анализа.

Подводя итог этому подразделу отметим, что основным механизмом описания результатов поиска являются фасеты, дополненные иерархическими классификаторами областей исследования. Более мощные средства предоставляются оффлайновыми инструментами картографирования науки, которые поддерживают и кластеризацию (в том числе с автоматической генерацией меток), и графическую визуализацию всех результатов с учетом автоматически вычисляемой важности отдельный статей. Однако оффлайновые системы не позволяют переформулировать поисковый запрос и не имеют доступа к полной базе статей, вынуждая пользователей постоянно проходить цикл поиска, экспорта данных, автоматического анализа и снова поиска с уточненными параметрами, переключаясь между двумя системами.

5.3. Инструменты поддержки длительного поиска в современных системах поиска результатов научных исследований

Рассматривая инструменты поддержки длительного поиска, реализованные в исследуемых системах, мы выделили четыре группы: поддержка совместной работы, подписка на обновления, история поиска и поддержка рабочих мест.

Поддержка совместного проведения исследований в рассмотренных системах реализована слабо. Только в трех системах удалось обнаружить функции, предназначенные для работы в команде. BibSonomy позволяет создавать группы. Каждый член группы, взаимодействуя с системой – создавая новый пост, или делясь ссылкой, может ограничить видимость своего действия конкретной группой. ResearchGate позволяет создавать так называемые

проекты, в которые можно добавить нескольких участников, оставить список релевантной литературы и список вопросов, на которые нужно получить ответы. Кроме того, в рамках проекта можно вести ленту обновлений. На проект можно подписаться и получать уведомления об изменениях. Mendeley тоже позволяет создавать группы, похожие по своей функциональности на проекты в ResearchGate. В табл. 4 мы использовали два значения: *общие проекты* для трех описанных выше систем и *нет* для всех остальных. Ни одна система не поддерживает собственно одновременный групповой поиск в духе SearchTogether.

Следующая группа инструментов для поддержки длительного поиска – это подписки. Основной вариант использования для подписок – поддержание осведомленности о состоянии предметной области. Рассмотренные системы поиска для ученых поддерживают следующие виды подписок в разных сочетаниях:

1. подписки на новые статьи выбранных авторов (*авторы*);
2. подписки на новые цитирования выбранных статей (*цитирования*);
3. подписки на новые результаты поисковых запросов (*запросы*).

Есть некоторые индивидуальные особенности: Semantic Scholar не позволяет подписываться на результаты поиска, но позволяет подписаться на обновления одной из поддерживаемых областей исследований, Google Scholar позволяет подписаться на цитирования любой статьи выбранного автора. Если достаточно тщательно настроить все виды подписок одновременно, то можно быть почти уверенным, что никакие новые статьи по интересующим вас темам не будут упущены. Для известных авторов можно подписаться на все их новые статьи. Чтобы получать статьи не только известных авторов, можно подписаться на все цитирования важных в области статей. Также можно подписаться на все цитирования собственных статей, чтобы отслеживать обратную связь на свои исследования. Наконец, можно подписаться на обновления результатов поисковых запросов, чтобы иметь возможность заметить новые исследования в смежных областях, которые могут не процитировать важные статьи, на которые вы уже подписаны.

Следующий инструмент, который мы рассмотрим, более важен для исследования новой области. Это истории поисковых запросов.

Процесс изучения новой предметной области можно сравнить с поиском в пространстве состояний. Пользователь начинает с некоторых поисковых запросов, просматривает результаты поиска на некоторую глубину, делает уточняющие запросы, когда хочет исследовать некоторую подзадачу полнее, и возвращается назад, когда хочет вернуться к основной задаче. История поисковых запросов должна помогать в этом процессе, позволять быстро переключаться между анализом результатов разных поисковых запросов. Многие поисковые системы позволяют создавать сложные запросы с большим количеством ограничений по разным параметрам статей. Создание таких

запросов может быть достаточно трудоемким и важно, чтобы, однажды составив их, пользователь мог легко к ним вернуться.

Табл. 4. Инструменты поддержки длительного поиска в современных системах поиска результатов научных исследований
Tab. 4. Tools to support long search in modern systems for search of research results

Система	Совместная работа	Подписки	Истории	Рабочие места
aMiner	нет	авторы	автоматически	теги
CiteseerX	нет	цитирования	нет	несколько папок
Google Scholar	нет	авторы, запросы, цитирования	вручную	теги
MS Academic	нет	авторы, цитирования	нет	одна папка
Semantic Scholar	нет	авторы, цитирования	нет	одна папка
Academia.edu	нет	авторы	автоматически	одна папка
BibSonomy	общие проекты	нет	нет	теги
Mendeley	общие проекты	авторы	нет	несколько папок
ResearchGate	общие проекты	авторы, цитирования	нет	несколько папок
ACM DL	нет	нет	вручную	несколько папок
arXiv.org	нет	нет	нет	нет
eLibrary	нет	нет	автоматически	несколько папок
PubMed	нет	авторы, запросы	автоматически	несколько папок
ScienceDirect	нет	нет	нет	нет
Scopus	нет	авторы, запросы, цитирования	вручную	несколько папок
Springerlink	нет	запросы	нет	нет
WOS	нет	авторы, запросы, цитирования	вручную	несколько папок

Частично сохранение истории поисковых запросов может быть реализовано средствами браузера, возможно поэтому около половины рассмотренных систем никак не реализуют эту функциональность. Тем не менее, возможностей браузера может быть недостаточно (как минимум, в истории браузера не сохранятся POST-запросы) и многие системы позволяют сохранять историю. Мы выделили два варианта поддержки истории:

1. ручной режим, когда пользователь должен самостоятельно сообщить системе, что конкретный поисковый запрос должен быть сохранен;
2. автоматический режим, когда система сама запоминает все запросы пользователя и позволяет ему просматривать эту историю и возвращаться к ранее сделанным запросам.

К сожалению, ни в одной из систем нам не удалось обнаружить возможности вернуться к конкретной странице результатов поиска, чтобы продолжить с того места, где работа была прервана. Таким образом, в этой части остается очевидное пространство для дальнейших улучшений.

Наконец, последний вид инструментов поддержки длительного поиска – это поддержка рабочих мест.

Ядром рабочего места пользователя в поисковых системах для научных публикаций является отобранный им список литературы, т.е. промежуточный результат выполняемого им длительного поиска. Поддержка рабочего места реализована не везде, а где есть, то обеспечивается в достаточно примитивном виде: это либо единый список статей, либо с разбивкой по папкам, либо с разбивкой по тегам. Единый список статей может быть полезен, только если ученый работает ровно в одной, достаточно узкой области или, например, использует систему для написания одной статьи. Когда количество отобранных статей начинает приближаться к сотне, единый список не может оставаться удобным и требуется какая-то возможность его рубрикации.

Очевидный способ – разделение списка на части аналогично разделению файлов по папкам. Этот способ организации наиболее популярен и поддерживается в половине рассмотренных систем. Несколько папок для отобранных статей удобнее, чем один плоский список, но остается существенное ограничение: каждая статья может быть сохранена только в одной папке, что часто неудобно.

Наиболее гибкий способ организации хранения отобранных статей – это тэги. Каждую статью можно пометить произвольным набором тегов и затем легко отбирать подмножества статей с заданными тэгами, в некотором смысле динамически формируя виртуальные папки. Google Scholar поддерживает поиск по статьям из библиотеки, что может быть очень удобно для поиска с целью обучения, когда надо найти уже встречавшуюся ранее статью.

Подытоживая этот подраздел, мы вынуждены констатировать, что хотя многие рассмотренные системы достаточно хорошо поддерживают поиск с целью поддержки осведомленности (в первую очередь за счет подписок), исследование новой предметной области поддерживается значительно хуже. Только в социальных сетях для ученых есть некоторые достаточно ограниченные средства организации командной работы, остальные поисковые системы не поддерживают ее вовсе. Поддержка рабочих мест также оставляет желать лучшего, следует отметить разве что Google Scholar с его возможностями организации статей с помощью тегов и локальному поиску среди отобранных статей.

6. Выводы и дискуссия

Мы предприняли попытку объединить результаты исследований поискового поведения ученых с наработками, полученными при изучении

исследовательского поиска. Исторически исследовательский поиск определялся, отталкиваясь от традиционного поиска и в противовес ему. Определение исследовательского поиска достаточно размыто, и многие инструменты, относимые к исследовательскому поиску, сегодня внедрены в традиционные поисковые системы.

Мы придерживались определения исследовательского поиска, данного в работе Уайта и Рота [2]: исследовательский поиск – это поиск, контекст проблемы которого является открытым, постоянным и многоаспектным, а процесс поиска является оппортунистическим, итеративным и мультитактическим. Опираясь на это определение, мы проанализировали основные задачи поиска, возникающие перед учеными, и обнаружили, что большая их часть может быть уверено отнесена к исследовательскому поиску.

Поисковое поведение ученых исследуется не первое десятилетие. Мы кратко проследили эти исследования, начиная от работ Эллиса и др. [4-5] и заканчивая достаточно современной работой Атукоралы и др. [10], в которой исследовалось поисковое поведение ученых в области компьютерных наук и то, как они используют современные электронные поисковые системы.

Обнаружив, что поисковое поведение ученых в большинстве случаев относится к исследовательскому поиску, мы выяснили, какие существуют инструменты, предназначенные для исследовательского поиска, и выделили среди них три группы:

1. инструменты помощи при формулировании запроса;
2. инструменты помощи при анализе результатов;
3. инструменты поддержки длительного поиска.

Для каждой из групп мы кратко описали относящиеся к ней инструменты.

Наконец, мы рассмотрели ряд специализированных поисковых систем для ученых, чтобы выяснить, насколько широко в них применяются инструменты исследовательского поиска и какие именно. Мы проанализировали адаптации универсальных поисковых систем к домену научных публикаций: Google Scholar и Microsoft Academic Search; экспериментальные академические разработки: aMiner, CiteseerX, SemanticScholar; поисковые системы, интегрированные в электронные библиотеки: ACMDL, arXiv.org, eLibrary, ScienceDirect, Scopus, SpringerLink, WOS; социальные сети для ученых: Academia.edu, BibSonomy, Mendeley, ResearchGate. Основная часть анализа поисковых систем была проведена в конце 2017 – начале 2018 года и затем актуализирована в конце 2018 года.

Из проведенного анализа можно сделать несколько выводов. И наиболее частые, и наиболее трудоемкие поисковые потребности ученых связаны с исследовательским поиском. При этом специализированные поисковые системы для ученых поддерживают проведение исследовательского поиска лишь частично.

Инструменты помощи при составлении запроса почти нигде не продвинулись дальше автодополнения, а во многих системах нет даже и этого. Таким

образом, порог входа остается высоким, и неспециалисту придется потратить существенное время на то, чтобы подобрать запрос, соответствующий его потребностям.

Среди инструментов обзора результатов поиска наиболее развиты фасеты, с помощью которых опытный ученый сможет эффективно отобрать интересующие его статьи. Однако инструменты визуализации сравнительно примитивны и сильно уступают реализованным в инструментах картографирования науки. Насколько можно судить, не реализованы ни методы кластеризации, ни анализ графа цитирования, ни построение текстовых описаний.

Методы поддержки длительного поиска в лучшем случае сводятся к возможности откладывать релевантные статьи в именованные коллекции, аналогичные папкам на файловой системе. Кроме того, можно подписываться на новые статьи избранных авторов, цитирования избранных статей и обновления результатов заранее подобранных поисковых запросов. Этих инструментов в целом достаточно для реализации сценария поиска с целью поддержки осведомленности, но для групповой работы над одной задачей придется использовать общий аккаунт и внешние средства коммуникации.

Таким образом, наиболее перспективными направлениями для дальнейшей работы выглядят исследования по следующим направлениям:

1. понижение порога входа для начинающих исследователей: реализация специфических видов запросов, прежде всего рекомендаций;
2. интеграция методов картографирования науки для обзора результатов поиска;
3. интеграция методов поддержки совместного поиска.

Список литературы

[1]. G. Marchionini. Exploratory search: from finding to understanding. Communications of the ACM, vol. 49, no. 4, 2006, pp. 41–46.

[2]. R.W. White and R.A. Roth. Exploratory search: Beyond the query-response paradigm. Synthesis lectures on information concepts, retrieval, and services, Morgan and Claypool Publishers, 2009, 98 p.

[3]. G. Marchionini. Information seeking in electronic environments. Cambridge University Press, 1997, 240 p.

[4]. D. Ellis, D. Cox, and K. Hall. A comparison of the information seeking patterns of researchers in the physical and social sciences. Journal of documentation, vol. 49, no. 4, 1993, pp. 356–369.

[5]. D. Ellis. A behavioural approach to information retrieval system design. Journal of documentation, vol. 45, no. 3, 1989, pp. 171–212.

[6]. X. Niu, B. M. Hemminger, C. Lown, S. Adams, C. Brown, A. Level, M. McLure, A. Powers, M. R. Tennant, and T. Cataldo. National study of information seeking behavior of academic researchers in the United States. Journal of the Association for Information Science and Technology, vol. 61, no. 5, 2010, pp. 869–890.

[7]. X. Niu and B. M. Hemminger. A study of factors that affect the information-seeking behavior of academics cientists. Journal of the American Society for Information Science and Technology, vol. 63, no. 2, 2012, pp. 336–353.
[8]. L.I. Meho and H.R. Tibbo. Modeling the information-seeking behavior of social scientists: Ellis's study revisited. Journal of the Association for Information Science and Technology, vol. 54, no. 6, 2003, pp. 570–587.
[9]. E. Palagi, F. Gandon, A. Giboin, and R. Troncy. A survey of definitions and models of exploratory search. In Proc. of the 2017 ACM Workshop on Exploratory Search and Interactive Data Analytics, 2017, pp. 3–8.
[10]. K. Athukorala, E. Hoggan, A. Lehtio, T. Ruotsalo, and G. Jacucci. Information-seeking behaviors of computer scientists: Challenges for electronic literature search tools. In Proc. of the Proceedings of the 76th ASIS&T Annual Meeting of Association for Information Science and Technology, vol. 50, 2013, pp. 1–11.
[11]. T. Jiang. Exploratory search: a critical analysis of the theoretical foundations, system features, and research trends. In Library and Information Sciences, Springer, 2014, pp. 79–103.
[12]. А.О. Янина, К.В. Воронцов. Мультимодальные тематические модели для разведочного поиска в коллективном блоге. Машинное обучение и анализ данных, том 2, вып. 2, 2016, стр. 173-186.
[13]. J. Beel, B. Gipp, S. Langer, and C. Breitinger. Research-paper recommender systems: a literature survey. International Journal on Digital Libraries, vol. 17, no. 4, 2016, pp. 305–338.
[14]. П.А. Пархоменко, А.А. Григорьев, Н.А. Астраханцев. Обзор и экспериментальное сравнение методов кластеризации текстов. Труды ИСП РАН, том. 29, вып. 2, 2017, стр. 161–200. DOI: 10.15514/ISPRAS-2017-29(2)-6.
[15]. H. Tian and H. H. Zhuo. Paper2vec: Citation-context based document distributed representation for scholar recommendation. arXiv preprint arXiv:1703.06587, 2017.
[16]. X. Kong, M. Mao, W. Wang, J. Liu, and B. Xu. Voprec: Vector representation learning of papers with text information and structural identity for recommendation. IEEE Transactions on Emerging Topics in Computing, 2018.
[17]. M.R. Morris and E. Horvitz. Searchtogether: an interface for collaborative web search. In Proc. of the 20th Annual ACM symposium on User interface software and technology, 2007, pp. 3–12.
[18]. M.R. Morris. Collaborative search revisited. In Proc. of the 2013 Conference on Computer supported cooperative work. ACM, 2013, pp. 1181–1192.
[19]. C. Chen, F. Ibekwe-SanJuan, and J. Hou. The structure and dynamics of cocitation clusters: A multiple-perspective cocitation analysis. Journal of the American Society for information Science and Technology, vol. 61, no. 7, 2010, pp. 1386–1409.
[20]. L. McCay-Peet, A. Quan-Haase, and D. Kern. Exploratory search in digital libraries: a preliminary examination of the use and role of interface features. In Proc. of the 78th Annual Meeting of Association for Information Science and Technology, vol. 52, 2015, pp. 1–4.
[21]. H. Li, I. Councill, W.-C. Lee, and C.L. Giles. Citeseerx: an architecture and web service design for an academic document search engine. In Proc. of the 15th International Conference on World Wide Web, 2006, pp. 883–884.
[22]. J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In Proc. of the 14th ACM SIGKDD International conference on Knowledge discovery and data mining, 2008, pp. 990–998.

[23]. W. Ammar, D. Groeneveld, C. Bhagavatula et al. Construction of the literature graph in semantic scholar. arXiv preprint arXiv:1805.02262, 2018.

[24]. H. Zaugg, R.E. West, I. Tateishi, and D.L. Randall. Mendeley: Creating communities of scholarly inquiry through research collaboration. TechTrends, vol. 55, no. 1, 2011, pp. 32–36.

Exploratory search for scientific articles

[1]*Y.R. Nedumov <yaroslav.nedumov@ispras.ru>*
[1,2,3,4] *S.D. Kuznetsov <kuzloc@ispras.ru>*
[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia*
[3] *Moscow Institute of Physics and Technology (State University) 9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia*
[4] *National Research University Higher School of Economics (HSE) 11 Myasnitskaya Ulitsa, Moscow, 101000, Russia*

Abstract. It is intuitively clear that the search for scientific publications often has many characteristics of a research search. The purpose of this paper is to formalize this intuitive understanding, explore which research tasks of scientists can be attributed to research search, what approaches exist to solve a research search problem in general, and how they are implemented in specialized search engines for scientists. We researched existing works regarding information seeking behavior of scientists and the special variant of a search called exploratory search. There are several types of search typical for scientists, and we showed that most of them are exploratory. Exploratory search is different to information retrieval and demands special support from search systems. We analyzed seventeen actual search systems for academicians (from Google Scholar, Scopus and Web of Science to ResearchGate) from the exploratory search support aspect. We found that most of them didn't go far from simple information retrieval and there is a room for further improvements especially in the collaborative search support.

Keywords: exploratory search; academic search engines; information seeking behaviour

DOI: -10.15514/ISPRAS-2018-30(6)-10

For citation: Nedumov Y.R., Kuznetsov S.D. Exploratory search for scientific articles. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 171-198 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-10

References

[1]. G. Marchionini. Exploratory search: from finding to understanding. Communications of the ACM, vol. 49, no. 4, 2006, pp. 41–46.

[2]. R.W. White and R.A. Roth. Exploratory search: Beyond the query-response paradigm. Synthesis lectures on information concepts, retrieval, and services, Morgan and Claypool Publishers, 2009, 98 p.

[3]. G. Marchionini. Information seeking in electronic environments. Cambridge University Press, 1997, 240 p.
[4]. D. Ellis, D. Cox, and K. Hall. A comparison of the information seeking patterns of researchers in the physical and social sciences. Journal of documentation, vol. 49, no. 4, 1993, pp. 356–369.
[5]. D. Ellis. A behavioural approach to information retrieval system design. Journal of documentation, vol. 45, no. 3, 1989, pp. 171–212.
[6]. X. Niu, B. M. Hemminger, C. Lown, S. Adams, C. Brown, A. Level, M. McLure, A. Powers, M. R. Tennant, and T. Cataldo. National study of information seeking behavior of academic researchers in the United States. Journal of the Association for Information Science and Technology, vol. 61, no. 5, 2010, pp. 869–890.
[7]. X. Niu and B. M. Hemminger. A study of factors that affect the information-seeking behavior of academics cientists. Journal of the American Society for Information Science and Technology, vol. 63, no. 2, 2012, pp. 336–353.
[8]. L.I. Meho and H.R. Tibbo. Modeling the information-seeking behavior of social scientists: Ellis's study revisited. Journal of the Association for Information Science and Technology, vol. 54, no. 6, 2003, pp. 570–587.
[9]. E. Palagi, F. Gandon, A. Giboin, and R. Troncy. A survey of definitions and models of exploratory search. In Proc. of the 2017 ACM Workshop on Exploratory Search and Interactive Data Analytics, 2017, pp. 3–8.
[10]. K. Athukorala, E. Hoggan, A. Lehtio, T. Ruotsalo, and G. Jacucci. Information-seeking¨ behaviors of computer scientists: Challenges for electronic literature search tools. In Proc. of the Proceedings of the 76th ASIS&T Annual Meeting of Association for Information Science and Technology, vol. 50, 2013, pp. 1–11.
[11]. T. Jiang. Exploratory search: a critical analysis of the theoretical foundations, system features, and research trends. In Library and Information Sciences, Springer, 2014, pp. 79–103.
[12]. A. Yanina, K. Vorontsov. Multimodal topic modeling for exploratory search in collective blog. In Journal of Machine Learning and Data Analysis, vol. 2, no. 2, 2016, pp. 173-186 (in Russian).
[13]. J. Beel, B. Gipp, S. Langer, and C. Breitinger. Research-paper recommender systems: a literature survey. International Journal on Digital Libraries, vol. 17, no. 4, 2016, pp. 305–338.
[14]. P.A. Parhomenko, A.A. Grigorev, N.A. Astrakhantsev A survey and an experimental comparison of methods for text clustering: application to scientific articles. Trudy ISP RAN/Proc. ISP RAS, 2017, vol. 29, issue 2, pp. 161-200 (in Russian). DOI: 10.15514/ISPRAS-2017-29(2)-6
[15]. H. Tian and H. H. Zhuo. Paper2vec: Citation-context based document distributed representation for scholar recommendation. arXiv preprint arXiv:1703.06587, 2017.
[16]. X. Kong, M. Mao, W. Wang, J. Liu, and B. Xu. Voprec: Vector representation learning of papers with text information and structural identity for recommendation. IEEE Transactions on Emerging Topics in Computing, 2018.
[17]. M.R. Morris and E. Horvitz. Searchtogether: an interface for collaborative web search. In Proc. of the 20th Annual ACM symposium on User interface software and technology, 2007, pp. 3–12.
[18]. M.R. Morris. Collaborative search revisited. In Proc. of the 2013 Conference on Computer supported cooperative work. ACM, 2013, pp. 1181–1192.

[19]. C. Chen, F. Ibekwe-SanJuan, and J. Hou. The structure and dynamics of cocitation clusters: A multiple-perspective cocitation analysis. Journal of the American Society for information Science and Technology, vol. 61, no. 7, 2010, pp. 1386–1409.

[20]. L. McCay-Peet, A. Quan-Haase, and D. Kern. Exploratory search in digital libraries: a preliminary examination of the use and role of interface features. In Proc. of the 78th Annual Meeting of Association for Information Science and Technology, vol. 52, 2015, pp. 1–4.

[21]. H. Li, I. Councill, W.-C. Lee, and C.L. Giles. Citeseerx: an architecture and web service design for an academic document search engine. In Proc. of the 15th International Conference on World Wide Web, 2006, pp. 883–884.

[22]. J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In Proc. of the 14th ACM SIGKDD International conference on Knowledge discovery and data mining, 2008, pp. 990–998.

[23]. W. Ammar, D. Groeneveld, C. Bhagavatula et al. Construction of the literature graph in semantic scholar. arXiv preprint arXiv:1805.02262, 2018.

[24]. H. Zaugg, R.E. West, I. Tateishi, and D.L. Randall. Mendeley: Creating communities of scholarly inquiry through research collaboration. TechTrends, vol. 55, no. 1, 2011, pp. 32–36.

Методы анализа информационных потоков в сети Интернет

[1,2] *Аветисян А. А.<a.a.avetisyan@ispras.ru>*
[1,3] *Дробышевский М. Д.<drobyshevsky@ispras.ru>*
[1,2,4] *Турдаков Д. Ю.<turdakov@ispras.ru>*

[1]*Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2]*Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1*
[3]*Московский физико-технический институт (государственный университет), 141701, Московская область, г. Долгопрудный, Институтский переулок, д.9*
[4] *НИУ "Высшая школа экономики", 101000, Россия, г. Москва, ул. Мясницкая, д. 20*

Аннотация. Распространение информации – это фундаментальный процесс, происходящий в сети Интернет. Ежедневно мы можем наблюдать публикацию различной информации и ее дальнейшее распространение через новостные агентства и сообщения обычных пользователей. И хотя сам процесс можно наблюдать явно, определить отдельные пути передачи очень сложно. Проникновение глобальной информационной среды во все сферы жизни человечества радикально меняет скорость и пути распространения информации. В этом обзоре мы исследуем модели распространения информационных потоков в сети Интернет, разделяя их на две группы: объяснительные, предполагающие наличие сети влияния между информационными узлами, и предсказательные, ставящие своей задачей изучение распространения отдельных частей информации. Несмотря на всю сложность, изучение глубинных свойств распространения информации необходимо для понимания общих процессов, происходящих в современном информационном обществе.

Ключевые слова: распространение информации, информационные каскады, пути распространения информации

DOI: 10.15514/ISPRAS-2018-30(6)-11

Для цитирования: Аветисян А.А., Дробышевский М.Д., Турдаков Д.Ю. Методы анализа информационных потоков в сети Интернет. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 199-220. DOI: 10.15514/ISPRAS-2018-30(6)-11

1. Введение

Ежедневно в сети Интернет происходят различные процессы распространения информации: от публикации статей новостными агентствами до обмена сообщениями обычных пользователей. Учитывая огромное влияние распространения информации на жизнь современного общества, анализ и моделирование информационных потоков имеют важное значение для понимания происходящих процессов и изучения способов влияния на них. Высокоточные масштабные модели распространения и эволюции онлайновой информации необходимы как для анализа стратегических кампаний в информационном пространстве, так и для предоставления критически важной информации населению в ходе операций по оказанию помощи при бедствиях, а также могут потенциально способствовать решению других критических задач в онлайновой информационной области. При этом возникает ряд задач, связанных со сбором данных из разнородных источников, их анализом и построением математических моделей. Например, если появление новостей или сообщений можно наблюдать явно, то определить, кто является источником, в каждом случае часто затруднительно. Предположение о возможных сетях влияния можно сделать, собрав данные из блогов или новостных статей и проанализировав времена появления похожих сообщений в различных источниках [1].

Основные платформы для распространения информационных потоков включают: системы мгновенного обмена сообщениями, приложения для ведения блогов и микро-блогов, электронную почту, мобильные сети коммуникации и другие приложения, большинство которых используются на смартфонах [2]. При этом виды информации и способы ее распространения можно разделить на несколько категорий [2].

- Широковещание – кроме традиционного радио и телевидения, активно используются социальные медиа для маркетинговых кампаний и продвижений, рассылки предупреждающих сообщений, привлечения пользователей и так далее.
- Обмен контентом в онлайновых соцсетях (sharing), доступный каждому пользователю: посты, репосты, ретвиты и другие, – согласно исследованию на основе Фейсбука [3], главными мотивами становятся «получение удовольствия» и «раскрытие информации, необходимой/полезной для извлечения выгоды из Фейсбук».
- Краудсорсинг – предполагает привлечение широких групп людей для решения задач путем обмена идеями и контентом, особенно внутри онлайн-сообществ.
- С развитием соцсетей активно используется вирусный маркетинг (viral marketing), целью которого является продвижение товаров и услуг среди пользователей путем распространения рекламы в видео, изображениях, блогах, текстовых сообщениях при условии передачи их между людьми.

В результате упомянутых процессов происходит формирование мнений (opinion formation) как у отдельных индивидов, так и у целых сообществ относительно новостей, политических событий или новых продуктов [4]. Аналогичным образом происходит распространение инноваций (innovation diffusion), например, в результате перенятия людьми идей от друзей в социальной сети [5].

- Отдельно стоит выделить распространение вредоносной информации (malicious spreading), такой как компьютерные вирусы, вредоносная реклама, слухи и сплетни. Благодаря высокой скорости передачи через Интернет сети, массовое заражение компьютерными вирусами представляют серьезную опасность [6], а распространение дезинформации может иметь неприятные последствия для общества [7].

Данная статья посвящена обзору методов анализа информационных потоков в сети Интернет, разработанных за последние годы. В разд. 2 приведены важные определения и основные свойства информационных потоков. Далее, в разд. 3 описаны методы и модели, в соответствии с разными аспектами задачи. В разд. 4 говорится о приложениях этих моделей, и затем следует заключение.

2. Основные свойства информационных потоков

Центральным понятием при изучении информационных потоков является **информационный каскад**. В сети появляется новая единица информации – сообщение, вброс, новость или вирус – которая на некоторый ограниченный период времени вызывает повышенное внимание. Тема последовательно появляется в различных узлах сети, захватывая таким образом все больший круг узлов и образуя каскад. Обычно отдельный каскад моделируется деревом, корнем в котором становится пользователь, первым опубликовавший сообщение, а информация распространяется от родителя к потомкам. Однако в других случаях могут быть несколько входящих ребер, например, когда один пользователь агрегирует множество рекомендаций.

2.1 Временн*ые* динамические свойства

Наиболее общей характеристикой информационных потоков служит последовательность их появления, бурного роста и спада (the rise and fall pattern), притом, что разные темы непрерывно сменяют друг друга. Это проиллюстрировано на рис. 1, где приведены 50 наиболее масштабных новостных тем за период с 1 августа по 31 октября 2008 года, собранных с медийных сайтов Google News (20,000 сайтов) и 1.6 млн блогов, форумов и других сайтов. В первых попытках описания динамических паттернов используется пуассоновский процесс, а функции роста и падения приближаются с помощью степенного закона с экспонентой от -0.1 до -2.5 [9]. При анализе хэштегов Твиттера было выявлено несколько классов временных паттернов, выражающих активность до и во время пика, во время и после

пика, симметрично по обе стороны от пика и на следующий день после пика [10]. Паттерн взлета и падения также может быть описан стохастическими дифференциальными уравнениями при ряде допущений и упрощений [11].

Однако динамические свойства информационных потоков пока остаются мало изученными, что сильно препятствует возможности предсказания каскадов.

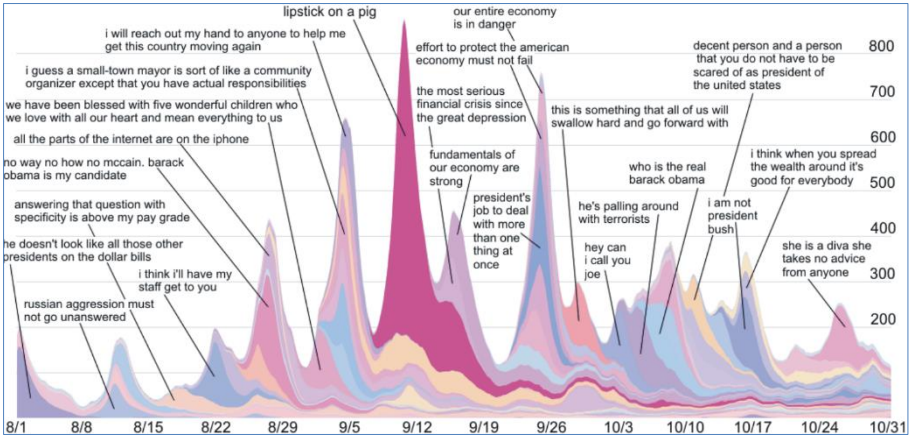


Рис. 1. 50 наиболее масштабных потоков в новостных циклах за период 1 августа – 31 октября 2008 года [8]. Каждый поток состоит из всех новостных заметок и постов в блогах, содержащих в своем тексте определенные фразы (некоторые приведены на рисунке)

Fig. 1. 50 most large-scale threads in news cycles for the period August 1 – October 31, 2008 [8]. Each thread consists of all news notes and blog posts that contain certain phrases in their text (some are shown in the figure)

Большое разнообразие паттернов обусловлено сложным устройством социальных систем. С одной стороны, это тип контента: было показано наличие существенных различий в характере распространения хэштегов в Твиттере в зависимости от их темы [12], а также в динамике каскадов в Фейсбуке в зависимости от категории распространяемой новости [13].

Вторым фактором является среда распространения, наиболее важную роль здесь играет структура лежащей в основе сети. В целом, пользователи с большим количеством друзей/подписчиков имеют большее влияние как на скорость распространения, так и на популярность передаваемой информации [14].

Также важную роль играют и другие свойства графа: распределение степеней, эффект ``малого мира'', структура эго-сетей, сила связей, идеологическая гомофилия при образовании связей. Например, было обнаружено, что, хотя сильные связи оказываются влиятельными локально, более многочисленные слабые связи ответственны за передачу новой информации [15].

Еще бо́льшую трудность несет тот факт, что топология сетей непостоянна во времени, ребра могут менять свою активность в рассматриваемых процессах, а

также само распространение информации оказывает влияние на эволюцию связей в сети.

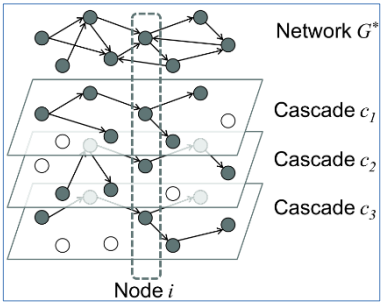


Рис. 2. Сеть распространения информации, полученная объединением путей передачи индивидуальных каскадов [1]
Fig. 2. Information diffusion network obtained by combining the paths of transmission of individual cascades

2.2 Структурные свойства

Имея данные о множестве информационных каскадов для одних и тех же участников, можно построить общую сеть распространения (diffusion network), объединив индивидуальные пути передачи из отдельных каскадов (рис. 2). В таких сетях экспериментально исследуются частота встречаемости каскадов определенной структуры и распределение их размеров. Несмотря на существование «вирусных» каскадов, достигающих огромных размеров, при анализе различных доменов – от коммуникационных платформ до игровых сетей и микроблогов – было обнаружено, что подавляющая доля (73-95%) каскадов состоит лишь из одного узла, то есть распространения не происходит [16]. Распределения и размера каскада, и его глубины (высоты дерево), измеренные для разных платформ (Facebook, Digg, Sina Weibo, Slashdot и других), показывают перекос в сторону каскадов малого размера и малой глубины. Обычно полагают, что обе зависимости приблизительно описывается степенным законом. Таким образом, большие каскады представлены относительно редко.

Для характеристики структуры каскада была предложена структурная виральность (structural virality, известная также как Wiener [17]):

$$v = \frac{1}{n(n-1)} \sum_i \sum_j d_{ij}$$

выражающая среднее расстояние d_{ij} между всеми парами вершин каскада. Низкое значение виральности говорит о распространении через узлы-хабы (больше в ширину), а высокое значение – о наличии длинных путей (больше в

глубину). Для многих систем были показаны низкие значения виральности вместе с низкой корреляцией значения виральности и размера сети распространения.

В [18] предлагается индекс потенциала виральности (Virality Potential Index), учитывающий для каждого узла m сети долю его потомков $Follow(m)$, которые участвуют в каскаде: $Infected(m)$. В предположении, что вероятность передачи следующим потомкам одинакова и равна «потенциалу» передачи $\frac{Infected(m)}{Follow(m)}$, индекс определен как минус логарифм от вероятности для всего дерева:

$$VPI(m) = \sum_{m} -|Infected(m)| \, log \frac{|Infected(m)|}{|Follow(m)|}$$

Индекс минимален, когда эта доля равна нулю (никто ничего не передал) или единице (нет потенциала передать больше). Индекс для каскада считается как средний индекс по всем его элементам m.

Кроме распространения информации в сети от одного узла к другому, имеет место приток из внешних источников, таких как масс-медиа, новостные сайты и телевидение. Например, анализ упоминаний URL в Твиттере показал, что на P2P взаимодействия приходится около 71% объема информации [19], а на популярность хэштегов внешние источники оказывают даже большее влияние [10].

2.3 Свойства поведения пользователей

Разнообразное поведение пользователей в социальной сети также будет генерировать различные шаблоны распространения информации. Влияние поведения пользователей может зависеть от разных факторов. Пользователи с одинаковыми предпочтениями с большей вероятностью будут взаимодействовать друг с другом, что существенно влияет на распространение информации. Анализ временны*ы*х признаков в системе мобильной связи выявляет, что социальное взаимодействие более часто встречается у лиц с аналогичными признаками, такими как пол и возраст. В работе [20] авторы наблюдают идеологическую связь в сетях друзей, основываясь на данных Facebook, где консерваторы/либералы с большей вероятностью дружатся с человеком с той же политической принадлежностью. Помимо влияния на структуру социальной сети, интерес пользователей или предпочтение также будет напрямую влиять на распространение информации. Как правило, пользователи хотели бы оценивать и передавать информацию, которая согласуется с их интересами, а разные предпочтения обычно интерпретируются как вероятность распространения. И факторы среды распространения, такие как информационный контент, социальное влияние или даже эмоции, будут существенно влиять на предпочтения пользователей.

Одним из факторов влияния поведения пользователей является динамическое событие с временными увеличениями интереса. Обычно поведение индивидов демонстрирует временный всплеск, где часто появляющиеся события наблюдаются в течение очень коротких периодов, за которыми следуют длительные периоды бездействия. Васкес и др. [21] анализируют распространения почтовых червей среди пользователей электронной почты. Предполагается, что временной интервал между двумя последовательными сообщениями, отправленными одним и тем же пользователем почты, является непуассоновским процессом. Эта гипотеза согласуется с наблюдаемыми процессами распространения вирусов, которые показывают время угасания, близкое к году, в отличие от однодневного угасания, предсказанного моделями, основанными на пуассоновском процессе.

2.4 Другие закономерности

Определение силы влияния для каждого из узлов дает увидеть некоторые закономерности в распространении информации. Например, после определения функции влияния различных веб-сайтов обнаружено, что они в значительной степени зависят от типа веб-сайта и темы информации. Например, если передача коротких текстовых фраз, связанных с новостями, сильно зависит от влияния нескольких крупных медиа-сайтов, то передача хэштегов Twitter регулируется гораздо большим набором активных пользователей, каждый из которых имеет относительно меньшее влияние. Также отмечено, что пользователи с наибольшим числом подписчиков не являются наиболее влиятельными в распространении хэштегов [22].

Предсказание большого каскада затруднено из-за множества факторов, влияющих на процесс диффузии информации. Проблема предсказания большого каскада рассматривается из-за редкого появления больших каскадов во многих системах и непредсказуемости в социальных системах. В работе [23] пришли к выводу, что никакая мера качества не может точно предсказать величину распространения, с помощью искусственного эксперимента «музыкального рынка», который показал, что песни с одинаковыми начальными условиями могут достичь очень разных уровней популярности.

В [24] авторы показали возможность предсказания того, что каскад достигнет наибольшей величины, с высокой точностью с учетом временных и структурных признаков начального процесса расширения. Более точное предсказание можно было бы дать, имея больше информации о каскаде, например, о большей части пути в каскаде распространения. Но до какой степени можно предсказать распространение информации и какие функции являются наиболее значимыми для прогнозирования информационных каскадов, все еще неясно. Хотя было бы очень трудно идентифицировать внутренние свойства для предсказания информационного каскада, некоторые закономерности получаются на основе эмпирического анализа. Например, важность индивидуумов, которые участвуют в каскаде на начальных этапах, и

ширина (а не глубина) каскадной структуры хорошо коррелирует с окончательным размером каскадов.

3. Метрики

В данном разделе описываются данные, используемые в изучении методов анализа информационных потоков, и различные модели распространения информации

3.1 Датасеты

В основной части исследований моделей предсказания используются данные, по которым легко отследить пути передачи информации, поэтому в большинстве случаев данными являются наборы твитов из Twitter или Sina Weibo или посты Facebook. Большинство данных находится в закрытом доступе. Для каждого твита или поста определен автор и первоначальный источник информации, а передачей информации от одного пользователя другому считается ретвит (репост) информации. В исследованиях также используют информационные каскады, собранные из новостных блогов и СМИ. Пути распространения информации от одного узла к другому создаются с помощью инструмента Memetracker [25], который строит карты ежедневного цикла новостей, анализируя около 900 000 новостных сообщений и сообщений в блоге в день из 1 миллиона онлайн-источников, а также гиперссылки между новостными сайтами, ссылающимися на одну и ту же информацию.

3.2 Объяснительные модели

В этом подразделе описываются модели, в которых предлагаются различные механизмы, объясняющие наблюдаемые информационные потоки. Пороговые (п. 3.2.1), каскадные модели (п. 3.2.2) и модели эпидемий (п. 3.2.3) основываются на существующей сети связей между узлами. При отсутствии такой сети в явном виде, другой тип моделей (п. 3.2.4) предполагает наличие скрытых отношений между узлами.

3.2.1 Пороговые модели

Основная идея заключается в том, что люди в сети принимают решения, основанные на действиях своих соседей. Такая стратегия принятия решений подразумевает, что пороговая модель (threshold model) содержит память об истории распространения. Описывается она следующим образом: на первом шаге произвольным образом выбирается небольшая часть людей, которая получила информацию с самого начала. В течение каждого временного шага для пользователя i в неактивном состоянии определяется ϕ_i как пороговое значение i. Состояние i изменится на активное, если доля его соседей в активном состоянии равна или больше, чем ϕ_i. Все узлы в активном

состоянии остаются без изменений. Процесс диффузии заканчивается, когда число пользователей в активном состоянии не меняется.

Чентола и др. [26] применяют пороговую модель для различных социальных сетей. В этой статье пришли к выводу, что, в отличие от результатов моделей независимого взаимодействия, где влияние пары узлов между собой не связано с влиянием других, случайные передачи информации между удаленными узлами уменьшают способность сетей распространять информацию.

Одно из применений пороговой модели заключается в изучении влияния структуры сообщества на распространение информации. Так, используя линейную пороговую модель, Нематзаде и др. [27] раскрыли роль модульной структуры в распространении информации, которая указывает на то, что влиятельные сообщества могут усилить локальное распространение, а слабые сообщества могут улучшить глобальное распространение. Более того, найдена оптимальная модульная структура, которая может способствовать как локальному, так и глобальному распространению.

В различных статьях предлагаются обобщения пороговой модели. Например, Доддс и Уоттс [28] формулируют модель, которая учитывает память о прошлых воздействиях на узлы сети. Модель также включает в себя Independent interaction model (SIS и SIR из п. 3.2.3) в качестве частного случая. Брамитт и др. [29] предполагают, что узел становится активным, если доля активных соседей в любом слое превышает пороговое значение. Авторы [30] модифицируют линейную пороговую модель, предполагая, что на каждого пользователя влияют его соседи в течение некоторого периода времени.

3.2.2 Каскадные модели

Каскадная модель (cascade model) основывается на двух гипотезах. Во-первых, попарные влияния узлов друг на друга независимы, во-вторых, любой активный узел i имеет только один шанс передать информацию своему соседу j, независимо от результата на последующих шагах узел i не будет влиять на j. На первом шаге, как и в пороговой модели выбирается некоторый набор узлов, который активируют. На новом шаге k каждый узел u, получивший информацию на предыдущем шаге, пытаются активировать всех неактивированных соседей j с вероятностью p_{ij}. Процесс останавливается, когда новые пользователи, получившие информацию, не появляются.

В реальных сетях возникает проблема предсказания вероятности всех ребер распространения. Одно из решений этой проблемы предложено в работе Диккенса и др. [31], в которой рассматриваются масштабируемые методы как для обучения информационных потоков на основе моделей независимых каскадов, так и для предсказания нового потока. Определяются два типа

данных: данные, для которых известны пути предыдущих потоков, и данные, для которых известны только конечные точки распространения.

При рассмотрении задачи распространения информации формулируется две проблемы разного типа: проблема максимизации влияния и минимизации «заражения». В первом случае стоит задача максимизировать конечный размер информационного каскада, во втором требуется минимизировать его влияние на сеть.

Предположим, что A – множество изначально активных узлов, ожидаемое число активных узлов в конечном состоянии обозначается как $\sigma(A)$. Задача максимизации влияния требует найти множество узлов S размера k такое, что $\sigma(A)$ максимально. Кемп и др. [32] изучают проблему максимизации влияния как дискретную задачу оптимизации, основанная на двух моделях: независимой каскадной модели и линейной пороговой модели, для которых применяется жадный алгоритм поиска восхождением. Для жадного алгоритма требуется большое количество вычислений, так как предельный выигрыш для $\sigma(A)$ должен вычисляться много раз при разных наборах начальных узлов. Кимура и др. [33] предложили эффективный метод оценки $\sigma(A)$, основанный на теории перколяции и теории графов, который применяется к приближенному решению задачи максимизации по жадному алгоритму. Эксперименты, проведенные на крупномасштабных сетях реального мира, показали, что этот метод может сократить вычислительную сложность.

Для решения проблемы «заражения» в ранних работах [34] из сети удалялись узлы. Кимура и др. [35] предложили новый метод блокировки ограниченного числа звеньев в сети для уменьшения размера каскада. В этой работе определяется степень «заражения» $C(G)$ на графе $G = (V, E)$, которая является средним от степеней влияния всех узлов v в G, где степень влияния узла v в G, определяется как ожидаемое число активных узлов в конце процесса распространения для начального активного узла v. Поскольку решить данную проблему в больших сетях вычислительно трудно, предлагается использовать жадный алгоритм для получения приближенного решения этой проблемы.

3.2.3 Модели эпидемий

В моделях эпидемии (epidemic models) люди разбиваются на различные группы, и каждой группе назначается какое-то состояние. Три состояния являются наиболее часто используются в процессе диффузии:

- S – восприимчивое состояние (susceptible), в котором узел еще не заражен;
- I – зараженное состояние (infectious), представляющее уже активированных пользователей, которые могут передать вирус другим;
- R – восстановленное состояние (recovered), в котором узел заражен, но не является активным и больше не будет передавать вирус другим.

Различные комбинации этих состояний могут приводить к различным моделям, таким как SI, SIS и SIR.

- Простейший случай модели эпидемии – модель SI, в которой рассматриваются два состояния: S и I. Эта модель похожа на пороговую модель и каскадную модель, поскольку, когда узел становится активированным, он остается в этом состоянии навсегда.
- Модель SIS используется для характеристики эпидемий, имеющих временный иммунитет. Это означает, что зараженные люди снова становятся восприимчивыми.
- Модель SIR вводится для объяснения эпидемий с постоянным иммунитетом среди населения. В отличие от модели SI и SIS, в этой модели приводится восстановленное состояние R. Узлы S будут активированы пользователями состояния I с вероятностью β, а узлы I будут восстанавливаться до состояния R с вероятностью восстановления γ.

Гомес-Родригес и др. [1] также предполагали, что их метод применим не только для моделирования информационных потоков, но и для предсказания распространения вирусов. Однако более поздние работы [36] показали, что процесс распространения информации в социальных медиа отличается от процессов распространения вирусов. Так, например, люди с высокой степенью связности (большим количеством подписок) с меньшей вероятностью передают информацию дальше. Как следствие, существующие модели, основанные на моделировании процесса заражения, не могут в полной мере описать процесс распространения информации.

3.2.4 Модели на основе скрытых связей

Для объяснения распространения информации предполагается наличие некоторых независимых скрытых отношений между пользователями, которые необходимо изучить. Так, Гомес-Родригес и др. [1] делают предположение о том, что существует некоторая базовая статическая сеть, по которой распространяется информация. Можно наблюдать, когда поток достигает узла сети, но неизвестно, откуда информация к нему попала. Анализируется набор независимых информационных каскадов, для которых известен набор пользователей, получивших информацию и время ее получения. Вычисляется вероятность наблюдения набора каскадов в выбранной скрытой сети, и задачей является построение наиболее вероятной скрытой сети, объясняющей распространение данных каскадов. Вводится функция правдоподобия, задача поиска максимума которой эквивалентна нахождению наиболее вероятного графа. Предлагается итеративный жадный алгоритм NetInf максимизации функции правдоподобия, где на первом шаге выбирается пустой граф на вершинах-пользователях сети, а на каждом новом шаге добавляется ребро с наибольшим вкладом в функцию правдоподобия.

В отличие от Netinf, фиксирующего скорость распространения информации между узлами, в работе Гомеса-Родригеса и др. [37] предлагается модель Netrate, которая дополняет Netinf тем, что допускает передачу информации по ребрам с разной скоростью. Вероятность узла, активирующего другой в данный момент времени, моделируется функцией плотности вероятности в зависимости от времени активации и скорости передачи между двумя узлами. Алгоритм Netrate, помимо построения графа распространения, вычисляет скорости передачи информации между узлами.

Недостаток предыдущих моделей в том, что их сеть влияния пользователей остается статической с течением времени. По этой причине Гомес-Родригес и др. [38] расширяют модель Netrate и предлагают алгоритм InfoPath, в котором используются стохастические градиенты построения графа влияния в каждый момент времени (раз в день).

Буриго и др. [39] предлагают модель для предсказания распространения информации и влияния узлов друг на друга. Ее отличие состоит в том, что для каждого пользователя определяется вектор в пространстве. Вероятности влияния выводятся из относительных положений векторов соответствующих пользователей в этом пространстве. Вектора пользователей вычисляются на основе данных каскадов.

3.3 Предсказательные модели

Другой класс моделей нацелен на предсказание того, как определенный информационный поток будет распространяться в данной сети на основе свойств предыдущих диффузий. Например, предсказание количества репостов для данного сообщения или дальнейшей динамики диффузии. Среди моделей для предсказания распространения информации можно выделить два подхода: модели на основе ролей пользователей (п. 3.3.2) и предсказание на основе признаков (п. 3.3.1).

3.3.1 Модели на основе признаков

Модели на основе признаков решают задачу предсказания распространения информационного каскада на основе некоторых свойств диффузии. Авторы, описывающие модели на основе признаков, используют данные, в которых легко отследить передачу информации от одного пользователя к другому. Такими, например, являются Twitter, Facebook, Sina-Weibo. В дальнейшем в этом разделе прием информации одним пользователем от другого будем называть репостом.

Одной из задач изучения распространения информации, является предсказание популярности новости. Для обучения моделей используются свойства новости и атрибуты пользователя и применяются стандартные бинарные классификаторы [40-41] или методы глубокого обучения [42-43]. Чэн и др. [24] предлагают следующую формулировку: для уже

распространяющегося информационного каскада предсказать, удвоится ли его размер. Результаты показывают, что точность такого предсказания растет с размером каскада; каскады, больше растущие в ширину, чем в глубину, с более высокой вероятностью достигнут больших размеров.

Шульман и др. [44] пытаются для потока новостей предсказать, станет ли он более популярен, чем определенный процент других новостей, учитывая набор новостей и данных об истории их распространения. Авторы статьи приходят к выводу, что временные признаки потоков наиболее важные в предсказании популярности распространения, при снижении влияния временных признаков точность значительно уменьшается.

В [45-46] представлен частный случай предсказания популярности: предсказание, будет ли у новости хотя бы один репост. В работе Петровича и др. [45] используется passive-aggressive алгоритм. Помимо глобальной модели обучаются 24 локальные модели для каждого часового периода в сутках, что позволило улучшить качество. Чжан и др. [46] для предсказания репоста предложил сверточную нейронную сеть, в которой объединены признаки пользователя, автора, пользовательские интересы, содержание твита и сходство между интересами пользователя и твитом. Цзян и др. [47] предлагают модель предсказания репостов, основанную на вероятностном методе матричной факторизации, путем интеграции наблюдаемых данных, социального влияния пользователей друг на друга и семантики сообщений.

3.3.2 Модели, учитывающие внешние источники информации

Один из способов улучшить предсказание распространения информации – понять, какую роль играет каждый пользователь в сети распространения. Ян [48] предлагает социальную модель распространения информации RAIN для изучения социальных ролей пользователей и моделирования распространения информации одновременно. RAIN определяет распределение социальных ролей каждого пользователя в соответствии с его структурными атрибутами и его поведением в процессе диффузии. Пользователи разделяются на три социальные группы: лидеры мнений (opinion leaders), пользователи, соединяющие разные сообщества (structural hole spanners), и обычные пользователи (ordinary users). Вводятся параметры для каждой роли ρ_r и λ_r как вероятность того, что пользователи, играющие роль *r*, активируют другого пользователя успешно и соответственно вызовут задержку распространения в течении одной временной отметки. Модель состоит из двух основных частей: сначала для каждого пользователя предсказывается одна из трех возможных ролей, после чего используется функция диффузии (например, пороговая или каскадная функцию), параметризованная ρ_r и λ_r, чтобы определить, станет ли пользователь активным.

Чубдар и др. [49] также предложили модель разбиения по ролям. Пользователей делят на пять ролей влияния, основываясь на различных

структурных свойствах (рейтинге влиянию на другие узлы, рейтинге принятия информации от других узлов, возрасте пользователя и т.д). Модель отличается от [48] тем, что роли пользователей непостоянны. Предложена динамическая модель кластеризации, которая объединяет свойства пользователя и его поведение в течение времени.

Если предыдущие модели предлагают классификацию пользователей по ролям, то Янг и Лесковец [22] рассматривают модель, в которой у каждого пользователя нет конкретной роли, но есть уровень влияния. По сети распространяются независимые каскады, и формулируется линейная модель влияния LIM (linear influence model), исходя из предположения, что количество узлов, вновь получивших информацию, зависит от того, у каких узлов эта информация была в прошлом. Затем число вновь активированных узлов моделируется как функция от времени, когда другие узлы получали информацию в прошлом. В этой модели каждый узел имеет связанную с ним функцию влияния. Тогда число вновь активированных узлов в момент времени *t* является функцией влияния узлов, зараженных до времени *t*.

4. Приложения

Изучение проблемы распространения информационных потоков в сети применимо во многих областях деятельности. Вирусный маркетинг является одним из наиболее важных приложений. Маркетинг из уст в уста – это новая форма маркетинга продукта, которая позволяет максимально использовать сетевые эффекты (например, через различные социальные сети) для повышения осведомленности о конкретном продукте и достижения рекламных целей. Он происходит в различных формах, включая изображения, видео, электронные письма, текстовые сообщения, твиты, игры и блоги. Компания может предоставить определенный продукт выбранному числу влиятельных лиц бесплатно, надеясь, что они порекомендуют продукт другим, если они будут им удовлетворены. Продавцы также могут предлагать скидки, основанные на влиятельности людей, в результате чего доход может даже быть увеличен. Целевая иммунизация – еще один пример изучения информационных потоков в сети, когда для новости требуется иммунизировать небольшое подмножество влиятельных узлов, чтобы уменьшить распространение передаваемой информации.

Распространение информации через социальные сети оказалось мощным инструментом во многих ситуациях, например, исследовались влияние Twitter на выборы президента [50] и Facebook в арабской весне 2010 года [51]. Другим важным исследованием в этой области являются эпидемии и распространение вирусов, которые привлекли многих ученых в области экологии и биологии. Идентификация узлов с наиболее важной ролью в распространении информации будет иметь множество потенциальных приложений в различных областях. Например, путем определения наиболее влиятельных распространителей в преступных группах, соответствующий

разведывательный орган может лучше контролировать преступность и осуществлять превентивные меры.

Диффузия инноваций – еще одна тема, которая часто изучается в сети. Новые идеи, технологии и способы выполнения действий могут быстро распространяться по сети. Было показано, что люди склонны принимать технологию (или продукт) с большей вероятностью, основываясь на мнении своих друзей и соседей (которые определяются на основе их связей в сети), которым они доверяют.

5. Заключение

В обзоре были рассмотрены методы анализа информационных потоков. Были выделены две основные группы рассматриваемых задач в процессе распространения информации.

Первая группа задач основана на предположении о существовании некоторой сети связей между узлами, по которой переходит информация. Пороговые и каскадные модели для существующих сетей пытаются определить, как будет распространяться каскад, основываясь на гипотезах получения информации, зависящих от доли активированных соседей и вероятности одного узла передать информацию другому. Модель, основанная на скрытых связях, предполагает наличие скрытой сети и пытается по информационным каскадам сети ее построить.

Во задачах второй группы не предполагается наличие скрытой сети, ее задачей является предсказание распространения определенного информационного потока на основе его признаков диффузии. Предлагается модель на основе признаков, которая, анализируя свойства информации и атрибуты пользователей, получивших ее, для уже распространяющегося информационного каскада предсказывает его популярность. Также, чтобы улучшить предсказание распространения, предложена модель, которая изучает влияние пользователя на другие узлы и предполагает, какую роль играет пользователь в распространении.

Несмотря на обилие работ в области, все они обладают рядом ограничений и недостатков. Большинство работ основаны на явно наблюдаемых механизмах социальных сетей (репосты), применяются только к одной социальной сети: Twitter, Sina Weibo или Facebook. В силу того, что большая часть данных, используемых в исследованиях, находится в закрытом доступе, предсказательные модели тестируются на разных данных, включая те, которые решают одну и ту же задачу. Это создает проблему сравнения результатов. Также не используются методы глубокого анализа текстов ввиду сложности разработки инструментов анализа текстов, способных обрабатывать большие объемы данных.

Благодарности

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта №18-07-01059.

Список литературы

[1] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 1019-1028.

[2] Z.-K. Zhang, C. Liu, X.-X. Zhan, X. Lu, C.-X. Zhang, and Y.-C. Zhang. Dynamics of information diffusion and its applications on complex networks. *Physics Reports*, vol. 651, 2016, pp. 1-34.

[3] Acquisti and R. Gross. Imagined communities: Awareness, information sharing, and privacy on the facebook. In Proc. of the International workshop on privacy enhancing technologies. Springer, 2006, pp. 36-58.

[4] S. Fournier and J. Avery. The uninvited brand. Business horizons, vol. 54, no. 3, pp. 193-207, 2011.

[5] G.E. Kreindler and H.P. Young. Rapid innovation diffusion in social networks. In Proceedings of the National Academy of Sciences, vol. 111, supplement 3, 2014, pp. 10 881-10 888.

[6] T. Holz, M. Steiner, F. Dahl et al. Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. In Proc. of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, 2008.

[7] B. Doerr, M. Fouz, and T. Friedrich. Why rumors spread so quickly in social networks. Communications of the ACM, vol. 55, no. 6, 2012, pp. 70-75.

[8] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009, pp. 497-506.

[9] Y. Sano, K. Yamada, H. Watanabe, H. Takayasu, and M. Takayasu. Empirical analysis of collective human behavior for extraordinary events in the Blogosphere. Physical Review E, vol. 87, no. 1, 2013.

[10] J. Lehmann, B. Goncalves, J.J. Ramasco, and C. Cattuto. Dynamical classes of collective attention in twitter. In Proc. of the 21st international conference on World Wide Web, 2012, pp. 251–260.

[11] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos. Rise and fall patterns of information diffusion: model and implications. In Proc. of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 6–14.

[12] D. M. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In Proc. of the 20th international conference on World wide web, 2011, pp. 695–704.

[13] M. Del Vicario, A. Bessi, F. Zollo, F. Petroni, A. Scala, G. Caldarelli, H.E. Stanley, and W. Quattrociocchi. The spreading of misinformation online. In Proceedings of the National Academy of Sciences, vol. 113, no. 3, 2016, pp. 554–559.

[14] S. Ardon, A. Bagchi, A. Mahanti, A. Ruhela, A. Seth, R. M. Tripathy, and S. Triukose. Spatio-temporal and events based analysis of topic popularity in twitter. In Proc. of the 22nd ACM international conference on Information & Knowledge Management, 2013,

pp. 219–228.
[15] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. In Proc. of the 21st international conference on World Wide Web, 2012, pp. 519–528.
[16] S. Goel, D.J. Watts, and D.G. Goldstein. The structure of online diffusion networks. In Proc. of the 13th ACM conference on electronic commerce, 2012, pp. 623–638.
[17] S. Goel, A. Anderson, J. Hofman, and D. J. Watts. The structural virality of online diffusion. Management Science, vol. 62, no. 1, 2015, pp. 180–196.
[18] S. Krishnan, P. Butler, R. Tandon, J. Leskovec, and N. Ramakrishnan. Seeing the forest for the trees: new approaches to forecasting cascades. In Proc. of the 8th ACM Conference on Web Science, 2016, pp. 249–258.
[19] S.A. Myers, C. Zhu, and J. Leskovec. Information diffusion and external influence in networks. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 33–41.
[20] E. Bakshy, S. Messing, and L.A. Adamic. Exposure to ideologically diverse news and opinion on facebook. Science, vol. 348, no. 6239, 2015, pp. 1130–1132.
[21] A. Vazquez, B. Racz, A. Lukacs, and A.-L. Barabasi. Impact of non-poissonian activity patterns on spreading processes. Physical review letters, vol. 98, no. 15, 2007.
[22] J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In Proc. of the 2010 IEEE 10th International Conference on Data Mining (ICDM), 2010, pp. 599–608.
[23] M.J. Salganik, P.S. Dodds, and D.J. Watts. Experimental study of inequality and unpredictability in an artificial cultural market. Science, vol. 311, no. 5762, 2006, pp. 854–856.
[24] J. Cheng, L.Adamic, P.A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In Proc. of the 23rd international conference on World wide web, 2014, pp. 925–936.
[25] MemeTracker data. Режим доступа: http://www.memetracker.org/data.html, дата обращения 20.11.2018.
[26] D. Centola, V. M. Eguıluz, and M.W. Macy. Cascade dynamics of complex propagation. Physica A: Statistical Mechanics and its Applications, vol. 374, no. 1, 2007, pp. 449–456.
[27] A. Nematzadeh, E. Ferrara, A. Flammini, and Y.-Y. Ahn. Optimal network modularity for information diffusion. Physical review letters, vol. 113, 2014, no. 8.
[28] P.S. Dodds and D.J. Watts. A generalized model of social and biological contagion. Journal of theoretical biology, vol. 232, no. 4, 2005, pp. 587–604.
[29] C.D. Brummitt, K.-M. Lee, and K.-I. Goh. Multiplexity-facilitated cascades in networks. Physical Review E, vol. 85, no. 4, 2012.
[30] F. Karimi and P. Holme. Threshold model of cascades in empirical temporal networks. Physica A: Statistical Mechanics and its Applications, vol. 392, no. 16, 2013, pp. 3476–3483.
[31] L. Dickens, I. Molloy, J. Lobo, P.-C. Cheng, and A. Russo. Learning stochastic models of information flow. In Proc. of the 2012 IEEE 28th international conference on data engineering, 2012, pp. 570–581.
[32] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 137–146.
[33] M. Kimura, K. Saito, and R. Nakano. Extracting influential nodes for information

diffusion on a social network. In Proc. of the Twenty-Second AAAI conference on Artificial intelligence, vol. 2, 2007, pp. 1371–1376.

[34] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. Computer networks, vol. 33, no. 1-6, 2000, pp. 309–320.

[35] M. Kimura, K. Saito, and H. Motoda. Minimizing the spread of contamination by blocking links in a network. in Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, 2008, pp. 1175–1180.

[36] K. Lerman. Information is not a virus, and other consequences of human cognitive limits. Future Internet, vol. 8, no. 2, 2016.

[37] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. arXiv preprint arXiv:1105.0697, 2011.

[38] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. Structure and dynamics of information pathways in online media. In Proc. of the sixth ACM international conference on Web search and data mining, 2013, pp. 23–32.

[39] S. Bourigault, S. Lamprier, and P. Gallinari. Representation learning for information diffusion through social networks: an embedded cascade model. In Proc. of the Ninth ACM International Conference on Web Search and Data Mining, 2016, pp. 573–582.

[40] M. Jenders, G. Kasneci, and F. Naumann. Analyzing and predicting viral tweets. In Proceedings of the 22nd international conference on world wide web, 2013, pp. 657–664.

[41] Y. Zhang, Z. Xu, and Q. Yang. Predicting popularity of messages in twitter using a feature-weighted model. Режим доступа: http://www.nlpr.ia.ac.cn/2012papers/gjhy/gh154.pdf, дата обращения 20.11.2018.

[42] Q. Cao, H. Shen, K. Cen, W. Ouyang, and X. Cheng. Deephawkes: Bridging the gap between prediction and understanding of information cascades. In Proc. of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1149–1158.

[43] C. Li, J. Ma, X. Guo, and Q. Mei. Deepcas: An end-to-end predictor of information cascades. In Proc. of the 26th International Conference on World Wide Web, 2017, pp. 577–586.

[44] B. Shulman, A. Sharma, and D. Cosley. Predictability of popularity: Gaps between prediction and understanding. in Proc. of the Tenth International AAAI Conference on Web and Social Media (ICWSM 2016), 2016, pp. 348–357.

[45] S. Petrovic, M. Osborne, and V. Lavrenko. Rt to win! predicting message propagation in twitter. In Proc. of the Fifth International AAAI Conference on Weblogs and Social Media, 2011, pp. 586–589.

[46] Q. Zhang, Y. Gong, J. Wu, H. Huang, and X. Huang. Retweet prediction with attention-based deep neural network. In Proc. of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 75–84.

[47] B. Jiang, Z. Lu, N. Li, J. Wu, and Z. Jiang. Retweet prediction using social-aware probabilistic matrix factorization. Lecture Notes in Computer Science, vol. 10860, 2018, pp. 316–327.

[48] Y. Yang, J. Tang, C. W.-k. Leung, Y. Sun, Q. Chen, J. Li, and Q. Yang. Rain: Social role-aware information diffusion. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 367–373.

[49] S. Choobdar, P. Ribeiro, S. Parthasarathy, and F. Silva. Dynamic inference of social roles in information cascades. Data mining and knowledge discovery, vol. 29, no. 5,

2015, pp. 1152–1177.
[50] L. Hughes and L. Palen. Twitter adoption and use in mass convergence and emergency events. International journal of emergency management, vol. 6, no. 3-4, 2009, pp. 248–260.
[51] P. N. Howard, A. Duffy, D. Freelon, M. M. Hussain, W. Mari, and M. Maziad. Opening closed regimes: what was the role of social media during the arab spring? Режим доступа: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2595096, дата обращения 20.11.2018.

Methods for Information Spread Analysis

[1,2] *Avetisyan A. A.<a.a.avetisyan@ispras.ru>*
[1,3] *Drobyshevskiy M. D.<drobyshevsky@ispras.ru>*
[1,2,4] *Turdakov D.Yu.<turdakov@ispras.ru>*

[1]*Ivannikov Institute for System Programming of the RAS,*
25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia
[2]*Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation
[3]*Moscow Institute of Physics and Technology,*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russian Federation
[4] *National Research University Higher School of Economics (HSE)*
11 Myasnitskaya Ulitsa, Moscow, 101000, Russia

Abstract. Spread of information is a fundamental process taking place on the Internet. Every day we can observe the publication of various information and its further dissemination through news articles and messages from ordinary users. Although the process itself can be observed explicitly, it is difficult to determine individual propagation paths. The increase of global information in all spheres of human life radically changes the speed and ways of disseminating information. In this review, we study models of information flows on the Internet and divide them into two groups: explanatory models, which suggest the existence of the underlying network over which information propagates, and predictive models, studying spread of individual information flows. Despite all the complexity, the study of the important properties of information spread is necessary for understanding the general processes occurring in the modern information society.

Keywords: information diffusion, information cascades, networks of diffusion

DOI: 10.15514/ISPRAS-2018-30(6)-11

For citation: Avetisyan A.A., Drobyshevskiy M. D., Turdakov D.Yu. Methods for Information Spread Analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 199-220 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-11

References

[1] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 1019-1028.
[2] Z.-K. Zhang, C. Liu, X.-X. Zhan, X. Lu, C.-X. Zhang, and Y.-C. Zhang. Dynamics of

information diffusion and its applications on complex networks. *Physics Reports*, vol. 651, 2016, pp. 1-34.

[3] Acquisti and R. Gross. Imagined communities: Awareness, information sharing, and privacy on the facebook. In Proc. of the International workshop on privacy enhancing technologies. Springer, 2006, pp. 36-58.

[4] S. Fournier and J. Avery. The uninvited brand. Business horizons, vol. 54, no. 3, pp. 193-207, 2011.

[5] G.E. Kreindler and H.P. Young. Rapid innovation diffusion in social networks. In Proceedings of the National Academy of Sciences, vol. 111, supplement 3, 2014, pp. 10 881-10 888.

[6] T. Holz, M. Steiner, F. Dahl et al. Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. In Proc. of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, 2008.

[7] B. Doerr, M. Fouz, and T. Friedrich. Why rumors spread so quickly in social networks. Communications of the ACM, vol. 55, no. 6, 2012, pp. 70-75.

[8] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009, pp. 497-506.

[9] Y. Sano, K. Yamada, H. Watanabe, H. Takayasu, and M. Takayasu. Empirical analysis of collective human behavior for extraordinary events in the Blogosphere. Physical Review E, vol. 87, no. 1, 2013.

[10] J. Lehmann, B. Goncalves, J.J. Ramasco, and C. Cattuto. Dynamical classes of collective attention in twitter. In Proc. of the 21st international conference on World Wide Web, 2012, pp. 251–260.

[11] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos. Rise and fall patterns of information diffusion: model and implications. In Proc. of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 6–14.

[12] D. M. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In Proc. of the 20th international conference on World wide web, 2011, pp. 695–704.

[13] M. Del Vicario, A. Bessi, F. Zollo, F. Petroni, A. Scala, G. Caldarelli, H.E. Stanley, and W. Quattrociocchi. The spreading of misinformation online. In Proceedings of the National Academy of Sciences, vol. 113, no. 3, 2016, pp. 554–559.

[14] S. Ardon, A. Bagchi, A. Mahanti, A. Ruhela, A. Seth, R. M. Tripathy, and S. Triukose. Spatio-temporal and events based analysis of topic popularity in twitter. In Proc. of the 22nd ACM international conference on Information & Knowledge Management, 2013, pp. 219–228.

[15] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. In Proc. of the 21st international conference on World Wide Web, 2012, pp. 519–528.

[16] S. Goel, D.J. Watts, and D.G. Goldstein. The structure of online diffusion networks. In Proc. of the 13th ACM conference on electronic commerce, 2012, pp. 623–638.

[17] S. Goel, A. Anderson, J. Hofman, and D. J. Watts. The structural virality of online diffusion. Management Science, vol. 62, no. 1, 2015, pp. 180–196.

[18] S. Krishnan, P. Butler, R. Tandon, J. Leskovec, and N. Ramakrishnan. Seeing the forest for the trees: new approaches to forecasting cascades. In Proc. of the 8th ACM Conference on Web Science, 2016, pp. 249–258.

[19] S.A. Myers, C. Zhu, and J. Leskovec. Information diffusion and external influence in networks. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 33–41.
[20] E. Bakshy, S. Messing, and L.A. Adamic. Exposure to ideologically diverse news and opinion on facebook. Science, vol. 348, no. 6239, 2015, pp. 1130–1132.
[21] A. Vazquez, B. Racz, A. Lukacs, and A.-L. Barabasi. Impact of non-poissonian activity patterns on spreading processes. Physical review letters, vol. 98, no. 15, 2007.
[22] J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In Proc. of the 2010 IEEE 10th International Conference on Data Mining (ICDM), 2010, pp. 599–608.
[23] M.J. Salganik, P.S. Dodds, and D.J. Watts. Experimental study of inequality and unpredictability in an artificial cultural market. Science, vol. 311, no. 5762, 2006, pp. 854–856.
[24] J. Cheng, L. Adamic, P.A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In Proc. of the 23rd international conference on World wide web, 2014, pp. 925–936.
[25] MemeTracker data. Режим доступа: http://www.memetracker.org/data.html, дата обращения 20.11.2018.
[26] D. Centola, V. M. Eguıluz, and M.W. Macy. Cascade dynamics of complex propagation. Physica A: Statistical Mechanics and its Applications, vol. 374, no. 1, 2007, pp. 449–456.
[27] A. Nematzadeh, E. Ferrara, A. Flammini, and Y.-Y. Ahn. Optimal network modularity for information diffusion. Physical review letters, vol. 113, 2014, no. 8.
[28] P.S. Dodds and D.J. Watts. A generalized model of social and biological contagion. Journal of theoretical biology, vol. 232, no. 4, 2005, pp. 587–604.
[29] C.D. Brummitt, K.-M. Lee, and K.-I. Goh. Multiplexity-facilitated cascades in networks. Physical Review E, vol. 85, no. 4, 2012.
[30] F. Karimi and P. Holme. Threshold model of cascades in empirical temporal networks. Physica A: Statistical Mechanics and its Applications, vol. 392, no. 16, 2013, pp. 3476–3483.
[31] L. Dickens, I. Molloy, J. Lobo, P.-C. Cheng, and A. Russo. Learning stochastic models of information flow. In Proc. of the 2012 IEEE 28th international conference on data engineering, 2012, pp. 570–581.
[32] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 137–146.
[33] M. Kimura, K. Saito, and R. Nakano. Extracting influential nodes for information diffusion on a social network. In Proc. of the Twenty-Second AAAI conference on Artificial intelligence, vol. 2, 2007, pp. 1371–1376.
[34] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. Computer networks, vol. 33, no. 1-6, 2000, pp. 309–320.
[35] M. Kimura, K. Saito, and H. Motoda. Minimizing the spread of contamination by blocking links in a network. in Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, 2008, pp. 1175–1180.
[36] K. Lerman. Information is not a virus, and other consequences of human cognitive limits. Future Internet, vol. 8, no. 2, 2016.
[37] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal

dynamics of diffusion networks. arXiv preprint arXiv:1105.0697, 2011.

[38] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. Structure and dynamics of information pathways in online media. In Proc. of the sixth ACM international conference on Web search and data mining, 2013, pp. 23–32.

[39] S. Bourigault, S. Lamprier, and P. Gallinari. Representation learning for information diffusion through social networks: an embedded cascade model. In Proc. of the Ninth ACM International Conference on Web Search and Data Mining, 2016, pp. 573–582.

[40] M. Jenders, G. Kasneci, and F. Naumann. Analyzing and predicting viral tweets. In Proceedings of the 22nd international conference on world wide web, 2013, pp. 657–664.

[41] Y. Zhang, Z. Xu, and Q. Yang. Predicting popularity of messages in twitter using a feature-weighted model. Available at: http://www.nlpr.ia.ac.cn/2012papers/gjhy/gh154.pdf, accessed 20.11.2018.

[42] Q. Cao, H. Shen, K. Cen, W. Ouyang, and X. Cheng. Deephawkes: Bridging the gap between prediction and understanding of information cascades. In Proc. of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1149–1158.

[43] C. Li, J. Ma, X. Guo, and Q. Mei. Deepcas: An end-to-end predictor of information cascades. In Proc. of the 26th International Conference on World Wide Web, 2017, pp. 577–586.

[44] B. Shulman, A. Sharma, and D. Cosley. Predictability of popularity: Gaps between prediction and understanding. in Proc. of the Tenth International AAAI Conference on Web and Social Media (ICWSM 2016), 2016, pp. 348–357.

[45] S. Petrovic, M. Osborne, and V. Lavrenko. Rt to win! predicting message propagation in twitter. In Proc. of the Fifth International AAAI Conference on Weblogs and Social Media, 2011, pp. 586–589.

[46] Q. Zhang, Y. Gong, J. Wu, H. Huang, and X. Huang. Retweet prediction with attention-based deep neural network. In Proc. of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 75–84.

[47] B. Jiang, Z. Lu, N. Li, J. Wu, and Z. Jiang. Retweet prediction using social-aware probabilistic matrix factorization. Lecture Notes in Computer Science, vol. 10860, 2018, pp. 316–327.

[48] Y. Yang, J. Tang, C. W.-k. Leung, Y. Sun, Q. Chen, J. Li, and Q. Yang. Rain: Social role-aware information diffusion. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 367–373.

[49] S. Choobdar, P. Ribeiro, S. Parthasarathy, and F. Silva. Dynamic inference of social roles in information cascades. Data mining and knowledge discovery, vol. 29, no. 5, 2015, pp. 1152–1177.

[50] L. Hughes and L. Palen. Twitter adoption and use in mass convergence and emergency events. International journal of emergency management, vol. 6, no. 3-4, 2009, pp. 248–260.

[51] P. N. Howard, A. Duffy, D. Freelon, M. M. Hussain, W. Mari, and M. Maziad. Opening closed regimes: what was the role of social media during the arab spring? Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2595096, accessed 20.11.2018.

Автоматический поиск фрагментов, содержащих биографическую информацию, в тексте на естественном языке[1]

А.В. Глазкова <a.v.glazkova@utmn.ru>
Тюменский государственный университет,
625003, Россия, г. Тюмень, ул. Володарского, д.6

Аннотация. Поиск и классификация текстовых документов применяются во многих практических приложениях и являются одними из ключевых задач информационного поиска. Методы поиска и классификации текстов находят применение в поисковых системах, электронных библиотеках и каталогах, системах сбора и обработки информации, платформах для онлайн-обучения и многих других. Существует большое количество частных применений указанных методов, однако каждая подобная практическая задача отличается, как правило, слабой формализуемостью, узкой предметностью и, следовательно, требует индивидуального изучения и собственного подхода к решению. В данной работе рассматривается задача автоматического поиска и типизации текстовых фрагментов, содержащих биографическую информацию. Ключевой проблемой при решении указанной задачи является проведение мультиклассовой классификации текстовых фрагментов в зависимости от наличия и типа содержащейся в них биографической информации. Проведя обзор научной литературы по рассматриваемому вопросу, авторы сделали вывод о перспективности и широте применения нейросетевых методов для решения подобных задач. Исходя из данного вывода, в работе проведено сравнение различных архитектур нейросетевых моделей, а также основных способов представления текстов (Bag-of-Words, Bag-of-Ngrams, TF-IDF, Word2Vec) на предварительно собранном и размеченном корпусе биографических текстов. В статье описываются этапы подготовки обучающего множества текстовых фрагментов для обучения моделей, способы представления текстов и методы классификации, выбранные для решения задачи. Также приводятся результаты мультиклассовой классификации текстовых фрагментов и показаны примеры автоматического поиска фрагментов, содержащих биографическую информацию, в текстах, не участвовавших в процессе обучения моделей.

Ключевые слова: классификация текстов; обработка естественного языка; векторные представления слов; нейронные сети; биографический текст.

[1] Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18-37-00272 «Автоматизированное извлечение биографических фактов из текстов на естественном языке».

DOI: 10.15514/ISPRAS-2018-30(6)-12

Для цитирования: Глазкова А.В. Автоматический поиск фрагментов, содержащих биографическую информацию, в тексте на естественном языке. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 221-236. DOI: 10.15514/ISPRAS-2018-30(6)-12

1. Введение

Постоянное увеличение количества и объема мировых информационных ресурсов требует разработки и совершенствования технологий информационного поиска. Решение задач информационного поиска связано, как правило, с обработкой естественного языка и направлено на анализ, обобщение и упорядочение неструктурированной документальной (в том числе текстовой) информации.

Одной из прикладных задач информационного поиска является задача автоматического поиска биографической информации в текстах, написанных на естественном языке. Извлечение биографической информации выполняется не только при построении текста биографической справки в поисковых системах, но и при проведении биографических исследований, которые подразумевают работу с фактами, касающимися жизни человека. Поиск биографических фактов в тексте на естественном языке имеет ряд особенностей. Так, факты сами по себе могут касаться различных сфер жизни – социальной, политической, экономической или личной. Тексты, содержащие биографические факты, включают в себя как строго документальные (автобиографии, резюме), так и нестрого документальные (воспоминания, очерки, хроники) [1]. Во втором случае какая-то часть биографической информации может встречаться в тексте в неявном виде, среди другой информации, не относящейся к биографической. Эти особенности вынуждают исследователя просматривать большое количество электронных документов в поисках значимых для его работы фактов.

В данной статье рассматривается задача автоматического поиска фрагментов, содержащих биографическую информацию, в тексте, написанном на естественном языке. Рассматриваемая задача может быть представлена как задача тематической классификации текстов. При этом текстовые фрагменты, подлежащие классификации, оцениваются с точки зрения того, содержат ли они биографическую информацию, и если содержат, то к какому тематическому классу эта информация относится.

Вопросы извлечения фактов различного характера из текста на естественном языке и их классификации довольно широко освещаются в научной литературе. При этом особенностью задачи извлечения фактов является ее слабая формализуемость и узкая предметность [2]. Существующие подходы к извлечению фактов, хотя они и охватывают довольно широкий спектр прикладных задач, сложно сравнить по качеству и результативности с подходом, представленным в данной работе. Это связано как со спецификой каждой отдельной задачи, так и с различием текстовых коллекций,

использованных для оценки результатов, и особенностями обрабатываемых естественных языков. Так, работа И.М. Адамовича и О.И. Волкова [3] посвящена извлечению биографических фактов из исторических документов. Авторы описывают технологию, которая представляет факт как древовидную структуру. Корнем такого дерева является факт (например, «рождение»), связанные с данным фактом сущности сохраняются в листьях. Также извлечение биографических фактов обсуждается в статьях [4-5]. В статье [6] авторы решают задачу классификации отношений между словами текста (по сути выделение фактов и «не-фактов») с использованием сверточной нейронной сети, которая выполняет классификацию путем ранжирования (CR-CNN). В работе P. Meerkampd и Z. Zhou [7] представлена архитектура системы извлечения информации из текста, которая сочетает в себе возможности синтаксического анализа (парсинга) текста и нейронной сети. В работе Y. Homma и др. [8] описана иерархическая нейронная сеть для классификации предложений для извлечения фактов о продукте из товарных документов. Сеть классифицирует каждое предложение в документе по классам атрибутов и условий на основе последовательностей слов и предложений в документе.

Стоит отметить, что в большинстве представленных работ используются методы машинного обучения. Для решения различных задач обработки естественного языка в работах российских и зарубежных исследователей [9-14] неоднократно применялись нейросетевые технологии, основанные на рекуррентности и долгой краткосрочной памяти.

2. Представление текстов

2.1 Текстовая коллекция

В качестве текстового фрагмента в рамках данной работы рассматривается предложение. Данная языковая единица представляет собой грамматически организованное соединение слов (или слово), обладающее смысловой законченностью [15]. При этом возможны ситуации, когда несколько предложений текста в смысловом плане отражают один и тот же биографический факт. Вопросы объединения взаимодополняющих и удаления дублирующихся фактов являются дальнейшими задачами данного исследования.

В работе использовались тексты, находящиеся в открытом доступе в онлайн-энциклопедии «Википедия» [16]. Ранее [17] авторами был описан корпус биографических текстов, собранный на основе биографических текстов, представленных в «Википедии», и размеченный в полуавтоматическом режиме. Каждому предложению собранной коллекции биографических текстов сопоставлен один из классов в соответствии со следующей таксономией:

- содержит биографические факты:
  - информация о родительской семье;

- личные события;
- место жительства или пребывания;
- место работы или службы;
- национальность;
- образование;
- профессиональные события (встречи, награждения и т.д.);
- род занятий;
- рождение;
- семья (женитьба, замужество, дети);
- смерть;
- прочие биографические факты;

- не содержит биографические факты.

В корпус включены 200 биографий персоналий, живших или живущих в 20-21 веках. Основные количественные характеристики корпуса представлены в табл. 1.

Табл. 1. Количественные характеристики корпуса
Table 1. Quantitative characteristics of the corpus

Характеристика	**Значение**
Среднее количество слов в текстах	225
Среднее количество предложений в текстах	19
Доля типов биографических фактов (%)	
Информация о родительской семье	5,23
Личные события	4,17
Место жительства	3,99
Место работы	2,15
Национальность	1,51
Образование	6,46
Профессиональные события	13,80
Род занятий	14,67
Рождение	4,68
Семья	34,94
Смерть	4,49
Прочие биографические факты	3,90

Корпус биографических текстов в формате .xml доступен по ссылке [18]. Для формирования коллекции предложений, не содержащих биографические

факты, была использована выборка случайных небиографических статей из «Википедии».

Итоговая коллекция текстовых фрагментов, использованная для обучения моделей, включила в себя предложения, относящиеся к 11 классам: «Информация о родительской семье», «Личные события», «Место жительства», «Место работы», «Образование», «Профессиональные события», «Род занятий», «Рождение», «Семья», «Смерть» и «Фрагменты, не содержащие биографическую информацию». Класс «Национальность» был объединен с классом «Информация о родительской семье» ввиду семантического сходства входящих в них обучающих примеров. Класс «Прочие биографические факты» был исключен, поскольку входящие в него фрагменты содержат информацию, косвенно относящуюся к биографической, однако не принадлежащую ни к одному из конкретных классов.

Для выравнивания количества обучающих элементов в различных классах был проведен простой оверсэмплинг – дублирование элементов миноритарных классов. Итоговое количество предложений, содержащих биографические факты, составило 6773.

Предобработка текстовой коллекции включала в себя следующие этапы:

- удаление знаков препинания и специальных символов;
- перевод символов в нижний регистр;
- удаление стоп-слов;
- лемматизация (использовались средства библиотеки pymorphy2 [19]).

2.2 Признаковое пространство

В данной работе были рассмотрены четыре способа представления текстов:

- модель Bag-of-Words;
- Bag-of-Words + TF-IDF;
- Bag-of-Ngrams + TF-IDF;
- Word2Vec.

В ходе построения модели Bag-of-Words текстовая коллекция была представлена в виде матрицы, количество строк которой равно количеству документов, а количество столбцов – количеству слов в коллекции (за исключением списка стоп-слов). На пересечении строки и столбца хранится количество вхождений слова в текст конкретного документа.

Для определения наиболее характерных для классов слов использовалась мера TF-IDF. Списки слов, имеющих наибольшие значения TF-IDF для каждого класса, представлены в таблице 2. В целях удобства отображения в каждом классе показаны по 5 наиболее значимых слов и соответствующие им значения TF-IDF.

Табл. 2. Наиболее значимые слова для классов текстовой коллекции
Table 2. The most important words for text collection classes

Название класса	**Наиболее значимые слова**		**Название класса**	**Наиболее значимые слова**	
	Слово	**TF-IDF**		**Слово**	**TF-IDF**
Личные события	познакомиться	0,13	Профессио-нальные события	быть	0,19
	религия	0,08		наградить	0,16
	обвинение	0,08		работа	0,15
	уголовный	0,08		орден	0,15
	приговор	0,07		звание	0,1
Место жительства	жить	0,41	Род занятий	заместитель	0,31
	переехать	0,27		назначить	0,26
	город	0,13		начальник	0,2
	провести	0,12		должность	0,19
	вернуться	0,12		работать	0,15
Место работы	общество	0,18	Рождение	родиться	0,86
	член-корреспондент	0,14		семья	0,26
	избрать	0,14		город	0,13
	состоять	0,08		область	0,12
	являться	0,07		губерния	0,08
Образование	окончить	0,63	Семья	супруг	0,12
	факультет	0,22		замужем	0,12
	защитить	0,2		младший	0,08
	институт	0,19		брак	0,06
	диссертация	0,17		совместный	0,06
Происхождение (объединение классов «Информация о родительской семье» и «Национальность»)	мать	0,46	Смерть	умереть	0,65
	семья	0,32		кладбище	0,41
	отец	0,26		похоронить	0,35
	родиться	0,17		скончаться	0,33
	принадлежать	0,11		расстрелять	0,08

При представлении текстов в виде модели Bag-of-Words с использованием TF-IDF на пересечении строки и столбца находится значение значимости слова в данном документе, рассчитанное при помощи TF-IDF.

Представление текста в виде Bag-of-Ngrams + TF-IDF аналогично представлению Bag-of-Words + TF-IDF, однако для расчета TF-IDF вместо слов используются N-граммы символов. В данной работе лучший результат для данного способа представления текстов был получен при N=4.

Word2Vec [20] в настоящее время служит одним из наиболее популярных и эффективных способов представления слов в векторном виде, пригодном для машинного обучения (word embeddings). Этот способ построен на частоте совстречаемости слов в пределах одного контекста.

В данной работе использовались векторные представления, полученные по алгоритму Word2Vec на основе текстов русскоязычной «Википедии» за 2018 год с использованием алгоритма обучения Skip-gram. В ходе экспериментов был выбран размер результирующего контекстного вектора, равный 300.

3. Методы

Для классификации фрагментов текстов были выбраны следующие типы нейронных сетей:

- сеть прямого распространения (feedforward network, FNN);
- сеть долгой краткосрочной памяти (long short-term memory, LSTM);
- двунаправленная сеть долгой краткосрочной памяти (bidirectional long short-term memory, BLSTM).

Сети прямого распространения применялись в качестве модели для классификации текстов с использованием представлений документов в виде моделей Bag-of-Words, Bag-of-Words + TF-IDF и Bag-of-Ngrams + TF-IDF, а также в качестве фрагментов каскадных моделей. Для классификации текстов на основании векторных представлений слов использовались сети долгой краткосрочной памяти.

В отличие от классических архитектур нейронных сетей, в рекуррентном слое сети долгой краткосрочной памяти предусмотрен механизм хранения долгосрочных зависимостей, позволяющий избежать проблемы затухания градиента [21]. Архитектура ячейки сети долгой краткосрочной памяти представлена на рис. 1 [22].

Пусть xt и yt – входной и выходной сигналы соответственно в момент времени t, а ct и mt – состояние ячейки и выхода в момент t. Преобразование входного сигнала в выходной при этом происходит следующим образом:

$$i_t = \sigma\left(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i\right),$$
$$f_t = \sigma\left(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f\right),$$
$$o_t = \sigma\left(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_{t-1} + b_o\right),$$

$$m_t = o_t \odot h(c_t),$$
$$y_t = \phi(W_{ym} m_t + b_y),$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx} x_t + W_{cm} m_{t-1} + b_c),$$

где W_{cx}, W_{ix}, W_{fx}, W_{ox} – веса входов, W_{cm}, W_{im}, W_{fm}, W_{om} – веса состояний ячеек, b_o, b_i, b_f – смещения, W_{ic}, W_{fc}, W_{oc} – веса связей между ячейками и слоем выходного фильтра. W_{ym} и b_y – вес и смещение для выхода. σ, g, h представляют собой некоторые нелинейные функции.

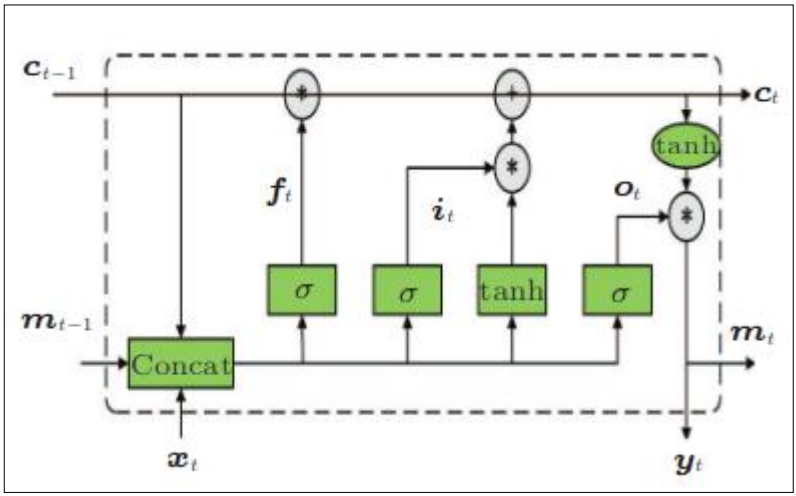


Рис. 1. Структура ячейки LSTM
Fig. 1. The structure of LSTM cell

Двунаправленная сеть долгой краткосрочной памяти комбинирует классическую LSTM-сеть, которая обрабатывает последовательность данных от её начала до конца, с другой LSTM-сетью, которая рассматривает последовательность в обратном порядке.

Анализ работ по близкой тематике показал, что указанные нейросетевые архитектуры широко применяются в задачах обработки естественного языка (см. разд. 1). Эксперименты, проведенные на используемом в работе текстовом корпусе, подтвердили эффективность использования LSTM- и BLSTM-сетей при проведении классификации на основании векторных представлений слов в сравнении с сетями прямого распространения и классическими рекуррентными сетями. В табл. 3 представлен результат сравнения нейросетевых моделей для случая бинарной классификации текстов (классификация в зависимости от того, содержит ли предложение биографическую информацию).

Табл. 3. Сравнение архитектур нейронных сетей на примере бинарой классификации
Table 3. The comparison of neural architectures for the task of binary classification

Архитектура сети	Accuracy (%)	Precision (%)	Recall (%)	F-мера(%)
FNN	86	86,01	93,89	89,78
RNN	89,5	89,86	94,66	92,19
LSTM	91,5	92,18	95,66	93,89
BLSTM	91,5	91,99	95,42	93,63

При проведении классификации фрагментов текстов в данной работе использовались нейросетевые модели, реализованные при помощи средств библиотеки Keras [23]. В качестве функций активации для рекуррентных сетей были выбраны гиперболический тангенс на внутренних слоях и функция Softmax для выходного слоя. Для сетей прямого распространения – логистическая функция на всех слоях. Размер обрабатываемых фрагментов данных (batch size) – 8. Использованный оптимизационный алгоритм – adaptive moment estimation (the Adam optimization). При обучении сетей проводилась дропаут-регуляризация с вероятностью 0,5. Количество нейронов в рекуррентных слоях варьировалось от 16 до 128 при глубине сети в 1-2 скрытых слоя. В итоге для каждой архитектуры были обучены несколько сетей, из которых по результатам на обучающей выборке была выбрана модель, допущенная до экзамена на тестовой выборке. Фрагменты исходного кода, использованные для построения моделей, можно получить по ссылке [24].

4. Результаты

В табл. 4 представлены результаты мультиклассовой классификации по 11 результирующим классам («Фрагменты, не содержащие биографическую информацию» и 10 классов фрагментов, содержащих биографические сведения).

Оценка качества классификации проводилась с использованием следующих метрик: точности (accuracy, то есть количества совпадений фактического и прогнозируемого классов, %) и F-меры (F-score, которая в случае мультиклассовой классификации определялась как средняя величина значений F-меры, рассчитанных для каждого класса по показателям точности (precision) и полноты (recall)).

Расчет точности классификации:

$$Accuracy = T / N ,$$

где T – количество фрагментов, по которым классификатор принял верное решение, N – общее количество документов.

Расчет F-меры:

$$Precision_n = TP / (TP + FP),$$

$$Recall_n = TP / (TP + FN),$$

$$F\_score_n = 2 * Precision * Recall / (Precision + Recall),$$

$$F\_score = \frac{1}{N} \sum_{n=1}^{N} F\_score_n ,$$

где TP – истинно-положительное решение, FP – ложно-положительное решение, FN – ложно-отрицательное решение, n –номер конкретного класса, N – количество классов.

Табл. 4. Результаты мультиклассовой классификации
Table 4. The results of multi-class classification

Способ представления текстов	**Архитектура сети**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F-мера (%)**
Bag-of-Words	FNN	80,2	86,64	85,17	85,9
Bag-of-Words + TF-IDF	FNN	84,16	86,87	88,27	87,57
Bag-of-Ngrams + TF-IDF	FNN	82,18	86,44	86,6	86,52
Word2Vec	LSTM	89,11	91,46	90,79	91,13
Word2Vec	BLSTM	90,1	92,49	91,8	92,15

Наилучшие результаты при проведении классификации по всем классам были достигнуты с использованием двунаправленной сети долгой краткосрочной памяти и представления текстов при помощи Word2Vec.

Результаты модели, показавшей минимальные ошибки на тестовой выборке (далее – Модель А), были сравнены с результатами двух каскадных архитектур нейросетевых моделей (рис. 2).

Во втором случае (Модель Б) на первом этапе проводится бинарная классификация фрагментов текстов. Далее предложения, классифицированные как содержащие биографическую информацию, поступают на вход модели для мультиклассовой классификации. В итоге для каждого фрагмента сначала определяется, содержит ли он биографическую информацию, и если содержит, то определяется тип этой информации.

В третьем случае (Модель В) результирующий вектор сети для мультиклассовой классификации подается на вход сети прямого распространения одновременно с представлением текста в виде модели Bag-of-Words + TD-IDF. Сравнение результатов трех моделей для мультиклассовой классификации представлено в табл. 5.

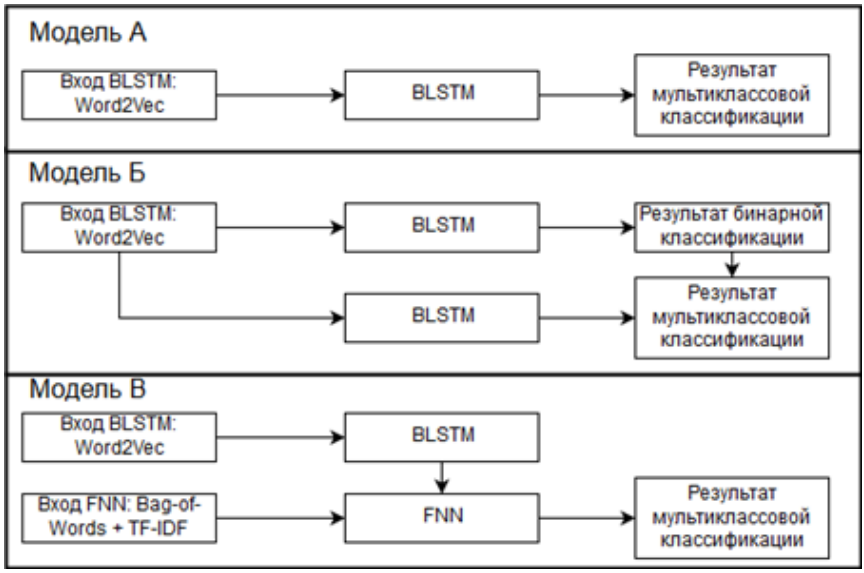


Рис. 2. Модели мультиклассовой классификации
Fig. 2. The models for multi-class classification

Табл. 5. Сравнение моделей для мультиклассовой классификации
Table 5. The comparison of multi-class classification models

Модель	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F-мера `(%)**
А	90,1	92,49	91,8	92,15
Б	93,07	95,82	93,35	94,57
В	94,06	96,37	94,36	95,36

В табл. 6 приводятся примеры автоматической классификации предложений при помощи модели, показавшей лучшие результаты на тестовой выборке. Первый текст представляет собой биографическую статью из онлайн-энциклопедии «Википедия», не входящую в корпус. Второй текст является новостью, размещенной на портале «газета.ru» [25].

Табл. 6. Примеры автоматической классификации предложений
Table 6. The examples of automatic sentences classification

Предложение	**Тип предложения (определен автоматически)**
Пример 1. «Дьяконов-Дьяченков, Георгий Иванович» (источник – «Википедия», 2018)	

Георгий Иванович Дьяконов родился 17 марта 1924 году в Москве в театральной семье.	Рождение
Отец Иван Дьяконов, родом из Оренбургской области, был артистом оперетты, который играл во многих театрах включая Москву, где у него и родился сын.	Информация о родительской семье
В 1941 году, приписав себе год, добровольцем ушёл на фронт, воевал зенитчиком в 205-м зенитно-артиллерийском полку 73-й зенитной дивизии РГК, был тяжело ранен, отличился в боях.	Род занятий
Карьеру актёра начал в своем родном городе Бугуруслане.	Место жительства
На гастролях по Украине выступал под псевдонимом Дьяченков, который позже включил в официальную фамилию.	Профессиональные события
С 1950 года выступал в Тюменском драматическом театре, где за 34 года сыграл около 200 ролей.	Место работы
Его талант отмечали на столичных гастролях Михаил Ульянов и Юрий Яковлев.	Профессиональные события
Умер 4 февраля 1991 года в Тюмени.	Смерть
Пример 2. «Зампред ЦБ Торшин уходит с поста» (источник – «газета.ru», 30.11.2018)	
Зампред Банка России Александр Торшин покидает свой пост в связи с выходом на пенсию, говорится в сообщении ЦБ.	Профессиональные события
Торшин с 1999 по 2001 годы занимал должность статс-секретаря-заместителя генерального директора государственной корпорации «Агентство по реструктуризации кредитных организаций».	Род занятий
С 2001 по 2015 годы являлся членом Совета Федерации.	Место работы
В январе 2015 года он был назначен зампредседателем Центрального банка России.	Место работы
Ранее ФБР сообщало о наличии электронной переписки россиянки Марии Бутиной, подозреваемой в шпионаже в пользу России, с высокопоставленным сотрудником Центробанка России — по сообщениям СМИ, с заместителем председателя ЦБ Александром Торшиным.	Личные события

5. Заключение

В работе предложен подход к автоматическому поиску фрагментов, содержащих биографическую информацию в тексте на естественном языке, основанный на применении нейронных сетей. Для обучения продемонстрированных в работе нейросетевых моделей был составлен корпус

биографических текстов, содержащий биографические статьи, размещённые в онлайн-энциклопедии «Википедия». На основании разработанного корпуса были протестированы различные методы представления текстов и различные архитектуры нейронных сетей. Предложенный в работе подход демонстрирует достаточно высокие результаты на тестовой выборке (F-мера – 95,36%, точность классификации – 94,06%).

В перспективе планируется провести экспериментальные исследования на других данных, в том числе на биографических текстах, не отличающихся явной хронологией изложения, а также осуществить извлечение фактов из предложений, содержащих биографическую информацию, в структурированном виде.

Список литературы

[1]. Терпугова А. В. Биографический текст как объект лингвистического исследования. Автореферат дис. кандидата филологических наук. Ин-т языкознания РАН, Москва, 2011, 26 стр.

[2]. Manning C., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge University Press, 2008. 506 p.

[3]. Адамович И. М., Волков О. И. Система извлечения биографических фактов из текстов исторической направленности. Системы и средства информатики, том 25, вып. 3, 2015 г., стр. 235-250.

[4]. Cybulska, A., Vossen, P. Historical Event Extraction From Text. In Proc. of 5th ACL-HLT Workshop on Language Technology on Cultural Heritage, 2011, pp. 39–43.

[5]. Hienert D., Luciano F. Extraction of Historical Events from Wikipedia. Lecture Notes in Computer Science, vol. 7540, 2015, pp. 16–28.

[6]. Santos C., Xiang B., Zhou B. Classifying Relations by Ranking with Convolutional Neural Networks. In Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 2015, pp. 626-634.

[7]. Meerkamp P., Zhou Z. Information Extraction with Character-level Neural Networks and Free Noisy Supervision. Cornell University Library [электронный ресурс]. 2016. URL: https://arxiv.org/abs/1612.04118 (дата обращения 21.09.2018).

[8]. Homma Y., Sadamitsu K., Nishida K., Higashinaka R., Asano H., Matsuo Y. A Hierarchical Neural Network for Information Extraction of Product Attribute and Condition Sentences. In Proc. of the Open Knowledge Base and Question Answering (OKBQA), 2016, pp. 21-29.

[9]. Arkhipenko K., Kozlov I., Trofimovich J., Skorniakov K., Gomzin A., Turdakov D. Comparison of Neural Architectures for Sentiment Analysis of Russian Tweets. In Proc. of the International Conference “Dialogue 2016”, 2016, pp. 50-58.

[10]. Андрианов И.А., Майоров В.Д., Турдаков Д.Ю. Современные методы аспектно-ориентированного анализа эмоциональной окраски. Труды ИСП РАН, том 27, вып. 5, 2015 г., стр. 5-22. DOI: 10.15514/ISPRAS-2015-27(5)-1.

[11]. Пархоменко П.А., Григорьев А.А., Астраханцев Н.А. Обзор и экспериментальное сравнение методов кластеризации текстов. Труды ИСП РАН, том 29, вып. 2, 2017 г., стр. 161-200. DOI: 10.15514/ISPRAS-2017-29(2)-6.

[12]. Ravuri S., Stolcke A. A Comparative Study of Recurrent Neural Network Models for Lexical Domain Classification. In Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 6075-6079
[13]. Yogatama D., Dyer C., Ling W., Blunsom P. Generative and discriminative text classification with recurrent neural networks. arXiv preprint arXiv:1703.01898, 2017.
[14]. Chen G., Ye D., Xing Z., Chen J., Cambria E. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In Proc. of the International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2377-2383.
[15]. Валгина Н.С., Розенталь Д.Э., Фомина М.И. Современный русский язык. Учебник. 6-е изд., перераб. и доп. Москва, Логос, 2002, 528 стр.
[16]. Википедия. Свободная энциклопедия. URL: https://ru.wikipedia.org/ (дата обращения: 26.11.2018).
[17]. Глазкова А. В. Формирование текстового корпуса для автоматического извлечения биографических фактов из русскоязычного текста. Современные информационные технологии и ИТ-образование, том 14, вып. 4, 2018 г.
[18]. Корпус биографических текстов, URL https://sites.google.com/site/utcorpus/ (дата обращения: 01.12.2018).
[19]. Морфологический анализатор pymorphy2, URL: https://pymorphy2.readthedocs.io/en/latest/ (дата обращения: 01.12.2018).
[20]. Mikolov T., Sutskever I., Chen K., Corrado G. S., Dean J. Distributed representations of words and phrases and their compositionality. In Proc. of the 26th International Conference on Neural Information Processing Systems, vol. 2, 2013, pp. 3111-3119.
[21]. Hochreiter S., Schmidhuber J. Long Short-term Memory. Neural computation, vol. 9, № 8, 1997, pp. 1735-1780.
[22]. Bai T., Dou H. J., Zhao W. X., Yang D. Y., Wen J. R. An Experimental Study of Text Representation Methods for Cross-Site Purchase Preference Prediction Using the Social Text Data. Journal of Computer Science and Technology, vol. 32, №. 4, 2017, pp. 828-842.
[23]. Keras: The Python Deep Learning library. URL: https://keras.io/ (дата обращения: 17.11.2018).
[24]. URL: https://github.com/oldaandozerskaya/biographical_samples.git (дата обращения: 27.12.2018).
[25]. газета.ru. URL: https://www.gazeta.ru/ (дата обращения: 09.12.2018).

Automatic search for fragments containing biographical information in a natural language text

A.V. Glazkova <a.v.glazkova@utmn.ru>
University of Tyumen,
6, Volodarsky st., Tyumen, 625003, Russia

Abstract. The search and classification of text documents are used in many practical applications. These are the key tasks of information retrieval. Methods of text searching and classifying are used in search engines, electronic libraries and catalogs, systems for collecting and processing information, online education and many others. There are a large number of particular applications of these methods, but each such practical task is characterized, as a rule, by weak formalizability and narrow objectivity. Therefore, it requires individual study and its own approach to the solution. This paper discusses the task of automatically searching

and typing text fragments containing biographical information. The key problem in solving this problem is to conduct a multi-class classification of text fragments, depending on the presence and type of biographical information contained in them. After reviewing the related works, the author concluded that the use of neural network methods is promising and widespread for solving such problems. Based on this conclusion, the paper compares various architectures of neural network models, as well as basic text presentation methods (Bag-Of-Words, TF-IDF, Word2Vec) on a pre-assembled and marked corpus of biographical texts. The article describes the steps involved in preparing a training set of text fragments for teaching models, methods for text representation and classification methods chosen for solving the problem. The results of the multi-class classification of text fragments are also presented. The examples of automatic search for fragments containing biographical information are shown for the texts that did not participate in the model learning process.

Keywords: text classification; natural language processing; word embedding; neural networks; biographical text.

DOI: 10.15514/ISPRAS-2018-30(6)-12

For citation: Glazkova A.V. Automatic search for fragments containing biographical information in a natural language text. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 221-236 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-12

References

[1]. Terpugova A.V. Biographical text as an object of linguistic researchю. Author's abstract of the PhD thesis. Institute of Linguistics RAS, Moscow, 2011, 26 p. (in Russian).
[2]. Manning C., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge University Press, 2008. 506 p.
[3]. Adamovich I.M., Volkov O.I. The system of facts extraction from historical texts. Sistemy i sredstva informatiki [Systems and Means of Informatics], vol. 25, № 3, 2015, p. 235-250 (in Russian).
[4]. Cybulska, A., Vossen, P. Historical Event Extraction From Text. In Proc. of 5th ACL-HLT Workshop on Language Technology on Cultural Heritage, 2011, pp. 39–43.
[5]. Hienert D., Luciano F. Extraction of Historical Events from Wikipedia. Lecture Notes in Computer Science, vol. 7540, 2015, pp. 16–28.
[6]. Santos C., Xiang B., Zhou B. Classifying Relations by Ranking with Convolutional Neural Networks. In Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 2015, pp. 626-634.
[7]. Meerkamp P., Zhou Z. Information Extraction with Character-level Neural Networks and Free Noisy Supervision. Cornell University Library [электронный ресурс]. 2016. URL: https://arxiv.org/abs/1612.04118 (дата обращения 21.09.2018).
[8]. Homma Y., Sadamitsu K., Nishida K., Higashinaka R., Asano H., Matsuo Y. A Hierarchical Neural Network for Information Extraction of Product Attribute and Condition Sentences. In Proc. of the Open Knowledge Base and Question Answering (OKBQA), 2016, pp. 21-29.
[9]. Arkhipenko K., Kozlov I., Trofimovich J., Skorniakov K., Gomzin A., Turdakov D. Comparison of Neural Architectures for Sentiment Analysis of Russian Tweets. In Proc. of the International Conference "Dialogue 2016", 2016, pp. 50-58.

[10]. Andrianov I., Mayorov V., Turdakov D. Modern Approaches to Aspect-Based Sentiment Analysis. Trudy ISP RAN/Proc. ISP RAN, vol. 27, №. 5, 2015 г., p. 5-22 (in Russian). DOI: 10.15514/ISPRAS-2015-27(5)-1.
[11]. Parhomenko P.A., Grigorev A.A., Astrakhantsev N.A. A survey and an experimental comparison of methods for text clustering: application to scientific articles. Trudy ISP RAN/Proc. ISP RAN, vol. 29, №. 2, 2017 г., p. 161-200 (in Russian). DOI: 10.15514/ISPRAS-2017-29(2)-6.
[12]. Ravuri S., Stolcke A. A Comparative Study of Recurrent Neural Network Models for Lexical Domain Classification. In Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 6075-6079
[13]. Yogatama D., Dyer C., Ling W., Blunsom P. Generative and discriminative text classification with recurrent neural networks. arXiv preprint arXiv:1703.01898, 2017.
[14]. Chen G., Ye D., Xing Z., Chen J., Cambria E. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In Proc. of the International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2377-2383.
[15]. Valgina N.S., Rosental D.E., Fomina M.I. Modern Russian Language. Moscow, Logos, 2002, 528 p. (in Russian).
[16]. Wikipedia. The free encyclopedia. URL: https://ru.wikipedia.org/, accessed 26.11.2018.
[17]. Glazkova A. V. Building a text corpus for automatic biographical facts extraction from Russian texts. Sovremennyye informatsionnyye tekhnologii i IT-obrazovaniye [Modern Information Technologies and IT-education], vol 14, №. 4, 2018 (in Russian).
[18]. The corpus of biographical texts, URL https://sites.google.com/site/utcorpus/, accessed 01.12.2018.
[19]. Morphological analyzer pymorphy2, URL: [19]. https://pymorphy2.readthedocs.io/en/latest/, accessed 01.12.2018.
[20]. Mikolov T., Sutskever I., Chen K., Corrado G. S., Dean J. Distributed representations of words and phrases and their compositionality. In Proc. of the 26th International Conference on Neural Information Processing Systems, vol. 2, 2013, pp. 3111-3119.
[21]. Hochreiter S., Schmidhuber J. Long Short-term Memory. Neural computation, vol. 9, № 8, 1997, pp. 1735-1780.
[22]. Bai T., Dou H. J., Zhao W. X., Yang D. Y., Wen J. R. An Experimental Study of Text Representation Methods for Cross-Site Purchase Preference Prediction Using the Social Text Data. Journal of Computer Science and Technology, vol. 32, №. 4, 2017, pp. 828-842.
[23]. Keras: The Python Deep Learning library. URL: https://keras.io/, accessed 17.11.2018.
[24]. URL: https://github.com/oldaandozerskaya/biographical_samples.git, accessed 27.12.2018.
[25]. [gazeta.ru]. URL: https://www.gazeta.ru/, accessed 09.12.2018.

Система операторов для пространственно-временного анализа динамических сцен

[1] *К.С. Петрищев <k_petrishchev@ispras.ru>*
[1] *В.А. Золотов <vladislav.zolotov@ispras.ru>*
[1,2,3] *В.А. Семенов <sem@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*
[2] *Московский физико-технический институт, 141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9*
[3] *НИУ Высшая школа экономики, 101978, Россия, г. Москва, ул. Мясницкая, д. 20*

Аннотация. В работе предлагается развитая система топологических, метрических, ориентационных и временных операторов для комплексного анализа пространственно-временных данных. Система допускает комбинированное использование методов количественного и качественного анализа, необходимых как для установления первичных фактов, так и для продукции новых знаний на основе установленных фактов. Система операторов представляется перспективной для решения задач пространственно-временного (4D) моделирования и планирования индустриальных проектов и, в частности, для спецификации и обнаружения нетривиальных конфликтов в календарно-сетевых графиках проектов.

Ключевые слова: качественный анализ; пространственно-временной анализ; моделирование динамических сцен

DOI: 10.15514/ISPRAS-2018-30(6)-13

Для цитирования: Петрищев К.С., Золотов В.А., Семенов В.А. Система операторов для пространственно-временного анализа сцен. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 237-258. DOI: 10.15514/ISPRAS-2018-30(6)-13

1. Введение

Стремительный рост объемов информации и необходимость ее всестороннего анализа приводят к развитию новых методов работы с многомерными и пространственно-временными данными. Класс приложений, в которых возникает необходимость поиска, анализа и визуализации больших пространственно-временных данных, чрезвычайно широк и охватывает системы компьютерной графики, анимации, робототехники, автоматизации

проектирования и производства, геоинформатики, визуального моделирования и планирования проектной деятельности. При этом сами данные часто представляются составными геометрическими моделями, что позволяет интерпретировать их как иерархически организованные сцены, состоящие из динамических многомерных объектов.

Для работы с такими данными довольно часто применяются средства качественного анализа (Qualitative Spatial Reasoning QSR), предназначенные для извлечения (продукции) новых знаний на основе установленных тем или иным способом фактов [1]. Несмотря на разнообразие имеющихся формальных систем анализа, наборы операторов не позволяют идентифицировать сложные пространственно-временные отношения между объектами. Существующие программные инструменты, такие как SparQ, GQR, QAT, CLP(QS), сфокусированы на анализе одного из аспектов и применимы в основном для интервального анализа временных рядов событий и продукции выводов о некоторых пространственных отношениях. На практике инструменты, как правило, применяются для анализа простых отношений в несложных сценах. Отсутствие интерфейсов ограничивает их комбинированное использование со средствами количественного анализа, необходимыми, например, для установления первичных фактов, и применение в рамках концепций 4D (пространственно-временного), 5D (пространственно-временного и стоимостного) и multi-D (многомерного) моделирования [2] [3].

Очевидно, что задачи анализа пространственно-временных данных вовсе не ограничиваются приведенными постановками, а их решение должно основываться на более развитом математическом аппарате, применимом к приложениям из разных предметных областей. По-видимому, такой аппарат должен обеспечивать комплексный анализ данных, предусматривать совместное использование методов качественного и количественного анализа, допускать применение к большим данным на основе развитых индексных структур. Вопросы индексирования иерархически организованных пространственно-временных данных, а также способы эффективного исполнения запросов к ним достаточно подробно рассматривались в работах [4] [5] [6] [7].

Настоящая статья адресована проблемам качественного и количественного анализа пространственно-временных данных и выработки общего математического аппарата для этих целей. На основе рассмотрения существующих формальных систем качественного анализа в статье описывается система топологических, метрических, ориентационных и временных операторов для комплексного анализа пространственно-временных данных. В разд. 2 дается обзор существующих популярных систем и формализмов, в разд. 3 подробно описываются предложенные группы операторов.

2. Обзор существующих формальных систем качественного анализа

Формальные системы качественного анализа (Qualitative Reasoning), как правило, основаны на интуитивных понятиях пространственных и временных отношений между физическими объектами окружающего мира [8]. Системы используют фиксированные наборы операторов для установления соответствующих фактов об объектах и вывода на основе них новых знаний. В отличие от количественного анализа (Quantitative Reasoning), основанного на точных числовых оценках взаимного расположения объектов в пространстве и времени, качественный анализ более естественен в использовании и проще в реализации. В ряде случаев он позволяет делать более быстрые выводы благодаря формализации в виде алгебр абстрактных отношений и их реализации с использованием символьных вычислений вместо трудоемких численных расчетов.

За последние три десятка лет было предложено большое число формальных алгебраических систем. Как правило, они ориентированы на пространственные или временные отношения и, тем самым, исключают возможность проведения комплексного анализа. Другим их принципиальным ограничением является невозможность получения первичных знаний об объектах сцены, которые часто доступны лишь в результате моделирования и проведения соответствующих численных расчетов. Данные недостатки суживают возможности применения формальных систем качественного анализа в индустриальных приложениях. Тем не менее, они представляют несомненный интерес с точки зрения выбора базовых операторов и построения единой, функционально развитой системы для комплексного анализа пространственно-временных данных, представимых динамическими сценами пространственных объектов.

В данном разделе кратко остановимся на существующих формальных системах качественного анализа. Наиболее известными примерами временных исчислений являются интервальная алгебра Аллена [9], темпоральные модальные логики [10] и точечная алгебра Вилайна и Каутца [11]. Например, классическая алгебра Аллена строится на семи базовых отношениях, которые могут быть установлены между двумя интервалами на множестве рациональных чисел. С учетом обратных отношений общее число становится равным тринадцати. Данный формализм хорошо подходит для приложений, в которых парные события определяют состояния объектов сцены, например, появление и удаление, изменение положения и т.п.

Примерами популярных исчислений направлений служат система относительных ориентаций Франка и Фрексы [12, 13], система ориентированных точек (Oriented Point Relation Algebra, OPRA), DCC, FlipFlop, QTC, dipole or extended objects [14, 15]. К сожалению, данные системы применимы только для точечных объектов, поскольку строго

сегментируются области возможного расположения основного объекта относительно точки расположения опорного объекта (см. рис. 1).

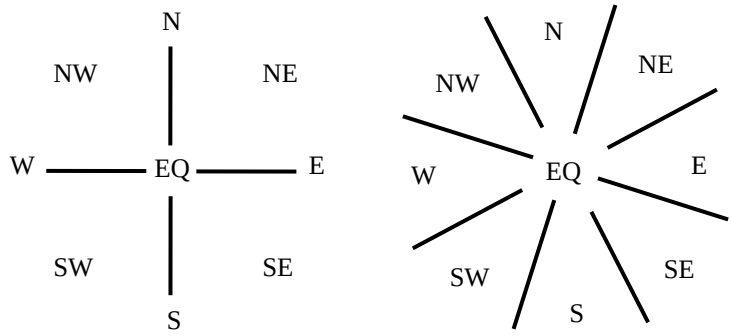


Рис.1 Системы отношений ориентаций Франка и Фрексы
Fig.1 Frank and Freksa's relatve orientation calculi

Для объектов с протяженными границами наиболее распространенными являются алгебра прямоугольников Rectangle Algebra (RA) [16] и система главных направлений Cardinal Direction Calculus (CDC) [17, 18]. В алгебре прямоугольников границы опорного и основного объекта аппроксимируются минимальными ограничивающими прямоугольниками (minimum bounding rectangle (MBR)), а к их главным проекциям применяются отношения интервальной алгебры. В системе главных направлений аппроксимируются границы опорного объекта, а относительное расположение основного объекта определяется на основе тех же интервальных отношений.

Топологическими системами являются мереотопологическая система, системы исчисления пересечений 4-Intersection Calculus и 9-Intersection Calculus [19], системы исчисления связности областей RCC-5, RCC-8 (Region Connection Calculi, RCC) [20], RCC-3D (VRCC-3D++) [21]. Эти формальные системы популярны среди исследователей из-за их простоты и легкости имплементации [22, 23, 24].

На рис.2 приведены восемь возможных отношений связанности RCC-8, которые могут быть установлены между двумя областями. Ниже они перечисляются с соответствующими принятыми аббревиатурами отношений:

- не связаны — disconnected (DC);
- связаны внешне — externally connected (EC);
- частично перекрыты — partial overlap (PO);
- равны — equal (EQ);
- первая область касается второй, находясь внутри — tangential proper part (TPP);

- вторая область касается первой, находясь внутри — tangential proper part inverse (TPPi);
- первая область лежит внутри второй — non-tangential proper part (NTPP);
- вторая область лежит внутри первой — non-tangential proper part inverse (NTPPi).

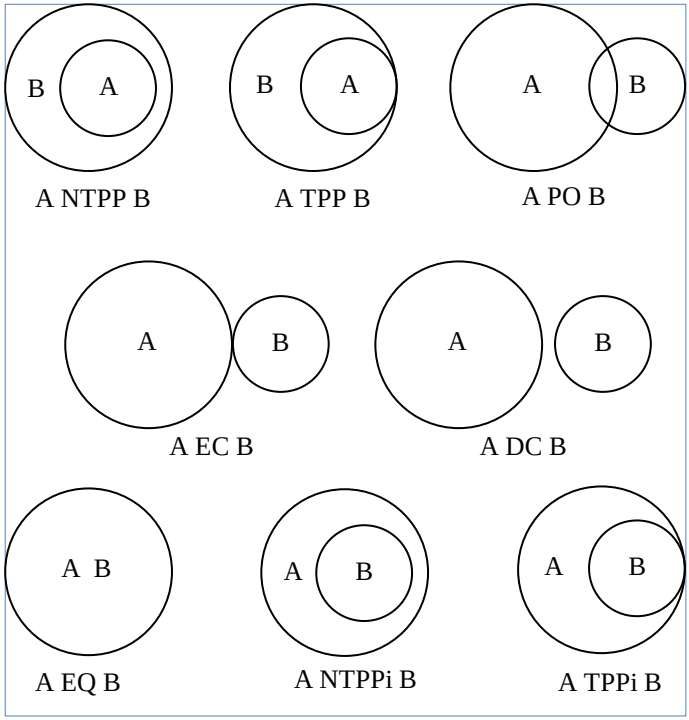


Рис.2 Система топологических операторов RCC-8
Fig.2 RCC-8 topoloigical operators

Приведенные отношения являются попарно исключающими (pairwise disjoint), а сама система – исчерпывающей (jointly exhaustive) в том смысле, что любая физически реализуемая конфигурация областей может быть описана одним из отношений. При этом система не является избыточной и противоречивой, поскольку никакая пара областей не может быть описана более чем одним отношением. Отношения могут сочетаться друг с другом, например, касание областей может быть определено как объединение TPP и NTPP отношений.

Другой популярной системой является RCC-5, которая предполагает только пять базовых отношений, не различая случаи касаний областей. Две другие системы RCC-23 и RCC-62, предложенные позже, наоборот расширили набор

отношений, что существенно повысило их наглядность, однако лишило возможности применения автоматических средств вывода.

В RCC-23 учитывается выпуклость областей. Анализ выпуклых и вогнутых областей различается, причем в системе отношений участвуют и их выпуклые оболочки. Базовые отношения отсутствия связанности DC (disconnected) и внешней связанности EC (externally connected) дополняются предикатами, определяющими находится ли область снаружи outside (O), частично внутри partially inside (P) или полностью внутри completely inside (I) другой области. Например, OOE отношение означает, что первая область лежит внутри выпуклой оболочки второй области, вторая – внутри выпуклой оболочки первой и обе области внешне связаны.

В системе RCC-62 области представляются своими границами (boundaries), внутренними частями (interiors), внешними частями (exteriors), а невыпуклые области – и смешанными частями (insides). 62 типа отношений порождается пересечением соответствующих частей представления. Очевидно, что чем больше типов отношений допускается системой анализа, тем больше пространственных конфигураций может быть описано.

К менее известным следует отнести систему исчисления измерений D-Calculus [25], систему исчисления положений Ternary Point Configuration Calculus [26], а также системы исчисления перекрытий (Occlusion Calculi, OCC), LOS-14, ROC-20, OCS-14.

Упомянутые выше системы ограниченны для практического использования, поскольку оперируют только одним из пространственных аспектов. В связи с этим рядом авторов предприняты попытки обобщения и анализа пространственно-временных отношений. В основе систем STCC (SpatioTemporal Constraint Calculus) [27] и ARCC-8 лежит идея синтеза интервальной алгебры Аллена и топологических отношений RCC-8. Другим примером подобных обобщений является система исчисления траекторий Qualitative Trajectory Calculus (QTC), которая позволяет делать выводы о движущихся объектах [28].

3. Организация и состав системы для качественного и количественного анализа сцен

Поскольку одной из задач является разработка системы операторов для качественного и количественного пространственно-временного анализа сцен, рассмотренные выше формальные алгебраические системы могут служить основой для подобных построений. В разрабатываемой системе удобно выделить следующие группы операторов:

- временные;
- топологические;
- метрические;

- ориентационные.

Заметим, что реализация самих операторов может базироваться на численных расчетах, что не должно препятствовать их дальнейшему использованию при получении качественных оценок. Таким образом, для идентификации нетривиальных пространственно-временных конфликтов методы качественного и количественного анализа могут и должны применяться совместно.

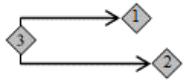
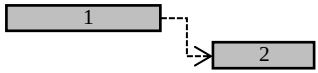
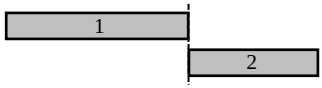
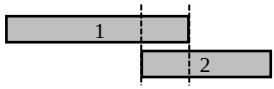
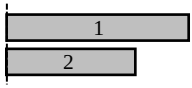
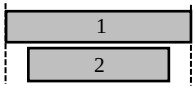
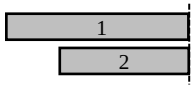
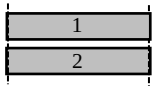
Далее мы ограничимся семантикой этих операторов, опуская детали их возможной реализации. Формальные определения операторов и иллюстрации сведены в табл. 1-4.

3.1 Временные операторы

Прежде всего, заметим, что изменения в динамической сцене могут происходить как мгновенно в дискретные моменты времени, так и непрерывно на протяжении некоторых временных интервалов, концы которых явно или неявно задаются дискретными событиями. Поэтому выделим две группы временных операторов.

Табл. 1. Темпоральные операторы
Table 1. Temporal operators

Оператор	**Формальное представление и иллюстрация**
Before (E1,E2) *E1 < E2*	*E1.time < E2.time* 
AtOnce (E1,E2) *E1 = E2*	*E1.time = E2.time* 
After (E1,E2) *E1 > E2*	*E1.time > E2.time* 
Next (E1,E2)	*neither E1.time ≤ E2.time* *nor ∃E3 E1.time > E3.time > E2.time* 
Previous (E1,E2)	*neither E1.time ≥ E2.time* *nor ∃E3 E1.time < E3.time < E2.time*

	
Fork (E1,E2)	$\exists E3\ E3.time < E1.time \wedge E3.time < E2.time$ 
Independent (E1,E2)	*neither* $\exists E1.time \wedge \exists E2.time$ *nor* $\exists E3\ E1.time < E3.time < E2.time$
Before (I1,I2)	$I1.start < I1.end < I2.start < I2.end$ 
Meets (I1,I2)	$I1.start < I1.end = I2.start < I2.end$ 
Overlaps (I1,I2)	$I1.start < I2.start < I1.end < I2.end$ 
Starts (I1,I2)	$I1.start = I2.start < I1.end < I2.end$ 
During (I1,I2)	$I1.start < I2.start < I2.end < I1.end$ 
Finishes (I1,I2)	$I2.start < I1.start < I1.end = I2.end$ 
AtOnce (I1,I2)	$I1.start = I2.start < I1.end = I2.end$ 

Первая группа включает в себя операторы *Before*, *AtOnce*, *After*, *Next*, *Previous*, *Fork* и *Independent*, которые непосредственно применяются к событиям и используются для того, чтобы определить, как пара или множество событий связаны друг с другом они. К примеру, можно узнать, произошло ли событие *E1* до события *E2* (*Before*), одновременно с ним (*AtOnce*) или после него (*After*). Дополнительные операторы *Next*, *Previous* помогают уточнить, случилось ли одно из этих событий сразу после другого или же какие-то события могли произойти между ними. Операторы *Fork* и *Independent* предназначены для идентификации случаев, когда у событий *E1* и *E2* общий предшественник *E3* или их можно рассматривать как независимые во времени.

Вторая группа повторяет алгебру Аллена с тем отличием, что отношения определяются не между временными интервалами, а между парами событий, которые собственно и задают интервалы.

Например, пару событий *E1,E2 (Before(E1,E2))* можно ассоциировать с интервалом *I1=I(E1,E2)*, а события *E3,E4 (Before(E3,E4))* — с интервалом *I2=I(E3,E4)* и отношения устанавливать непосредственно между *I1* и *I2*.

Следующие операторы *Before*, *Meets*, *Overlaps*, *Starts*, *During*, *Finishes*, и *AtOnce* позволяют установить семь базовых отношения алгебры, которыми могут быть связаны два интервала *I1* и *I2*. Формальные определения и графические иллюстрации операторов приводятся в табл. 1. Формальные параметры *start* и *end* здесь обозначают начало и конец соответствующих временных интервалов, которые задаются временными штампами ассоциируемых событий. Приведенный набор операторов является достаточно развитым для анализа сложных сцен.

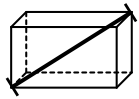
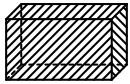
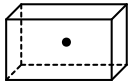
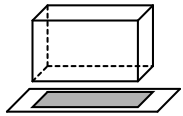
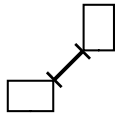
3.2 Метрические операторы и функции измерения

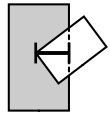
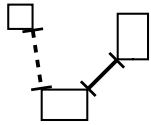
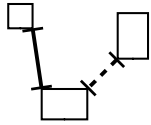
Довольно часто в анализе сцен участвуют геометрические свойства твердых тел, такие как диаметр, площадь поверхности, объем, центр масс и проекции тел на плоскости. Для объектов, граничное представление которых задается простыми геометрическими формами, подобные свойства рассчитываются аналитически. В общем случае это не так, поэтому на практике используется упрощенное полиэдральное представление, полученное в результате аппроксимации (tessellation) исходной геометрии и допускающее подсчет требуемых характеристик. Независимо от способов вычислений будем считать, что для твердых объектов сцены определены функции подсчета диаметра *Diameter(A)*, площади поверхности *Area(A)*, объема *Volume(A)*, центра масс *CentreOfMass(A)* и проекции на заданную плоскость *Projection(A,D)*, определяемую нормалью *D*. Также следует предусмотреть функцию *Distance(A,B)* для расчета расстояния между объектами *A,B* и функцию *Depth(A,B)* для определения глубины взаимопроникновения, т.е. расстояния, на которое твердые объекты *A,B* перекрывают друг друга.

Специальные метрические операторы *CloserThan(A,B,C)* and *FartherThan(A,B,C)* позволяют определить, какой из двух объектов *A,B* находится ближе к третьему объекту *C* и какой из *A,B* находится дальше от *C*. Эти операторы, как правило, используются в тех случаях, когда вердикт может быть получен сравнительно быстро с использованием качественных оценок. Графические иллюстрации, поясняющие данные функции и операторы приведены в табл. 2.

Поскольку положение объектов в сцене и их геометрическое представление подвержены изменениям с течением времени, все перечисленные измерения выполняются для определенного фиксированного момента времени, называемого в дальнейшем модельным временем (focus time) *T*. Модельное время *T* является неявным параметром приведенных выше операторов и функций. В дальнейшем в случаях, когда требуется подчеркнуть зависимость от времени, параметр *T* включается в сигнатуру явно.

Табл. 2. Метрические операторы и функции измерения
Table 2. Metric operators and measurement functions

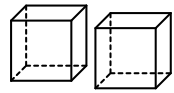
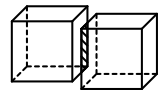
Функции и операторы	**Иллюстрация**
Diameter (A)	
Area (A)	
Volume (A)	
CentreOfMass (A)	
Projection (A, D)	
Distance (A, B)	

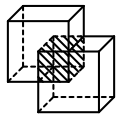
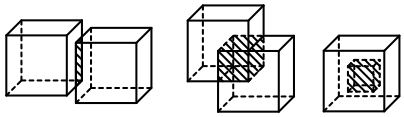
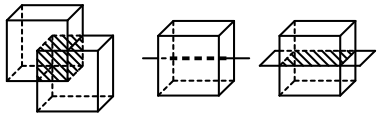
Depth (A, B)	
CloserThan (A, B, C)	
FartherThan (A, B, C)	

3.3 Топологические операторы

Определяется набор топологических операторов, который, в определенной степени, основывается на формальных системах топологических отношений, рассмотренных в обзорном разделе. Выбор операторов обусловлен необходимостью вычислительно эффективной реализации и возможностью их применения при выводе качественных суждений. В таблице 3 приведены формальные определения и иллюстрации для каждого из операторов набора.

Табл. 3. Топологические операторы
Table 3. Topological operators

Оператор 9IM	**Формализация и иллюстрация отношения**
Equal (A, B) $\begin{bmatrix} * & F & F \\ F & * & F \\ F & F & * \end{bmatrix}$	$\sim A^0 \partial B \wedge \sim A^0\, B^- \wedge \sim \partial A B^0 \wedge \sim \partial A B^- \wedge \sim A^- B^0 \wedge A^- \partial B$ 
Disjoint (A, B) $\begin{bmatrix} F & F & * \\ F & F & * \\ * & * & * \end{bmatrix}$	$\sim A^0\, B^0 \wedge \sim \partial A\, B^0 \wedge \sim A^0\, \partial B \wedge \sim \partial A\, \partial B$ 
Touches (A, B) $\begin{bmatrix} F & F & * \\ F & T & * \\ * & * & * \end{bmatrix}$	$\sim A^0\, B^0 \wedge \sim \partial A\, B^0 \wedge \sim A^0\, \partial B \wedge \partial A\, \partial B$ 

Within (A, B) $\begin{bmatrix} * & * & F \\ * & F & F \\ * & * & * \end{bmatrix}$	$\sim A^0 B^- \wedge \sim \partial A\ \partial B \wedge \sim \partial A\ B^-$ 
Contains (A, B) $\begin{bmatrix} * & * & * \\ * & F & * \\ F & F & * \end{bmatrix}$	$\sim \partial A\ \partial B \wedge \sim A^-\ B^0 \wedge \sim A^- \partial B$ *Within (B, A)*
Overlaps(A, B) $\begin{bmatrix} 3 & * & T \\ * & * & * \\ T & * & * \end{bmatrix}$	$A^0\ B^0 \wedge A^-\ B^0 \wedge A^0\ B^-$ 
Intersects (A, B)	$A^0\ B^0 \vee \partial A\ B^0 \vee A^0\ \partial B \vee \partial A\ \partial B$ *Not Disjoint (A, B)* 
Clashes (A, B)	$A^0\ B^0 \vee \partial A\ B^0 \vee A^0\ \partial B$ 

Предложенный набор отчасти повторяет известную пространственно-расширенную модель, основанную на девяти пересечениях (Dimensionally Extended nine-Intersection Model, DE-9IM) и предназначенную для качественного анализа геометрических объектов, для которых определена внутренность, граница и наружная часть. В нотации топологических пространственных операторов эти области обозначаются как A^0, ∂A, и A^- соответственно. Модель абстрактно описывает геометрические объекты через их возможные отношения друг с другом, используя матрицы размера 3×3, называемые матрицами пересечения. Для объектов A и B элементы матрицы хранят максимальную размерность пересечений их областей:

$$\begin{bmatrix} \dim A^0 \cap B^0 & \dim A^0 \cap \partial B & \dim A^0 \cap B^- \\ \dim \partial A \cap B^0 & \dim \partial A \cap \partial B & \dim \partial A \cap B^- \\ \dim A^- \cap B^0 & \dim A^- \cap \partial B & \dim A^- \cap B^- \end{bmatrix}$$

Размерность пустого множества далее обозначается как F (false), непустого — как T (true) или более конкретно: 0 для точек, 1 для линий, 2 для поверхностей и 3 для объемов.

Основные различия между предложенным набором операторов и моделью DE-9IM заключаются в следующем:

- понятия точек, линий и поверхностей интерпретируются в терминах общей топологии, т.е. как имеющие границу, но не имеющие внутренности;
- геометрические объекты предполагаются замкнутыми и ограниченными, при этом они могут состоять из нескольких несвязанных частей;
- операторы предназначены для двумерных и трехмерных геометрических объектов;
- операторы и соответствующие топологические отношения формализованы с использованием оригинальных матриц пересечения;
- операторы применяются для анализа объектов сцены на текущее модельное время.

3.4 Операторы направленности (относительного положения)

Направление – бинарное отношение между упорядоченной парой объектов *A* и *B* в заданной системе отсчета, где *A* – исходный объект, а *B* – целевой объект. В случае точечных объектов для качественного анализа обычно применяют модель кардинальных направлений Франка [29] или, так называемую, звездчатую модель [30]. В двухмерном случае обе модели используют разбиение плоскости на угловые сегменты, с общей вершиной в исходном объекте *A* и сторонами, опирающимися на линии запад-восток и юг-сервер. В зависимости от линии или сегмента, которым может принадлежать целевой объект *B*, фиксируется девять отношений расположения объекта *B* относительно объекта *A* (см. таблицу 4). Однако у обеих моделей есть принципиальные недостатки: в трехмерном случае значительно вырастает число отношений, а в случае объектов с протяженными границами возникают неопределенности.

Чтобы преодолеть первый недостаток и избавиться от необходимости поддержки огромного числа отношений, можно ввести обобщенное отношение и параметризовать его с помощью дополнительного операнда, задающего вектор направления, относительно которого следует определять статус относительного расположения объектов. Второй недостаток носит более фундаментальный характер, поскольку в случае протяженных границ идентификация взаимного расположения объектов становится нетривиальной и требует определения отношения направленности иным образом.

Табл. 4. Модель кардинальных направлений Франка
Таблица 4. Frank's cardinal direction model

Отношение и формализация	**Иллюстрация**
NorthOf (A, B) $A_x = B_x \ \wedge A_y > B_y$	 A B
NorthEastOf (A, B) $A_x > B_x \ \wedge A_y > B_y$	 A B
EastOf (A, B) $A_x > B_x \ \wedge A_y = B_y$	 B A
SouthEastOf (A, B) $A_x > B_x \ \wedge A_y < B_y$	 B A
SouthOf (A, B) $A_x = B_x \ \wedge \ A_y < B_y$	 B A
SouthWestOf (A, B) $A_x < B_x \ \wedge A_y < B_y$	 B A
WestOf (A, B) $A_x < B_x \ \wedge A_y = B_y$	 A B
NorthWestOf (A, B) $A_x < B_x \ \wedge A_y > B_y$	 A B
Equal (A, B) $A_x = B_x \ \wedge A_y = B_y$	 A B

На рис. 3 представлено несколько характерных случаев, в которых идентификация отношений направленности между основным объектом *A* и другими объектами сцены *B,C,D,E* затруднена и требует формализации, отличающейся от той, которая применяется для точечных объектов. Следует отметить, что ранее предпринимались попытки работы с подобными объектами [31]. В их основе обычно лежит идея сепарации объектов плоскостью, перпендикулярной заданному направлению. Если удается провести такую плоскость, то устанавливается строгое отношение направленности расположения объектов. Если сепарирующая плоскость может быть проведена только для отдельных частей объектов, то устанавливается, так называемое, нестрогое (релаксируемое) отношение. Во всех остальных случаях отношение не определено.

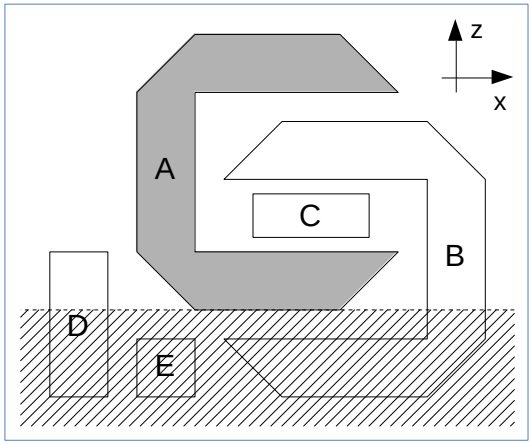


Рис.3 Сцена с неопределенностями в отношениях между объектами
Figure 3 A scene with uncertain relations between objects

Построение сепарирующей плоскости также допускает альтернативы. Обычно используются модели, использующие проекции (projection-based model) и полупространства (halfspace-based model). В первой модели исходный объект протягивается в заданном направлении и ищется пересечение с целевым объектом. Во второй модели строится ограничивающий параллелепипед AABB (Axis Aligned Bounding Box) основного объекта и определяется местоположение второго объекта в политопах, образуемых плоскостями ограничивающего параллелепипеда.

Для примера рассмотрим систему отношений *LeftOf*, *RightOf*, *BackOf*, *FrontOf*, *Above*, *Below*. Для краткости ограничимся формальными определениями отношения *Above*. Другие типы отношений определяются аналогично.

Пусть A и B – объекты в трехмерном эвклидовом пространстве (твердые тела, поверхности, кривые и точки), а $a \in A$, $b \in B$ — точки, принадлежащие объектам A и B соответственно.

Определение 1. (Полупространственная модель, нестрогая интерпретация отношения)

$$Above(A,B) \Leftrightarrow \forall a: \exists b: a_z < b_z$$

Определение 2. (Полупространственная модель, строгая интерпретация отношения)

$$Above(A,B) \Leftrightarrow \forall a,b: a_z < b_z$$

Определение 3. (Проекционная модель, нестрогая интерпретация отношения)

$$Above(A,B) \Leftrightarrow \exists a,b: a_x = b_x \wedge a_y = b_y \wedge a_z < b_z$$

Определение 4. (Проекционная модель, строгая интерпретация отношения)

$$Above(A,B) \Leftrightarrow \forall a: \left(\exists b: a_x = b_x \wedge a_y = b_y \wedge a_z < b_z\right) \wedge \left(\nexists b: a_x = b_x \wedge a_y = b_y \wedge a_z \geq b_z\right)$$

Вернемся к рис. 3. Согласно приведенным определениям идентификация отношений направленности может существенно различаться в зависимости от принятой модели и строгости интерпретации. Например, в рамках проекционной модели объекты *B* и *E* лежат ниже объекта *A* при нестрогой интерпретации отношения, и только объект *E* лежит ниже объекта *A* при строгой интерпретации отношения. Применяя полупространственную модель, объекты *B*, *D*, *E* лежат ниже объекта *A* при нестрогой интерпретации, и только объект *E* лежит ниже объекта *A* при строгой интерпретации.

Табл. 5. Отношения направленности в обобщенной форме
Table 5. Directional relations

Отношение	**Формализация**
InDirectionOf (A, B, D)	$\forall b \in B\ \exists a \in A: \left(\overline{a} \cdot \overline{D}\right) > (\overline{b} \cdot \overline{D})$
StrictlyInDirectionOf (A, B, D)	$\forall a \in A, b \in B: \left(\overline{a} \cdot \overline{D}\right) > (\overline{b} \cdot \overline{D})$

Вместо определения базовых отношений *LeftOf*, *RightOf*, *BackOf*, *FrontOf*, *Above*, *Below* и дополнительных отношений, порождаемых произвольно выбранными ограничивающими параллелепипедами OBB (Oriented Bounding Box), определим отношения направленности в обобщенной форме. Для этого используем дополнительный операнд *D*, обозначающий направление, в котором ведется проверка отношения. Задавая направление *D*, можно смоделировать частные типы отношений.

Формальные определения обобщенных отношений в нестрогой и строгой интерпретациях приводятся в таблице 5. Скалярные и векторные произведения двух векторов, заданные точкой $a \in A$ и направлением *D* здесь и в дальнейшем обозначаются как $\left(\overline{a} \cdot \overline{D}\right)$ и $\overline{a} \times \overline{D}$ соответственно. Первое отношение *InDirectionOf* — слабая форма, обладающая свойством транзитивности. Второе отношение *StrictlyInDirectionOf* — строгая форма, оно транзитивно, иррефлексивно и ассиметрично. Это просто понять, если зафиксировать направление *D*, к примеру, вертикально вверх. В этом случае приведенные формальные условия редуцируются к частным определениям 1,2, рассмотренным ранее. При обобщениях можно было бы основываться и на проекционной модели таким образом, чтобы частные условия совпали с определениями 3,4. Однако это представляется избыточным для обсуждаемых моделей пространственно-временных конфликтов.

3.5 Комбинирование операторов

Операторы рассмотренных видов допускают совместное использование (комбинирование), что обеспечивает задание сложных пространственно-временных условий. Для этого пространственные операторы

интерпретируются в терминах временных событий и интервалов, на протяжении которых они принимают истинное значение. Действительно, при моделировании динамических сцен пространственные отношения между объектами возникают только во временном контексте. Например, чтобы описать конфликт, обусловленный пересечением двух движущихся пространственных объектов *A(t)*, *B(t)* на некотором временном интервале *T* можно воспользоваться следующей конструкцией:

$$Clashes(A,B) \vee Overlaps(T) \Leftrightarrow$$
$$\exists t \in T\ Clashes\big(A(t), B(t)\big)$$

Условие бесконфликтности перемещения объектов на интервале *T* тогда приобретает вид:

$$!\,Clashes(A,B) \vee During(T) \Leftrightarrow$$
$$\forall t \in T\ !\,Clashes\big(A(t), B(t)\big)$$

Для комбинирования нескольких пространственных или временных условий используются обычные логические операторы. Например, условие устойчивости пирамидальной конструкции, состоящей из пространственных объектов *A*, *B*, *C*, можно представить следующим образом:

$$Above\ (A,B) \vee Above\ (B,C)$$

Условие нарушения устойчивости конструкции тогда приобретает вид

$$!\,Above\ (A,B) \wedge !\,Above\ (B,C)$$

Пространственные и временные операторы широко применяются в специализированных СУБД. В работе [31] описывается пространственное расширение языка запросов SQL, обеспечивающее содержательную работу с информационными моделями зданий и сооружений. В работах [32] и [33] описываются реализации СУБД, учитывающие как пространственный, так и временной аспект работы с данными. Тем не менее, реализации имеют ряд ограничений, связанных с дискретным по времени представлением пространственных объектов (snapshots). Ограничения не являются принципиальными для типовых геоинформационных приложений, однако оказываются критичными для моделирования динамических сцен с непрерывно перемещаемыми объектами. Более развитые инструменты для качественного пространственно-временного анализа представлены в статье [34], авторы которой также попытались доработать язык SQL.

В [27] делаются теоретические выводы относительно совместного применения интервальной алгебры Аллена и системы топологических отношений RCC-8 без учета других пространственных отношений. Мюллер [35] попытался построить формальную модель для пространственно-временных регионов на основе RCC-8, в дальнейшем доработанную Дэвисом [36].

Приведенные работы показывают, что существующие формальные системы имеют серьезные ограничения для качественного пространственно-временного анализа. Предложенная в данной статье система операторов,

наоборот, является довольно развитой и при этом допускает содержательное комбинирование пространственных и временных операторов.

4. Заключение

Таким образом, проведен обзор существующих формальных систем качественного анализа. На его основе предложена развитая система топологических, метрических, ориентационных и временных операторов для комплексного анализа пространственно-временных данных. Система допускает комбинированное использование методов количественного и качественного анализа, необходимых как для установления первичных фактов, так и для продукции новых знаний на основе установленных фактов. Система операторов представляется перспективной для решения задач 4D моделирования индустриальных проектов и, в частности, для идентификации нетривиальных пространственных конфликтов в календарно-сетевых графиках.

Список литературы

[1]. C. Freksa. Spatial computing. Cognitive and Linguistic Aspects of Geographic Space, Heidelberg, Springer Berlin, 2013, pp. 23-42.

[2]. D. Heesom, L. Mahdjoubi. Trends of 4D CAD applications for construction planning. Construction Management and Economics, vol. 22, 2004, pp. 171-182.

[3]. M.L.A.E. Borges, I.C. de Souza, S. Melo, J.P. Giesta. 4D Building Information Modelling: A Systematic Mapping Study. In Proc. of the 35th International Symposium on Automation and Robotics in Construction, Berlin, 2018.

[4]. В. Золотов, В. Семенов. Современные методы поиска и индексации многомерных данных в приложениях моделирования больших динамических сцен. Труды ИСП РАН, т. 24, 2013 г, стр. 381-416. DOI: 10.15514/ISPRAS-2013-24-17.

[5]. V. Semenov, K. Kazakov, S. Morozov, O. Tarlapan, V. Zolotov, T. Dengenis. 4D modeling of large industrial projects using spatio-temporal decomposition. In eWork and eBusiness in Architecture, Engineering and Construction, 2010, pp. 89-95.

[6]. Золотов В.А., Семенов В.А. Исследование и развитие метода декомпозиции для анализа больших пространственных данных. Труды ИСП РАН, том 25, 2013 г., стр. 121-166. DOI: 10.15514/ISPRAS-2013-25-8.

[7]. Золотов В.А., Семенов В.А. Перспективные схемы пространственно-временной индексации для визуального моделирования масштабных индустриальных проектов. Труды ИСП РАН, том 26, вып. 2, 2014 г., стр. 175-196. DOI: 10.15514/ISPRAS-2014-26(2)-8.

[8]. A.G. Cohn, S.M. Hazarika. Qualitative Spatial Representation and Reasoning: An Overview. Fundamenta Informaticae, vol. 46, № 1-2, 2001, pp. 1-29.

[9]. J.F. Allen. Maintaining knowledge about temporal intervals. Communications of the ACM, vol. 26, issue 11, 1983, pp. 832-843.

[10]. E.A. Emerson. Chapter 16 - Temporal and Modal Logic. In Handbook of Theoretical Computer Science, B: Formal Models and Semantics, Elsevier, 1990, pp. 995-1072.

[11]. M.B. Vilain, H. Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In Proc. of the 5th National Conference on Artificial Intelligence, 1986, pp. 377-382.

[12]. A.U. Frank. Qualitative spatial reasoning about distances and directions in geographic space. Journal of Visual Languages & Computing, vol. 3, № 4, 1992, pp. 343-371.

[13]. C. Freksa. Using orientation information for qualitative spatial reasoning. Lecture Notes in Computer Science, vol. 639, 1992, pp. 162-178.
[14]. G.F. Ligozat. Qualitative triangulation for spatial reasoning. Lecture Notes in Computer Science, vol. 716, 1993, pp. 54-68.
[15]. R. Moratz, J. Renz, D. Wolter. Qualitative Spatial Reasoning About Line Segments. In Proc. of the 14th European Conference on Artificial Intelligence, 2000, pp. 234-238.
[16]. P. Balbiani, J.-F. Condotta, L.F. del Cerro. A new tractable subclass of the rectangle algebra. In Proc. of the 16th International joint conference on Artifical Intelligence, vol. 1, 1999, pp. 442-447.
[17]. R.K. Goyal, M.J. Egenhofer. Similarity of cardinal directions. Lecture Notes in Computer Science, vol. 2121, 2001, pp. 36-55.
[18]. S. Skiadopoulos, M. Koubarakis. On the consistency of cardinal direction constraints. Artificial Intelligence, vol. 163, № 1, 2005, pp. 91-135.
[19]. M.J. Egenhofer, R.D. Franzosa. Point-set topological spatial relations. International Journal of Geographical Information Systems, vol. 5, № 2, 1991, pp. 161-174.
[20]. D. A. Randell, Z. Cui, A. G. Cohn. A spatial logic based on regions and connection. In Proc. of the 1st International Conference Principles of Knowledge Representation and Reasoning, 1992, стр. 165-176.
[21]. N. Eloe. VRCC-3D+: Qualitative spatial and temporal reasoning in 3 dimensions. PhD Thesis, Missouri University of Science and Technology, 2015.
[22]. A.G. Cohn, J. Renz. Qualitative Spatial Representation and Reasoning. In Handbook of Knowledge Representation, Elsevier, 2008, pp. 551-596.
[23]. M.J. Egenhofer, J. Sharma, D.M. Mark. A Critical Comparison of the 4-Intersection and 9-Intersection Models for Spatial Relations: Formal Analysis. In Proc. of the International Research Symposium on Computer-based Cartography, AutoCarto 11, 1993, pp. 1-12.
[24]. T. Mossakowski, R. Moratz. Qualitative Reasoning about Relative Direction of Oriented Points. Artificial Intelligence, vol. 180-181, 2012, pp. 34-45.
[25]. K. Zimmermann. Measuring without measures the D-calculus. Lecture Notes in Computer Science, vol. 988, 1995, pp. 59-67.
[26]. R. Moratz, B. Nebel, C. Freksa. Qualitative spatial reasoning about relative position. Lecture Notes in Computer Science, vol. 2685, 2002, pp. 385-400.
[27]. A. Gerevini, B. Nebel. Qualitative Spatio-Temporal Reasoning with {RCC-8} and Allen's Interval. In Proc. of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, 2002, pp. 312-316.
[28]. N. Van de Weghe, A. Cohn, G. De Tre, P. De Maeyer. A qualitative trajectory calculus as a basis for representing moving objects in geographical information systems. Control and Cybernetics, vol. 35, № 1, 2006, pp. 97-119.
[29]. A. Frank. Qualitative Spatial Reasoning: Cardinal Directions as an Example. International Journal of Geographical Information Systems, vol. 10, issue 3, 1996, pp. 269-290.
[30]. J. Renz, D. Mitra. Qualitative Direction Calculi with Arbitrary Granularity. Lecture Notes in Computer Science, vol. 3157, 2004, pp. 65-74.
[31]. A. Borrmann, E. Rank. Query support for BIMs using semantic and spatial conditions. In Handbook of Research on Building Information Modeling and Construction Informatics: Concepts and Technologies, IGI Global, 2009, pp. 405-450.
[32]. A. Carvalho, C. Ribeiro, A. Augusto de Sousa. A Spatio-temporal Database System Based on TimeDB and Oracle Spatial. Research and Practical Issues of Enterprise

Information Systems, IFIP International Federation for Information Processing, vol. 205, 2006, pp. 11-20, Boston, MA, Springer US.

[33]. C. Xinmin Chen, C. Zaniolo. SQLST : A Spatio-Temporal Data Model and Query Language. Lecture Notes in Computer Science, vol. 1920, 2000, pp. 96-111.

[34]. M. Erwig, M. Schneider. Developments in spatio-temporal query languages. In Proc. of the Tenth International Workshop on Database and Expert Systems Applications, DEXA 99, 1999, pp. 441-449.

[35]. P. Muller. A Qualitative Theory of Motion Based on Spatio-Temporal Primitives. In Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning *(*KR '98), 1998, pp. 131-141.

[36]. E. Davis. Describing spatial transitions using mereotopological relations over histories. Technical Report #2000-809, New York University, 2000.

A system of operators for spatial-temporal analysis of dynamic scenes

[1] *K.S. Petrishchev <k_petrishchev@ispras.ru>*
[1] *V.A. Zolotov <vladislav.zolotov@ispras.ru>*
[1,2,3] *V.A. Semenov <sem@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *Moscow Institute of Physics and Technology (State University), 9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia*
[3] *National Research University, Higher School of Economics 20, Myasnitskaya Ulitsa,Moscow, 101978, Russia*

Abstract. The rapid growth in the volume of information and the need for its comprehensive analysis lead to the development of new methods of working with multidimensional and space-time data. To work with such data, Qualitative Spatial Reasoning is often used to extract (produce) new knowledge based on facts established in one way or another. Despite the variety of formal analysis systems available, sets of operators do not allow for the identification of complex space-time relationships between objects. Existing software tools, such as SparQ, GQR, QAT, CLP (QS), are focused on analyzing one aspect and are applicable mainly for interval analysis of time series of events and production of conclusions about some spatial relationships. In practice, tools are usually used to analyze simple relationships in simple scenes. The lack of interfaces limits their combined use with quantitative analysis tools, necessary, for example, to establish primary facts, and use within the framework of the concepts of 4D (space-time), 5D (space-time and cost) and multi-D (multidimensional) modeling. This paper proposes a system of topological, metric, directional and temporal operators for complex spatial-temporal analysis of dynamic scenes. This system allows combined usage of qualitative and quantitative analysis methods, which is essential not only for determining initial facts, but also for producing new knowledge based on these facts. The system of operators proposed is deemed prospective for problems of spatial-temporal (4D) modeling and planning of industrial projects, and particularly for specifying and detecting of non-trivial conflicts in project schedules.

Keywords: qualitative reasoning; spatial-temporal reasoning; dynamic scene modeling.

DOI: 10.15514/ISPRAS-2018-30(6)-13

For citation: Petrishchev K.S., Semenov V.A., Zolotov V.A. A system of operators for spatial-temporal scene analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 237-258 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-13

References

[1]. C. Freksa. Spatial computing. Cognitive and Linguistic Aspects of Geographic Space, Heidelberg, Springer Berlin, 2013, pp. 23-42.

[2]. D. Heesom, L. Mahdjoubi. Trends of 4D CAD applications for construction planning. Construction Management and Economics, vol. 22, 2004, pp. 171-182.

[3]. M.L.A.E. Borges, I.C. de Souza, S. Melo, J.P. Giesta. 4D Building Information Modelling: A Systematic Mapping Study. In Proc. of the 35th International Symposium on Automation and Robotics in Construction, Berlin, 2018.

[4]. V. Zolotov, V. Semenov. Modern search and indexing methods for multi-dimensional data in modelling large-scale dynamic scene applications. Trudy ISP RAN/Proc. ISP RAS, vol. 24, 2013, pp. 381-416 (in Russian).

[5]. V. Semenov, K. Kazakov, S. Morozov, O. Tarlapan, V. Zolotov, T. Dengenis. 4D modeling of large industrial projects using spatio-temporal decomposition. In eWork and eBusiness in Architecture, Engineering and Construction, 2010, pp. 89-95.

[6]. V. Zolotov, V. Semenov. On application of spatial decomposition method for large data sets indexing. Trudy ISP RAN/Proc. ISP RAS, vol. 25, 2013, pp. 121-166 (in Russian). DOI: 10.15514/ISPRAS-2013-25-8.

[7]. V. Zolotov, V. Semenov. Prospective schemes of temporal-spatial indexing for visual modeling. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 2, 2014, pp. 175-196 (in Russian). DOI: 10.15514/ISPRAS-2014-26(2)-8.

[8]. A.G. Cohn, S.M. Hazarika. Qualitative Spatial Representation and Reasoning: An Overview. Fundamenta Informaticae, vol. 46, № 1-2, 2001, pp. 1-29.

[9]. J.F. Allen. Maintaining knowledge about temporal intervals. Communications of the ACM, vol. 26, issue 11, 1983, pp. 832-843.

[10]. E.A. Emerson. Chapter 16 - Temporal and Modal Logic. In Handbook of Theoretical Computer Science, B: Formal Models and Semantics, Elsevier, 1990, pp. 995-1072.

[11]. M.B. Vilain, H. Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In Proc. of the 5th National Conference on Artificial Intelligence, 1986, pp. 377-382.

[12]. A.U. Frank. Qualitative spatial reasoning about distances and directions in geographic space. Journal of Visual Languages & Computing, vol. 3, № 4, 1992, pp. 343-371.

[13]. C. Freksa. Using orientation information for qualitative spatial reasoning. Lecture Notes in Computer Science, vol. 639, 1992, pp. 162-178.

[14]. G.F. Ligozat. Qualitative triangulation for spatial reasoning. Lecture Notes in Computer Science, vol. 716, 1993, pp. 54-68.

[15]. R. Moratz, J. Renz, D. Wolter. Qualitative Spatial Reasoning About Line Segments. In Proc. of the 14th European Conference on Artificial Intelligence, 2000, pp. 234-238.

[16]. P. Balbiani, J.-F. Condotta, L.F. del Cerro. A new tractable subclass of the rectangle algebra. In Proc. of the 16th International joint conference on Artifical Intelligence, vol. 1, 1999, pp. 442-447.

[17]. R.K. Goyal, M.J. Egenhofer. Similarity of cardinal directions. Lecture Notes in Computer Science, vol. 2121, 2001, pp. 36-55.

[18]. S. Skiadopoulos, M. Koubarakis. On the consistency of cardinal direction constraints. Artificial Intelligence, vol. 163, № 1, 2005, pp. 91-135.

[19]. M.J. Egenhofer, R.D. Franzosa. Point-set topological spatial relations. International Journal of Geographical Information Systems, vol. 5, № 2, 1991, pp. 161-174.
[20]. D. A. Randell, Z. Cui, A. G. Cohn. A spatial logic based on regions and connection. In Proc. of the 1st International Conference Principles of Knowledge Representation and Reasoning, 1992, стр. 165-176.
[21]. N. Eloe. VRCC-3D+: Qualitative spatial and temporal reasoning in 3 dimensions. PhD Thesis, Missouri University of Science and Technology, 2015.
[22]. A.G. Cohn, J. Renz. Qualitative Spatial Representation and Reasoning. In Handbook of Knowledge Representation, Elsevier, 2008, pp. 551-596.
[23]. M.J. Egenhofer, J. Sharma, D.M. Mark. A Critical Comparison of the 4-Intersection and 9-Intersection Models for Spatial Relations: Formal Analysis. In Proc. of the International Research Symposium on Computer-based Cartography, AutoCarto 11, 1993, pp. 1-12.
[24]. T. Mossakowski, R. Moratz. Qualitative Reasoning about Relative Direction of Oriented Points. Artificial Intelligence, vol. 180-181, 2012, pp. 34-45.
[25]. K. Zimmermann. Measuring without measures the D-calculus. Lecture Notes in Computer Science, vol. 988, 1995, pp. 59-67.
[26]. R. Moratz, B. Nebel, C. Freksa. Qualitative spatial reasoning about relative position. Lecture Notes in Computer Science, vol. 2685, 2002, pp. 385-400.
[27]. A. Gerevini, B. Nebel. Qualitative Spatio-Temporal Reasoning with {RCC-8} and Allen's Interval. In Proc. of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, 2002, pp. 312-316.
[28]. N. Van de Weghe, A. Cohn, G. De Tre, P. De Maeyer. A qualitative trajectory calculus as a basis for representing moving objects in geographical information systems. Control and Cybernetics, vol. 35, № 1, 2006, pp. 97-119.
[29]. A. Frank. Qualitative Spatial Reasoning: Cardinal Directions as an Example. International Journal of Geographical Information Systems, vol. 10, issue 3, 1996, pp. 269-290.
[30]. J. Renz, D. Mitra. Qualitative Direction Calculi with Arbitrary Granularity. Lecture Notes in Computer Science, vol. 3157, 2004, pp. 65-74.
[31]. A. Borrmann, E. Rank. Query support for BIMs using semantic and spatial conditions. In Handbook of Research on Building Information Modeling and Construction Informatics: Concepts and Technologies, IGI Global, 2009, pp. 405-450.
[32]. A. Carvalho, C. Ribeiro, A. Augusto de Sousa. A Spatio-temporal Database System Based on TimeDB and Oracle Spatial. Research and Practical Issues of Enterprise Information Systems, IFIP International Federation for Information Processing, vol. 205, 2006, pp. 11-20, Boston, MA, Springer US.
[33]. C. Xinmin Chen, C. Zaniolo. SQLST : A Spatio-Temporal Data Model and Query Language. Lecture Notes in Computer Science, vol. 1920, 2000, pp. 96-111.
[34]. M. Erwig, M. Schneider. Developments in spatio-temporal query languages. In Proc. of the Tenth International Workshop on Database and Expert Systems Applications, DEXA 99, 1999, pp. 441-449.
[35]. P. Muller. A Qualitative Theory of Motion Based on Spatio-Temporal Primitives. In Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR '98), 1998, pp. 131-141.
[36]. E. Davis. Describing spatial transitions using mereotopological relations over histories. Technical Report #2000-809, New York University, 2000.

Проблемно-ориентированная библиотека SOWFA для решения прикладных задач ветроэнергетики

М.В. Крапошин <m.kraposhin@ispras.ru>
С.В. Стрижак <strijhak@yandex.ru>

Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. В статье рассматриваются возможности открытой библиотеки SOWFA. Проблемно-ориентированная библиотека, работающая в составе открытого пакета OpenFOAM v.2.4.0, предназначена для решения задач ветроэнергетики. В связи со строительством новых ветропарков на территории РФ (Ульяновская область, республика Адыгея) вопросы проектирования, моделирования работы ветропарков и ветроустановок являются актуальными в настоящее время. В статье приводится описание структуры библиотеки SOWFA и некоторых ее классов. Изучение динамики самоорганизованных турбулентных вихревых структур и оценка их размеров важны с точки зрения максимизации вырабатываемой мощности ветроэлектрическими установками (ВЭУ), для анализа оптимального расположения ВЭУ в ветропарке. При этом необходимо детально изучать процесс эжекции воздуха, процесс смещения двух сред, в котором одна среда, находясь под давлением, оказывает воздействие на другую и увлекает ее в требуемом направлении, в районе ветропарке. Явление эжекции играет положительную роль и позволяет восстановить дефицит скорости в следе за ВЭУ, следовательно повлиять на вырабатываемую мощность ветропарка. Явление эжекции можно изучать с помощью движения твердых частиц. В статье описывается пример добавления нового класса KinematicCloud в решатель ABLSolver, который описывает кинематическое облако частиц и пример решения прикладной задачи ветроэнергетики для модельного ветропарка. Расчетная область для модельного ветропарка имела форму параллелепипеда с заданными размерами. Неструктурированная сетка содержала 6 миллионов ячеек. Для определения начального распределения параметров использовалось приближение нейтрального атмосферного пограничного слоя, рассчитанное с применением Precursor метода, реализованного в решателе ABLSolver. Математическое моделирование параметров течения в ветропарке было проведено с использованием решателя pisoFoamTurbine и метода Actuator Line Model. В ходе расчета для случая модельного ветропарка с 12 ветроустановки были получены поля осредненных и пульсационных величин для скорости, для давления, подсеточная вязкость, тензор напряжени, завихренность. В статье выполнено сравнение значения безразмерной горизонтальной скорости в двух различных сечениях со значениями, полученными в эксперименте. Вычисления проведены с использованием ресурсов вычислительного кластера web-лаборатории UniCFD ИСП РАН.

Ключевые слова: OpenFOAM; SOWFA; библиотека; модель турбулентности, облако частиц, ветроустановка, ветропарк, ветроэнергетика

DOI: 10.15514/ISPRAS-2018-30(6)-14

Для цитирования: Крапошин М.В., Стрижак С.В. Проблемно-ориентированная библиотека SOWFA для решения прикладных задач ветроэнергетики. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 259-274. DOI: 10.15514/ISPRAS-2018-30(6)-14

1. Введение

Открытая библиотека OpenFOAM предоставляет широкие возможности для решения задач механики сплошной среды. Библиотека поставляется в виде исходного кода, написанного на языке программирования C++. Общий объем исходного кода составляет более 1 миллиона строк. Для решения уравнений в частных производных используется метод контрольного объема. Для инженеров существует возможность работы с готовыми решателями, для исследователей – возможность разработки собственных решателей, служебных утилит, а также интеграции библиотеки OpenFOAM с другими открытыми пакетами.

Одно из направлений в механике сплошной среды связано с решением задач в области вычислительной гидродинамики. К там задачам можно отнести задачи в области геофизической гидродинамики, в частности моделирование процессов в атмосферном пограничном слое (АПС). Известно, что для моделирования физических процессов в АПС используются вихреразрешающие методы. Крупномасштабные вихревые структуры рассчитываются при помощи интегрирования фильтрованных уравнений неразрывности, Навье-Стокса, уравнения энергии. Мелкие вихри, размер которых не превышает шага расчетной сетки, моделируются с помощью различных моделей турбулетности, например, лагранжевой динамической модели Смагоринского для подсеточной вязкости. Аппроксимация слагаемых в исходных уравнениях выполняется с первым и вторым порядком точности по времени и пространству. Полученные уравнения для связи скорости и давления решались с помощью итерационного алгоритма PIMPLE. Полученная система алгебраических уравнений для скорости, давления, тензора напряжений и параметров модели турбулентности решается итерационным методом сопряженных градиентов с предобуславливателем.

Основным преимуществом библиотеки OpenFOAM является открытое вычислительное ядро на основе языка программирования C++ [1-2], позволяющее описывать основные объекты вычислительной механики сплошной среды (МСС) и отношения между ними: расчетную область, алгебраические преобразования тензорной и линейной алгебры, алгоритмы интегрирования систем уравнений в частных производных (решатели), численные схемы аппроксимации дифференциальных операторов, вектора, физические поля, матрицы, сетки, численные схемы, модели турбулентности. За счет использования средств метапрограммирования исходный код

решателей OpenFOAM представляет собой описание исследуемых систем уравнений на языке близком к естественному.

2. Обзор библиотеки SOWFA

В настоящее время ветроэнергетика является одним из актуальных направлений в области возобновляемых источников энергии. Математическое моделирование позволяет рассчитать мощность ветроустановки и мощность ветропарка в целом с учетом орографии местности, погодных условий, особенностей конструктивного исполнения ветряных турбин, турбулентного переноса импульса. Разработанная в Национальной лаборатории возобновляемой энергии (NREL) на базе математической модели несжимаемых течений OpenFOAM 2.4.0 библиотека SOWFA является реализацией алгоритмов и численных инструментов для комплексного моделирования ветропарков. В составе библиотеки SOWFA имеется несколько проблемно-ориентированных решателей, в том числе для расчёта атмосферного пограничного слоя, решатель ABLSolver, и физических параметров в ветропарке, решатель windPlantSolver, с использованием метода крупных вихрей [5-6]. В составе библиотеки имеется несколько моделей для подсеточной турбулентной вязкости и специальные граничные условия для задания скорости и температуры. Модель Actuator Line Model (ALM), или модель плоских сечений, может быть использована для расчёта течения вблизи вращающихся лопастных турбин на фиксированной расчётной сетке, что значительно экономит вычислительные ресурсы и упрощает процесс счёта. На рис. 1 представлена общая структурная схема библиотеки SOWFA, на рис. 2 – структура класса для описания моделей турбулентности.

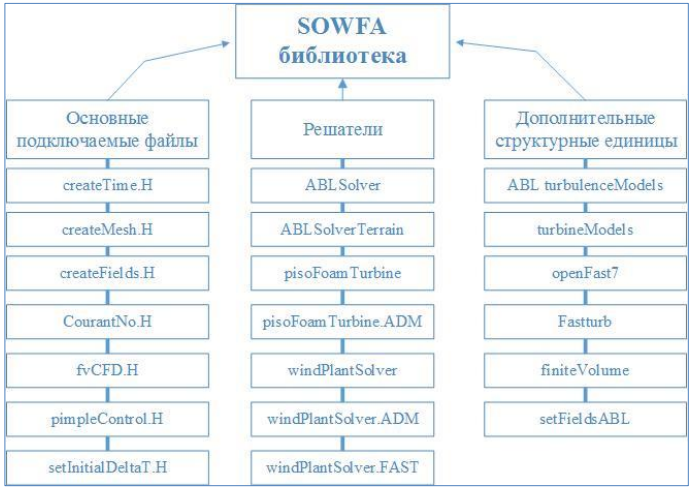


Рис.1 Структура библиотеки SOWFA
Fig.1 The structure of library SOWFA

Инженерная методика, используемая в модели расчета сил от ветроэнергетической установки (ВЭУ), основана на представлении вращающихся лопастей в виде набора профилей с табулированными коэффициентами аэродинамических сил, в то время как их действие на поток моделируется посредством добавления в уравнение движения аэродинамической силы, спроецированной на расчётную сетку.

В состав базовых модулей входят модули для задания расчетного времени и начального шага по времени, для работы с сеткой, с различными полями, определения числа Куранта, использование алгоритма PIMPLE (гибридный алгоритм PISO/SIMPLE) и другие.

В состав дополнительных структурных единиц входят компоненты для задания моделей подсеточной турбулентной вязкости с учетом влияния атмосферного пограничного слоя, описания моделей турбин (рис. 3), определения правила взаимодействия работы с различными пакетами задач аэроупругости (рис. 4), задания специальных граничных условий, задания параметров горизонтальной ВЭУ.

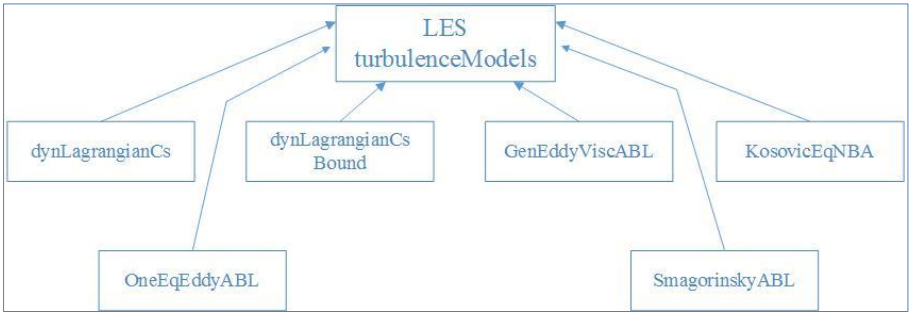


Рис.2 Производные классы SOWFA для описания моделей турбулентности
Fig.2 SOWFA derived classes for turbulence models

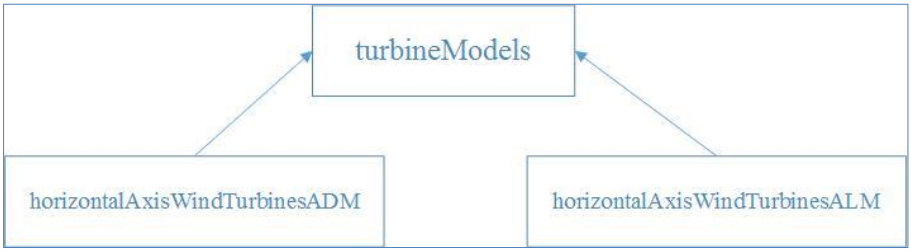


Рис.3 Класс turbineModels для моделей турбин
Fig.3 The turbineModels class for models of turbine

На рис. 5 показана блок схема для класса, свяанного с заданием граничных условия для скорости и температуры с учетом влияние атмосферного пограничного слоя.

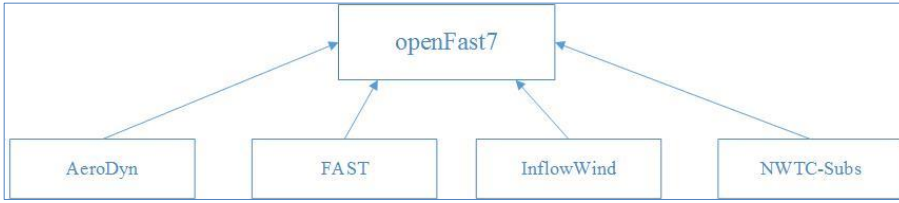


Рис.4 Класс openFast7 для работы с моделями аэроупругости
Fig.4 The openFast7 class for different FSI solvers

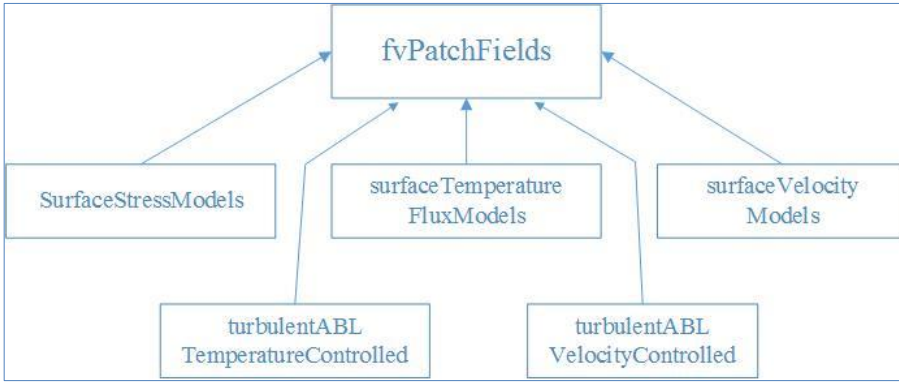


Рис.5 Некоторые специальные граничные условия, используемые в библиотеке SOWFA для описания аэродинамических полей
Fig.5 Some particular boundary conditions of SOWFA library for aerodynamic fields

3. Модель облака частиц в составе пакета OpenFOAM

В пакете OpenFOAM для описания движения частиц используется класс KinematicCloud. Схема, иллюстрирующая взаимосвязь основных классов облака частиц, представлена на рис. 6, где стрелками показано наследование классов.

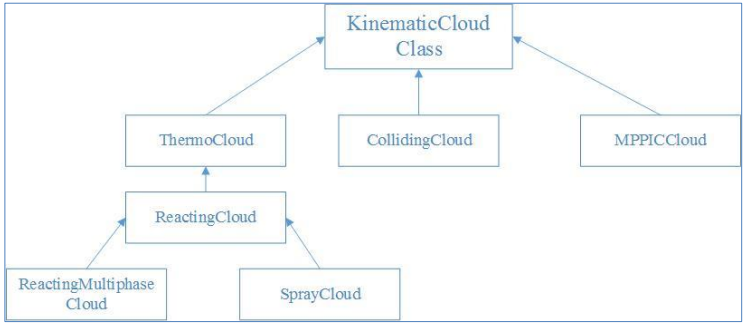


Рис.6 Класс KinematicCloud и производные классы
Fig.6 The KinematicCloud class and its children

Как видно из схемы, класс KinematicCloud является «базовым» при описании остальных классов, характеризующих различные типы облака частиц. Для единообразного управления облаком частиц в классе введена функция move(),унаследованная от класса KinematicParcel, которая производит физическое перемещение отдельной частицы в соответствии с заданным уравнением движения.

Для описания движения в классе введены структуры, учитывающие различные силы, действующие на частицы:

- выталкивающая сила;
- сила трения;
- разность давления.

Физические модели, используемые при описании облака частиц включают в себя следующие модели и соответствующие им классы:

- модель дисперсии частиц;
- модель инжекции частиц;
- модель взаимодействия частиц с поверхностями;
- стохастическая модель соударений частиц;
- модель поверхностной пленки.

Ниже приведены заложенные в пакете OpenFOAM опции физических моделей класса (указываются в дальнейшем при настройке расчетных кейсов в файлах, описывающих свойства дисперсной фазы):

- CollisionModel – класс моделей соударения; он хранит информацию о точности определения сил плоского взаимодействия с поверхностью, о модели попарного взаимодействия для расчета взаимодействия между двумя частицами, о модели стенки для расчета взаимодействия частиц со стенкой, о списках ячеек, с которыми происходит взаимодействие для каждой ячейки, участвующей во взаимодействиях.
- DispersionModel – класс моделирования дисперсии.

3.1 Модель для описания облака частиц

На рис. 7 приведена общая структура класса Injection Model.

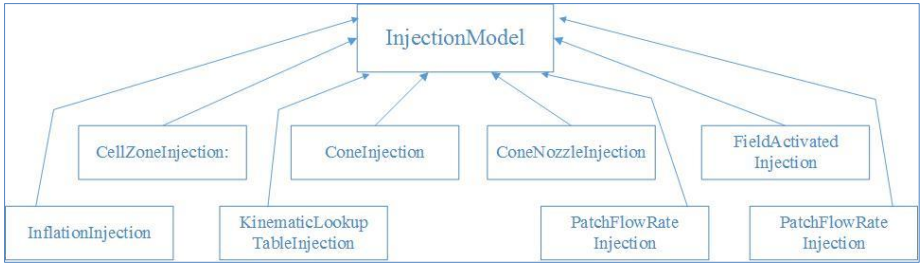


Рис.7 Модель ввода частиц
Fig. 7 The model for particle's injection

Данный класс включает в себя следующие модели.

1. *CellZoneInjection*: характеризуется плотностью распределения частиц по заданным ячейкам. Исходными данными являются:
   - эффективная плотность распределения частиц в ячейках;
   - общая инжектируемая масса;
   - начальная скорость пакета частиц;
   - диаметры пакетов частиц, получаемые из заданной функции плотности вероятности.
2. *ConeInjection:* описывает инжекцию из конуса. Исходными данным модели являются:
   - время начала инжекции;
   - список координат и направлений инжекторов (по оси инжекции);
   - число пакетов частиц, инжектируемых каждым инжектором;
   - скорости пакетов частиц;
   - внутренний и внешний углы полураствора конусов
   - диаметры пакетов частиц, определяемые в распределении капель по размерам.
3. *FieldActivatedInjection:* описывает инжекцию в заданных точках при выполнении определенного условия.
4. *InflationInjection:* описывает ввод новых частиц путем разделения существующих частиц в генерирующих ячейках.
5. *KinematicLookupTableInjection:* источники частиц задаются в виде таблицы, содержащей параметры о распределении частиц по диаметрам, о массовом расходе и положении инжекции.
6. *ManualInjection:* описывает ручной способ ввода частиц в расчетную область.
7. *PatchFlowRateInjection, PatchInjection* описывают ввод частиц через заданную внешнюю границу расчетной области.

3.2 Классы описывающие силы, действующие на частицу

Силы сопротивления частицы рассчитываются с использованием следующих моделей:

- модель трения для твердых сфер;
- модель трения для несферических частиц.

Для расчета подъемной силы используются следующие модели:

- модель подъемной силы, применимая для сферических частиц;
- модель подъемной силы, применимая для пузырей изменяемой формы.

На частицу могут действовать и другие силы, описываемые следующими классами:

- *NonInertialFrame:* используется для расчета центробежной силы, связанной с неинерциальностью системы отсчета;

- *Paramagnetic:* используется для расчета силы, действующей на частицы со стороны магнитного поля;
- *PressureGradient:* используется для расчёта силы, вызванной перепадом давлений;
- *SRF:* используется для расчета центробежной силы, прикладываемой к частице при движении во вращающейся системе координат (Single Rotating Frame of Reference);
- *VirtualMass:* используется для расчета силы, прилагаемой к частице вследствие образования дополнительной (виртуальной) массы.

3.3. Взаимодействие частиц

Частицы могут взаимодействовать друг с другом и с границами расчетной области. Для учета этого взаимодействия введены следующие классы:

- *Rebound:* модель простого упругого взаимодействия частицы с границей.
- *StandardWallInteraction:* модель взаимодействия со стенкой, включает в себя три опции:
- *rebound:* упругое взаимодействие, при необходимости возможно определить коэффициенты упругости и восстановления;
- *stick:* мгновенное прикрепление частицы к поверхности;
- *escape:* уход частицы, пересекшей границу из расчетной области;
- *StochasticCollisionMode:* описание стохастических соударений частиц.
- *SurfaceFilmModel*: для описания взаимодействия частиц с поверхностью, покрытой тонкой плёнкой.

3.4 Классы, наследуемые от класса KinematicCloud

Помимо базового класса KinematicCloud, реализующего базовый функционал для переноса роя частиц, существуют и другие классы, выполняющие специализированные функции:

- класс многофазного моделирования с использованием метода Частица-в-Ячейке (Particle-in-Cell)
- модель облака соударяющихся частиц;
- модель термически неравновесного облака частиц;
- модель облака реагирующих частиц;
- модель термически неравновесного облака реагирующих частиц различного состава;
- модель облака частиц, имитирующего распад (атомизацию) струи.

4. Добавление модели облака частиц в исходные уравнения движения и температуры

Изучение динамики самоорганизованных турбулентных вихревых структур и оценка их характерных размеров важны с точки зрения максимизации

вырабатываемой мощности ВЭУ, анализа оптимального расположения ВЭУ в ветропарке. Необходимо детально исследовать процесс эжекции воздуха в районе ветропарке. Явление эжекции позволяет восстановить дефицит скорости в следе за ВЭУ и сделовательно положительно повлиять на вырабатываемую мощность ветропарка.

В работе [7] было предложено исследовать трехмерное турбулентное движение воздуха в окрестности ветропарка с помощью облака частиц. В библиотеке SOWFA отсутствует возможность моделирования турбулентного движения газа с частицами, в связи с чем предлагается её добавить. Внесение изменений в реализацию библиотеки в соответствии с предлагаемой модификацией модели также позволяет наглядно продемонстрировать преимущество объектно-ориентированного подхода, в соответствии с которым возможно представление разностных аналогов уравнений в виде исходного кода на языке, близком к естественному.

Другим важным приложением использования моделей движения частиц может быть исследование процесса лёдообразования на лопастях ВЭУ, оказывающего значительное влияние на эксплуатацию ветропарков в областях земного шара, для которых характерными являются времена года с низкими температурами и высокой влажностью воздуха.

С целью объединения моделей аэродинамики ветропарка и облака частиц, добавим в исходный код решателей ABLSolver и windPlantSolver слагаемые, учитывающие движение частиц и их взаимодействие с воздушным потоком.

Внесём изменения в модуль UEqn.H, описывающий уравнение движения воздуха в атмосферном пограничном слое в приближении Буссинеска (выделено жирным шрифтом с подчеркиванием):

```
// Solve the momentum equation
    #include "computeCoriolisForce.H"
    fvVectorMatrix UEqn
    (
        fvm::ddt(U)                 // time derivative
      + fvm::div(phi, U)            // convection
      + turbulence->divDevReff(U)   // momentum flux
      + fvc::div(Rwall)
      - fCoriolis                   // Coriolis force
      + gradP                       // driving pressure gradient
      - turbines.force()            // turbine body forces
      - parcels.SU(U)               // momentum from particles
    );
```

Выделенная строка указывает на добавление слагаемое, отвечающего за вклад в баланс импульса воздуха от частиц.

Поскольку частицы обладать тепловой энергией отличной от тепловой энергии воздуха, необходимо изменить модуль Teqn.H для уравнения температуры воздуха (выделено жирным шрифтом с подчеркиванием):

```
    kappat = turbulence->nut()/Prt;
```

```
    kappat.correctBoundaryConditions();
volScalarField kappaEff("kappaEff",
                        turbulence->nu()/Pr + kappat);
fvScalarMatrix TEqn
    (
     fvm::ddt(T)
     + fvm::div(phi, T)
     - fvm::laplacian(kappaEff, T)
     - fvc::div(qwall)
     - parcels.Sh(T)/(rho*Cp)
    );
```

Предполагается, что теплоемкость воздуха Cp и плотность меняются слабо.

Наконец, для подключения библиотеки переноса облака частиц к решателям необходимо:

- указать соответствующий заголовочный файл: #include KinematicCloud.H;
- вызвать процедуру интегрирования уравнений движения частиц облака после решения уравнений аэродинамики: parcels.evolve().

5. Пример расчета с использованием библиотеки SOWFA

Для тестирования возможностей библиотеки SOWFA была выбрана модельная задача с ветропарком в составе 12 имитаторов ветроустановок (ВЭУ), расположенных в 4 ряда. Размеры модельных ветроустановок выбирались на основе данных из эксперимента работы [8]. Пространственная вычислительная область представляла собой прямоугольный параллелепипед с размерами 6 метров по оси X, 5 метров по оси Y, 1 метр по оси Z (по вертикали). Размер получившейся неструктурированной сетки составил 6 миллионов ячеек. На верхней границе задавалось граничное условие проскальзывания, а на боковых границах - циклическое граничное условие. На входе расчетной области задавался логарифмический профиль скорости, полученный из данных ветромониторинга, на твердой поверхности использовалась модель пристеночной функции согласно теории подобия Монина-Обухова.

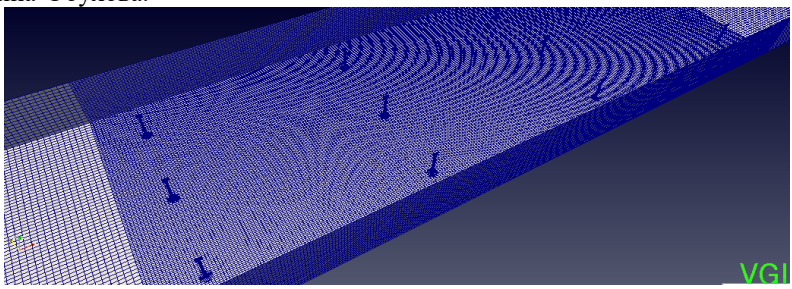


Рис. 8. Расчетная область и сетка с 12 ВЭУ
Fig. 8 Numerical domain and grid with 12 wind turbines

Неструктурированная сетка имела 6 миллионов ячеек. По результатам расчета были получены поля скорости, завихренности, давления, турбулентной вязкости, спектр кинетической энергии турбулентности. Более подробную информацию по этой задаче можно найти в работе [11]. Результаты расчета (профили безразмерной скорости в различных сечениях в следе за ветроустановками, а также по значения коэффициентов мощности и осевой силы тяги ветроустановки) сравнивались с результатами эксперимента [9,10]. Данные исследования проводились с целью разработки методики для расчета параметров течения для действующего ветропарка в Ульяновской области РФ с 28 ВЭУ.

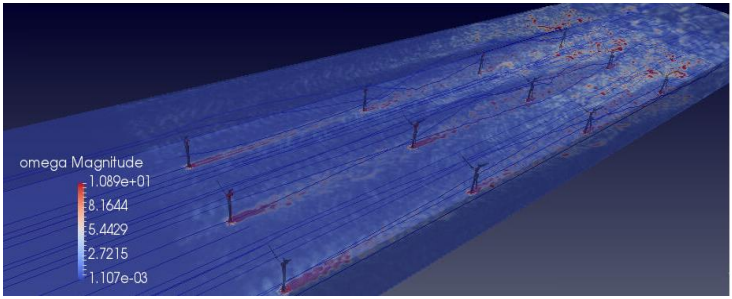


Рис.9 Поле модуля завихренности в момент времени t=20 секунд
Fig. 9. The vorticity magnitude field at time = 20 seconds

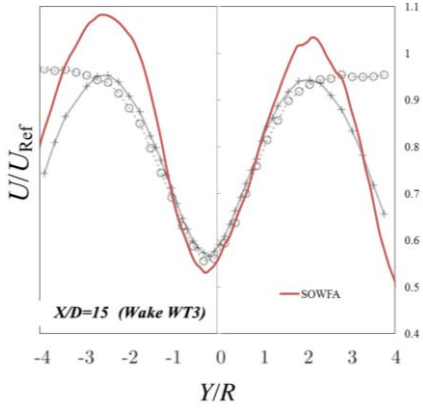


Рис.10. Профиль безразмерной горизонтальной скорости на расстоянии X/D=15, где X является расстоянием от ВЭУ, D является диаметром ВЭУ. Красная линия – расчет с использованием SOWFA, линия с круглыми маркерами – эксперимент для ряда из четырех турбин, линия с крестами – эксперимент для массива из 12 (3x4) турбин
Fig. 10. The normalized horizontal velocity U/Uref profiles behind the wind turbines of the central row in section X/D =15, where X is the distance from the first wind farm, D is the diameter of the wind turbines. Red line – calculation with SOWFA, line with circles – experiment for single line of 4 turbines, line with crosses – experiment for array of 12 (3x4) turbines

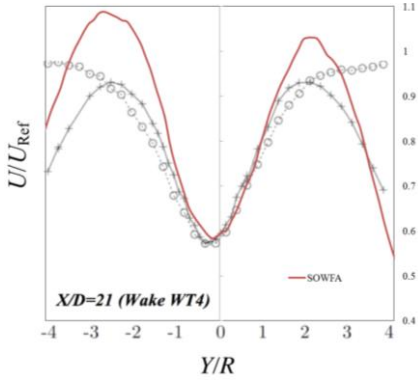


Рис.11 Профиль безразмерной горизонтальной скорости на расстоянии X/D=15, где X является расстоянием от ВЭУ, D является диаметром ВЭУ. Красная линия – расчет с использованием SOWFA, линия с круглыми маркерами – эксперимент для ряда из четырех турбин, линия с крестами – эксперимент для массива из 12 (3x4) турбин
Fig. 11 The normalized horizontal velocity U/Uref profiles behind the wind turbines of the central row in section X / D =15, where X is the distance from the first wind farm, D is the diameter of the wind turbines. Red line – calculation with SOWFA, line with circles – experiment for single line of 4 turbines, line with crosses – experiment for array of 12 (3x4) turbines

Для определения начального распределения физических величин использовалось приближение нейтрального атмосферного пограничного слоя, рассчитанное с применением Precursor метода, реализованного в решателе ABLSolver. Таким образом, для задания начальных данных на входе расчетной области с заданными размерами использовались значения нестационарных физических величин. Время моделирования задавалось исходя из условия 2-х кратного прохождения потоком расчётной области.

Математическое моделирование параметров течения в ветропарке было проведено с использованием решателя pisoFoamTurbine и метода Actuator Line Model. Достоверность и точность выбранного решателя проверялась на задачах с одной, двумя и двенадцатью модельными ветроустановками [9-11].

Дополнительно был проведен расчет значения безразмерной горизонтальной скорости в различных сечениях и проведено сравнение с результатами эксперимента (рис. 10, 11). В результате получено хорошее совпадение с результатами эксперимента.

В работах [10-11] с помощью программы ImaCalc проводилась оценка величины фрактальной размерности турбулентного течения в вихревом следе для ВЭУ и оценка мультифрактального спектра турбулентности в разных сечениях за ВЭУ. Данные параметры могут быть использованы в формуле для расчета спектра кинетической энергии в ветропарке с учетом влияния стратификации в атмосферном пограничном слое.

6. Заключение

В статье рассмотрены возможности открытого пакета OpenFOAM для решения прикладных задач механики сплошной среды в области энергетики, в том числе возобновляемой. Исследование процессов турбулентного движения в ветропарке предложено осуществлять с применением средств отслеживания облака частиц. Приведено описание структуры открытой библиотеки SOWFA для решения задач ветроэнергетики. В статье приведен пример добавления модели облака частиц в состав решателя ABLSolver и windPlantSolver.

Выполнено описание базового класса KinematicCloud для описания движения облака частиц. Для демонстрации работы библиотеки SOWFA представлены результаты расчета поля завихренности, безразмерной горизонтальной скорости для модельного ветропарка с 12 ВЭУ. Вычисления были проведены с использованием ресурсов вычислительного кластера web-лаборатории UniCFD ИСП РАН. Для расчета одного примера было использовано от 72 до 96 вычислительных ядер.

Благодарности

Работа выполнена при финансовой поддержке РФФИ (грант № 17-07-01391) и Программы Президиума РАН № 26.

Список литературы

[1] Прата С. Язык программирования C++. Лекции и упражнения. Вильямс, 2012, 1248 стр.

[2] Страуструп Б. Язык программирования C++. Специальное издание. Бином, 2017, 1136 с.

[3] Weller H.G., Tabor G., Jasak H., Fureby C. A tensorial approach to computational continuum mechanics using object oriented techniques. Computers in Physics, vol.12, № 6, 1998, pp. 620-631.

[4] Churchfield M.J., Lee S., Michalakes J., Moriarty P.J. A numerical study of the effects of atmospheric and wake turbulence on wind turbine dynamics. Journal of Turbulence, vol. 13, no. 14, 2012, pp. 1–32.

[5] Breton S.-P., Sumner J., Sørensen J.N., Hansen K.S., Sarmast S., Ivanell S. A survey of modelling methods for high-fidelity wind farm simulations using large eddy simulation. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 375, issue 2091, 2017.

[6] Sagaut P. Large eddy simulation for incompressible flows: an introduction. Springer, Berlin, 2002, 426 p.

[7] Andersen S.J., Sørensen J.N., Mikkelsen R.F. 2017 Turbulence and entrainment length scales in large wind farms. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 375, issue 2091, 2017.

[8] Hancock P.E., Farr T.D. Wind-tunnel simulations of wind-turbine arrays in neutral and non-neutral winds. Journal of Physics: Conference Series, vol. 524, conference 1, 2014.

[9] Strijhak S., Redondo J.M., Tellez-Alvarez J. Multifractal analysis of a wake for a single wind turbine. In Proc. of the Conference on Topical Problems of Fluid Mechanics, 2017. pp. 275-284.

[10] Kryuchkova A., Tellez-Alvarez J., Strijhak S., Redondo J.M. Assessment of Turbulent Wake Behind two Wind Turbines Using Multi-Fractal Analysis. In Proc. of the Ivannikov ISPRAS Open Conference, 2017. DOI: 10.1109/ISPRAS.2017.00025.
[11] Strijhak S.V., Koshelev K.B., Kryuchkova A.S. Studying parameters of turbulent wakes for model wind turbines. AIP Conference Proceedings, 2018, vol. 2027, issue 1.

The problem-oriented library SOWFA for solving the applied tasks of wind energy

M.V. Kraposhin <m.kraposhin@ispras.ru>,
S.V. Strijhak <strijhak@yandex.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. The article discusses the possibilities of the open source library SOWFA. The problem-oriented library, operating as part of the open source package OpenFOAM v.2.4.0, is intended for solving wind energy's problems. In connection with the construction of new wind farms in the Russian Federation (Ulyanovsk region, The Republic of Adygea), the issues of designing and modeling the operation of wind farms and wind turbines are currently relevant. The article describes the structure of the SOWFA library and some of its classes. The study of the dynamics of self-organized turbulent structures and the assessment of their size are important from the point of view of maximizing the power generated by wind turbines, for analyzing the optimal location of wind turbines in wind farm. At the same time, it is necessary to study in detail the process of air's ejection, the process of displacement of two media, in which one medium, being under pressure, affects the other and carries it in the required direction, in the area of the wind farm. The phenomenon of ejection plays a positive role and allows restoring the velocity's deficit in the wake of the wind turbine, therefore, affects the wind capacity of wind farm. The phenomenon of ejection can be studied using the motion of solid particles. The article describes an example of adding a new KinematicCloud class to the ABLSolver solver, which describes a kinematic cloud of particles and an example of solving an applied wind energy problem for a model wind farm. The numerical domain for the model wind farm had the shape of a parallelepiped with given dimensions. The unstructured mesh contained 6 million cells. To determine the initial distribution of parameters, we used the neutral atmospheric boundary layer approximation, calculated using the Precursor method, implemented in the ABLSolver solver. The mathematical modeling of the flow parameters in the wind farm was done using the pisoFoamTurbine solver and the Actuator Line Model. In the course of calculation, for the case of a model wind farm with 12 wind turbines, the fields of averaged and pulsation values were obtained for velocity, pressure, subgrid scale viscosity, stress tensor, vorticity. The article compares the values of the dimensionless horizontal velocity in two different sections with the values obtained in the experiment. The calculations were performed using the resources of the high performance cluster of the UniCFD web-laboratory in ISP RAS.

Keywords: OpenFOAM; SOWFA; library; model of turbulence; cloud of particles; wind turbine; wind farm; wind energy

DOI: 10.15514/ISPRAS-2018-30(6)-14

For citation: Kraposhin M.V., Strijhak S.V. The problem-oriented library SOWFA for solving the applied tasks of wind energy. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 259-274 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-14

References

[1] Prata S. C++ Primer Plus. Addison-Wesley Professional, 2012, 1440 p.

[2] Stroustrup B. The C++ Programming Language. Addison-Wesley Professional, 2013, 1376 p.

[3] Weller H.G., Tabor G., Jasak H., Fureby C. A tensorial approach to computational continuum mechanics using object oriented techniques. Computers in Physics, vol.12, № 6, 1998, pp. 620-631.

[4] Churchfield M.J., Lee S., Michalakes J., Moriarty P.J. A numerical study of the effects of atmospheric and wake turbulence on wind turbine dynamics. Journal of Turbulence, vol. 13, no. 14, 2012, pp. 1–32.

[5] Breton S.-P., Sumner J., Sørensen J.N., Hansen K.S., Sarmast S., Ivanell S. A survey of modelling methods for high-fidelity wind farm simulations using large eddy simulation. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 375, issue 2091, 2017.

[6] Sagaut P. Large eddy simulation for incompressible flows: an introduction. Springer, Berlin, 2002, 426 p.

[7] Andersen S.J., Sørensen J.N., Mikkelsen R.F. 2017 Turbulence and entrainment length scales in large wind farms. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 375, issue 2091, 2017.

[8] Hancock P.E., Farr T.D. Wind-tunnel simulations of wind-turbine arrays in neutral and non-neutral winds. Journal of Physics: Conference Series, vol. 524, conference 1, 2014.

[9] Strijhak S., Redondo J.M., Tellez-Alvarez J. Multifractal analysis of a wake for a single wind turbine. In Proc. of the Conference on Topical Problems of Fluid Mechanics, 2017. pp. 275-284.

[10] Kryuchkova A., Tellez-Alvarez J., Strijhak S., Redondo J.M. Assessment of Turbulent Wake Behind two Wind Turbines Using Multi-Fractal Analysis. In Proc. of the Ivannikov ISPRAS Open Conference, 2017. DOI: 10.1109/ISPRAS.2017.00025.

[11] Strijhak S.V., Koshelev K.B., Kryuchkova A.S. Studying parameters of turbulent wakes for model wind turbines. AIP Conference Proceedings, 2018, vol. 2027, issue 1.

Многомасштабный подход к моделированию сложных переходных процессов движения жидкостей в технических системах

М.В. Крапошин, <m.kraposhin@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН, Россия, Москва, ул. А. Солженицына, д. 25

Аннотация. В работе приводится описание многомасштабного подхода для моделирования процессов течений двухфазных сред в сложных технических системах. В основе многомасштабного подхода лежит как разделение расчетной области на подобласти с собственной системой уравнений, так и расщепление исходной системы уравнений на несколько подсистем для каждого из рассматриваемого масштабов. В качестве примера возможного применения многомасштабной модели рассматривается задача определения акустического шума в дальнем поле при старте ракеты-носителя с учетом подачи воды в газовые струи двигательной установки. Другими областями применения многомасштабной модели можно указать задачи нефтегазовой отрасли: глушение газодобывающих скважин, расположенных на большой глубине, глушение нефтяных скважин с высоким газовым фактором на месторождениях. Предлагаемая многомасштабная математическая модель включает в себя 5 подмоделей: 1) подмодель газодинамики высокоскоростных многокомпонентных течений смеси газов; 2) подмодель гидродинамики течения двухфазной смеси в гомогенном приближении с учетом сжимаемости газовой фазы и обмена массой между фазами; 3) подмодель переноса межфазной границы; 4) подмодель переноса облака капель и его взаимодействия с газожидкостной средой; 5) подмодель оценки шума в дальнем поле на основе акустической аналогии Ффоукс Вильямса-Хоукинга. Предложенная в рамках многомасштабного подхода модель может быть расширена для включения дополнительных моделей – таких, например, как Эйлерова-Лагранжева модель атомизации струй на основе уравнения эволюции плотности межфазной поверхности. Реализация подмоделей может быть выполнена на основе пакетов с открытым исходным кодом: OpenFOAM, Nektar++, ITHACA-FV. Подмодели акустики и гибридный алгоритм решения уравнений сжимаемой гомогенной двухфазной среды реализованы в виде модулей libAcoustics и hybridCentralSolvers на базе открытого пакета OpenFOAM. Использование платформы OpenFOAM в качестве базы для реализации программы позволяет получить архитектуру со взаимозаменяемыми элементами. Исходный код разрабатываемой модели свободно доступен через проект GitHub https://github.com/unicfdlab.

Ключевые слова: многомасштабные модели; численное моделирование; численные схемы; сжимаемые течения; многофазные течения; акустика; вычислительная гидро- аэро- и газодинамика; свободное программное обеспечение; метод контрольного объёма; разрывный метод Галёркина; libAcoustics; hybridCentralSolvers; OpenFOAM; Nektar++; Volume Of Fluid; Lagrangian Particle Tracking.

DOI: 10.15514/ISPRAS-2018-30(6)-15

Для цитирования: Крапошин М.В. Многомасштабный подход к моделированию сложных переходных процессов движения жидкостей в технических системах. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 275-292. DOI: 10.15514/ISPRAS-2018-30(6)-15

1. Список сокращений и обозначений

ELSA – Eulerian-Lagrangian Spray Atomization
LPT – Lagrangian Particle Tracking
NASA – National Aero Space Agency
PISO – Pressure Implicit With Splitting Operators
SIMPLE – Semi-Implicit Method for Pressure Linked Equations
VOF – Volume-of-Fluid Method
PIMPLE – PISO/SIMPLE
КА – Космический Аппарат
ПО – Программное Обеспечение
ПУ – Пусковая Установка
РН – Ракета-Носитель
ТС – Техническая Система
УБИ – Уравнение Баланса Импульса
УСМ – Уравнение Сохранения Массы
УБЭ – Уравнение Баланса Энергии
УСС – Уравнение Состояния Среды
УПМП – Уравнение Переноса Межфазной Поверхности

2. Введение

Актуальность применения многомасштабных подходов к моделированию течений жидких сред в технических системах связана с возрастающим спросом со стороны промышленности на разработку так называемых цифровых двойников – программного обеспечения, имитирующего реальные изделия и позволяющего проводить компьютерную отработку и испытания технических устройств.

Разработка такого прикладного ПО сопряжена с необходимостью математического моделирования сложного движения двухфазных сред, объединяющих такие промышленных задачи как старт ракеты-носителя

класса, движение жидкости в технических системах космических аппаратов, унос грунта при посадке КА, процесс производства чугуна в доменных печах, глушение газовых и нефтяных скважин и пр. Эти задачи характеризуется крайней пространственной неоднородностью по числу Маха, многофазностью, «разбрызгиванием» водяных (жидких) струй, наличием физико-химических процессов, взаимодействием и отражением волн от элементов конструкции стартового комплекса РН, большими различиями в масштабах моделируемых элементов, фазовыми превращениями.

Многие из известных базовых моделей (том числе и реализованные в коммерческих зарубежных пакетах) используют допущения и методы, основанные на переносе отдельных частиц либо же многожидкостные подходы, которые удобны для моделирования отдельных явлений, но малопригодны для описания происходящих в указанных выше случаях сложных переходных процессов и образующихся разномасштабных структур.

В настоящее время рядом как российских, так и зарубежных коллективов ведутся работы по объединению различных подходов с целью расширения области использования моделей течений многофазных сред – например, модели, включающие в себя перенос межфазной границы и отдельных капель.

В работе предлагается гибридный, иерархический подход для решения актуальных промышленных задач. Такие подходы являются узкоспециализированными и в зависимости от постановки задачи должен корректироваться состав математической модели. Таким образом, поскольку сама модель не может рассматриваться в отрыве от прикладной задачи, то в данной работе в качестве примера применения взята задача исследования акустического шума при старте ракеты носителя.

3. Постановка прикладной задачи

В настоящее время одной из актуальных задач в ракетно-космической технике является снижение акустического шума от струй работающих двигателей. Для решения задачи снижения шума используются различные методы – пассивные и активные. Одним из таких методов является подача струй воды в зону газовой струи за срезом сопла двигателя ракеты-носителя (РН). Данная технология послужила предметом исследований, находящих свое отражение в ряде недавних статей авторов, представляющих ЦИАМ им. П.И. Баранова, ЦЭНКИ НИИСК, ФГУП ЦНИИМаш, NASA Glenn Research Center, NASA Marshall Space Flight Center.

Детальное изучение применимости этой технологии и сопутствующих газодинамических и акустических процессов в натурных условиях требует использования инструментов численного моделирования. Неоднородность области по числу Маха, многофазность, «разбрызгивание» водяных струй, наличие физико-химических процессов, взаимодействие и отражение волн от

элементов конструкции стартового комплекса РКН, большие различия в масштабе моделируемых узлов – все эти особенности обуславливают необходимость разработки гибридной модели газодинамическо-акустического решателя, способной корректно воспроизводить и прогнозировать данные явления при изменении конструкторской документации.

Анализ акустики дальнего поля (в области полезного груза) при старте ракеты-носителя сопряжен с анализом явлений существенно различных масштабов:

- многофазные течения в области подачи воды под соплами двигательной установки (ДУ);
- распространение газокапельной смеси в газоходе;
- турбулентное движение трансзвуковой струи на выходе из газохода;
- излучение звуковых волн;
- перенос (распространение) акустических колебаний в сторону носовой части ракеты-носителя.

Объем пространства моделирования определяется габаритами стартового стола и сооружений, составляющими порядка сотни метров по каждому направлению, в то время как характерный размер расчетной сетки по пространству может составлять около 0,01 м. Таким образом, число неизвестных в задаче будет составлять величину порядка 10^{12} ...10^{13}, что делает предъявляемые требования к вычислительным мощностям невыполнимыми в обозримом будущем. Для воспроизведения же капельного потока средствами VOF метода (как одного из наиболее частого используемого и надёжного) потребуются ещё более мелкие сетки, поскольку для разрешения капель необходимо порядка 10 ячеек на диаметр, а характерный размер капли, в зависимости от состава жидкости – 0.1 – 1 мм.

Решение такой задачи требует совместного применения разномасштабных моделей, каждая из которых будет действовать в своей подобласти, получаемой разделением области моделирования на подобласти (рис. 1). С другой стороны, для согласованного учета распространения капель в сверхзвуковом потоке понадобится расщепить описание газокапельного потока на две системы, действительные для своих масштабов: континуальную и дискретную (описываемую в переменных Лагранжа).

Расчётная область (геометрическая модель) газодинамики и акустики старта РН с учетом водоподачи включает стартовое сооружение (ПУ), РН и патрубки системы подачи воды и в соответствии с принятыми допущениями разбивается на три подобласти:

a) взаимодействия сверхзвуковых газовых струй с водяными струями;
b) турбулентного до- и транс- звукового течения струй горячего сжимаемого совершенного газа;

c) распространения акустических колебаний в воздухе (дальнее поле).

Подобласть a) включает в себя пространство внутри газохода от среза сопел РН до выходного сечения. Для корректного разрешения взаимодействия газовых и капельных струй необходимо совместное использование моделей на основе континуального и Лагранжева подхода. Пульсации газодинамических параметров на выходном срезе газохода являются исходными данными для подобласти b). При взаимодействии потока продуктов сгорания и струй воды, догорание компонентов смеси и конденсация не учитываются.

Подобласть b) включает открытую часть ПУ от выходного сечения ПУ вниз по потоку на расстоянии приблизительно 20-30 диаметров сопла. Пульсации газодинамических параметров (давления, плотности и скорости) в данной подобласти используются для определения акустических нагрузок в дальнем поле (на поверхности РН).

Подобласть c) соответствует дальнему полю распространения акустических колебаний в равномерной невозмущённой среде и включает в себя поверхность РН с расположенными на ней микрофонами.

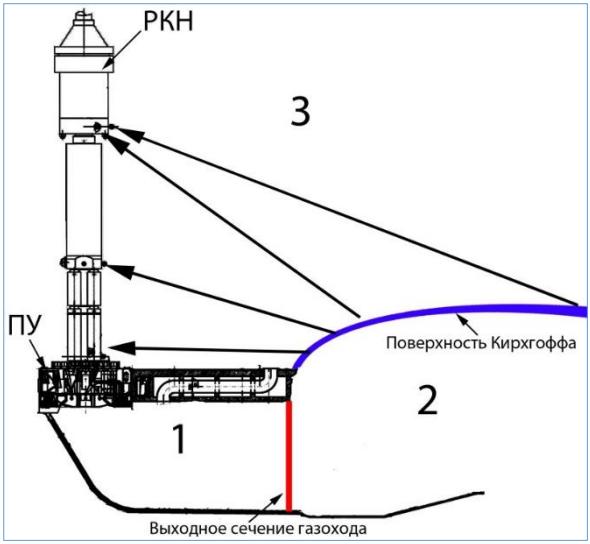


Рис. 1. Схема расположения подобластей расчетной области при моделировании старта РН

Fig. 1. Sketch of subregions introduced for accounting different phenomena during rocket lift-off

4. Математические модели

С учётом принятых допущений и описания расчетной схемы, математическая модель состоит из:

- в подобласти a) – трехмерных уравнений Навье-Стокса для турбулентных сверх-, транс- и дозвуковых течений сжимаемой газокапельной смеси, включая уравнения сохранения массы, импульса и энергии; переноса турбулентных величин, переноса компонент смеси, переноса капель, описание механизмов взаимодействия водяных и газовых потоков (испарение, обмен импульсом);
- в подобласти b) – трехмерных уравнений Навье-Стокса для турбулентных до- и трансзвуковых течений, включая уравнения сохранения массы, импульса и энергии; переноса турбулентных величин, переноса компонент смеси;
- в подобласти c) – интегрального решения уравнения Лайтхила, получаемого с помощью аналогии (уравнения) Фоукс Уильямс-Хокинга.

Кроме того, вместо решения уравнений Навье-Стокса в подообласти №2 и аналогии Фоукс Уильямс-Хокинга в подобласти c) возможно применение инженерно-эмпирического подхода, аналогичного разработанного в NASA для оценки шума от свободных струй [1-4]. Данный подход напрямую неприменим для рассматриваемой задачи, поскольку условия истечения струи существенно отличаются от исходных допущений автора.

Впоследствии точное разрешение процессов газодинамики и динамики капельного потока на основе многомасштабной модели может быть использовано для построения упрощённых инженерных подходов, аналогичных [1-4] или [7].

4.1. Модель газокапельной среды

В рассматриваемой задаче процессы обмена импульсом и внутренней энергией между водяными струями и газовым потоком исходящим из сопел ракеты-носителя являются определяющими при подавлении шума во время старта. Согласно [5-7] основной вклад в общее снижение энергии акустических пульсаций вносят именно механизмы:

- отбора кинетической и внутренней энергии от среднего потока каплями воды;
- интенсификации перемешивания в ядре струи, препятствующие развитию крупномасштабных вихревых структур (неустойчивостей) в слое смешения.

Скорость протекания процессов обмена импульсом и энергией между газовой и водяной фазами будет во многом определять мощность потока на выходе из

газохода, а также, следовательно, и уровень акустического шума в дальнем поле.

Согласно [8], в зависимости от коэффициента вязкости и поверхностного натяжения, диаметр самых маленьких капель, на которые разрывается струя на три порядка меньше характерного размера (например, диаметра струи) – таким образом, для насадок 0.1 – 1 м диаметр капель должен составлять порядка 0.1 – 1 мм. С учетом требований, предъявляемых современными численными моделями к сеточному разрешению [8], согласно которым на 1 диаметр капли должно приходиться от 8-12 ячеек, становится понятно, что прямое численное разрешение подобных течений не может быть применено в реальных приложениях. Более того, становится очевидно, что требования численного метода к пространственному разрешению для решения задачи эволюции межфазной поверхности отдельных капель являются более «вычислительно затратными» по сравнению с требованиями модели распространения акустических волн и газодинамики (см. подраздел 4.2).

В то же время, полностью отказаться от моделей на основе VOF не представляется возможным, поскольку:

- эти модели предъявляют наименьшие требования к вычислительным ресурсам по сравнению с многожидкостными или лагранжевым (LPT) подходами;
- в расчетной области могут присутствовать объёмы воды (водяной струи), сопоставимые по размерам с масштабами газовых струй.

Таким образом, разрабатываемая модель должна удовлетворять ряду критериев:

- не предъявлять существенно более высокие требования к вычислительным ресурсам по сравнению с моделью газодинамики;
- обеспечивать возможность переноса энергии и импульса в случаях, когда объёмная доля газа низкая (менее 1-5%, например) средствами моделей на основе LPT;
- поддерживать атомизацию (дробление струи) за счет взаимодействия с потоком газа;
- поддерживать взаимодействие капель с крупными объёмами жидкости, разрешаемыми с помощью модели VOF явно;
- осуществлять перенос импульса, энергии и массы из жидкой в газообразную фазу (фазы);
- учитывать сжимаемость газовой составляющей и движение со сверхзвуковыми скоростями;
- обеспечивать возможность учета многокомпонентной смеси;
- быть масштабируемой и расширяемой.

Количество частиц может быть весьма значительным, расстояния между ними могут достаточно малыми, а дисперсионный состав облака разнообразный,

что может сделать модель крайне сложной (столкновения, слипания, образование вторичных брызг, воздействие с помощью спутного следа и пр). В этом случае интерес представляют модели смеси на основе эволюции объёмной плотности межфазной поверхности [9], которая были специально разработана для больших значений чисел Re и We. Тем не менее в рассматриваемом случае применение этой модели сопряжено со значительными затруднениями:

- имеется большое количество эмпирических соотношений, которой необходимо верифицировать для случая подачи водяных струй в сверхзвуковые струи газа;
- в исходной модели предполагаются дозвуковые струи потока, что говорит о необходимости доработки для совместного решения с учетом баланса энергии потока.

В данной работе за основу взята модель, соединяющая подходы VOF и LPT [8, 10] и модель сжимаемого течения на основе схемы Курганова-Тадмора и алгоритма PIMPLE [11]. При этом в дальнейшем она может быть расширена для совместного использования с многожидкостными моделями или моделью на основе ELSA [9].

В основе предлагаемой многомасштабной модели лежит техника переноса маркера жидкости (объёмной доли жидкости) для определения положения межфазной границы с помощью балансного уравнения:

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot \left(\vec{U}\, \alpha_l\right) - (\alpha_l + K_C)\nabla \cdot \left(\vec{U}\right) = \dot{v}_l\,, \tag{1}$$

где α_l – объёмная доля жидкости, $\vec{U}$ – среднемассовое поле скорости газожидкостной смеси, $\dot{v}_l$ – источник или сток объёма жидкости в континуальной системе, либо в капельную систему, K_C – коэффициент сжимаемости двухфазной смеси, определяемый в соответствии с [12, 13]:

$$K_C = \frac{\rho_{gm} c_{gm}^2 - \rho_l c_l^2}{\dfrac{\rho_{gm} c_{gm}^2}{\alpha_{gm}} + \dfrac{\rho_l c_l^2}{\alpha_l}}, \tag{2}$$

где индекс gm соответствуют смеси газов, индекс l – жидкости, ρ – истинная (термодинамическая) плотность жидкой или газообразной фазы, c – скорость звука жидкой или газообразной фазы, α_{gm} – объёмная доля смеси газов (газовой фазы).

Источниковое слагаемое $\dot{v}_l$ состоит из двух частей (3): $\dot{m}_d$ - переход массы жидкости из континуального описания в переменные Лагранжа (в систему капель) и обратно и $\dot{m}_v$ – объёмное вскипание.

$$\dot{v}_l = \frac{1}{\rho_l}\dot{m}_d + \frac{1}{\rho_l}\,\dot{m}_v\,. \tag{3}$$

В случае образования структур жидкой фазы, размеры которых меньше сеточного масштаба или сеточное разрешение не позволяет воспроизводить их с достаточным качеством (например, менее 7 ячеек на диаметр), эти структуры переводятся в описание в переменных Лагранжа (добавляются к облаку частицам) с помощью слагаемого $\dot{m}_d$. Если же частицы сталкиваются с жидкостью, то они «переводятся» в континуальное описание жидкой фазы с помощью слагаемого $\dot{m}_d$. Подробно этот подход описан в работах [8, 10]. Согласно (3) жидкость может как испаряться с поверхности капель, так и вскипать в объёме и переходить в газообразное состояние.

Движение капель описывается с помощью системы обыкновенных дифференциальных уравнений, составленных для каждой из капель (или кластера капель), и включающая в себя уравнения движения центра масс капель, баланса масс капель, баланса импульса капель и баланса энергии капель.

Предполагается, что газожидкостная система, описываемая континуальным подходом, находится в механическом и термодинамическом равновесии и тогда можно записать уравнения баланса массы, энергии (полной энтальпии) и импульса:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \left(\vec{U}\,\rho\right) = \dot{m}_d\,, \tag{4}$$

$$\frac{\partial \rho h^{tot}}{\partial t} + \nabla \cdot \left(\vec{U}\,\rho h^{tot}\right) - \frac{\partial p}{\partial t} - \sum_i \nabla \cdot (\rho D_i \nabla Y_i\, h_i) = -\nabla \cdot \vec{q} + \dot{m}_d e_d + \nabla \cdot \left(\hat{\Pi} \cdot \vec{U}\right), \tag{5}$$

$$\frac{\partial \rho \vec{U}}{\partial t} + \nabla \cdot \left(\vec{U}\,\rho \otimes \vec{U}\right) = \dot{m}_d \vec{U}_d - \nabla p + \nabla \cdot \left(\hat{\Pi}\right), \tag{6}$$

где $h^{tot} = h + \frac{1}{2}\vec{U} \cdot \vec{U}$ – удельная полная энтальпия смеси, $h = u + p/\rho$ – удельная энтальпия смеси, u – удельная внутренняя энергия смеси. Удельная энтальпия смеси является взвешенной суммой энтальпий её составляющих: $h = Y_l h_l + Y_v h_v + Y_g h_g + Y_a h_a$, p – статическое давление в смеси, $\hat{\Pi}$ – тензор вязких напряжений в смеси, e_d – удельная полная энергия капель перешедших в/из континуальное состояние, $\vec{U}_d$ – скорость капель перешедших в/из континуальное состояние, Y_i – массовая доля компоненты газожидкостной смеси, h_i – удельная энтальпия компоненты газожидкостной смеси, D_i – коэффициент диффузии компоненты газожидкостной смеси, $\vec{q}$ – вектор теплового потока.

Тензор вязких напряжений $\hat{\Pi}$ с учетом гипотезы Стокса имеет вид:

$$\hat{\Pi} = \mu\left(\nabla\vec{U} + \left(\nabla\vec{U}\right)^T\right) - \frac{2}{3}\mu I \nabla \cdot \vec{U}\,, \tag{7}$$

где μ – коэффициент динамической вязкости смеси рассчитываемый по таблицам свойств среды в зависимости от температуры и состава смеси, I – единичный тензор.

Вектор теплового потока вычисляется в соответствии с законом Фурье:

$$\vec{q} = -\lambda \nabla T \,, \tag{8}$$

где λ – коэффициент теплопроводности смеси рассчитываемый по таблицам свойств среды в зависимости от температуры и состава смеси.

Система дополняется уравнениями баланса массы компонент газовой смеси в диффузионном приближении для учета взаимного движения компонентов:

$$\frac{\partial \rho Y_a}{\partial t} + \nabla \cdot \left(\vec{U}\, \rho\, Y_a\right) - \nabla \cdot (\rho\, D_a \nabla Y_a) = 0 \,. \tag{9}$$

$$\frac{\partial \rho Y_g}{\partial t} + \nabla \cdot \left(\vec{U}\, \rho\, Y_g\right) - \nabla \cdot \left(\rho\, D_g \nabla Y_g\right) = 0 \,. \tag{10}$$

$$\frac{\partial \rho Y_v}{\partial t} + \nabla \cdot \left(\vec{U}\, \rho\, Y_v\right) - \nabla \cdot (\rho\, D_v \nabla Y_v) = \dot{m}_v + \dot{m}_{dv} \,, \tag{11}$$

где $\dot{m}_{dv}$ – источник пара за счет испарения с поверхности капель.

Коэффициенты диффузионного переноса зависят от состава среды (равны нулю для случая соседства с жидкой фазой), и рассчитываются, например с использованием бинарных коэффициентов. Плотность среды ρ вычисляется через массовые доли и истинные плотности компонент:

$$\frac{1}{\rho} = \frac{Y_l}{\rho_l} + \frac{Y_v}{\rho_v} + \frac{Y_a}{\rho_a} + \frac{Y_g}{\rho_g}. \tag{12}$$

Плотность жидкости ρ_l вычисляется с использованием линейного уравнения состояния: $\rho_l = \rho_{l0} + \psi_l(p - p_{l0})$, где ρ_{l0} и p_{l0} – референсные значения плотности и давления, ψ_l – коэффициент сжимаемости жидкости. Плотности пара, воздуха и продуктов сгорания ρ_v , ρ_a , ρ_g вычисляются в соответствии с уравнением совершенного газа и законом Дальтона. Уравнения состояния компонент смеси в основные балансные уравнения (1) – (10) непосредственно не входят и, следовательно, принципиально возможно использование этой модели совместно с более точными способами термодинамического описания фаз.

При необходимости для замыкания модели турбулентности может использоваться метод крупных вихрей.

4.2. Модель газодинамики

Модель газодинамики многокомпонентного потока применяется в подобласти b) после выхода из газохода и включает в себя:

- уравнение сохранения массы смеси (4) без источников водяной фазы;
- уравнение баланса энергии (5);
- уравнение баланса импульса (6);
- уравнений переноса массовый концентраций газообразных компонент смеси воздуха, пара и продуктов сгорания (9) – (11);
- уравнений состояния совершенного газа для компонент смеси;
- замыкающих соотношений для вычисления вязких потоков внутренней энергии и импульса (7) и (8).

При необходимости для замыкания модели турбулентности может использоваться метод крупных вихрей. Для корректного разрешения масштабов, связанных с распространением акустических волн требуется сеточное разрешение от 10 ячеек (в случае высокоточных численных методов) до примерно 60-80 ячеек (в случае схемы второго порядка) на длину волны.

4.3. Модель акустики в дальнем поле

Для учета акустики в дальнем поле используется интегральная модель на основе решения уравнений аналогии Фоукс Вильямса-Хокинга в постановке Фарассат 1A [14] с учётом отложенного времени. Рассматривается распространение возмущений в воздушной среде при нормальных условиях.

5. Реализация модели

Реализуемая модель акустики старта ракеты носителя включает в себя последовательное выполнение трёх расчетных этапов (подмоделей):

- расчёт взаимодействия газовых и водяных струй в газоходе и накопление статистики по газодинамическим величинам (давление, температура, скорость) на выходном срезе газохода;
- расчёт с учётом накопленной на предыдущем этапе статистики выхода турбулентной многокомпонентной струи газа из газохода в область над плоскостью стартового стола и накопление статистики по газодинамическим величинам на контрольной поверхности Кирхгофа;
- определение акустического поля с использованием аналогии Фоукс Вильямса-Хокинга и накопленной на втором этапе статистики.

Для эффективной обработки статистических данных на границах областей возможно использование методов анализа данных, таких как POD [15], прототип реализации которых выполнен в открытой библиотеке ITHACA-FV [16, 17].

Модель распространения турбулентной сжимаемой смеси над стартовым столом может быть реализована с применением разрывного метода Галёркина, реализованного в открытой библиотеке Nektar++ [18,19]. Для реализации

модели акустики дальнего поля целесообразно адаптировать библиотеку libAcoustics [20], разрабатываемую в ИСП РАН.

Моделей и их открытых реализаций, которые бы полностью соответствовали процессам, происходящим при взаимодействии газовых струй с водяными струями во время старта ракеты-носителя, в настоящее время не существует. При этом можно утверждать, что в зрелой стадии разработки находятся следующие модели, описывающие различные стороны этого процесса:

- модель газодинамики продуктов сгорания и гомогенной газоводяной смеси на основе гибридной метода Курганова-Тадмора и алгоритма связи скорости и давления проекционного типа PIMPLE [11, 21-23];
- модель движения несжимаемой двухфазной смеси с разделом фаз и фазовыми превращениями [24, 25];
- модель движения облака частиц [26], модель движения и тепломассообмена распыленной водяной струи в горячей газовой среде [27].
- модель турбулентного течения на основе метода крупных вихрей, реализованная в пакете OpenFOAM;
- многомасштабная модель движения двухфазной струи с атомизацией на основе Эйлер-Лагранжевого описания [8, 10].

Таким образом, с использованием перечисленных разработок, реализация модели взаимодействия газовых и водяных струй в газоходе включает в себя алгоритм интегрирования уравнений газожидкостной смеси (1) – (12), под-модель переноса фронта воды, под-модель взаимодействия жидкой фазы из континуального представления с каплями, под-модель движения облака капель, под-модель испарения капель, под-модель вскипания объёма жидкости, под-модель турбулентного переноса на основе метода крупных вихрей. С учетом превалирования в данном списке реализаций подмоделей на основе пакета OpenFOAM, целесообразно осуществлять разработку на основе этой библиотеки.

Алгоритм интегрирования уравнений модели и принципиальная схема взаимодействия подмоделей представлены на рис. 2. Из рисунка видно, что на каждом временном шаге моделируемого физического процесса можно выделить три основных этапа:

- перенос межфазной границы и облака капель;
- решение уравнений модели турбулентности;
- решение уравнений баланса массы, энергии и импульса газожидкостной смеси.

Последний этап строится в соответствии с алгоритмом, предложенным в [11].

6. Заключение

В настоящее время всё большую популярность набирает направление развития информационных технологий, связанное с разработкой цифровых двойников технических систем – специализированных программ для имитации происходящих в ТС процессов.

Рис. 2. Общий алгоритм интегрирования уравнений в модели взаимодействия газовых и водяных струй

Fig. 2. Multiscale model equations solution sequence and sub-models interaction diagram

К алгоритмам и программам, разрабатываемым в рамках данного направления, предъявляются требования, обусловленные особенностями эксплуатации такого программного обеспечения: точность, быстродействие, надёжность и мультифизичность. Последнее требование сопряжено с многосторонностью эксплуатируемой технической системы, выражающееся в

разнообразии протекающих в ней процессов характеризуемых большим разбросом масштабов определяющих физических величин.

В качестве примеров технических систем и происходящих в них процессов приведены выход газовой струи из газодобывающей скважины на дне океана, старт ракеты носителя, глушение скважин, эксплуатирующих пласты с высоким газовым фактором, процессы изготовления чугуна в доменных печах. Отмечено, что решение перечисленных задач невозможно без совместного и согласованного привлечения иерархии математических моделей разработанных для описания явлений различного масштаба.

На примере задачи исследования акустического шума при старте ракеты-носителя дано развёрнутое описание многомасштабной математической модели начального этапа подъёма ракеты, включающей использование:

- модели акустики дальнего поля на основе акустической аналогии Ффоукс Вильямса Хокинга;
- модели турбулентной газовой струи – источника шума, основанной на решении осреднённых по пространству уравнений Навье-Стокса, баланса энергии и переноса компонент смеси сжимаемых газов;
- модели взаимодействия сверхзвуковых газовых и дозвуковых водяных струй в газоходе с учётом: ударных волн, волн разрежения, контактных разрывов, движения крупномасштабных и маломасштабных (капель) элементов водяных струй, а также процессов фазовых превращений.

Предложенное описание происходящих при старте РН процессов включает в себя 5 математических моделей, позволяющих описывать пространственные масштабы от 100 мкм до 10 м, временные масштабы от 0.01 мс до нескольких секунд, плотности среды от 1 кг/м^3 до 1000 кг/м^3, скорости сред, соответствующие значениям числа Маха от 0.01 до 5-6.

Реализация таких комплексных алгоритмов может быть существенно упрощена за счет привлечениях библиотек на основе открытого исходного кода, таких как Nektar++, OpenFOAM, LIGGGHTS и др.

Список литературы

[1] C.K.W. Tam, K. Viswanathan, K.K. Ahuja, J. Panda. The sources of jet noise: experimental evidence. Journal of Fluid Mechanics, vol. 615, 2008, vp. 253-292.

[2] C.K.W. Tam. Theoretical aspects of supersonic jets noise. In Proc. of the First Annual High-Speed Research Workshop, Part 2, 1992, pp. 647-662.

[3] C.K.W. Tam, M. Golebiowsky, J.M. Seiner. On the two components of turbulent mixing noise from supersonic jets. In Proc. of the Aeroacoustics Conference, 1996.

[4] C.K.W. Tam, H. Shen, G. Raman. Screech tones of supersonic jets from beveled rectangular nozzles. AIAA Journal, vol. 35, issue 7, 1997, pp. 1119-1125.

[5] P. Rajput, S. Kumar. Jet noise reduction by downstream fluidic injection: effect of injection pressure ratio and number of injection ports. In Proc. of the 2018 AIAA Aerospace Sciences Meeting, 2018.

[6] D.G. Crighton. Basic principles of aerodynamic noise generation. Progress in Aerospace Sciences, vol. 16, issue 1, 1975, pp 31-96.
[7] M. Kandula, B. Vu. On the scaling laws for jet noise in subsonic and supersonic flow. NASA Preprint No. KSC-2003-040, 2003.
[8] Y. Ling, S. Zaleski, R. Scardovelli. Multiscale simulation of atomization with small droplets represented by a Lagrangian point-particle model. International Journal of Multiphase Flow, Elsevier, vol. 76, 2015, pp. 122-143.
[9] J.M. Garcia-Oliver, J.M. Pastor, A. Pandal, N. Trask, E. Baldwin, D.P. Schmidt. Disel spray CFD simulations based on the Sigma-Y Eulerian atomization model. Atomization and Sprays, vol. 23, no. 1, 2003, pp. 71-95.
[10] J.L. Estivalezes, D. Zuzio, B. DiPierro. An improved multiscale Eulerian-Lagrangian method for simulation of atomization process. Computers & Fluids, vol. 176, 2018, pp. 285-301.
[11] M. Kraposhin, M. Banholzer, I. Marchevsky, M. Pfitzner. A hybrid pressure-based solver for nonideal single-phase fluid flows at all speeds. International Journal for Numerical Methods in Fluids, vol. 88, issue 2, 2018, pp. 79-99.
[12] R. Saurel, O. Le. Metayer, J. Massoni, S. Gavrilyuk. Shock jump relations for multiphase mixtures with stiff mechanical relaxation. Shock Waves, vol. 16, issue 3, 2007, pp 209–232.
[13] F. Denner, C.-N. X., Xiao, B.G.M. van Wachem Pressure-based algorithm for compressible interfacial flows with acoustically-conservative interface discretization. Journal of Computational Physics, vol. 367, 2018, pp. 192-234.
[14] F. Farassat. Derivation of Formulation 1 and 1A of Farassat. NASA Report NASA/TM-2007-214853, March, 2007
[15] М. Калугин, В. Корчагова, М. Крапошин, И. Марчевский, В. Морева. Использование инструментов анализа больших данных при решении задач газовой динамики и акустики. Вестник МГТУ им. Н.Э. Баумана, том 78, №3, 2018.
[16] G. Stabile, S. Hijazi, A. Mola, S. Lorenzi, G. Rozza. POD-Galerkin reduced order methods for CFD using Finite Volume Discretisation: vortex shedding around a circular cylinder. Communications in Applied and Industrial Mathematics, vol. 8, no. 1, 2017, pp. 210-236.
[17] G. Stabile G. Rozza. Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier-Stokes equations. Computers & Fluids, vol. 173, 2018, pp. 273-284
[18] C. D. Cantwell, D. Moxey, A. Comerford et al. Nektar++: An open-source spectral/hp element framework. Computer physics communications, vol. 192, 2015, pp. 205-219.
[19] Nektar++: spectral/hp element framework. Режим доступа: https://www.nektar.info/downloads/, дата обращения 12.12.2018.
[20] A. Epikhin, I. Evdokimov, M. Kraposhin, M. Kalugin, S. Strijhak. Development of a Dynamic Library for Computational Aeroacoustics Applications Using the OpenFOAM Open Source Package. Procedia Computer Science, vol. 66, 2015, pp. 150-157.
[21] M. Kraposhin, A. Bovtrikova, S. Strijhak. Adaptation of Kurganov-Tadmor Numerical Scheme for Applying in Combination with the PISO Method in Numerical Simulation of Flows in a Wide Range of Mach Numbers. Procedia Computer Science, vol. 66, 2015, pp. 43-52.

[22] М.В. Крапошин. Возможности гибридного метода аппроксимации конвективных потоков при моделировании течений сжимаемых сред. Труды ИСП РАН, том 28, вып. 3, 2016, стр. 267-326. DOI: 10.15514/ISPRAS-2016-28(3)-16.
[23] M. Kraposhin, V. Korchagova, S.Strizhak, J. Beilke, A. Al-Zoubi. Comparison of the performance of open-source and commercial CFD packages for simulating supersonic compressible jet flows. In Proc. of Ivannikov Memorial Workshop'2018, Erevan, 2018.
[24] S.S. Deshpande, L. Anumolu, M.F. Trujillo. Evaluating the performance of the two-phase flow solver interFoam. Computational Science & Discovery, vol. 56, 2012.
[25] J. Roenby, B.E. Larsen, H. Bredmose, H. Jasak. A new volume-of-fluid method in OpenFOAM. In Proc. of the 7th Internatinal Conference on Computational Methods in Marine Engineering, 2017
[26] OpenFOAM Lagrangian solvers. Режим доступа: https://www.openfoam.com/documentation/cpp-guide/html/group__grpLagrangianSolvers.html, дата обращения 12.12.2018.
[27] Цой А.С. Режимы и механизмы подавления пламени распылённой водой. Диссертация кандидата технических наук, Санкт-Петербургский политехнический университет Петра Великого, 2016, 177 стр.

Multiscale approach for simulation of complex transient processes of fluid flows in technical systems

M.V. Kraposhin <m.kraposhin@ispras.ru>

Ivannikov Institute for System Programming of Russian Academy of Sciences, Russia, Moscow, Solzhenitsyna str., 25

Annotation. The paper presents a multiscale approach for simulation of the two-phase flows processes in complex technical systems. The multiscale approach is based both on the division of the computational domain into subdomains with their own system of equations, as well as on the splitting of the initial system of equations into several subsystems each is valid to the corresponding scale under consideration. The problem of the far field acoustic noise calculation during the launch of a rocket vehicle is considered as an example of the possible use of a multiscale model. The case is studied with account to noise suppression due to water supply into the gas jets of the propulsion system. Other areas of application of the multiscale model include the cases of the oil and gas industry: killing gas-producing wells located at great depth, killing oil wells with a high gas factor at the fields. The proposed multi-scale mathematical model includes 5 sub-models: 1) gas dynamics of high-speed multicomponent gas mixture flows; 2) the hydrodynamics of a two-phase mixture flow in a homogeneous approximation with the account for the compressibility of the gas phase and the mass exchange between the phases; 3) the liquid-gas interface transport; 4) the transport of a cloud of droplets and its interaction with a gas-liquid medium; 5) noise calculation in the far field using the Ffowks Williams-Hawking acoustic analogy. The model can be extended to include additional sub-models, such as the Eulerian-Lagrange Jet Atomization. The implementation of the submodels can be done on the basis of open source packages: OpenFOAM, Nektar ++, ITHACA-FV. The acoustics library and the hybrid algorithm for compressible homogeneous two-phase flow are implemented as libAcoustics and hybridCentralSolvers modules based on the OpenFOAM open package. The source code of the developed model is freely available through the GiHub project https://github.com/unicfdlab.

Keywords: multiscale models; numerical simulation; numerical schemes; compressible flows; multiphase flows; acoustics; computational hydro- aero- and gas-dynamics; open source software; finite volume method; discontinuous Galerkin method; libAcoustics; hybridCentralSolvers; OpenFOAM; Nektar++; Volume Of Fluid; Lagrangian Particle Tracking

DOI: 10.15514/ISPRAS-2018-30(6)-15

For citation: Kraposhin M.V. Multiscale approach for simulation of complex transient processes of fluid flows in technical systems. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 275-292 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-15

References

[1] C.K.W. Tam, K. Viswanathan, K.K. Ahuja, J. Panda. The sources of jet noise: experimental evidence. Journal of Fluid Mechanics, vol. 615, 2008, vp. 253-292.

[2] C.K.W. Tam. Theoretical aspects of supersonic jets noise. In Proc. of the First Annual High-Speed Research Workshop, Part 2, 1992, pp. 647-662.

[3] C.K.W. Tam, M. Golebiowsky, J.M. Seiner. On the two components of turbulent mixing noise from supersonic jets. In Proc. of the Aeroacoustics Conference, 1996.

[4] C.K.W. Tam, H. Shen, G. Raman. Screech tones of supersonic jets from beveled rectangular nozzles. AIAA Journal, vol. 35, issue 7, 1997, pp. 1119-1125.

[5] P. Rajput, S. Kumar. Jet noise reduction by downstream fluidic injection: effect of injection pressure ratio and number of injection ports. In Proc. of the 2018 AIAA Aerospace Sciences Meeting, 2018.

[6] D.G. Crighton. Basic principles of aerodynamic noise generation. Progress in Aerospace Sciences, vol. 16, issue 1, 1975, pp 31-96.

[7] M. Kandula, B. Vu. On the scaling laws for jet noise in subsonic and supersonic flow. NASA Preprint No. KSC-2003-040, 2003.

[8] Y. Ling, S. Zaleski, R. Scardovelli. Multiscale simulation of atomization with small droplets represented by a Lagrangian point-particle model. International Journal of Multiphase Flow, Elsevier, vol. 76, 2015, pp. 122-143.

[9] J.M. Garcia-Oliver, J.M. Pastor, A. Pandal, N. Trask, E. Baldwin, D.P. Schmidt. Disel spray CFD simulations based on the Sigma-Y Eulerian atomization model. Atomization and Sprays, vol. 23, no. 1, 2003, pp. 71-95.

[10] J.L. Estivalezes, D. Zuzio, B. DiPierro. An improved multiscale Eulerian-Lagrangian method for simulation of atomization process. Computers & Fluids, vol. 176, 2018, pp. 285-301.

[11] M. Kraposhin, M. Banholzer, I. Marchevsky, M. Pfitzner. A hybrid pressure-based solver for nonideal single-phase fluid flows at all speeds. International Journal for Numerical Methods in Fluids, vol. 88, issue 2, 2018, pp. 79-99.

[12] R. Saurel, O. Le. Metayer, J. Massoni, S. Gavrilyuk. Shock jump relations for multiphase mixtures with stiff mechanical relaxation. Shock Waves, vol. 16, issue 3, 2007, pp 209–232.

[13] F. Denner, C.-N. X., Xiao, B.G.M. van Wachem. Pressure-based algorithm for compressible interfacial flows with acoustically-conservative interface discretization. Journal of Computational Physics, vol. 367, 2018, pp. 192-234.

[14] F. Farassat. Derivation of Formulation 1 and 1A of Farassat. NASA Report NASA/TM-2007-214853, March, 2007

[15] M. Kalugin, V. Korchagova, M. Kraposhin, I. Marchevsky, V. Moreva. Using Big Analytics Tools in Performance of Gas Dynamics and Acoustics Tasks. Herald of the Bauman Moscow State Technical University, vol. 78, no. 3, 2018 (in Russian).

[16] G. Stabile, S. Hijazi, A. Mola, S. Lorenzi, G. Rozza. POD-Galerkin reduced order methods for CFD using Finite Volume Discretisation: vortex shedding around a circular cylinder. Communications in Applied and Industrial Mathematics, vol. 8, no. 1, 2017, pp. 210-236.

[17] G. Stabile G. Rozza. Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier-Stokes equations. Computers & Fluids, vol. 173, 2018, pp. 273-284

[18] C. D. Cantwell, D. Moxey, A. Comerford et al. Nektar++: An open-source spectral/hp element framework. Computer physics communications, vol. 192, 2015, pp. 205-219.

[19] Nektar++: spectral/hp element framework. Available at: https://www.nektar.info/downloads/, accessed 12.12.2018.

[20] A. Epikhin, I. Evdokimov, M. Kraposhin, M. Kalugin, S. Strijhak. Development of a Dynamic Library for Computational Aeroacoustics Applications Using the OpenFOAM Open Source Package. Procedia Computer Science, vol. 66, 2015, pp. 150-157.

[21] M. Kraposhin, A. Bovtrikova, S. Strijhak. Adaptation of Kurganov-Tadmor Numerical Scheme for Applying in Combination with the PISO Method in Numerical Simulation of Flows in a Wide Range of Mach Numbers. Procedia Computer Science, vol. 66, 2015, pp. 43-52.

[22] M.V. Kraposhin. Study of capabilities of hybrid scheme for advection terms approximation in mathematical models of compressible flows. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 3, 2016, pp. 267-326 (in Russian). DOI: 10.15514/ISPRAS-2016-28(3)-16.

[23] M. Kraposhin, V. Korchagova, S.Strizhak, J. Beilke, A. Al-Zoubi. Comparison of the performance of open-source and commercial CFD packages for simulating supersonic compressible jet flows. In Proc. of Ivannikov Memorial Workshop'2018, Erevan, 2018.

[24] S.S. Deshpande, L. Anumolu, M.F. Trujillo. Evaluating the performance of the two-phase flow solver interFoam. Computational Science & Discovery, vol. 56, 2012.

[25] J. Roenby, B.E. Larsen, H. Bredmose, H. Jasak. A new volume-of-fluid method in OpenFOAM. In Proc. of the 7th Internatinal Conference on Computational Methods in Marine Engineering, 2017

[26] OpenFOAM Lagrangian solvers. Available at: https://www.openfoam.com/documentation/cpp-guide/html/group__grpLagrangianSolvers.html, accessed 12.12.2018.

[27] Tsoy A.S. Regimes and mechanisms for suppressing flames from sprayed water. PhD Thesis, Peter the Great St. Petersburg Polytechnic University, 2016, 177 p. (In Russian).

Минимальный базис модуля сизигий старших членов

А.В. Шокуров <shok@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. Системы полиномиальных уравнений – один из наиболее универсальных математических объектов. Почти все задачи криптографического анализа можно свести к поиску решений систем полиномиальных уравнений. Соответствующее направление исследований принято называть алгебраическим криптоанализом. С точки зрения вычислительной сложности, системы полиномиальных уравнений охватывают весь диапазон возможных вариантов, от алгоритмической неразрешимости диофантовых уравнений до хорошо известных эффективных методов решения линейных систем. Метод Бухбергера приводит систему алгебраических уравнений к системе специального вида, определяемой базисом Гребнера исходной системы уравнений, позволяющему использовать исключение зависимых переменных. Фундаментом определения базиса Гребнера является допустимое упорядочение на множестве термов. Множество допустимых упорядочений на множестве термов бесконечно и даже континуально. Наиболее трудоемким этапом при нахождении базиса Гребнера с помощью алгоритма Бухбергера является доказательство приводимости к нулю всех S-многочленов. Известно, что достаточно провести такую проверку только для любого подмножества таких многочленов, представляющих систему образующих K[X]-модуля S-многочленов. Возникает естественная задача нахождения такой минимальной системы образующих. Существование такой системы образующих следует из теоремы Накаямы. Предложен алгоритм построения такого базиса для любого упорядочения.

Ключевые слова: кольцо многочленов; поле; идеал; сизигия; базис Гребнера; алгоритм Бухбергера; допустимый порядок

DOI: 10.15514/ISPRAS-2018-30(6)-16

Для цитирования: Шокуров А.В. Минимальный базис модуля сизигий старших членов. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 293-304. DOI: 10.15514/ISPRAS-2018-30(6)-16

1. Основные определения и обозначения

Пусть K – поле и $K[x_1, \dots, x_n]$ – кольцо многочленов от независимых переменных $X = \{x_1, \dots, x_n\}$. Термом на переменных X будем называть выражения вида $x_1^{m_1} \dots x_n^{m_n}$, где $m_1, \dots, m_n$ – неотрицательные целые числа.

Для термов в дальнейшем будем использовать обозначение $x_1^{m_1}\dots x_n^{m_n} = x^{\omega}$, где $\omega = (m_1, \dots, m_n) \in \mathbb{Z}_+^n$. Множество всех термов на множестве переменных X обозначим через $T\langle X\rangle$. Множество всех термов является коммутативным моноидом относительно умножения. Заметим, что каждый многочлен $f \in [x_1, \dots, x_n]$ представим в виде

$$\boldsymbol{f = \sum_{t \in T\langle X\rangle} \alpha_t t, \alpha_t \in K,} \qquad (1)$$

причем только конечное число коэффициентов $\boldsymbol{\alpha_t \neq 0}$. Обозначим через $\boldsymbol{T_f\langle X\rangle}$ множество ненулевых термов многочлена $\boldsymbol{f}$

$$\boldsymbol{T_f\langle X\rangle = \{t \in T\langle X\rangle | \alpha_t \neq 0\},}$$

причем только конечное число коэффициентов $\boldsymbol{\alpha_t \neq 0}$. Обозначим через $\boldsymbol{T_f\langle X\rangle}$ множество ненулевых термов многочлена $\boldsymbol{f}$

$$\boldsymbol{T_f\langle X\rangle = \{t \in T\langle X\rangle | \alpha_t \neq 0\}.}$$

Определение 1.1. Допустимым порядком на множестве термов $\boldsymbol{T\langle X\rangle}$ называется линейный порядок $\prec$, удовлетворяющий свойствам

$$\boldsymbol{\forall t, t_1, t_2 \in T\langle X\rangle\ t_1 \prec t_2 \Rightarrow tt_1 \prec tt_2,}$$
$$\boldsymbol{\forall t \in T\langle X\rangle\ t \neq 1 \Rightarrow 1 \prec t.}$$

Фиксируем допустимый порядок на $\boldsymbol{T\langle X\rangle}$. Отметим, что множество термов вполне упорядочено относительно $\prec$.

Определение 1.2. Старшим термом многочлена $\boldsymbol{f}$ называется максимальный элемент множества $\boldsymbol{T_f\langle X\rangle}$. Старший терм многочлена $\boldsymbol{f}$ будем обозначать через $\mathrm{HT}(\boldsymbol{f})$.

Определение 1.3. Старшим мономом многочлена $\boldsymbol{f}$, заданного формулой ([1]) называется моном $\boldsymbol{\alpha_t t}$, где $\boldsymbol{t}$ – его старший терм. Старший моном многочлена $\boldsymbol{f}$ будем обозначать через $\mathrm{HM}(\boldsymbol{f})$.

Определение 1.4. Базисом Гребнера идеала $\boldsymbol{I}$ называется его конечный базис $\boldsymbol{G}$, удовлетворяющий условию

$$\boldsymbol{\forall f \in I \exists g \in G \mathrm{HM}(g) | \mathrm{HM}(f).}$$

Определение 1.5. Пусть $\boldsymbol{f, h \in K[X]}$ и $\boldsymbol{G}$ – конечное подмножество в $\boldsymbol{K[X]}$. Будем говорить, что существует простая монотонная редукция $\boldsymbol{f}$ к $\boldsymbol{h}$ по модулю $\boldsymbol{G}$, если $\mathrm{HM}(\boldsymbol{g}) \prec \mathrm{HM}(\boldsymbol{f})$ и для некоторых $\boldsymbol{g, \mu_g \in K[X]}$ выполняется равенство $\boldsymbol{h = f - \mu_g g}$. Простую монотонную редукцию $\boldsymbol{f}$ к $\boldsymbol{h}$ по модулю $\boldsymbol{G}$ будем обозначать через $\boldsymbol{f \to_G h}$. Если имеется последовательность простых монотонных редукций

$$\boldsymbol{f \to_G h_1 \to_G \dots \to_G h_k,}$$

будем говорить что имеется монотонная редукция $\boldsymbol{f}$ к $\boldsymbol{h}$ по модулю $\boldsymbol{G}$ и записывать ее формулой $\boldsymbol{f \to_{G^*} h}$, где $\boldsymbol{h = h_k}$.

Определение 1.6. Пусть $\boldsymbol{f \to_{G^*} h}$ и из $\boldsymbol{f \to_{G^*} h}$ следует, что $\boldsymbol{h = h_1}$. В этом случае будем говорить, что (монотонная) редукция $\boldsymbol{f \to_{G^*} h}$ является полной, а

$\boldsymbol{f}$ приводится к нормальной форме $\boldsymbol{h}$ по модулю множества $\boldsymbol{G}$, и записывать формулой $\boldsymbol{f} \to_{\boldsymbol{G}^*} \underline{\boldsymbol{h}}$.

Напомним определение операции $\boldsymbol{S}(\boldsymbol{f},\boldsymbol{g})$ для многочленов $\boldsymbol{f},\boldsymbol{g} \in \boldsymbol{K}[\boldsymbol{X}]$. Введем обозначение

$$\boldsymbol{u}_{\boldsymbol{f},\boldsymbol{g}} = \frac{\text{НМ}(\boldsymbol{g})}{\text{НОД}(\text{НТ}(\boldsymbol{f}),\text{НТ}(\boldsymbol{g}))} \qquad (\mathbf{2}).$$

Тогда $\boldsymbol{S}(\boldsymbol{f},\boldsymbol{g}) = \boldsymbol{u}_{\boldsymbol{f},\boldsymbol{g}}\boldsymbol{f} - \boldsymbol{u}_{\boldsymbol{g},\boldsymbol{f}}\boldsymbol{g} \in \boldsymbol{K}[\boldsymbol{X}]$.

Пусть $\boldsymbol{G} = \{\boldsymbol{g}_1, \dots, \boldsymbol{g}_m\}$. Алгоритм нахождения базиса Гребнера основан на следующем критерии Бухбергера [1].

Теорема 1. Конечное множество $\boldsymbol{G} \subset \boldsymbol{K}[\boldsymbol{X}]$ является базисом Гребнера идеала $\boldsymbol{I} \subset \boldsymbol{K}[\boldsymbol{X}]$, порожденного множеством $\boldsymbol{G}$, тогда и только тогда, когда

$$\forall \boldsymbol{f},\boldsymbol{g} \in \boldsymbol{G}\ \ \boldsymbol{S}(\boldsymbol{f},\boldsymbol{g}) \to_{\boldsymbol{G}^*} \mathbf{0}.$$

В действительности нет необходимости проверять условие (2) на всех парах $(\boldsymbol{g}_i,\boldsymbol{g}_j)$. Достаточно проверить его только для пар, $\boldsymbol{S}$-операции на которых порождают $\boldsymbol{S}$-операции для остальных пар [2].

Теорема 2. Пусть $\boldsymbol{H} \subset \boldsymbol{G} \times \boldsymbol{G}$ такое подмножество, что выполнено

$$\forall \mathbf{1} \le \boldsymbol{i} < \boldsymbol{j} \le \boldsymbol{m} \forall (\mathbf{1} \le \boldsymbol{k} < \boldsymbol{l} \le \boldsymbol{m} | (\boldsymbol{g}_k,\boldsymbol{g}_l) \in \boldsymbol{H}) \exists \boldsymbol{f}_{k,l} | \boldsymbol{S}(\boldsymbol{g}_i,\boldsymbol{g}_j) = \sum_{(\boldsymbol{g}_k,\boldsymbol{g}_l)\in\boldsymbol{H}} \boldsymbol{f}_{k,l}\,\boldsymbol{S}(\boldsymbol{g}_k,\boldsymbol{g}_l).$$

Тогда

$$(\forall (\boldsymbol{g}_k,\boldsymbol{g}_l) \in \boldsymbol{H}\ \boldsymbol{S}(\boldsymbol{g}_k,\boldsymbol{g}_l) \to_{\boldsymbol{G}^*} \mathbf{0}) \Rightarrow\ \forall \mathbf{1} \le \boldsymbol{i} < \boldsymbol{j} \le \boldsymbol{m}\ \boldsymbol{S}\big(\boldsymbol{g}_i,\boldsymbol{g}_j\big) \to_{\boldsymbol{G}^*} \mathbf{0}.$$

Наиболее трудоемкой процедурой в алгоритме Бухбергера является проверка выполнения условий $\boldsymbol{S}\big(\boldsymbol{g}_i,\boldsymbol{g}_j\big) \to_{\boldsymbol{G}^*} \mathbf{0}$. Теорема 2 позволяет сократить число таких проверок. Однако чтобы воспользоваться этим критерием, необходимо уметь находить соответствующие множества $\boldsymbol{H}$. Алгебраическое решение этой задачи без вычисления $\boldsymbol{S}$-операций проведено в данной работе.

2. Сизигии старших членов

Пусть $\{\boldsymbol{f}_1, \dots, \boldsymbol{f}_m\}$ – базис идеала $\boldsymbol{I}$. Рассмотрим свободный $\boldsymbol{K}[\boldsymbol{X}]$-модуль с выделенным базисом $\boldsymbol{e}_1, \dots, \boldsymbol{e}_m$, соответствующим базису идеала $\boldsymbol{I} = (\boldsymbol{f}_1, \dots, \boldsymbol{f}_m)$.

Определение 2.1. Пусть $\boldsymbol{F} = (\boldsymbol{f}_1, \dots, \boldsymbol{f}_m) \in \boldsymbol{K}[\boldsymbol{X}]^{\boldsymbol{m}}$. Сизигией старших членов $\boldsymbol{F}$ называется такой $\boldsymbol{S} = (\boldsymbol{h}_1, \dots, \boldsymbol{h}_m) \in \boldsymbol{K}[\boldsymbol{X}]^{\boldsymbol{m}}$, что

$$\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{h}_i \boldsymbol{HM}\,(\boldsymbol{f}_i) = \mathbf{0}.$$

Множество всех сизигий (старших членов) $\boldsymbol{F}$ будем обозначать $\boldsymbol{\mathcal{S}}(\boldsymbol{F})$.

Согласно определению 2.1 выполняется включение $\boldsymbol{\mathcal{S}}(\boldsymbol{F}) \subset \boldsymbol{K}[\boldsymbol{X}]^{\boldsymbol{m}}$. Рассматривая $\boldsymbol{K}[\boldsymbol{X}]^{\boldsymbol{m}}$ как $\boldsymbol{K}[\boldsymbol{X}]$-модуль, получаем представление $\boldsymbol{S} = \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{h}_i\,\boldsymbol{e}_i$, где $\boldsymbol{e}_i$, $\boldsymbol{i} = \mathbf{1}, \dots, \boldsymbol{m}$, – стандартный $\boldsymbol{K}[\boldsymbol{X}]$-базис в $\boldsymbol{K}[\boldsymbol{X}]^{\boldsymbol{m}}$. Очевидно, $\boldsymbol{\mathcal{S}}(\boldsymbol{F})$

является подмодулем $\boldsymbol{K[X]}$-модуля $\boldsymbol{K[X]^m}$. Определение $\boldsymbol{S}$-операции может быть записано с использованием сизигий в виде скалярного произведения

$$\boldsymbol{S(f_i, f_j) = S_{f_i,f_j} \cdot F}, \qquad (3)$$

где $\boldsymbol{S(f_i, f_j) = u_{f_i,f_j} e_i - u_{f_j,f_i} e_j \in S(F)}$. Сизигии $\boldsymbol{S(f_i, f_j)}$ называются критическими.

Определение 2.2. Сизигия $S \in \boldsymbol{S(F)}$ называется однородной степени $\boldsymbol{\omega} \in \mathbb{Z}_+^{\boldsymbol{n}}$, если

$$\boldsymbol{S = (c_1 x_1^{\omega}, \dots, c_m x_m^{\omega}) = \sum_{i=1}^{m} c_i \mathbf{x}^{\omega_i} e_i, \omega_i \in \mathbb{Z}_+^n}, \qquad (4)$$

где $\boldsymbol{c_i \in K}$ и $\boldsymbol{HT(x^{\omega_i} f_i) = x^{\omega}}$ при $\boldsymbol{c_i \neq 0}$. Положим также по определению $\boldsymbol{\deg e_i = HT(f_i)}$.

Определение 2.3. Выражения вида $\boldsymbol{te_i}$, где $\boldsymbol{t \in T\langle X\rangle}$ и $\boldsymbol{i = 1, \dots, m}$, называются термами в $\boldsymbol{K[X]}$-модуле $\boldsymbol{K[X]^m}$. Множество термов в $\boldsymbol{K[X]}$-модуле $\boldsymbol{K[X]^m}$ обозначим через $\boldsymbol{T\langle X\rangle\langle e_1, \dots, e_m\rangle}$.

Лемма 1. Порядок на множестве термов в $\boldsymbol{K[X]}$-модуле $\boldsymbol{K[X]^m}$, заданный формулой

$$\boldsymbol{te_i < st_j \Leftrightarrow \Big(tHT(f_i) \prec sHT(f_j) \vee \Big(tHT(f_i) = sHT(f_j) \wedge HT(f_i) \succ HT(f_j)\Big)\Big)},$$

является линейным и допустимым, т.е.

$$\boldsymbol{\forall t_1, t_2, t_3 \in T\langle X\rangle\ t_1 e_i < t_2 e_j \Rightarrow t_3 t_1 e_i < t_3 t_2 e_j},$$
$$\boldsymbol{\forall V \subset T\langle X\rangle\langle e_1, \dots, e_m\rangle \exists v \in V : \forall v' \in Vv \neq v' \Rightarrow v < v'}.$$

Доказательство. Следует непосредственно из свойств линейности и допустимости порядка $\prec$ на множестве термов $T\langle X\rangle$.

Определение 2.4. Старшим термом сизигии $\boldsymbol{S = (h_1, \dots, h_m)}$ называется наибольший из ненулевых термов $\boldsymbol{HT(h_i)e_i}$. Соответственно, минимальным термом сизигии называется наименьший из ненулевых термов $\boldsymbol{HT(h_i)e_i}$. Старший терм обозначим через $\boldsymbol{HT(S)}$, а младший – $\boldsymbol{LT(S)}$.

В силу определения 2.2 для однородной сизигии $\boldsymbol{S}$ для всех $\boldsymbol{i = 1, \dots, m}$ выполняется равенство $\boldsymbol{dS = \omega_i + de_i}$. В частности сизигии $\boldsymbol{S_{f_i,f_j}}$ из формулы (3) являются однородными и $\boldsymbol{dS_{f_i,f_j} = du_{f_i,f_j} + de_i = du_{f_j,f_i} + de_j}$.

Лемма 2. Сизигии $\boldsymbol{S(F)}$ допускают единственное представление в виде суммы однородных сизигий.

Доказательство. Пусть $\boldsymbol{S = (h_1, \dots, h_m) \in S(F)}$. Для каждого $\boldsymbol{\omega \in \mathbb{Z}_+^n}$ определим $\boldsymbol{h_{i,\omega}}$ как слагаемое многочлена $\boldsymbol{h_i}$, для которого $\boldsymbol{deg(h_{i,\omega} f_i) = \omega}$. Тогда согласно определению сизигии $\boldsymbol{S_\omega = (h_{1,\omega}, \dots, h_{m,\omega})}$ также является сизигией и выполняется соотношение $\boldsymbol{S = \sum_{\omega \in \mathbb{Z}_+ n} S_\omega}$. Единственность такого представления очевидна.

Лемма 3. Любая однородная сизигия $\boldsymbol{S}$ представима в виде суммы $\boldsymbol{S} = \sum_{1 \le i < j \le m} \boldsymbol{g_{i,j} S_{f_i,f_j}}$, где $\boldsymbol{g_{i,j}}$ – однородные многочлены, причем $\boldsymbol{g_{i,j} = 0}$ при $\boldsymbol{dS \neq d(g_{i,j} S_{f_i,f_j})}$. Иными словами, критические сизигии составляют базис модуля сизигий.

Доказательство. Пусть утверждение леммы неверно. Пусть $\mathcal{G}$ – подмодуль модуля сизигий $\boldsymbol{F} = \boldsymbol{S(F)}$, порожденный всеми критическими сизигиями. В множестве сизигий $\boldsymbol{F \setminus \mathcal{G}}$ выберем элемент $\boldsymbol{S}$ с минимальным старшим термом. Пусть $\boldsymbol{t^{\omega_i} e_i}$ – ее старший терм и $\boldsymbol{a_i t^{\omega_i} e_i}$ – соответствующий старший моном. Тогда по определению сизигии у нее имеется меньший ненулевой терм вида $\boldsymbol{t^{\omega_j} e_j}$, для которого выполняется равенство $\boldsymbol{t^{\omega_j} HT(f_j) = t^{\omega_i} HT(f_i)}$. Тогда $\boldsymbol{t_i^{\omega} e_i - t_j^{\omega} e_j = t^{\alpha} S_{f_i,f_j}}$, сизигия $\boldsymbol{S - a S_{f_i,f_j}}$ неразложима по критическим сизигиям, а ее старший терм меньше $\boldsymbol{t_i^{\omega} e_i}$, что противоречит выбору сизигии $\boldsymbol{S}$.

Из леммы 2 следует, что модуль сизигий старших членов является однородным модулем с фильтрацией, определяемой множеством $\mathbb{Z}_+^{\boldsymbol{n}}$ относительно однородного кольца многочленов $\boldsymbol{K[X]}$ с той же фильтрацией. Такое однородное кольцо многочленов является локальным – его единственным максимальным однородным идеалом является идеал $\boldsymbol{(x_1, \dots, x_n)}$. Следовательно к модулю сизигий старших членов применима следующая лемма Накаямы [3].

Лемма 4. Пусть $\boldsymbol{A}$ – локальное кольцо и $\mathfrak{m}$ – его (единственный) максимальный идеал, $\boldsymbol{M}$ – конечнопорожденный $\boldsymbol{A}$-модуль и пусть $\boldsymbol{\varphi: M \to M/\mathfrak{m}M}$ – гомоморфизм факторизации. Элементы $\boldsymbol{m_1, \dots, m_s}$ порождают модуль $\boldsymbol{M}$ тогда и только тогда, когда их образы $\boldsymbol{\varphi(m_1), \dots, \varphi(m_s)}$ порождают фактормодуль $\boldsymbol{M/\mathfrak{m}M}$.

Определение 2.5. Базис $\boldsymbol{L \subset S(F)}$ в модуле сизигий $\boldsymbol{S(F)}$ называется минимальным, если любое его собственное подмножество не является базисом модуля сизигий.

Поскольку $\boldsymbol{K[X]/\mathfrak{m}M = K}$, и, следовательно, фактормодуль $\boldsymbol{M/\mathfrak{m}M}$ является векторным пространством над $\boldsymbol{K}$, получаем

Следствие 1. Число элементов минимального базиса модуля сизигий $\boldsymbol{S(F)}$ является его инвариантом.

Теперь из леммы 3 и следствия 1 вытекает

Следствие 2. Существует минимальный базис модуля сизигий $\boldsymbol{S(F)}$, состоящий из критических сизигий. Число элементов такого базиса не зависит от его выбора.

Далее предложен алгоритм нахождения такого минимального базиса. Заметим, что предложенный в работе алгоритм нахождения такого базиса не является полным. Доказательство неприводимости множества элементов $\boldsymbol{\Sigma^*}$ неточное, рассуждение о дальнейшем повторе выполняемой там процедуры по

индукции является неполным и некорректным. По тем же причинам некорректно доказательство по индукции минимальности базиса $\boldsymbol{\Sigma}^{**}$.

3. Разложимые сизигии

Введем несколько обозначений. Напомним, что $F = (f_1, \dots, f_m) \in K[X]^m$ Положим $T(F) = \{\tau_1, \dots, \tau_m\}$, где $\tau_i = HT(f_i)$. Поскольку K – поле, то без ограничения общности можно считать, что где $HM(f_i) = \tau_i$. Также, не ограничивая общность, можно считать, что $\tau_i \prec \tau_j$ при $i < j$. При $i < j$ положим $S_{f_i,f_j} = \alpha_{i,j}e_i + \beta_{i,j}e_j$. Обозначим множество критических сизигий для F через Σ. Согласно лемме 2 множество Σ является однородным базисом $K[X]$-модуля $\mathcal{S}(F)$. Введем обозначение

$$\boldsymbol{\Sigma_\omega} = \{\boldsymbol{s} \in \boldsymbol{\Sigma} | \boldsymbol{ds} \prec \boldsymbol{\omega}\}, \boldsymbol{\omega} \in \mathbb{Z}_+^{\boldsymbol{n}}$$

Определение 3.1. Однородная сизигия s степени $\omega \in \mathbb{Z}_+^n$ называется разложимой, если выполняется соотношение

$$\boldsymbol{s} = \sum_{\boldsymbol{\sigma} \in \boldsymbol{\Sigma_\omega}} \boldsymbol{c_\sigma}\, \boldsymbol{t_\sigma \sigma}, \boldsymbol{\omega} = \boldsymbol{d}(\boldsymbol{t_\sigma \sigma}), \boldsymbol{t_\sigma} \in \boldsymbol{T}(\boldsymbol{K}[\boldsymbol{X}]), \boldsymbol{c_\sigma} \in \boldsymbol{K}.$$

Непосредственно из определения 3.1 следует

Лемма 5. Сумма разложимых однородных сизигий одинаковой степени разложима.

Из определения 3.1. и леммы 5 следует

Лемма 6. Пусть Σ_0^ω – подмножество множества всех разложимых критических сизигий размерности $\omega \in \mathbb{Z}_+^n$, $\Sigma'_\omega = \Sigma_\omega \cup \Sigma_0^\omega$, s – однородная сизигия размерности ω и

$$\boldsymbol{s} = \sum_{\boldsymbol{\sigma} \in \boldsymbol{\Sigma'_\omega}} \boldsymbol{c_\sigma}\, \boldsymbol{t_\sigma \sigma}, \qquad \boldsymbol{\omega} = \boldsymbol{d}(\boldsymbol{t_\sigma \sigma}), \qquad \boldsymbol{t_\sigma} \in \boldsymbol{T}(\boldsymbol{K}[\boldsymbol{X}]), \qquad \boldsymbol{c_\sigma} \in \boldsymbol{K}. \qquad (\mathbf{5})$$

Тогда сизигия s – разложима.

В дальнейшем количество ненулевых элементов c_σ в разложении (5) будем называть длиной этого разложения.

Лемма 7. Пусть заданы число k, подмножество $L = \{l_1, \dots, l_p\}$ в $\{1, \dots, m\} \setminus \{k\}$, $\omega \in \mathbb{Z}_+^n$ и такая однородная сизигия s размерности ω

$$\boldsymbol{s} = \sum_{\boldsymbol{i} \in \boldsymbol{L}} \boldsymbol{a_i}\, \boldsymbol{t_i} \boldsymbol{S_{f_i,f_k}}, \qquad \boldsymbol{t_i} \in \boldsymbol{T}(\boldsymbol{K}[\boldsymbol{X}]), \qquad \boldsymbol{deg}(\boldsymbol{t_i} \boldsymbol{S_{f_i,f_k}}) = \boldsymbol{\omega},$$

что $a_i \neq 0$ при $i \in L$ и $\sum_{i \in L} a_i = 0$. Тогда имеется разложение

$$\mathbf{s} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{p-1}} \left(\sum_{\boldsymbol{j=1}}^{\boldsymbol{i}} \boldsymbol{a_{l_j}} \right) \mathbf{u_i} \mathbf{S_{f_{l_i},f_{l_{i+1}}}}, \quad \mathbf{u_i} \in \mathbf{T}(\mathbf{K}[\mathbf{X}]).$$

Согласно определению критических сизигий имеем

$$\boldsymbol{S_{f_i,f_j} = u_{i,j}e_{f_i} - u_{j,i}e_{f_j}},$$

причем $\boldsymbol{\deg S_{f_i,f_j} = \deg u_{i,j} + \deg e_{f_i} = \deg u_{j,i} + \deg e_{f_j}}$, $\boldsymbol{u_{i,j}, u_{j,i} \in T(K[X])}$.

По условию леммы для всех $i \in L$ выполняются равенства $\deg t_i + \deg u_{k,i} + \deg e_{f_k} = \omega$ и, следовательно, для всех пар $i, j \in L, i \neq j$

$$\mathbf{t_i S_{f_i,f_k} - t_j S_{f_j,f_k} = t_i(u_{i,k}e_{f_i} - u_{k,i}e_{f_k}) - t_j\left(u_{j,k}e_{f_j} - u_{k,j}e_{f_k}\right) =}$$
$$\mathbf{\left(t_i u_{i,k}e_{f_i} - t_j u_{j,k}e_{f_i}\right) + \left(t_j u_{k,j} - t_{iu_{k,i}}\right)e_{f_k} = t_i u_{i,k}e_{f_i} - t_j u_{j,k}e_{f_j} =}$$
$$\boldsymbol{t_{i,j}S_{f_i,f_j}},\ \text{где}\ \boldsymbol{t_{i,j} = x^{\omega - \deg S_{f_i,f_j}}}.$$

Следовательно**,**

$$\mathbf{s} = \sum_{i\in L} \boldsymbol{a_i t_i S_{f_i,f_k}} = \sum_{i=1}^{p-1}\left(\sum_{j=1}^{i} \boldsymbol{a_{l_j}}\right)\left(\boldsymbol{t_{l_i}S_{f_{l_i},f_{l_k}} - t_{l_{i+1}}S_{f_{l_{i+1}},f_{l_k}}}\right) =$$
$$\sum_{i=1}^{p-1}\left(\sum_{j=1}^{i} \boldsymbol{a_{l_j}}\right)\boldsymbol{t_{l_i,l_{i+1}}S_{f_{l_i},f_{l_{i+1}}}}.$$

Лемма 8. Множество неразложимых критических сизигий Σ^* получается с помощью следующего алгоритма:

Шаг 1. $\boldsymbol{S \coloneqq \Sigma}$.

Шаг 2. Найти сизигию $s \in S$, представимую в виде $s = t_u u + t_v v$, где $\forall w \in \{u, v\} \subset \Sigma$ выполняется $t_w \in T(K[X])$ и либо $\deg t_w \succ 1$, либо $\deg t_w = 1$ и $w \notin S$.

Шаг 3. Если сизигия s найдена, то $S \coloneqq S \setminus \{s\}$ и перейти к шагу 2.

Шаг 4. $\Sigma^* \coloneqq S$.

Доказательство. Без ограничения общности можно считать, что $\deg HT(f_1) \prec \deg HT(f_2) \prec \cdots \prec \deg HT(f_m)$. Определим порядок $<$ на Σ формулой:

$$\boldsymbol{\sigma_1 < \sigma_2 \Leftrightarrow (HT(\sigma_1) < HT(\sigma_2) \vee \left(HT(\sigma_1) = HT(\sigma_2) \wedge LT(\sigma_1) < LT(\sigma_2)\right)).}$$

Согласно лемме 3.1 все исключаемые сизигии разложимы. Достаточно проверить, что будут исключены все разложимые сизигии. Пусть это не так. Выберем в множестве Σ^* разложимую сизигию S_{f_i,f_j} наименьшей степени ω, минимальную относительно порядка $<$. Из ее возможных разложений вида (5), для которых $\Sigma_0^{\omega}=\Sigma^{\omega} \setminus \Sigma^*$, выберем разложение наименьшей длины. В этом разложении выберем максимальное относительно порядка $<$ ненулевое слагаемое $c_{\sigma_0}t_{\sigma_0}\sigma_0$, где $\sigma_0 = S_{f_k,f_l}$ и $k > l$ [1]. Тогда $j \leq k$.

[1] В этом месте доказательства неразложимости сизигий из Σ^* в работе [4] делается неправильное предположение о существовании слагаемого в правой части

При $i < j$ положим $S_{f_i,f_j} = \alpha_{i,j}e_i + \beta_{i,j}e_j$.

Пусть $j = k > l$. Тогда из равенства (5) и леммы 7 следует, что $S_{f_i,f_j} = t_{\sigma_0}S_{f_i,f_j} + t_2S_{f_i,f_l} = t_2S_{f_i,f_l} - t_{\sigma_0}$.Если $t_2 \neq 1$, то сизигия S_{f_i,f_j} должна быть удалена на шаге 3 и, следовательно, не принадлежит Σ^*. Пусть $t_2 = 1$, тогда $S_{f_i,f_l} = S_{f_i,f_j} + t_{\sigma_0}$. Тогда, поскольку $i < j$ и $l < j$, то $S_{f_i,f_l} < S_{f_i,f_j}$ и имеется разложение

$$\boldsymbol{S_{f_i,f_l} = \left(c_{\sigma_0} - 1\right)t_{\sigma_0} + \sum_{\sigma\in\Sigma'_\omega,\sigma\neq\sigma_0} c_\sigma\, t_\sigma \sigma,}$$

что противоречит свойству минимальности сизигии S_{f_i,f_j}.

Следовательно, $k > j > i$. Положим

$$\boldsymbol{\Sigma_\omega(\sigma_0) = \{\sigma \in \Sigma'_\omega \mid HT(t_\sigma) = t_{\sigma_0}t_0e_k, c_\sigma \neq 0\}.}$$

Из соотношения (5) и неравенства $k > j > i$ следует, что

$$\boldsymbol{\sum_{S_{f_l,f_k}\in\Sigma_\omega(\sigma_0)|k>l} c_{S_{f_l,f_k}}\,\mathbf{t}_{f_l,f_k}\alpha_{l,k}e_k = 0.}$$

Поэтому

$$\boldsymbol{\sum_{S_{f_l,f_k}\in\Sigma_\omega(\sigma_0)|k>l} c_{S_{f_l,f_k}} = 0.}$$

Тогда к разложению

$$\boldsymbol{\sum_{S_{f_l,f_k}\in\Sigma'_\omega|k>l} c_{S_{f_l,f_k}}\,\mathbf{t}_{f_l,f_k}S_{f_l,f_k}} \qquad (7)$$

применима лемма 7, позволяющая получить новое разложение

$$\boldsymbol{\sum_{S_{f_l,f_k}\in\Sigma_\omega(\sigma_0)|k>l} \mathbf{c}_{S_{f_l,f_k}}\,\mathbf{t}_{f_l,f_k}S_{f_l,f_k} = \sum_{S_{f_i,f_k}\in\Sigma'_\omega|i<k,j<k} d_{S_{f_l,f_k}}\,\mathbf{t}_{S_{f_i,f_j}}S_{f_i,f_j},} \qquad (8)$$

имеющее меньшее число ненулевых слагаемых. Поэтому, заменяя слагаемые вида (7) в разложении (5) с помощью формулы (8), получаем новое разложение сизигии s, имеющее меньшее число слагаемых, что противоречит выбору этой сизигии и ее разложения.

Из леммы 8 следует, что $\Sigma_d = \Sigma \setminus \Sigma^*$ – множество всех разложимых критических сизигий, а из алгоритма построения Σ^* вытекает, что Σ^* является базисом пространства сизигий $\mathcal{S}(G)$. При доказательстве леммы 8 был также определен порядок < на множестве критических сизигий.

Лемма 9. Простая редукция относительно множества Σ_d и порядка на множестве критических сизигий преобразует элемент $\sigma \in \Sigma^*$ в $\sigma' \in \Sigma^*$.

представления сизигии, старший член которой совпадает со старшим членом разложимой сизигии S_{f_i,f_j}

Доказательство. Пусть $\boldsymbol{\sigma} \to_{\boldsymbol{\Sigma_d}} \boldsymbol{\sigma'}$. Тогда $\sigma' \in \Sigma$ и выполняется равенство $\sigma' = \sigma - \sigma_1$, где $\sigma_1 \in \Sigma_d$. Пусть $\sigma' \in \Sigma_d$. Тогда $\sigma = \sigma' + \sigma_1 \in \Sigma_d$, что противоречит условию $\sigma \in \Sigma^* = \Sigma \setminus \Sigma_d$. Следовательно, $\sigma' \in \Sigma^*$.

Поэтому определена полная редукция сизигий из множества Σ^* относительно множества разложимых сизигий Σ_d, задающая отображение $\Sigma^* \to \Sigma^*$. Образ этого отображения обозначим через Σ^{**}. Относительно этого множества справедлива

Лемма 10. Множество Σ^{**} является базисом $K[X]$-модуля сизигий $S(F)$, и при выполнении соотношения

$$\sum_{\boldsymbol{\sigma}\in\boldsymbol{\Sigma}^{**}} \mathbf{a}_{\boldsymbol{\sigma}}\boldsymbol{\sigma} = \sum_{\boldsymbol{\sigma}\in\boldsymbol{\Sigma_d}} \boldsymbol{p}_{\boldsymbol{\sigma}}, \quad \textit{где } \boldsymbol{\Sigma}^{**} = \{\boldsymbol{\sigma} \in \boldsymbol{\Sigma}^{**} \mid \mathbf{deg}\,\boldsymbol{\sigma} = \boldsymbol{\omega}\}, \mathbf{a}_{\boldsymbol{\sigma}} \in \mathbf{K}, \qquad (\mathbf{9})$$

всегда

$$\sum_{\boldsymbol{\sigma}\in\boldsymbol{\Sigma_d}} \boldsymbol{p}_{\boldsymbol{\sigma}}\,\boldsymbol{\sigma} = \mathbf{0}.$$

Доказательство. Пусть это не так. Среди разложений вида (9) выберем соотношение, с ненулевой правой частью минимальной длины. Без ограничения общности можно считать, что $\deg(p_\sigma\sigma) = \omega$ и p_σ – мономы. Высотой критической сизигии $\sigma = \alpha e_f + \beta e_g$ будем называть e_f, если $f \succ g$, или e_g в противном случае. Высоту сизигии σ обозначим через $\mathrm{V}(\sigma)$. Положим

$$\boldsymbol{L} = \{\boldsymbol{\sigma} \in \boldsymbol{\Sigma_d} | \boldsymbol{p}_{\boldsymbol{\sigma}} \neq \mathbf{0}\},$$
$$\boldsymbol{e} = \max_{\boldsymbol{\sigma}\in\mathbf{L}} \mathrm{V}(\boldsymbol{\sigma}),$$

и

$$\boldsymbol{L_0} = \{\boldsymbol{\sigma} \in \boldsymbol{L} | \mathrm{V}(\boldsymbol{\sigma}) = \boldsymbol{e}\}.$$

Тогда из определения Σ^{**}следует, что

$$\forall \boldsymbol{\alpha e_f + \beta e_g} \in \boldsymbol{\Sigma}^{**} \forall \boldsymbol{\tau} \in \boldsymbol{L}\ \boldsymbol{e_f} \neq \boldsymbol{e}, \boldsymbol{e_g} \neq \boldsymbol{e}.$$

Поэтому из условия (9) следует, что $\sum_{\sigma\in L_0} p_\sigma \mathrm{HT}\sigma = 0$ и, следовательно, воспользовавшись леммой 7 для разложения

$$\boldsymbol{s} = \sum_{\boldsymbol{\sigma}\in\boldsymbol{L_0}} \boldsymbol{p}_{\boldsymbol{\sigma}}\,\boldsymbol{\sigma},$$

получим разложение

$$\boldsymbol{s} = \sum_{\boldsymbol{\sigma}\in\boldsymbol{L_1}} \boldsymbol{q}_{\boldsymbol{\sigma}}\,\boldsymbol{\sigma}, \text{ где } \boldsymbol{L_1} \subset \boldsymbol{\Sigma_d},$$

меньшей длины. Поэтому имеется соотношение вида (9) с правой частью меньшей длины.

Для любого $L \subset \Sigma^*$ определен максимальный элемент в L. Обозначим этот элемент через $M(L)$.

Лемма 11. Пусть Σ^{***} – результат следующего алгоритма:

Шаг 1. $\boldsymbol{S}^{***} \coloneqq \emptyset, \boldsymbol{T} \coloneqq \boldsymbol{\Sigma}^{**}$.

Шаг 2. Пусть $\boldsymbol{\sigma}$ – нормальная форма (см. определение 1.6) элемента $\boldsymbol{M(T)}$ относительно множества $\boldsymbol{T \setminus \{M(T)\}}$.
Шаг 3. Если $\boldsymbol{\sigma \neq 0}$, то $\boldsymbol{S^{***} := S^{***} \cup \{\sigma\}}$.
Шаг 4. $\boldsymbol{T := T \setminus \{M(T)\}}$.
Шаг 5. Если $\boldsymbol{T \neq \emptyset}$ перейти к шагу 2.
Шаг 6. $\boldsymbol{\Sigma^{***} := S^{***}}$.

Тогда Σ^{***} является минимальным базисом $K[X]$-модуля сизигий $\mathcal{S}(F)$.

Доказательство. Каждый раз на шаге 4 данного алгоритма множество сизигий $S^{***} \cup T$ является базисом $K[X]$-модуля сизигий $\mathcal{S}(F)$. Поэтому достаточно доказать минимальность базиса Σ^{***}.

Пусть $\Sigma^{***} = \{s_1, \dots, s_k\}$. В силу леммы Накаямы достаточно проверить линейную независимость элементов Σ^{***}, т.е. проверить, что любая однородная линейная комбинация элементов из Σ^{***}, неразложима. Пусть это не так. В силу леммы 10 это означает, что имеется нетривиальная линейная комбинация элементов из Σ^{***}, равная нулю

$$\sum_{i=1}^{k} \boldsymbol{\alpha_i s_i = 0, \alpha_i \in K}. \qquad (\mathbf{10})$$

Пусть e – старший терм этого разложения. Тогда этот терм является одновременно старшим термов по крайней мере двух слагаемых разложения (10). Без ограничения общности можно считать, что этими слагаемыми являются $\alpha_1 s_1$ и $\alpha_2 s_2$, что невозможно виду шага 2 алгоритма, поскольку этот старший член должен исчезнуть при приведении к нормальной форме. Следовательно, согласно лемме Накаямы Σ^{***} является минимальным базисом.

Список литературы

[1]. Gebauer R., Moller H.M. On an Instalation of Buchberger's Algorithm. Journal of Symbolic Computation, no. 6, 1987, pp. 257-286..
[2]. Caboara M., Kreuzer M., Robbiano L. Efficiently computing minimal sets of critical pairs, Journal of Symbolic Computation, No. 38, 2004, pp. 1169-1190.
[3]. Ленг С.. Алгебра. Москва, Мир, 1968.
[4]. Агиевич С.В. Усовершенствованый алгоритм Бухбергера. Труды Института математики НАН Беларуси, том 20, no. 1, 2012, стр. 3-13.
[5]. Buchberger B. Grobner Bases: An Algorithmic Method in Polynomial Ideal. In Multidimensional Systems Theory and Applications, 1985, pp. 184-232.

Minimal basis of the syzygies module of leading terms

A.V. Sokurov <shok@ispras.ru>
Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. Systems of polynomial equations are one of the most universal mathematical objects. Almost all the problems of cryptographic analysis can be reduced to finding solutions to systems of polynomial equations. The corresponding direction of research is called algebraic cryptanalysis. In terms of computational complexity, systems of polynomial equations cover the entire range of possible options, from algorithmic insolubility of Diophantine equations to well-known efficient methods for solving linear systems. The method of Buchberger [5] brings a system of algebraic equations to the system of a special type defined by the Gröbner original system of equations, allowing the use of the exception of the dependent variables. The basis for determining the Groebner basis is the permissible ordering on the set of terms. The set of admissible orderings on the set of terms is infinite and even continuum. The most time-consuming step in finding the Groebner basis using the Buchberger algorithm is to prove that all S-polynomials representing a system of generators of K[X]-module S-polynomials. There is a natural problem of finding such a minimal system of generators. The existence of such a system of generators follows from Nakayama's theorem. An algorithm for constructing such a basis for any ordering is proposed.

Keywords: polynomial ring; field; ideal; syzygy; Groebner basis; Buchberger algorithm; admissible order

DOI: 10.15514/ISPRAS-2018-30(6)-16

For citation: Sokurov A.V. Minimal basis of the syzygies module of leading terms. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 293-304 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-16

References

[1]. Gebauer R., Moller H.M. On an Instalation of Buchberger's Algorithm. Journal of Symbolic Computation, no. 6, 1987, pp. 257-286.
[2]. Caboara M., Kreuzer M., Robbiano L. Efficiently computing minimal sets of critical pairs. Journal of Symbolic Computation, no. 38, 2004, pp. 1169-1190.
[3]. Lang S. Algebra. Addison-Wesley Publishing Company Reading, 1965.
[4]. Agievich S. V. Improved Buchberger algorithm. Proceedings of the Institute of Mathematics, National Academy of Sciences of Belarus, vol. 20, issue 1, 2012, pp. 3-13 (in Russian).
[5]. Buchberger B. Grobner Bases: An Algorithmic Method in Polynomial Ideal. In Multidimensional Systems Theory and Applications, 1985, pp. 184-232.

Программирование цифрового линейно-фазового фильтра в архитектуре ARMv8

А.М. Водовозов <am.vodovozov@gmail.com>
Д.С. Полетаев <dimanzaec@yandex.ru>

Вологодский государственный университет,
160000, Россия, г. Вологда, ул. Ленина, д. 15

Аннотация. Рассматривается задача использования процессоров с архитектурой ARMv8 для ускорения работы алгоритмов мультимедиа и цифровой обработки при решении задач восстановления сигналов в процессе фильтрации. В качестве примера рассмотрена реализация алгоритма работы цифрового КИХ-фильтра с линейной фазо-частотной характеристикой. Предложены формулы расчета фильтра. Алгоритм оптимизирован с использованием векторных SIMD-инструкций архитектуры ARMv8. Представлена реализация алгоритма обработки сигнала на языке Си на чипе BCM2837 с процессором ARM Cortex-A53. Решение обеспечило эффективное восстановление частот, искаженных при передаче сигналов в звуковом диапазоне, и доказывает эффективность использования мобильных многоядерных процессоров ARMv8 для параллельной обработки данных в процессе решения сложных вычислительных задач. Результаты эксперимента показывают, что использование процессоров с архитектурой ARMv8 при решении задач фильтрации сигналов позволяет существенно ускорить работу мультимедиа и алгоритмов обработки сигналов, таких как видеокодер/декодер, 2D/3D графика, игры, обработка звука и речи, обработка изображений, телефония и звук

Ключевые слова: цифровая обработка сигналов, линейно-фазовый фильтр, конечная импульсная характеристика, ARMv8, SIMD.

DOI: 10.15514/ISPRAS-2018-30(6)-17

Для цитирования: Водовозов А.М., Полетаев Д.С. Программирование цифрового линейно-фазового фильтра в архитектуре ARMv8-A. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 305-314. DOI: 10.15514/ISPRAS-2018-30(6)-17

1. Введение

Появление 64-битного поколения процессоров с архитектурой ARMv8 и соответствующих инструментов компиляции открыло возможность повышения эффективности прикладных программ за счет использования преимуществ этой платформы. Одним из таких преимуществ является новый набор инструкций ARMv8, позволяющий выполнять несколько 128-битных операций векторизации параллельно. Причем для использованияе этих

операций не требуются каких-либо дополнительные усилия разработчиков, так как за всё отвечают встроенные в ядро аппаратные средства.

Одной из новых областей применения архитектуры ARMv8 можно считать решение задачи восстановления сигналов, искаженных при передаче по линиям связи, сопровождающей построение современных цифровых систем обработки информации и мультимедиа. Для решения такой задачи в структуре создаваемой системы обычно используются цифровые фильтры, которые позволяют восстановить исходный сигнал из принятого на основе данных о частотных характеристиках линии [1,2]. Цифровые фильтры влияют как на амплитуду сигнала, так и на его фазу, внося фазовые искажения.

В задачах управления фазовые искажения, как правило, нежелательны и указанный недостаток может быть исключен за счет использования линейно-фазового цифрового фильтра, который может быть реализован только при условии конечности его импульсной характеристики. В результате, при известных параметрах линий связи, для решения задачи используют фильтры с конечной импульсной характеристикой (КИХ-фильтры). Такие фильтры позволяют получить любую желаемую фазо-частотную характеристику, в том числе и линейную, при которой групповая задержка сигнала является величиной постоянной [3-6].

КИХ-фильтры могут быть реализованы нерекурсивно, т.е. с помощью прямой свертки, и всегда устойчивы. Однако для аппроксимации фильтров, частотные характеристики которых имеют значительный наклон, требуется импульсная характеристика с большим числом отсчетов и, при использовании в процессе реализации фильтра обычной свертки необходимо выполнять большой объем вычислений. Современная микропроцессорная архитектура ARM, активно используемая в задачах управления при создании встроенных приложений, открывает новые возможности при организации вычислений. Так, процессоры с архитектурой ARMv8 имеют мощные блоки SIMD, позволяющие эффективно выполнять параллельные вычисления в режиме реального времени, необходимые для реализации фильтров [7-9].

2. Алгоритм работы КИХ-фильтра

Реализация КИХ-фильтра предусматривает формирование сигнала на выходе путем свертки дискретного входного сигнала с дискретной импульсной характеристикой желаемой системы [10]. Операцию можно описать выражением:

$$y[n] = \sum_{i=0}^{N} h\,[i] \cdot x[n-i], \quad n = 0,\ 1,\ 2,..., \tag{1}$$

где:

- x – последовательность входных отсчетов (сигнал на входе);
- h – импульсная характеристика длины N желаемой системы;
- y – последовательность выходных отсчетов (сигнал на выходе системы).

Таким образом, проектирование КИХ-фильтра сводится к поиску нужной импульсной характеристики. Если известна дискретизированная по частоте комплексная частотная характеристика фильтра, то импульсную характеристику будем искать, применив к заданной частотной характеристике фильтра обратное дискретное преобразование Фурье (ОДПФ) [11]:

$$h[n] = \frac{1}{N}\sum_{k=0}^{N} X[k]\cdot e^{\frac{2\pi\cdot j}{N}\cdot k\cdot n}, \quad n = 0,\ 1,\ 2,\ \ldots,\ N-1, \tag{2}$$

где:

- X – комплексная частотная характеристика системы;
- h – импульсная характеристика системы;
- N – количество отсчетов импульсной характеристики, а также количество точек, взятых на частотной характеристике.

Расчет фильтра начинается с определения желаемой частотной характеристики, которую можно разбить на две составляющие – амплитудную и фазовую. Фазо-частотная характеристика (ФЧХ) задается линейной, а амплитудно-частотная (АЧХ) может быть задана произвольной. На рис. 1 изображен пример фильтра Баттерворта с АЧХ пятого порядка и линейной ФЧХ. Для удобства расчетов частота дискретизации принимается за 1, поэтому частотные характеристики задаются для частот от 0 до 0.5.

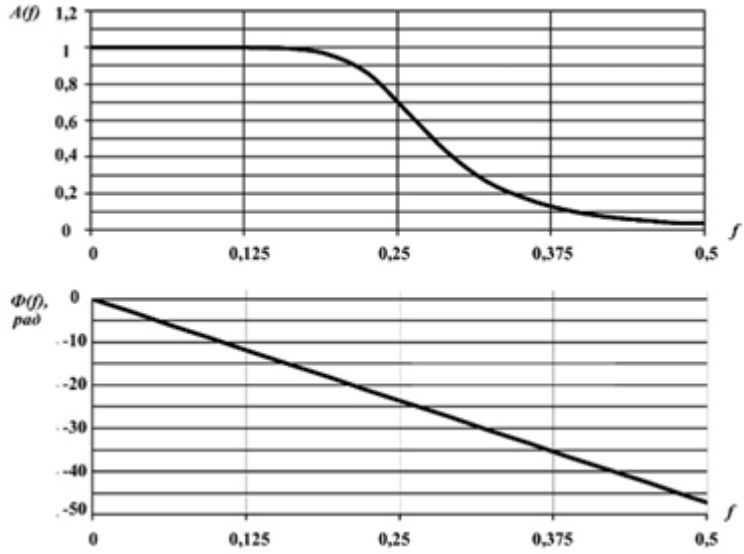


Рис. 1. Пример задания АЧХ и ФЧХ системы.
Fig. 1. Example of setting the frequency response and phase response of the system

Представленные частотные характеристики также можно описать комплексной функцией:

$$X(f) = A(f) \cdot e^{j \cdot \Phi(f)} = A(f) \cdot e^{-j \cdot 2\pi \cdot f \frac{N-1}{2}}, \quad (3)$$

где:

- f – относительная частота;
- A(f) – амплитудно-частотная характеристика системы;
- Ф(f) – фазо-частотная характеристика системы;
- N – количество коэффициентов КИХ-фильтра (на рисунке N=31);
- (N-1)/2 - задержка (число отсчетов), вносимая фильтром длины N.

Импульсная характеристика, получаемая через обратное преобразование Фурье, должна быть чисто вещественной, а это означает, что ее Фурье-образ является симметричным [12]. Поэтому перед дискретизацией и расчетом ОДПФ необходимо обеспечить симметрию АЧХ и ФЧХ (рис. 2).

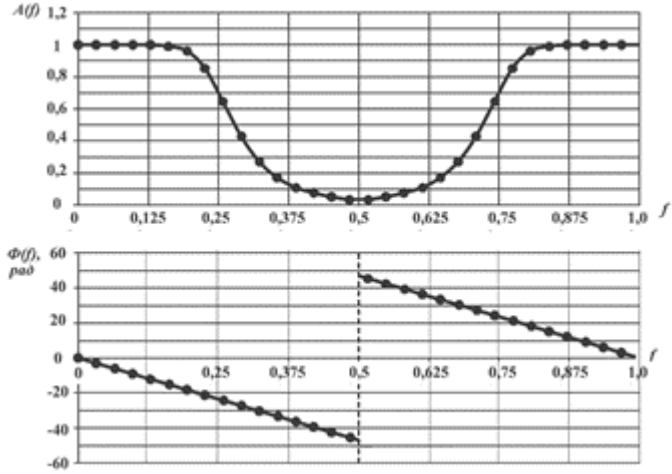


Рис. 2. Дискретизация комплексного коэффициента передачи для ОДПФ
Fig. 2. Discretization of complex transfer coefficient for Inverse Discrete Fourier Transform

Тогда значения комплексного коэффициента передачи будут определяться соотношением:

$$X[k] = \begin{cases} A\left(\frac{k}{N}\right) \cdot e^{-j \cdot 2\pi \cdot \frac{N-1}{2} \cdot \frac{k}{N}}, & \frac{k}{N} < 0.5 \\ A\left(1 - \frac{k}{N}\right) \cdot e^{-j \cdot 2\pi \cdot \frac{N-1}{2} \cdot \frac{N-k}{N}}, & \frac{k}{N} \geq 0.5 \end{cases}, \quad k = 0,\ 1,\ ...,\ N-1, \quad (4)$$

где:

- N – количество коэффициентов КИХ-фильтра, а также количество точек, взятых на частотной характеристике;
- k – индекс частоты;
- A(f) – желаемая амплитудно-частотная характеристика системы;
- X – дискретизированная комплексная частотная характеристика.

В результате применения к последовательности X обратного дискретного преобразования Фурье (2) получается импульсная характеристика системы h[n] (рис. 3).

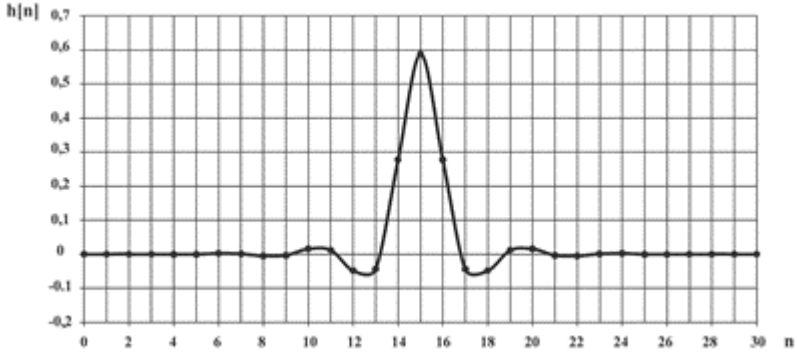


Рис. 3. Коэффициенты КИХ-фильтра, полученные через ОДПФ
Fig. 3. The coefficients of the FIR filter obtained through Inverse Discrete Fourier Transform

Если значения АЧХ системы не превышают 1, то коэффициенты фильтра лежат в диапазоне от -1 до 1. При целочисленной реализации алгоритма коэффициенты приводятся к диапазону целого знакового типа.

3. Программирование КИХ-фильтра

КИХ-фильтры требуют большого числа коэффициентов [2], поэтому для их реализации были использованы векторные SIMD-инструкции архитектуры ARMv8 [13]. Архитектура имеет поддержку набора команд SIMD с плавающей запятой, которые могут применять одну и ту же операцию к нескольким упакованным элементам одного типа и размера параллельно, что в несколько раз сокращает время выполнения. Программа реализована на языке Си с использованием ассемблерных команд архитектуры ARM.

В приведенном ниже листинге показан код функции, реализующей вычисление очередного отсчета выходной последовательности в соответствии с формулой (1).

```
//Реализация для ARMv8 с использованием SIMD инструкций
int16_t fir_compute_sample(int16_t* samples_buf_bottom,
                           int16_t* coeffs_arr, unsigned count)
{volatile int16_t result;
```

```
asm volatile(
"eor V1.16B, V1.16B, V1.16B\n\t" //очищается аккумулятор
//в основном цикле перемножаются параллельно 8 пар значений
"FIR_MAIN_CYCLE:\n\t"
"cmp %[pS]j %[smp_end_mult8]\n\t"
"bhs FIR_MAIN_CYCLE_END\n\t"
"ldl { V7.8H }, [%[pC]], #16\n\t”
//загружается 8 16-битных коэффициентов в V7.8H
"ldl { V5.8H [*[PS]], #16\n\t"
//загружается 8 16-битных отсчетов в V5.8H
"sqrdmulh V3.8H, V5.8H, V7.8H\n\t"
//перемножается 8 отсчетов на 8 коэффициентов
"add VI.8H, VI.8H, V3.8H\n\t"
//результаты накапливаются в 8 аккумуляторов
"b FIR_MAIN_CYCLE\n\t"
"FIR MAIN CYCLE END:\n\t"
""
"saddlv SI, V1.8H\n\t"
//складываются все 8 аккумуляторов в V1.S[0]
"sqxtn VI.4H, V1.4S\n\t"
//и результат преобразуется в 16 бит с контролем переполнения
//в доп. цикле перемножается все, что осталось,
//каждый раз только одна пара чисел
"FIR_ADD_CYCLE:\n\t"
"cmp %[pS], %[smp_end]\n\t"
"beq FIR_END\n\t"
"ldl { V7.H >[0], [%[pC]], #2\n\t"
//загружается коэффициент в V7.H[0]
"ldl { V5.H >[0], [%[pS]], #2\n\t"
//загружается отсчет в V5.H[0]
"sqrdmulh V3.4H, V5.4H, V7.4H\n\t"
//перемножаем отсчет с коэффициентом
"sqadd VI.4H, VI.4H, V3.4H\n\t"
//результат добавляется к аккумулятору
"b FIR_ADD_CYCLE\n\t"
"FIR_END:\n\t"
""
"stl { VI.H }[0], [%[result]]\n\t"
//аккумулятор выгружается в память
:
: [pC] "r" (coeffs_arr),
[pS] "r" (samples_buf_bottom),
[smp_end] "r" (samples_buf_bottom + count),
```

```
[smp_end_mult8] "r" (samples_buf_bottom + (count - count % 8)),
[result] "r" (&result)
);
return result;
}
```

Входные отсчеты реализуемой функции должны быть расположены последовательно, начиная с адреса «samples_buf_bottom». При обработке сигнала в реальном времени отсчеты могут поступать в циклический буфер. Вызов представленной функции каждый раз при поступлении нового отсчета позволит получить очередной выходной отсчет фильтра.

4. Заключение

Результаты экспериментов показывают, что реализация КИХ-фильтра в архитектуре ARMv8 с использованием SIMD-ускорения увеличивает скорость обработки сигнала примерно в 4 раза по сравнению с исходным кодом, позволяя в системах автоматического управления эффективно решать задачу восстановления сигналов датчиков. Например, на вычисление выходного отсчета КИХ-фильтра с длиной N = 1000 на чипе BCM2837 с процессором ARM Cortex-A53 требуется менее 1 мкс. Это означает, что, к примеру, при обработке в реальном времени сигнала звукового диапазона с частотой дискретизации 48 кГц и периодом в 21 мкс фильтр может состоять, приблизительно, из 15 – 20 тысяч коэффициентов, что совершенно достаточно для коррекции АЧХ во всем диапазоне заявленных частот. С ростом числа коэффициентов повышается допустимая крутизна АЧХ фильтра и расширяется его нижняя граница по полосе пропускания.

Эксперимент доказывает перспективность использования мобильных многоядерных процессоров ARMv8 для параллельной обработки данных и решения сложных вычислительных задач мультимедиа и обработки сигналов, таких как видеокодер/декодер, 2D/3D графика, игры, обработка звука и речи, обработка изображений, телефония и звук.

Список литературы

[1]. Лайонс Р. Цифровая обработка сигналов. М., Бином, 2006, 656 стр.
[2]. Dwivedi A.K., Ghosh S., Londhe N.D. Review and Analysis of Evolutionary Optimization-Based Techniques for FIR Filter Design. Circuits, Systems, and Signal Processing, vol. 37, no. 10, 2018, pp. 4409-4430.
[3]. Paquelet S., Savaux V. On the symmetry of FIR filter with linear phase. Digital Signal Processing: A Review Journal, vol. 81, 2018, pp. 57-60.
[4]. Строгонов А. КИХ-фильтры на параллельной распределенной арифметике. Компоненты и технологии, № 5, 2013, стр. 84-88.
[5]. Бычков Д.Б., Дождев С.Ю. Аспекты оценки эффективности процессорных архитектур. Электронная техника. Серия 3: Микроэлектроника, № 2, 2015, стр. 34-37.

[6]. Оппенгейм А., Шафер Р. Цифровая обработка сигналов. М., Техносфера, 2006, 856 стр.
[7]. Ибрагимов Т.Р., Мунерман В.И. Возможность использования процессоров ARMV8 для параллельных вычислений. Системы компьютерной математики и их приложения, № 19, 2018, стр. 152-157.
[8]. Belloch J.A., Alventosa F.J. Alonso P., Quintana-Ortí E.S., Vidal A.M. Accelerating multi-channel filtering of audio signal on ARM processors. Journal of Supercomputing, vol. 73, issue 1, 2017, pp. 203-214.
[9]. Водовозов А. М. Микроконтроллеры для систем автоматики. М., Инфра-Инженерия, 2016. 164 стр.
[10]. Winser A. Digital Signal Processing: Principles, Algorithms and System Design London, Academic Press, 2017, 617 p.
[11]. Рабинер, Л., Б. Гоулд Б. Теория и применение цифровой обработки сигналов. М., Мир, 1978, 835 стр.
[12]. Winser A., Cranos W. Digital Signal Processing: Principles, Algorithms and System Design. London, Academic Press, 2017, 617 p.
[13]. ARMv8 Instruction Set Overview. Доступно по ссылке: https://class.ee.washington.edu/469/peckol/doc/ARM/ARM_v8_Instruction_Set_Architecture_(Overview).pdf, (дата обращения: 18.04.18).

Programming of digital linear phase filter in ARMv8 architecture

A.M. Vodovozov < am.vodovozov@gmail.com >
D.S. Poletaev < dimanzaec@yandex.ru >

Vologda State University,
15, Lenin st., Vologda, 160000, Russia

Abstract. We consider the problem of using processors with an ARMv8 architecture to speed up the operation of multimedia algorithms and digital processing when solving problems of signal recovery in the filtering process. As an example, the implementation of the algorithm of a digital FIR filter with a linear phase-frequency characteristic is considered. The proposed formula for calculating the filter. The algorithm is optimized using vector SIMD instructions of the ARMv8 architecture. An implementation of the C signal processing algorithm on a BCM2837 chip with an ARM Cortex-A53 processor is presented. The solution provided effective recovery of frequencies distorted when transmitting signals in the audio range and proves the efficiency of using mobile multicore ARMv8 processors for parallel data processing in solving complex computational problems. Experimental results prove that the use of ARMv8 architecture processors when solving signal filtering problems significantly speeds up multimedia and signal processing algorithms such as video encoder / decoder, 2D / 3D graphics, games, sound and speech processing, image processing, telephony and sound.

Keywords: digital signal processing; linear phase filter; finite impulse response; FIR; ARMv8; SIMD

DOI: 10.15514/ISPRAS-2018-30(6)-17

For citation: Vodovozov A.M., Poletaev D.S. Programming of digital linear phase filter in ARMv8 architecture. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 305-314 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-17

References

[1]. Richard G. Lyons. Understanding Digital Signal Processing (2nd Edition). Prentice Hall, 2004, 665 p.

[2]. Dwivedi A.K., Ghosh S., Londhe N.D. Review and Analysis of Evolutionary Optimization-Based Techniques for FIR Filter Design. Circuits, Systems, and Signal Processing, vol. 37, no. 10, 2018, pp. 4409-4430.

[3]. Paquelet S., Savaux V. On the symmetry of FIR filter with linear phase. Digital Signal Processing: A Review Journal, vol. 81, 2018, pp. 57-60.

[4]. Strogonov A. FIR filters on parallel distributed arithmetic. Komponenty i tekhnologii [Components and technologies], 2013, no. 5, pp. 84-88 (in Russian).

[5]. Bychkov D.B., Dozhdev S.Yu. Aspects of evaluating the effectiveness of processor architectures. Elektronnaya tekhnika. Seriya 3: Mikroelektronika. [Electronic equipment. Series 3: Microelectronics], 2015, no. 2, pp. 34-37 (in Russian).

[6]. Oppenheim Alan V., Schafer Roland W., Buck John R. Discrete-Time Signal Processing, 2nd ed., Prentice-Hall, 1999, 870 p.

[7]. Ibragimov T.R., Munerman V.I. Ability to use ARMV8 processors for parallel computing. Sistemy komp'yuternoy matematiki i ikh prilozheniya [Computer mathematics systems and their applications], no. 19, 2018, pp. 152-157 (in Russian).

[8]. Belloch J.A., Alventosa F.J. Alonso P., Quintana-Ortí E.S., Vidal A.M. Accelerating multi-channel filtering of audio signal on ARM processors. Journal of Supercomputing, vol. 73, issue 1, 2017, pp. 203-214.

[9]. Vodovozov A.M. Microcontrollers for automation systems. M., Infra-Engineering, 2016, 164 p. (in Russian).

[10]. Winser A. Digital Signal Processing: Principles, Algorithms and System Design London, Academic Press, 2017, 617 p.

[11]. Рабинер, Л., Б. Гоулд Б. Теория и применение цифровой обработки сигналов. М., Мир, 1978, 835 стр.

[12]. Winser A., Cranos W. Digital Signal Processing: Principles, Algorithms and System Design. London, Academic Press, 2017, 617 p.

[13]. ARMv8 Instruction Set Overview. URL: https://class.ee.washington.edu/469/peckol/doc/ARM/ARM_v8_Instruction_Set_Architecture_(Overview).pdf, (accessed: 18.12.18).

Тестирование различных методов моделирования внутренних течений несжимаемой жидкости

В.Г. Мельникова <vg-melnikova@yandex.ru>
МГТУ им. Н.Э. Баумана,
105005, Россия, г. Москва, ул. 2-ая Бауманская, д. 5, стр. 1

Аннотация. Математическое моделирование течения в гидравлических элементах относится к отдельному классу задач внутреннего течения несжимаемой жидкости и имеет большую практическую значимость, в том числе при проектировании новых гидроагрегатов. Целью данной работы является обзор, тестирование и сравнение различных возможных численных методов расчета: метода контрольного объема, метода частиц в контрольных объемах и метода решёточных уравнений Больцмана на простых примерах внутреннего течения жидкости. В качестве тестовых задач были выбраны трубы круглого сечения различной формы: постоянного сечения, с внезапным расширением сечения, с внезапным сужением сечения и колено. Проводилось сравнение полей скоростей и давлений, точности решения и времени, затраченного на вычисления различными методами.

Ключевые слова: PFVM; LBM; FVM; OpenFOAM; внутреннее течение; вычислительная гидродинамика; численное моделирование

DOI: 10.15514/ISPRAS-2018-30(6)-18

Для цитирования: Мельникова В.Г. Тестирование возможностей различных методов расчета для моделирования внутренних течений жидкости. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 315-328. DOI: 10.15514/ISPRAS-2018-30(6)-18

1. Введение

При создании новых гидроагрегатов важную роль играет исследование характеристик течения с применением математического моделирования. Оно позволяет свести к минимуму число многочисленных дорогостоящих испытаний изделия и значительно сократить время его разработки. Течение в трубопроводах, клапанах, регуляторах и других гидравлических элементах относится к типу задач внутреннего течения несжимаемой жидкости. Для исследования таких течений обычно используются модели на основе уравнений Навье-Стокса аппроксимируемые с помощью метода контрольного объема (МКО) [1, 2, 3] или метода конечного элемента [4, 5]. Однако эти методы расчета зачастую неэффективны, сложны в настройке и плохо себя показывают при расчетах с подвижными сетками. Поэтому в настоящее время существует

необходимость поиска альтернативных методов решения задач подобного класса. Одним из важных требований к новым методам, помимо приемлемой точности, является требование высокой вычислительной производительности.

Целью данной работы является обзор, тестирование на примере простейших задач внутреннего течения жидкости и сравнение с методом контрольного объема других численных методов: малораспространенного метода частиц в контрольных объемах (PFVM – particle finite volume method) и метода решёточных уравнений Больцмана (LBM - Lattice Boltzmann method), который уже использовался для расчета клапанов [6, 7].

Для расчета методом частиц в контрольных объемах использовался решатель, реализованный на основе пакета с открытым исходным кодом OpenFOAM, что достаточно удобно, так как в пакете имеются все необходимые средства для автоматической генерации сетки, решения уравнений, подключения моделей турбулентности, параллельных вычислений, визуализации и анализа результатов.

Для расчетов методом решёточных уравнений Больцмана использовался программный комплекс XFlow, который обладает достаточно быстрой скоростью счета, возможностью моделирования нестационарных задач с подвижными граничными условиями, алгоритмами распараллеливания расчета, удобными средствами пре- и пост- процессинга.

В качестве тестовых задач моделировалось течение в круглых трубах различной формы: постоянного сечения, с внезапным расширением сечения, с внезапным сужением сечения и колено.

2. Описание метода частиц в контрольных объемах

Метод частиц в контрольных объемах относится к комбинированным лагранжево-эйлеровым методам, он сочетает в себе лучшие свойства метода контрольного объема и бессеточных методов с частицами. Метод является продолжением метода частиц в конечных элементах (PFEM и PFEM-2) [8].

Для описания движения несжимаемой, изотермической, вязкой жидкости необходимы следующие уравнения:

уравнение неразрывности:

$$\nabla \cdot \bar{u} = 0 \qquad (1)$$

уравнение Навье-Стокса:

$$\frac{\partial \bar{u}}{\partial t} + (\bar{u} \cdot \nabla)\bar{u} = -\frac{1}{\rho}\nabla p + \nu \Delta \bar{u} + \bar{f} \qquad (2)$$

где $\bar{u}$ – поле скоростей среды, t – время, ρ – плотность среды, p – поле давлений, ν – кинематическая вязкость среды, $\bar{f}$ – поле массовых сил.

Исследуемая область с жидкостью разбивается на конечно-объемную сетку с частицами, которые движутся совместно со сплошной средой из одной ячейки сетки в другую. Частицы служат для вычисления конвективного слагаемого в уравнениях Навье-Стокса (2), с помощью них определяются параметры самой жидкости (например, масса, энергия, скорость). Их перемещение со средой

рассчитывается явным методом с малым шагом по времени (число Куранта CFL ≈ 0.1). В то же время на эйлеровой сетке путем решения соответствующих линейных уравнений с большим числом Куранта CFL ≈ 1-10 определяются параметры поля (например, давление, плотность, температура).

Рис. 1. Схемы расчета методом частиц в контрольных объемах: верхняя – решатель particlePimpleFoam, нижняя – решатель particlePimpleFoamOld
u – поле скорости в центрах конечных объемов (КО), p – поле давления в центрах КО; x_p – положение частиц, u_p – скорости в частицах; u_{proj} – спроецированное поле скоростей в центрах КО

Fig. 1. PFVM simulation algorithms: top –particlePimpleFoam solver, bottom – particlePimpleFoamOld solver.
u – velocity field in the center of control volumes (CV), p – pressure field in the center of CV; x_p – particle location, u_p – particle velocity; u_{proj} – projected velocity field in the center of CV

Для связи фиксированной конечно-объемной сетки и облака частиц на каждом временном шаге необходимо использовать специальные операторы для проецирования и интерполяции характеристик, в частности поля скоростей, с частиц на сетку и обратно. Существуют две модификации данного метода

particlePimpleFoam и particlePimpleFoamOld, по-разному учитывающие в уравнениях спроецированное поле скорости (рис. 1).

В теории применение частиц для вычисления конвекции увеличивает временной шаг интегрирования уравнений гидродинамики, что позволяет снизить общее время расчета. Еще одним плюсом PFVM является то, что конвекция становится независимой от качества сетки, позволяя избежать типичных ошибок, например, при перекосе ячеек, встречающихся при расчете стандартным МКО.

3. Описание метода решёточных уравнений Больцмана

Метод решёточных уравнений Больцмана [9, 10] представляет собой лагранжев метод для численного расчета течений жидкостей, газов и плазмы. Он является одной из альтернатив методам, базирующимся на дискретизации уравнений Навье-Стокса. Метод хорошо зарекомендовал себя при моделировании многофазных течений, а также при расчете обтекания пористых тел. Суть LBM метода состоит в том, что жидкость рассматривается как совокупность небольшого числа частиц, находящихся внутри равномерных декартовых ячеек (решётки). В каждой точке пространства со временем плотность вероятности нахождения частиц жидкости в определенной точке фазового пространства и плотность вероятности распределения частиц по скоростям меняются согласно кинетическому вероятностному уравнению Больцмана [11]:

$$\frac{df}{dt} + \bar{v}\,\nabla f + \frac{\bar{F}}{m}\cdot\nabla_v f = -\frac{f-f^{eq}}{\tau} \qquad (3)$$

где $f = f(\bar{r}, \bar{v}, t)$ – функция распределения плотности вероятности частиц по координатам и скоростям в каждый момент времени,

t – время,

$\bar{v}$ – вектор скорости частицы,

$\bar{F}(\bar{r}, t)$ – поле сил, действующее на частицы в жидкости или газе,

m – масса частиц,

$-\frac{f-f^{eq}}{\tau}$ – оператор столкновения в виде модели приближения Батнагара-Гросса-Крука, которое представляет собой линейную релаксацию к локальному равновесию,

$f^{eq} = \frac{\rho}{(2\pi RT)^{3/2}} * \exp(-\frac{(\bar{v}-\bar{u})^2}{2RT})$ – равновесная функция распределения,

τ – время релаксации.

Основные макроскопические характеристики течения: плотность $\rho(\bar{r}, t)$ и скорость $\bar{u}(\bar{r}, t)$ среды являются моментами функции распределения и находятся посредством интегрирования по всем возможным скоростям $\bar{v}$:

$$\rho = \int f d\bar{v}, \qquad \rho\bar{v} = \int f\bar{v} d\bar{v} \qquad (4)$$

Дискретизация уравнения (3) происходит в два этапа: на первом осуществляется дискретизация в пространстве скоростей, а на втором:

дискретизация по времени и пространственным переменным. Так как LBM является явным методом и содержит только простейшие арифметические операции, то при его распараллеливании не возникает трудностей.

Метод решёточных уравнений Больцмана достаточно универсален и удобен благодаря его простоте, но его применение ограничено малыми скоростями потока (число Маха < 1).

4. Тестовые задачи

При сравнении численных методов моделировалось трехмерное течение в трубах круглого сечения четырех различных форм. Эти виды конструкции являются типовыми узлами, которые присутствуют в большинстве гидравлических систем. Рассматривалась нестационарная постановка задачи на интервале времени от 0 до 10 с. Так как течение во всех задачах достигало установившегося режима, результаты не осреднялись по времени и представлены для момента времени 10 с. В качестве граничных условий на концах труб был задан перепад относительного кинематического давления в 1 $м^2/с^2$. Начальные поля давления и скорости внутри расчетной области равны 0. Коэффициент кинематической вязкости жидкости равен 0.01 $м^2/с$. При таких условиях безразмерный критерий Рейнольдса в зависимости от формы трубы лежит в пределах от 4 до 20, что соответствует ламинарной области течения. На практике такое течение встречается в маслосистемах и гидропередачах, при движении по трубам вязких жидкостей, например, смазочных масел, глицериновых смесей и др.

Задачи были решены двумя модификациями метода частиц в контрольных объемах (решатели particlePimpleFoam и particlePimpleFoamOld), методом решёточных уравнений Больцмана и методом контрольного объема в стандартном решателе OpenFOAM (решатель pimpleFoam). Шаг по времени выбирался автоматически в соответствии со значением максимального числа Куранта 1, 2, 5 или 10. Для метода LBM было выбрано только одно значение максимального числа Куранта - 1, так как данный метод не обеспечивает численную устойчивость при высоких числах Куранта.

В методе PFVM распределение частиц в обеих модификациях задавалось таким образом, чтобы в начальный момент времени в каждой ячейке было по 3 частицы, что является минимальным числом для осуществления корректного переноса скорости с частиц на ячейки и обратно. Движение частиц по линиям тока проводилось методом Рунге-Кутты 2 порядка. Принадлежащие частицам значения скорости проецировались в центр конечно-объемной ячейки, где находится частица, и в центры ближайших ячеек-соседей с определенными весовыми коэффициентами. Тип проецирования – quinticRBF (quintic Radial Basis Function) – значения проецируемых параметров пропорциональны расстоянию между частицей и центром ячейки в 5 степени [12]. Для вычисления скорости на частицах в новый момент времени проводилась интерполяция разницы скорости между

текущим и предыдущим шагами по времени с центра ячейки, которой принадлежит частица. Интерполируемое значение пропорционально расстоянию от частицы до центра ячейки.

Для решения системы уравнений для давления применялся PCG метод (метод сопряженных градиентов с предобуславливанием) с предобуславлевателем DIC (метод неполного разложения Холецкого). Система уравнений для прогноза скоростей решалась методом smoothSolver. Для сглаживания использовался метод Гаусса-Зейделя. Описание всех используемых методов представлено в [13].

При решении всеми методами во всех задачах характерный размер ячейки сетки составил 0.01 м. Такое разрешение расчетной области выбрано исходя из условий баланса точности результатов расчета и времени счета: для всех задач сгущение сетки в два раза не дает изменения решения по скорости и давлению более чем на 2%, при увеличении времени моделирования, в некоторых задачах, в пять раз. Расчеты проводились на 1 ядре Intel(R) Core™ i7-4712HQ 2.30GHz.

4.1. Труба постоянного сечения

Рассмотрим задачу течения жидкости в прямой трубе длиной 1 м, диаметром 0.2 м. Профиль скорости вдоль поперечного сечения известен - закон Пуазейля [14]. Такое течение может наблюдаться только в трубах со сравнительно малым диаметром и при небольших скоростях. Скорость жидкости у стенок трубы равна нулю и, плавно увеличиваясь, достигает максимума на оси трубы, график изменения скоростей по поперечному сечению – парабола. Расчет всеми методами показал, что в решении проявляется незначительная пульсация модуля скорости потока на оси трубы в продольном направлении, а также изменение течения и завышение скорости потока в краевых зонах, как показано на рис.2.

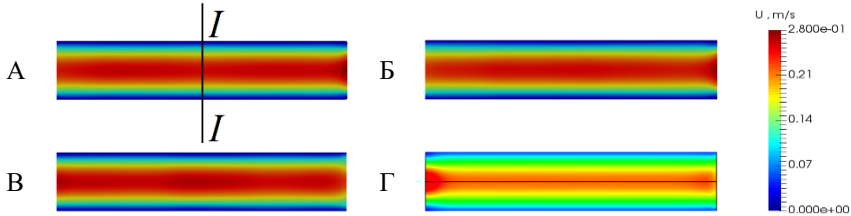


Рис. 2. Поле скоростей в трубе при решении задачи различными методами
Fig. 2. Velocity field in the pipe for different solving methods
А – pimpleFoam (CFL=2), Б – particlePimpleFoam (CFL=2),
В – particlePimpleFoamOld (CFL=2), Г – Lattice Boltzman method (CFL=1)

Из рис. 3 видно, что хорошая сходимость решения задачи обеспечивается при использовании МКО, а PFVM методами сходимость обеспечивается только при низких числах Куранта (CFL ≈ 1-2). Наибольшее отклонения от аналитического решения составляет около 20 % для метода PFVM и CFL = 5.

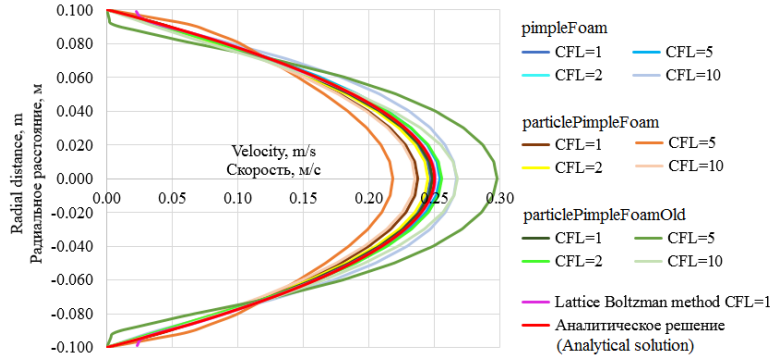


Рис. 3. Распределение скорости жидкости по сечению трубы I-I
Fig.3. The fluid velocity distribution over the I-I pipe section

По продолжительности расчёта наиболее быстрыми оказались решатели, использующий метод контрольного объема и LBM метод, за ними идет решатель particlePimpleFoamOld и на последнем месте решатель particlePimpleFoam (рис. 4).

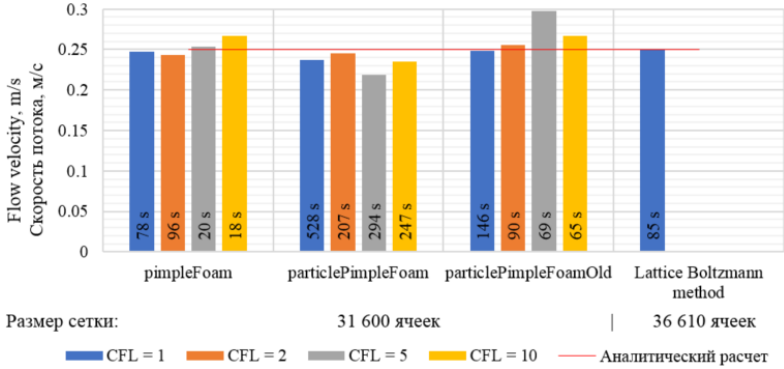


Рис. 4. Скорость течения в центре (на оси) среднего сечения трубы I-I. На столбцах указано время расчета
Fig. 4. Velocity in the center (on the axis) of the pipe middle section I-I. The simulation time is noted on the columns

4.2. Прямая труба с внезапным расширением сечения

В качестве второй тестовой задачи было выбрано течение в трубе с внезапно расширяющимся сечением. Длина трубы - 1 м, диаметры входа– 0.2 м и выхода 0.4 м. В трубе такой формы скорость течения по длине трубы уменьшается, поток срывается с уступа и расширяется не резко, как сечение, а постепенно (рис. 5), при этом в кольцевом зазоре между основным потоком и стенкой появляются вихревые структуры.

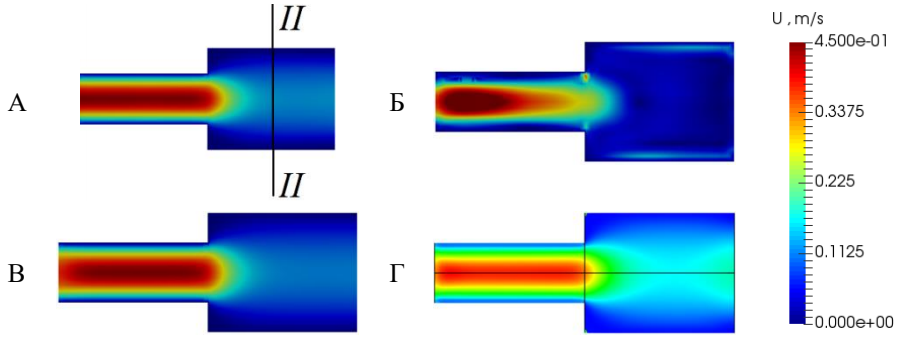


Рис. 5. Поле скоростей в трубе при решении задачи различными методами
Fig. 5. Velocity field in the pipe for different solving methods
А – pimpleFoam (CFL=10), Б – particlePimpleFoam (CFL=10),
В – particlePimpleFoamOld (CFL=10), Г – Lattice Boltzman method (CFL=1)

Расхождение по скорости менее 2% при всех числах Куранта показали решатели pimpleFoam, particlePimpleFoamOld и LBM метод, решатель particlePimpleFoam не справился с расчетом – расхождение по скорости составило более 20% (рис. 6, 7).

Время расчета методом контрольного объема и LBM методом для всех чисел Куранта оказалось значительно меньше, чем расчет методом частиц в контрольном объеме (рис.7).

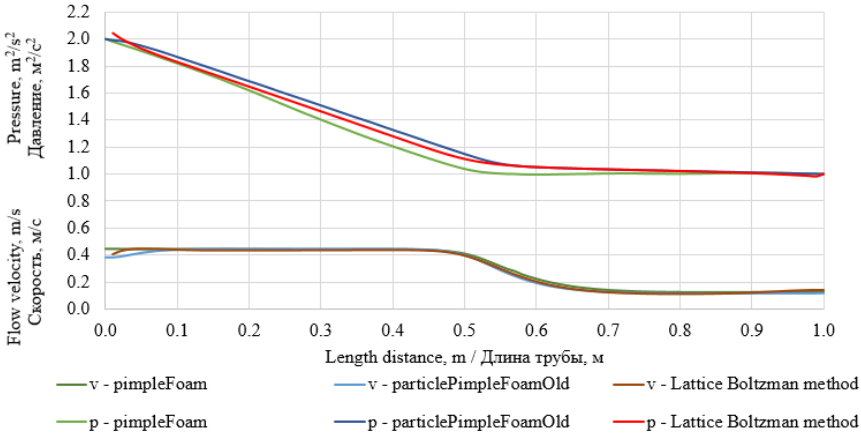


Рис. 6. Распределение давления и скорости жидкости по длине трубы при решении задачи различными методами (МКО и PFVM CFL=10, LBM CFL=1)
Fig. 6. The fluid velocity and pressure distribution over the pipe length for different solving methods (FVM и PFVM CFL=10, LBM CFL=1)

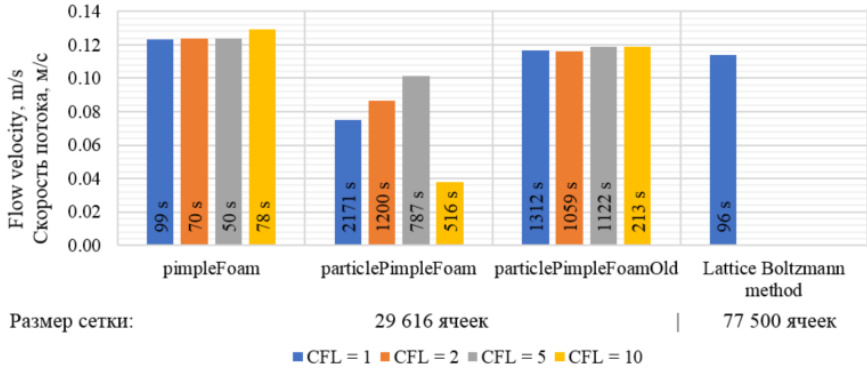


Рис. 7. Скорость течения в центре среднего сечения II-II широкой трубы. На столбцах указано время расчета
Fig. 7. Velocity in the center of the big pipe middle section II-II. The simulation time is noted on the columns

4.3. Прямая труба с внезапным сужением сечения

Рассмотрим течение в трубе с внезапно сужающимся сечением. Длина трубы – 1 м, диаметры входа – 0.2 м и выхода 0.4 м. При резком уменьшении диаметра, скорость течения по длине трубы возрастает, поток срывается с входного угла и сужается (рис.8). В кольцевом зазоре между сужающимся потоком и стенками канала образуется вихревое течение.

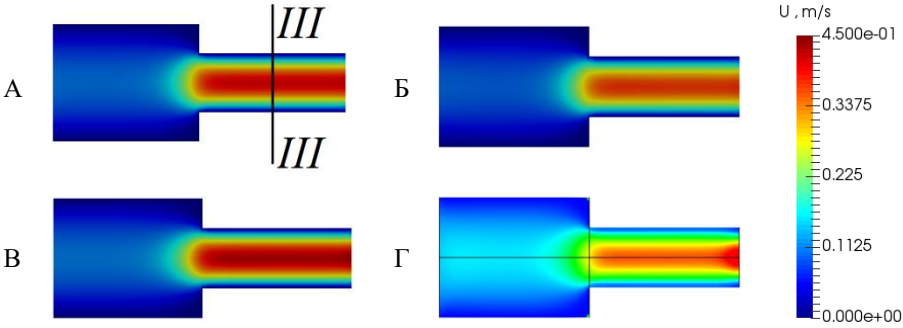


Рис. 8. Поле скоростей в трубе при решении задачи различными методами
Fig. 8. Velocity field in the pipe for different solving methods
А – pimpleFoam (CFL=5), Б – particlePimpleFoam (CFL=5),
В – particlePimpleFoamOld (CFL=5), Г – Lattice Boltzman method (CFL=1)

При числах Куранта от 1 до 5 решатели pimpleFoam и particlePimpleFoamOld дают достоверные результаты. Быстрее всего данная задача решается методом контрольного объема (рис. 9).

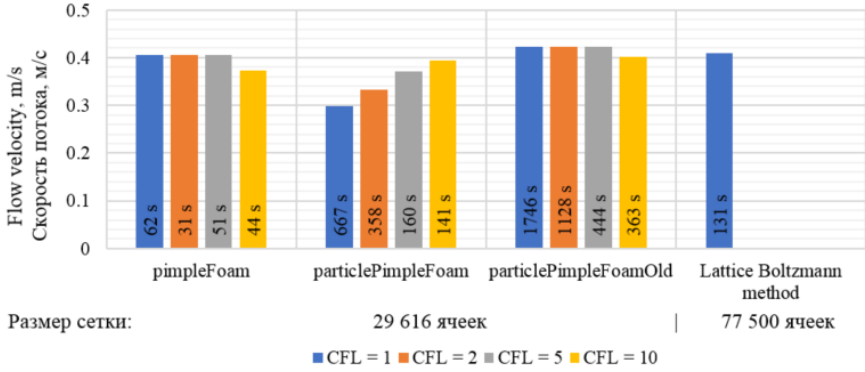


Рис. 9. Скорость течения в центре среднего сечения III-III узкой трубы. На столбцах указано время расчета
Fig. 9. Velocity in the center of the small pipe middle section III-III. The simulation time is noted on the columns

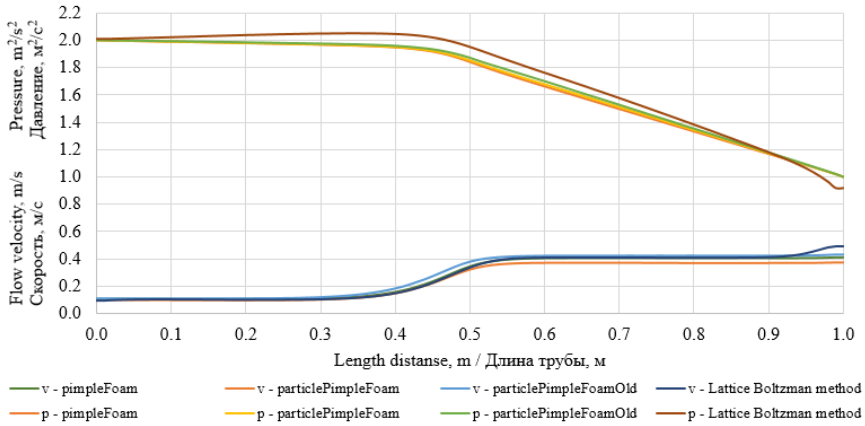


Рис. 10. Распределение давления и скорости жидкости по длине трубы при решении задачи различными методами (МКО и PFVM CFL=5, LBM CFL=1)
Fig. 10. The fluid velocity and pressure distribution over the pipe length for different solving methods (FVM и PFVM CFL=5, LBM CFL=1)

4.4. Колено

В качестве последнего тестового примера было выбрано течение в трубе с поворотом на 90 градусов. Длина прямых участков – 0.5 м, радиус сгиба – 0.2 м, диаметр трубы – 0.2 м. Из рис. 11 видно, что все методы не дают точного решения при больших значения CFL. Для CFL = 1 наиболее быстрым оказался решатель, использующий метод контрольного объема.

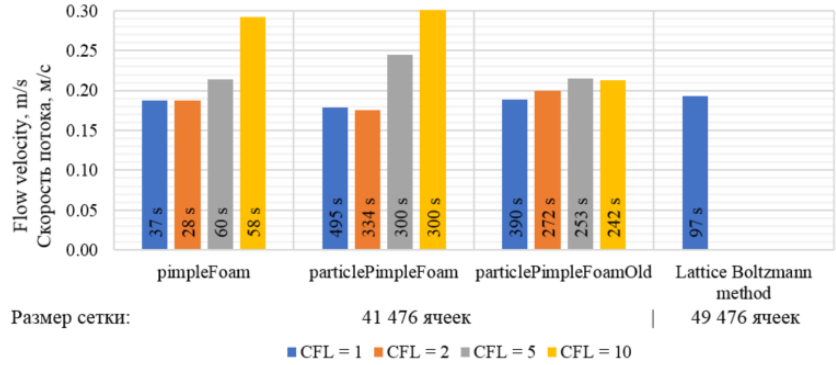


Рис. 11. Максимальная скорость течения в сечении трубы IV-IV (рис. 12). На столбцах указано время расчета
Fig. 11. Maximum velocity in the elbow-pipe section IV-IV (fig. 12). The simulation time is noted on the columns

Так как исследуемое колено закругленное, т.е. поворот потока происходит постепенно на данных скоростях, то вихреобразование в таком течении отсутствует (рис. 12).

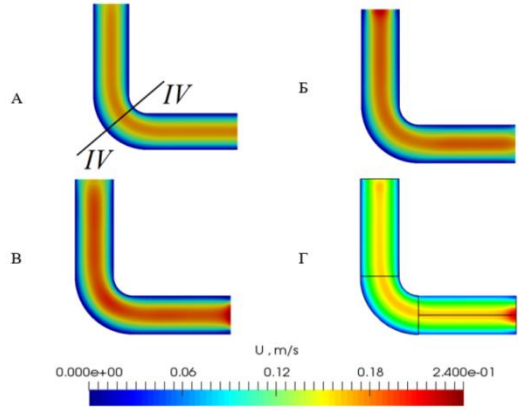


Рис. 12. Поле скоростей в трубе при решении задачи различными методами
Fig. 12. Velocity field in the pipe for different solving methods
А – pimpleFoam (CFL=2), Б – particlePimpleFoam (CFL=2),
В – particlePimpleFoamOld (CFL=2), Г – Lattice Boltzman method (CFL=1

4. Заключение

На примере четырех различных тестовых задач рассмотрены два решателя на базе метода частиц в контрольных объемах, реализованные в свободном пакете OpenFOAM: particlePimpleFoam и particlePimpleFoamOld и метод

решёточных уравнений Больцмана, реализованный в пакете XFlow в сравнении с базовым конечно-объемным решателем OpenFOAM - pimpleFoam.

Проделанная работа позволяет сделать вывод о том, что метод частиц в контрольных объемах на данный момент предназначен для ограниченного класса задач, и не дает выигрыша ни по времени расчета, ни по точности решения для задач внутренней гидродинамики по сравнению с методом контрольного объема. Метод решёточных уравнений Больцмана, в свою очередь, по вычислительной точности сопоставим с методом контрольных объемов, но проведенное тестирование реализации этого метода в пакете XFlow не показало выгоду по скорости расчета при вычислениях на 1 ядре. Также остаётся открытым вопрос эффективности использования метода решёточных уравнений Больцмана для задач со сложной геометрией и подвижными стенками, расчеты которых трудоемки для конечно-объемных методов.

Благодарности

Работа поддержана грантом РФФИ (проект № 17-08-01468 А).

Список литературы

[1]. Castilla R., Alemany I., Algar A., Gamez-Montero P.J., Roquet P., Codina E. Pressure-Drop Coefficients for Cushioning System of Hydraulic Cylinder With Grooved Piston: A Computational Fluid Dynamic Simulation. Energies, vol. 10, no. 11, 2016.

[2]. Fries C., Manhartsgruber B. A moving piston boundary condition including gap flow in OpenFOAM. Wseas Transactions on Fluid Mechanics, vol. 10, 2015, pp.95-104.

[3]. Мельникова В.Г., Коцур О.С., Щеглов Г.А. Особенности построения расчетной схемы для моделирования динамики стабилизатора расхода в пакете OpenFOAM. Труды ИСП РАН, том 29, вып. 1, 2017 г., стр. 53–70. DOI:10.15514/ISPRAS-2017-29(1)-4.

[4]. Степанченко Т.Е. Моделирование процессов движения жидкости в трубопроводе в пакете COMSOL 3.5 Multiphysics. Труды VIII Всероссийской научно-практическая конференция «Технологии Microsoft в теории и практике программирования», 2012, стр. 62-65.

[5]. Гобыш А.В. Моделирование внутренних течений вязкой несжимаемой жидкости методом конечных элементов с использованием противопотоковых схем. Автореф. дис. канд. ф-м. наук, Ин-т вычисл. технологий СО РАН, Новосибирск, 2007. 17 стр.

[6]. Mutabaruka P., Kamrin Ken A simulation technique for slurries interacting with moving parts and deformable solids with applications. Computational Particle Mechanics, vol. 5, issue 2, 2018, pp. 239–267. DOI: 10.1007/s40571-017-0166-3.

[7]. Manhartsgruber B. The Lattice Bolzmann Method used for fluid flow modeling in hydraulic components. In Proc. of the 15th Scandinavian International Conference on Fluid Power, Sweden, 2017.

[8]. J.M. Gimenez, H. Aguerre, N.M. Nigro, S. Idelson. PFEM based solvers implemented in the OpenFOAM suite. In Proc. of the V International Conference on Particle-Based Methods (PARTICLES 2017), Hanover. Germany. 2017.
[9]. Nourgaliev R.R., Dinh T.N., Theofanous T.G., Joseph D. The lattice Boltzmann equation method: theoretical interpretation, numerics and implications. International Journal of Multiphase Flow, vol. 29, issue 1, 2003, pp. 117−169.
[10]. Chen S., Doolen G.D. Lattice Boltzmann method for fluid flows. Annual Review of Fluid Mechanics, vol. 30, 1998, pp. 329–364.
[11]. Succi S. The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press, 2001, 288 p.
[12]. B. Fornberg, T.A. Driscoll, G. Wright, R. Charles. Observations on the Behavior of Radial Basis Function Approximations Near Boundaries. Computers and Mathematics with Applications. vol. 43, issues 3-5, 2002, pp. 473-490.
[13]. Moukalled F., Mangani L., Darwish M. The Finite Volume Method in Computational Fluid Dynamics. An Advanced Introduction with OpenFOAM and Matlab. Fluid Mechanics and Its Applications, vol. 113, Springer. 2015.
[14]. Ламб Г. Гидродинамика. М., Л., ОГИЗ Гостехиздат, 1947, 928 стр.

Testing different numerical methods opportunities for internal flows simulation

V.G. Melnikova <vg-melnikova@yandex.ru>

Bauman Moscow State Technical University,
5-1, 2-nd Baumanskaya st., Moscow, 105005, Russia

Abstract. Numerical simulation plays an important role in the design of new hydraulic units. It allows to minimize the number of expensive experimental tests of products and reduces development time. The flow in pipes, valves, regulators and other hydraulic elements belongs to the internal incompressible flow. Standard numerical methods such as a finite volume method (FVM) and finite element method (FEM) are already successfully used for incompressible internal flows modeling. However, in the case of domains with moving boundaries, these methods are hard to set up and sometimes inefficient. Therefore, now, there is a necessity of search of alternative methods for such class of problems. Requirements for new methods include acceptable accuracy and high computing efficiency. The aim of this study is an overview, testing and comparison different simulation methods for simplest types of internal flow: finite volume method, particle finite volume method (PFVM) and Lattice Boltzmann method (LBM). Different shapes of circular pipes were considered: the straight pipe with the constant area, the step pipe (abruptly increase of the diameter), the backward step pipe (abruptly decrease of the diameter) and the elbow pipe. The velocities and pressure fields, accuracy and simulation time were compared. Next solvers were used in the study: pimpleFoam as the OpenFOAM implementation of FVM, XFlow as the implementation of LBM, and ParticlePimpleFoam as OpenFOAM implementation of PFVM. Four values of the non-dimensional time step (Courant numbers) for PFVM and FVM methods: 1, 2, 5 and 10 were considered.

Keywords: PFVM; LBM; FVM; OpenFOAM; internal flow; computational fluid dynamics; numerical simulation.

DOI: 10.15514/ISPRAS-2018-30(6)-18

For citation: Melnikova V.G. Testing different numerical methods opportunities for internal flows simulation. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 315-328 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-18

References

[1]. Castilla R., Alemany I., Algar A., Gamez-Montero P.J., Roquet P., Codina E. Pressure-Drop Coefficients for Cushioning System of Hydraulic Cylinder with Grooved Piston: A Computational Fluid Dynamic Simulation. Energies, vol. 10, no. 11, 2016.

[2]. Fries C., Manhartsgruber B. A moving piston boundary condition including gap flow in OpenFOAM. Wseas Transactions on Fluid Mechanics, vol. 10, 2015, pp.95-104.

[3]. Melnikova V.G., Kotsur O.S., Shcheglov G.A. Numerical simulation of the flow rate regulator valve using OpenFOAM. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 1, 2017, pp. 53-70 (in Russian). DOI: 10.15514/ISPRAS-2017-29(1)-4.

[4]. Stepanchwnko T.E. Simulation of the flow in the pipeline in the COMSOL 3.5 Multiphysics package. In Proc. of the VIII Russian scientific-practical conference «Microsoft Technologies in the theory and practice of programming», 2012, pp. 62-65 (in Russian).

[5]. Gobish A.V. Modeling of internal viscous incompressible flow by finite element method using counterflow schemes. Abstract of the PhD Thesis, Institute of Computational Technologies of SB RAS, Novosibirsk, 2007, 17 p. (in Russian).

[6]. Mutabaruka P., Kamrin Ken A simulation technique for slurries interacting with moving parts and deformable solids with applications. Computational Particle Mechanics, vol. 5, issue 2, 2018, pp. 239–267. DOI: 10.1007/s40571-017-0166-3.

[7]. Manhartsgruber B. The Lattice Bolzmann Method used for fluid flow modeling in hydraulic components. In Proc. of the 15th Scandinavian International Conference on Fluid Power, Sweden, 2017.

[8]. J.M. Gimenez, H. Aguerre, N.M. Nigro, S. Idelson. PFEM based solvers implemented in the OpenFOAM suite. In Proc. of the V International Conference on Particle-Based Methods (PARTICLES 2017), Hanover. Germany. 2017.

[9]. Nourgaliev R.R., Dinh T.N., Theofanous T.G., Joseph D. The lattice Boltzmann equation method: theoretical interpretation, numerics and implications. International Journal of Multiphase Flow, vol. 29, issue 1, 2003, pp. 117−169.

[10]. Chen S., Doolen G.D. Lattice Boltzmann method for fluid flows. Annual Review of Fluid Mechanics, vol. 30, 1998, pp. 329–364.

[11]. Succi S. The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press, 2001, 288 p.

[12]. B. Fornberg, T.A. Driscoll, G. Wright, R. Charles. Observations on the Behavior of Radial Basis Function Approximations Near Boundaries. Computers and Mathematics with Applications. vol. 43, issues 3-5, 2002, pp. 473-490.

[13]. Moukalled F., Mangani L., Darwish M. The Finite Volume Method in Computational Fluid Dynamics. An Advanced Introduction with OpenFOAM and Matlab. Fluid Mechanics and Its Applications, vol. 113, Springer. 2015.

[14]. Lamb G. Hydrodynamics. Dover Publications, 1945, 768 p.

Математическая модель процесса дегазации полимерного покрытия в условиях открытого космоса

Н.А. Полибина <Natalya.d.2011@inbox.ru>

Корпорация космических систем специального назначения «Комета», 115280 Москва, Россия, ул. Велозаводская, 5

Аннотация. Разработана полуэмпирическая математическая модель, описывающая процесс элиминирования компонентов полимерного покрытия с поверхности элемента конструкции космического аппарата. Расчет производился с целью оценить толщину пленки, образующейся в результате неравновесной конденсации летучих компонентов на защищаемом диаметре. Исходными данными для модели служат установленные экспериментально временные характеристики процесса дегазации исследуемого полимерного покрытия, а также сведения о температурном режиме элемента конструкции и законе его изменения вследствие изменения ориентации космического аппарата по отношению к Солнцу. Для непосредственного расчета параметров дегазации учитывается, что геометрия элемента конструкции и защищаемого элемента близки к осесимметричным. В результате получены численные значения средней, а также максимальной и минимальной толщины образующейся пленки за весь срок активного существования космического аппарата.

Ключевые слова: математическое моделирование; вакуум; дегазация; полимерное покрытие; оптическая система; условия открытого космоса; потеря массы; деструкция.

DOI: 10.15514/ISPRAS-2018-30(6)-19

Для цитирования: Полибина Н.А. Математическое моделирование процесса дегазации полимерного покрытия в условиях открытого космоса. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 329-340. DOI: 10.15514/ISPRAS-2018-30(6)-19

1. Введение

Современные тенденции космического кораблестроения диктуют определенные облегченные способы компоновки непилотируемых космических аппаратов (КА). Одной из особенностей является негерметичное исполнение, которое позволяет в разы увеличить полезную нагрузку при сохранении массовых характеристик. Однако такой подход имеет определенные недостатки, в частности − делает уязвимыми чувствительные узлы аппарата для воздействия агрессивных факторов космического

пространства. Вследствие этого возникает риск снижения срока эксплуатации аппарата и преждевременный выход из строя отдельных его элементов.

Агрессивные факторы космического пространства (далее ФКП) – градиент температур, ионизированное излучение, сверхнизкое давление – вызывают интенсивные процессы дегазации неметаллических (а в отдельных случаях и металлических) конструкционных материалов аппарата. Такие процессы делятся на периодические и непрерывные [1-6]. К периодическим источникам дегазации относятся продукты неполного сгорания двигательных установок, а к непрерывным источникам – пылегазовые выделения поверхностей КА (выделение адсорбированных паров и газов), сублимация неметаллических материалов, утечки различного рода технических жидкостей и топлива.

Деструктивное воздействие ФКП на конструкционные материалы КА вызывает ускоренное элиминирование адсорбированных газов и паров из поверхностных слоев, частичную деструкцию и сублимацию низкомолекулярных компонентов самого материала. Как следствие, нарушаются физико-химические и поверхностные свойства таких материалов, что особенно опасно в нагружаемых узлах, ухудшаются оптические параметры терморегулирующих покрытий, красок, эмалей, происходит общая потеря массы материалов.

В ходе длительного пребывания аппарата на орбите локально создаются благоприятные условия для неравновесной конденсации газообразных продуктов. Это, в свою очередь, особенно вредит оптически чувствительным поверхностям (солнечные батареи, терморегулирующие покрытия, зеркала, линзы и т.п.) вследствие ухудшения их рабочих характеристик. По литературным данным [7-9] история космонавтики насчитывает несколько случаев нарушения работоспособности КА вследствие загрязнения его поверхностей продуктами газовыделения вплоть до вывода из строя приборов.

Процессы, происходящие в полимерных композиционных материалах под действием ФКП, характеризуются следующими этапами: потеря массы; диффузия низкомолекулярных компонентов из объема к поверхности материала; последующая их десорбция и образование летучих продуктов способных, в свою очередь, оседать на различных поверхностях аппарата.

Наибольшую опасность для чувствительного узла, с точки зрения его загрязнения, представляют неметаллические материалы, находящиеся в зоне прямой видимости.

2. Постановка задачи

Исходя из конструктивных особенностей конкретного аппарата, рассматриваемого в данной работе, оснащенного сверхчувствительной термостабилизированной оптической системой, которая в условиях эксплуатации функционирует при постоянной пониженной температуре, материалом, находящимся в зоне прямой видимости, является в первую

очередь внутреннее покрытие солнцезащитного узла – эмаль ЭКОМ-2 (черная).

Данная эмаль относится к терморегулирующим покрытиям (ТРП) класса «истинные поглотители» и представляет собой суспензию пигмента и наполнителей в акриловом сополимере и смеси растворителей.

В качестве амидосодержащей акриловой смолы используются:

- смола АС – продукт сополимеризации метакрилата и булилметакрилата;
- смола АСН – раствор сополимера метакриламида с бутилметакрилатом в смеси ацетона и изопропилового спирта или сополимера;
- смола С38 – раствор продукта сополимеризации метакрилата с бутилметакрилатом, акрилонитрилом и стиролом в смеси растворителей ацетона, ксилола, бутилацетата или бутилового спирта.

Толщина покрытия составляет 80-100 мкм.

Рассматриваемый элемент конструкции представлен на рис.1. Он состоит из двух солнцезащитных бленд, внутренняя – в виде цилиндра, внешняя в виде усеченного конуса.

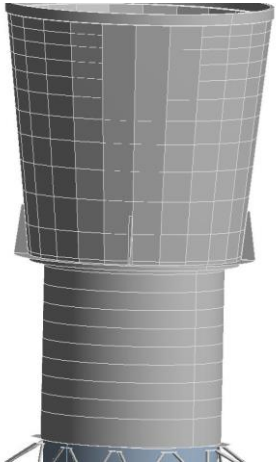


Рис. 1. Схематическое изображение внешнего вида солнцезащитного узла
Fig.1. Scematic view of sun shield external appearance

Защищаемый диаметр (ЗД) – внешние элементы оптической системы (расположены в центре нижнего основания цилиндрической бленды), общий диаметр которых составляет 96 см.

Целью данной работы являлась разработка математической модели для описания процессов дегазации полимерного покрытия рассматриваемого элемента конструкции в условиях открытого космоса и алгоритма расчета толщины загрязнения, которое образуется на защищаемом диаметре в результате неравновесной конденсации газообразных компонентов.

В ходе работы были решены следующие задачи.

- Экспериментально получены кинетические зависимости потери массы образцов покрытия ЭКОМ-2 (черная) от времени в изотермическом режиме при различных температурах (40, 60, 80 и 100 ºС).
- Экспериментально получена общая зависимость потери массы образца полимерного покрытия от температуры.
- Определена аппроксимирующая зависимость, описывающая процесс дегазации полимерного материала в условиях космического пространства.
- Разработан алгоритм расчета толщины загрязняющей пленки на защищаемом диаметре на основе эмпирических и теоретических данных.
- Получены значения максимальной, минимальной и средней толщины загрязняющей пленки, позволяющей оценить степень влияния загрязнения на работоспособность оптической системы.

3. Экспериментальная часть

3.1 Термогравиметрический анализ

Были получены кинетические кривые потери массы образцов черной эмали при различных температурах и продолжительности эксперимента 24 ч. Исследования проводились термогравиметрическим методом в изотермическом режиме. Обобщенные результаты эксперимента представлены на рис. 2, 3 и в табл. 1.

Суть метода термогравиметрического анализа заключается в измерении изменения массы образца при его выдержке при постоянной заданной температуре в специальной печи. Измерение массы образца производится с использованием кварцевых микровесов, которые позволяют фиксировать изменение веса с точностью до 0,01 мг.

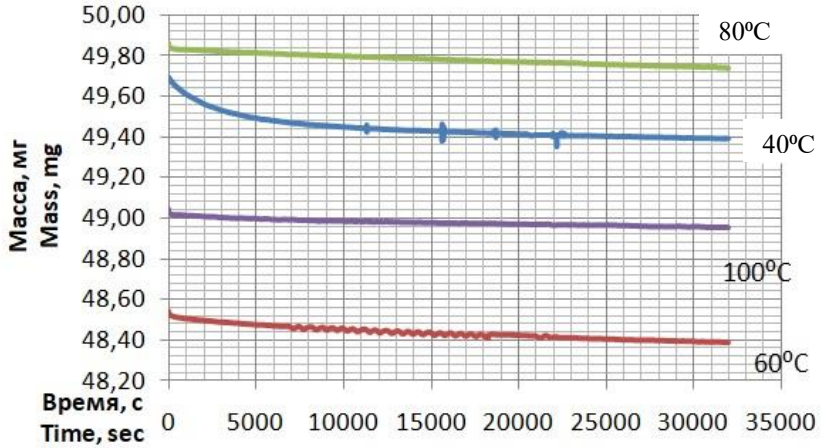


Рис. 2. Кинетические кривые потери массы образцов черной эмали при различных температурах
Fig. 2. Kinetic curves of cover samples mass loss at different temperatures

Табл. 1. Результаты термогравиметрического анализа покрытия ЭКОМ-2 (черная)
Table 1. The results of thermogravimetric analysis of black cover samples

T, °C	**Δ*, мг**	**ПМ, %**
30	0,550	1,120
40	0,337	0,678
45	0,148	0,253
50	0,075	0,125
55	0,208	0,357
60	0,200	0,412
70	0,353	0,582
80	0,215	0,431
100	0,221	0,450

*Δ = m-m0, где m – конечная масса образца, мг; m0 – начальная масса образца, мг

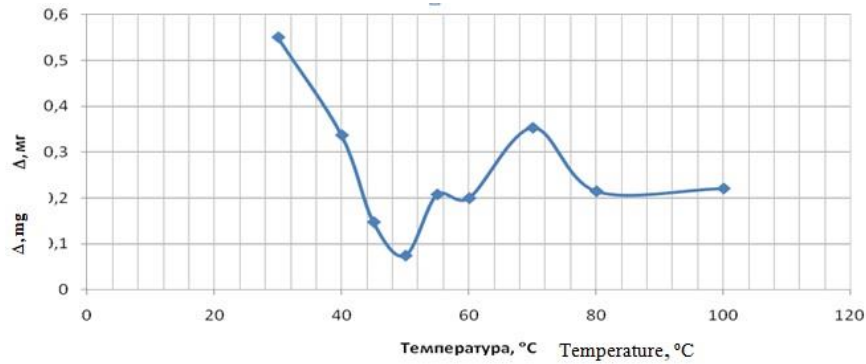


Рис. 3. Зависимость Δ (мг) образца от температуры
Fig.3. Function Δ (T)

Согласно полученным экспериментальным данным, на кривой Δ-температура присутствуют два максимума: при 30 и 70 градусах. Повышенное газовыделение при температуре 30 градусов обусловлено, по-видимому, выделением молекул растворителя, поскольку отверждение эмали происходит при комнатной температуре. Минимум потери массы при 50 °C говорит о прохождении процесса самопроизвольной полимеризации после выхода остатков растворителя.

Второй максимум при 70 градусах свидетельствует уже о выделении низкомолекулярных компонентов, входящих в состав самой эмали.

3.2 Жидкостная экстракция

Общее количество низкомолекулярных компонентов (включая остатки растворителя) было определено экспериментально методом жидкостной экстракции с использованием «сильных» и «слабых» растворителей:

- тетрагидрофуран (ТГФ) – универсальный «сильный» растворитель, применяется для экстракции как полярных, так и неполярных компонентов;
- ацетон – «сильный» растворитель, применяется для экстракции полярных компонентов;
- диэтиловыйэфир – «слабый» растворитель, применяется для экстракции неполярных компонентов.

Согласно методике, исследуемые образцы помещаются в закрытые бюксы с растворителем и выдерживаются в течение 48 ч. Затем высушиваются в течение 24 ч при температуре 50°С.

Процентное содержание вымытых из ткани компонентов (сухой остаток) определяется весовым методом и рассчитывается по формуле:

$$\frac{M2-M1}{M1} \cdot 100\%,$$

где М1 – масса образца до экстракции, М2 – масса образца после экстракции.

Результаты жидкостной экстракции представлены в табл. 2.

Табл. 2. Результаты жидкостной экстракции образцов ЭКОМ-2 (черная)

Table 2. The results of black cover samples liquid extraction

Наименование растворителя	**M_1, мг**	**M_2, мг**	**Сухой остаток, %**
ТГФ	0,54494	0,39130	28,2
Ацетон	0,59268	0,47509	19,8
Диэтиловый эфир	0,64377	0,59931	7,0

Таким образом, максимальная доля низкомолекулярных компонентов (полярных и неполярных), содержащаяся в материале и способная к элиминированию – 28,2%.

4. Расчетная часть

На рис. 4 красными линиями показаны полученные в экспериментах зависимости потери массы от времени при температурах 40 °С (а), 60 °С (б) и 80 °С (в). Анализ показал, что использование закона Аррениуса не позволяет с достаточной точностью описать кинетику дегазации, в то время как все приведенные кривые хорошо аппроксимируются зависимостью

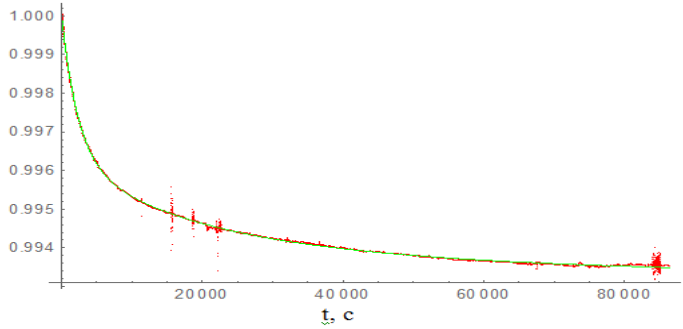
$$J = A + B \cdot \operatorname{arctg}(LT) \qquad (1)$$

где J – поток массы вещества, мг/с; T – время эксперимента, с; A, B, L – коэффициенты, определяемые для каждого эксперимента нелинейным

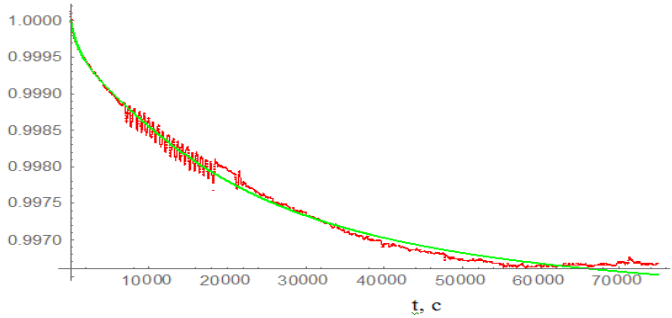
методом наименьших квадратов. Зеленые кривые на рис. 4 построены в соответствии с формулой (1).

Разработана полуэмпирическая математическая модель, описывающая процесс абляции с внутренней поверхности бленды компонентов эмали и позволяющая оценить толщину пленки, образующейся на защищаемом диаметре. Исходными данными для модели служат установленные экспериментально временные характеристики процесса дегазации, а также сведения о температурном режиме бленды и законе ее изменения вследствие изменения ориентации бленды по отношению к Солнцу.

a)



b)



c)

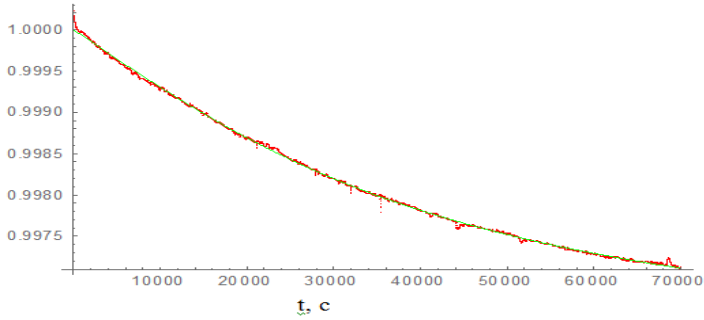


Рис. 4. Зависимость потери массы образца (%) от времени при температуре 40 ℃ (a), 60 ℃ (b), 80 ℃ (c)

Fig. 4. Mass loss relation to experimental duration at 40 ℃ (a), 60 ℃ (b), 80 ℃ (c)

При построении модели были приняты следующие упрощающие допущения и предположения:

1) с внутренней поверхности внутренней бленды выделения низкомолекулярных компонентов эмали не происходит вследствие низких для газовыделения температур; газовыделение имеет место только с внутренней поверхности внешней бленды;
2) за время активного существования аппарата все количество летучих веществ, способных к элиминированию, будет удалено из материала, причем основное газовыделение приходится на несколько первых дней работы аппарата на орбите;
3) разработанная математическая модель является стационарной, т.е. позволяет оценить толщину пленки, полученной после выделения из эмали всего газа и осаждения на защищаемый диаметр некоторой его части;
4) интенсивность газовыделения с поверхности эмали в разных направлениях принимается пропорциональной косинусу угла между рассматриваемым направлением и нормалью к поверхности, что согласуется с известными в литературе экспериментальными данными; коэффициент пропорциональности находится из условия выделения всего объема газа в полупространство;
5) считается, что выделенный в заданном направлении объем газа распространяется в этом направлении и осаждается на первой попавшейся на его пути преграде (по аналогии с распространением лучей в моделях геометрической оптики без их отражения);
6) в связи с изложенными в пп. 4) и 5) предположениями принимается в расчет только газовыделение с боковых (внутренних) сторон диафрагм (рис. 5); влияние газовыделения с внутренней поверхности бленды принимается незначительным.

Для непосредственного расчета параметров газовыделения учитывается, что геометрия бленды и защищаемого элемента оптической системы близки к осесимметричным, что несколько упростило методику расчета.

Защищаемый диаметр был разбит системой радиусов и концентрических окружностей на ячейки равной «ширины», аналогичным образом также были разбиты на ячейки поверхности диафрагм внешней бленды. Для каждой ячейки на защищаемом диаметре была построена система лучей, направленных на центры ячеек диафрагм. Если данный луч не пересекался другими диафрагмами, принималось, что количество газа, пропорциональное площадям ячеек на диафрагме и диаметре и косинусу угла между нормалью к ней и проведенным лучом, осаждалось на данной ячейке защищаемой поверхности:

$$\Delta m \sim S_{яч}^{диафр} \cdot S_{яч}^{диам} \cdot \cos\angle(\vec{n}_{диафр}, \vec{s});$$

если же луч пересекался другими диафрагмами – принималось, что выделяемый с поверхности диафрагмы газ на защищаемой поверхности не осаждался.

В расчетах по разработанной математической модели каждая диафрагма и защищаемый диаметр разбивались на 24 ячейки в окружном направлении, каждая такая ячейка на поверхности диафрагмы по ширине разбивались на 10 узких «полос» равной ширины, защищаемый круг также разбивался в радиальном направлении на 10 ячеек (рис. 5).

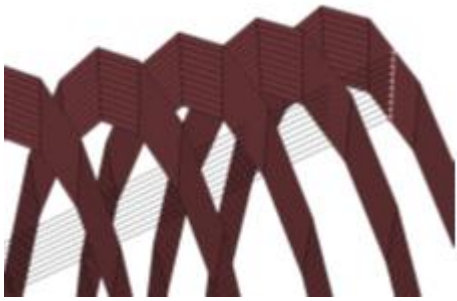


Рис. 5. Диафрагмы бленды и лучи, проведенные от точек на защищаемом диаметре к центрам ячеек на последней диафрагме. Все лучи, кроме двух нижних, пересекаются другими диафрагмами

Fig5. Blend apertures and rays drawn from points on the diameter being protected, to the centers of the cells on the last diaphragm. All the rays, except for the bottom two, intersect other diaphragms

Определение «видимости» вдоль всех лучей, как и все остальные расчеты по предложенной модели были произведены в системе Wolfram Mathematica, что позволило существенно упростить процесс программирования, сосредоточившись на содержательной стороне модели.

Также в модели была дополнительно учтена неравномерность нагрева внешней бленды Солнцем. Это оказывало влияние на суммарное количество газа, выделяемое с единицы площади поверхности диафрагм в зависимости от их температуры на основании полученных экспериментальных данных.

На основании полученных сведений об осажденном количестве газа на ячейках, введенных в расчете на поверхности защищаемого оптического элемента, была вычислена средняя толщина образующегося загрязнения, которая составила 2984 Å.

В результате расчетов были получены значения максимальной и минимальной толщины загрязнения, образующегося на поверхности защищаемого диаметра (оптической системы): min = 2880 Å; max = 3136 Å.

5. Заключение

Проведенные экспериментальные исследования и произведенные на их основании расчеты, позволяют сделать следующие выводы.

1. Температурная зависимость потери массы полимерного композиционного материала характеризуется двумя экстремумами: максимальное значение потери массы вещества при 30ºС свидетельствует об элиминировании адсорбированных газов и паров из поверхностных слоев материала; минимальное значение потери массы вещества при 50ºС свидетельствует о протекании процесса самопроизвольной полимеризации.
2. Показано, что кинетические зависимости потери массы образцов исследуемой эмали наилучшим образом аппроксимируются неэкспоненциальной зависимостью. Построены экспериментальные и теоретические кривые.
3. Экспериментально установлено, что максимальная доля вещества, способного к элиминированию из материала составляет 28,2%.
4. Разработана упрощенная математическая модель массопереноса газообразного вещества к защищаемому диаметру и алгоритм расчета толщины образующейся на ЗД пленки загрязнения. Расчет производился для всего срока активного существования аппарата.

Список литературы

[1]. Milinchuk V.K., Smirnova T.N. Properties of the polymeric films after natural exposure to the space environment on the orbital space station «MIR». In Proc. of the 8th International Symposium on Materials in a Space Environment, 2000.
[2]. Leger L.J. Oxygen atomic reaction with shuttle materials at orbital altitude-data and experiment status. AIAA paper, 83-0073, 1983.
[3]. Акишин А.И. Космическое материаловедение. Методическое и учебное пособие. М., НИИЯФ МГУ, 2007, 209 стр.
[4]. Silverman E.M. Space environmental effects on spacecraft: LEO materials selection guide. Nasa CR-4661, 1995, part 1-2.
[5]. Нусинов М.Д. Воздействие и моделирование космического вакуума. М., Машиностроение, 1982, 175 стр.
[6]. Акишин А.И., Новиков Л.С. Воздействие окружающей среды на материалы космических аппаратов. М., Знание, 1983, 64 стр.
[7]. Панасюк М.И., Новиков Л.С. Модель космоса. Том 2. М., КДУ, 2007, 1144 стр.
[8]. Хасаншин Р.Х., Тимофеев А.Н. О влиянии облучения на загрязнение поверхностей продуктами газовыделения полимерных композиционных материалов. Ракетостроение и космическая техника, №41, 2009.
[9]. Leger L.I., Visentine I.T., Kaminecz I.F. Low Earth orbit atomic oxygen effects on surfaces. AIAA Paper, 1984, № 548.

Mathematical modeling of polymeric cover outgassing process in open space conditions

N.A. Polibina <Natalya.d.2011@inbox.ru>
Special Space Systems Corporation «Kometa»
Velozavodskaya Str. 5, Moscow, 115280, Russian Federation

Abstract. A half-empiric mathematical model was elaborated to describe polymeric cover outgassing process in open space conditions. The calculation was carried out to estimate film thickness that arises as a result of gaseous products nonequilibrium condensation at a sensitive surface. Source data were: a) experimentally obtained time functions of polymeric cover mass loss at different temperatures; b) the construction element temperature regime data; c) the construction element temperature changes according to its location towards the Sun. To carry out the calculation the construction element and the sensitive surface geometry is considered to be axisymmetric. As a result, the numerical values of the average as well as maximum and minimum film thickness for the whole exploitation period of the spacecraft.

Keywords: spacecraft; contamination; outgassing; mass loss; temperature regime of the spacecraft; mathematical model; modeling.

DOI: 10.15514/ISPRAS-2018-30(6)-19

For citation: Polibina N.A. Mathematical modeling of polymeric cover outgassing process in open space conditions. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 329-340 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-19

References

[1]. Milinchuk V.K., Smirnova T.N. Properties of the polymeric films after natural exposure to the space environment on the orbital space station «MIR». In Proc. of the 8th International Symposium on Materials in a Space Environment, 2000.
[2]. Leger L.J. Oxygen atomic reaction with shuttle materials at orbital altitude-data and experiment status. AIAA paper, 83-0073, 1983.
[3]. Akishin A.I. Space Materials. Methodical and tutorial. M., INP MSU, 2007, 209 p. (in Russian)
[4]. Silverman E.M. Space environmental effects on spacecraft: LEO materials selection guide. Nasa CR-4661, 1995, part 1-2.
[5]. Nusinov M.D. Impact and modeling of cosmic vacuum. M., Mashinostroenie, 1982, 175 p. (in Russian)
[6]. Akishin A.I., Novikov L.S. The impact of the environment on the materials of spacecraft. M., Knowledge, 1983, 64 p. (in Russian)
[7]. Panasyuk M.I., Novikov L.S. Model of the cosmos. Volume 2. M., KDU, 2007, 1144 p. (in Russian)
[8]. Khasanshin R.Kh., Timofeev A.N. On the effect of irradiation on the contamination of surfaces by the products of gas evolution of polymer composite materials. Rocket and space technology, №41, 2009 (in Russian).
[9]. Leger L.I., Visentine I.T., Kaminecz I.F. Low Earth orbit atomic oxygen effects on surfaces. AIAA Paper, 1984, № 548.

О представлении модельного времени при помощи механизмов функционального программирования[1]

[1] *Д.В.Буздалов <buzdalov@ispras.ru>*
[1, 3, 4] *А.К.Петренко <petrenko@ispras.ru>*
[1, 2, 3, 4] *А.В.Хорошилов <khoroshilov@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский физико-технический институт, 141700, Московская область, г. Долгопрудный, Институтский пер., 9*
[3] *Московский государственный университет имени М. В. Ломоносова Москва, 119991, ГСП-1, Ленинские горы, д. 1*
[4] *НИУ "Высшая школа экономики", 101000, Россия, г. Москва, ул. Мясницкая, д. 20*

Аннотация. Функциональное программирования играет все большую роль в современном компьютеризированном мире. Такой подход позволяет создавать более надежный, абстрактный и автоматически проверяемый код. Однако при этом эти техники мало используются при создании систем проектирования и моделирования ответственных систем. Данная работа является попыткой применения удачных техник функционального программирования для создания системы моделирования на примере системы динамического моделирования поведения систем для проверки характеристик, связанных с временем.

Ключевые слова: моделирование архитектуры; ответственные системы; моделирование поведения; модельное время; функциональное программирование; монады

DOI: 10.15514/ISPRAS-2018-30(6)-20

Для цитирования: Буздалов Д.В., Петренко А.К., Хорошилов А.В. О представлении модельного времени при помощи механизмов функционального программирования. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 341-366. DOI: 10.15514/ISPRAS-2018-30(6)-20

[1] Исследования проводились при финансовой поддержке РФФИ в рамках проекта 17-01-00504.

1. Введение

Контекстом работы является создание моделей архитектуры ответственных систем с целью дальнейшей проверки этих моделей на соответствие функциональным и нефункциональным требованиям (требованиям отказоустойчивости, потребления ресурсов памяти, задержек при передаче информации и др.).

Архитектурные модели программно-аппаратных систем управления представляют структуру системы и набор спецификаций поведения отдельных компонентов и связей между ними, в том числе описывающих темпоральные свойства поведения. Соответственно, рассматриваются такие характеристики моделируемой системы, как длительность выполнения тех или иных задач моделируемой системой и размер задержек по передаче информации внутри системы.

В проводившихся ранее работах [1][2] авторы использовали для динамического анализа акторную модель, рассматривая отдельные компоненты системы, наделенные поведением, как отдельные независимые общающиеся акторы, работающие в модельном времени. При этом были выявлены следующие недостатки системы, основывающейся на акторной модели, при использовании в динамическом моделировании поведения моделей систем:

- нелокальность акторной системы
  - акторы рассчитывают на наличие определенных акторов (например, которые умеют обрабатывать определенные сообщения), при этом не могут гарантировать или проверить это;
  - нельзя автоматизированно статически (не тестированием) оценить корректность замены одного актора, на другой в случае, если третий актор, который пользуется заменяемым, рассчитывает на какие-нибудь свойства исходного (например, рассчитывает на возможность принять и полностью обработать определенный тип сообщения);
- нетипизированность акторной системы
  - принятие или непринятие сообщения решается только во время выполнения и не проверяется при компиляции;
- слабость средств для поддержки множественных, недетерминированных, вероятностных или интерактивных параметров на входе:
  - множественные параметры требуют множественных запусков с комбинаторным количеством запусков и повторением общей работы;
  - вероятностные и интерактивные параметры требуют специальной ad hoc обработки в каждом отельном работающим с этим акторе; это приводит к смешению собственной логики актора с реализацией получения этих параметров и приводит к заселению акторов посторонней логикой;

- недетерминизм требует или перезапусков (подобно множественным параметрам), или слишком раннего недетерминированного выбора отдельных параметров.

Акторная модель вычислений широко применяется в традиционном программировании, хотя при этом там не учитывается специальным образом модельное время, с которым необходимо работать при моделировании поведения систем реального времени (для проверки свойств модели, связанных со временем).

Одновременно с этим в программировании наблюдается тенденция противопоставления акторной модели вычисления функциональному программированию, которое позволяет решать (помимо других) те же проблемы, для которых была создана акторная модель. При этом функциональное программирование обладает рядом преимуществ. Аналогично тому, как взамен акторной модели применяются техники функционального программирования в областях, не связанных с моделированием времени, возникает идея попробовать применить аналогичные техники взамен акторной модели в контексте модельного времени, необходимого для поведенческого моделирования систем реального времени.

В данной работе предлагается способ организации поведенческих спецификаций для динамического моделирования систем реального времени, который учитывает модельное время и использует технику функционального программирования для решения проблем, сопутствующих моделированию при помощи акторов.

Само понятие функционального программирования не является четко определенным и порой интерпретируется несколько по-разному разными специалистами. В данной работе мы будем пользоваться понятиями, связанными с чистым типизированным функциональным программированием. Более детальное описание используемых терминов и понятий можно найти на странице проекта РФФИ[2]. Там читатель найдет краткое описание и примеры для таких понятий как *чистая функция*, *параметрический полиморфизм*, *функция высших порядков*, *функтор*, *монада*, *эффект (в чистом функциональном программировании)*, которые важны для понимания основного материала.

2. Эффекты в функциональном программировании

Так как данная работа использует технику моделирования эффектов в *чистом* функциональном программировании, следует остановиться на том, что мы понимаем под *эффектом* и как проходило развитие этого понятия в истории функционального программирования.

[2] Страница проекта гранта РФФИ 17-01-00504: https://forge.ispras.ru/projects/ductilejur

2.1 Побочные эффекты

Основным понятием в чистом функциональном программировании является понятие *чистой тотальной функции* [3][4], т.е. функции без *побочных эффектов*. Однако, в конечном итоге, программы должны производить побочные эффекты – например, читать и писать в базы данных, выводить информацию пользователю, читать и писать файлы, взаимодействовать с другими программами по сети и т.п. Таким образом, перед функциональным стилем программирования встает вопрос совмещения *чистоты* используемых функций и *побочных эффектов* для выполнения полезных действий.

Для того, чтобы совместить эти две, казалось бы, несовместимые вещи, в функциональном программировании используют тот факт, что *чистое* вычисление может возвращать достаточно произвольную структуру данных, в том числе структуру данных, которая внутри содержит описание вычисления, не являющегося *чистым* (т.е., *вычисление с побочным эффектом*).

При этом, если собственно запуск *вычисления с побочным эффектом* недоступен *чистому* коду, то не нарушается никаких предположений о чистом коде.

Таком образом, результатом работы чистого кода может быть описание *вычислений с побочным эффектом*, которое может быть далее выполнено внешним (по отношению к самой программе) исполнителем. Тем самым, производится разделение выполнения функциональной программы на две части:

- в первой части исходная программа, написанная на функциональном языке, вычисляет некую структуру данных, которая содержит помимо других вычислений инструкции, содержащие побочные эффекты, предназначенные для выполнения специальным исполнителем; на этом выполнение собственно программы заканчивается;
- во второй части специальный исполнитель исполняет инструкции с побочным эффектом из вычисленной специальной структуры данных.

В общем случае, может существовать несколько специальных исполнителей инструкций с побочным эффектом и, соответственно, структур данных, с которыми эти исполнители работают. Некоторые из них предназначены для выполнения произвольных внешних побочных эффектов, некоторые только для определенного ограниченного множества (что позволяет гранулировано контролировать их использование). Существуют также библиотеки, позволяющие абстрагироваться от конкретного исполнителя побочных эффектов, позволяя писать полиморфный код с указанием используемых эффектов.

- **Произвольные эффекты**

  Чаще всего, структуры данных, которые инкапсулируют в себе *вычисления с побочным эффектом*, называются `IO` или `Task` [5][6][7][8].

Соответственно, могут быть чистые функции, которые возвращают значения этих типов, например, функции строкового ввода и вывода:

```
def printLine(s: String): IO[Unit]
def readLine: IO[String]
```

В данном случае, функция `printLine` чистая и она лишь возвращает вычисление, которое распечатает строку на экран (в возвращаемом значении типа `IO[Unit]`). Аналогично, функция `readLine` возвращает не строку, введенную пользователем, а вычисление, которое может вернуть строку, введенную пользователем.

- **Гранулярные эффекты**

  Существуют способы описания вычислений, которые *полиморфны* по конкретному типу возвращаемого значения, но имеют гранулярный контроль за производимыми эффектами.

  В этом случае функция возвращает тип произвольного эффекта `F[_]`. При этом в сигнатуре функции указываются ограничения на возможные варианты эффектов, допустимые для осуществления этой функцией.

2.2 Композиция вычислений с эффектами

Промоделировав функции с побочным эффектом как чистые функции вида `A => IO[B]` или более гранулярно как `A => F[B]` для определенных `F`, мы получаем некие возможности обработки информации, полученной при помощи побочного эффекта, но их пока недостаточно.

Например, при помощи функций `readLine` и `printLine`, описанных выше, не получается распечатать строку, которая содержала бы информацию из считанной строки, потому что `IO[A]` – это лишь описание *вычисления с побочным эффектом*, возвращающего `A`, поэтому невозможно его получить вне `IO` в чистой функции.

Для решения проблем *композиции* функций вида `A => IO[B]` и `B => IO[C]` была применена техника, которая называется *монадами* [5] и которая имеет свои истоки в теории категорий [9][10].

Если какой-то тип данных `M[_]` является монадой, это означает, что, имея `M[A]`, мы можем применить к нему функцию `A => M[B]` и получим значение `M[B]`. Эта операция исторически имеет названия `bind`, `flatMap` или оператора >>=.

Для примера с `IO[A]`, описывающей вычисление с побочным эффектом, это означает, что, если мы применим функцию, которая обрабатывает значение типа `A`, полученное в результате побочного эффекта, к функции, которая его обработает и вернет `IO[B]`, мы сможем произвести их композицию с тем, чтобы получить `IO[B]`.

```
def f[A, B](a: IO[A], f: A => IO[B]): IO[B] = a >>= f
```

Другой важной чертой монад является то, что всегда есть возможность по чистому значению получить монадическое, то есть получить значение как бы с *нулевым эффектом*. Обычно такого рода функции называются `pure` или `point` и имеют вид `A => F[A]`.

Монадическая композиция по сути является последовательной композицией зависимых вычислений, где каждое вычисление может производить эффект. Для монад типа `IO` монадическая композиция является своего рода аналогом «точки с запятой» в императивных языках программирования, где точка с запятой часто разделяет отдельные операторы, каждый из которых может обладать побочным эффектом.

Как мы увидим далее, в функциональных языках монады позволяют более гибко управлять этой композицией. При этом такого рода композиция гарантирует упорядоченность возникновения побочных эффектов.

Однако не всегда вычисления с эффектами должны быть строго упорядоченными. Причиной могут быть как соображения *эффективности* (не всегда вычисления стоит делать последовательно), так и соображения *семантики* (не всегда нужна или определена именно последовательная композиция эффектов).

Для такого рода композиций была предложена абстракция *аппликативных функторов* [11]. Аппликативные функторы позволяют слить пару эффектов в эффект получения пары объектов:

```
def product[A, B](a: F[A], b: F[B]): F[(A, B)]
```

Так как аппликативный функтор является функтором, к значению вида `F[(A, B)]` можно применить функцию вида `(A, B) => C` с тем, чтобы получить значение типа `F[C]`.

Для эффектов, которые можно выполнять параллельно, реализация аппликативного функтора может позволить выполнять эти вычисления параллельно. В общем случае, абстракция аппликативного функтора позволяет осуществлять параллельную композицию вычислений с эффектами.

2.3 Эффекты в широком смысле

На понятие *эффекта* можно посмотреть шире: под эффектом можно понимать не только побочные эффекты в виде изменения состояния внешнего по отношению к программе мира или ее переменных.

В общем случае (в зависимости от потребностей точного моделирования семантики выполнения в чистом функциональном языке) под эффектом можно понимать даже обращение к значению переменной находящейся, быть может, на другом узле вычислительной сети при распределенном выполнении [12]. В более частных случаях под эффектом могут пониматься любые действия функции помимо вычисления и возврата простого единственного значения, примеры которых будут рассмотрены далее.

Важным свойством функций с побочным эффектом является то, что они могут делать еще что-то, кроме вычисления возвращаемого значения, а также пользоваться еще чем-либо, кроме своих аргументов для своей работы. При функциональном программировании вместо *функций с побочным эффектом* вида `A => B` рассматривают *чистые* функции вида `A => F[B]`, где `F` инкапсулирует сам эффект (например, произвольный побочный эффект, когда это тип `IO`). При более широком взгляде на понятие *эффекта* можно рассматривать функции вида `A => F[B]` как *чистые* аналоги *функций с эффектом* вида `A => B` для более широкого класса типов `F`, чем просто побочный эффект.

Рассмотрим некоторые примеры монад, которые моделируют те или иные *эффекты* в чистом функциональном программировании.

- **Монада произвольного побочного эффекта.** Упоминавшиеся ранее монады `IO` и `Task` и другие подобные позволяют описывать вычисления с произвольным побочным эффектом.
- **Тривиальная монада.** Монада `Id` описывает “нулевой” эффект. Функция обладающая таким эффектом (т.е., функция вида `A => Id[B]`) на самом деле является чистой (фактически, `A => B`). Последовательная композиция функций с тривиальным эффектом совпадает с простой композицией функций.

  На практике такую монаду используют в для полиморфного по эффектам кода, где в конкретном случае никакие эффекты не нужны или действие определенных эффектов моделируется чистым кодом (например, при тестировании).
- **Монада частичных вычислений.** В качестве эффекта можно рассматривать ситуацию, когда функция в некоторых случаях не может возвратить значение, то есть функция не является *тотальной* (является *частичной*). В таких случаях используют монаду `Option` (она же `Maybe`).

  Последовательная композиция функций с эффектом частичных вычислений обладает семантикой *раннего падения* (fail fast). Это означает, что стоит любой функции в последовательности функций с этим эффектом вернуть значение типа `None` (т.е., отсутствие результата), вычисление прекращается с тем же результатом.
- **Монада множественных вычислений.** Некоторые функции вычисляют несколько значений (в том числе и отсутствие значений). Это тоже можно рассматривать как эффект и композировать такие многозначные функции друг с другом. Композицией многозначных вычислений является аналог декартового произведения (при необходимости с учетом повторений). Обыкновенный список `List` часто является подходящей монадой для многозначных вычислений.

- **Монада вычислений с контекстом.** Монада `Reader` представляет собой вычисление, которое имеет доступ до дополнительной информации: контекст или конфигурация вычисления. Возможность такого доступа (явно не отраженного напрямую в сигнатуре функции) также является эффектом (в широком смысле).
- **Монада вычислений с метаинформацией.** В качестве эффекта рассматривают формирование дополнительной информации при вычислении основного значения. Эта информация является своего рода описанием (возможно пустым), приложенным к вычисленному значению. За это отвечает монада `Writer`.

  Для корректной работы этой монады, в частности, для обеспечения композируемости монадических вычислений, нужно обеспечить, чтобы соответствующие используемые описания, которые идут вместе с вычисленным значением, также композировались.

  Характерным примером использования этой монады является описание вычислений, которые параллельно с какими-то вычислениями, формируют лог событий. В этом случае, этот лог событий и будет являться описанием этих вычислений.
- **Вероятностная монада.** Еще одним примером эффекта можно рассмотреть возврат не единичного значения, а вероятностного распределения значений. При рассмотрении конечных дискретных распределений вероятности, это является обобщением эффекта множественных значений (где каждому значению приписана вероятность возникновения этого значения).

  Можно рассматривать и распределения, не являющиеся дискретными. В этом случае результирующее вероятностное распределение должно быть далее проинтерпретировано (например, при помощи генератора случайных чисел).

2.4 Наслаивание монад

2.4.1 Проблема

Очень часто возникает ситуация, когда либо одновременно используются две или более монад для одного и того же значения (то есть код производит несколько эффектов), либо имеется потребность в композировании функций, использующих различные монады (то есть функции, имеющие разные эффекты). Уже через небольшое время после использования монад для моделирования эффектов (в широком смысле) было показано, что в общем случае разные монады не композируются [13].

Также было показано, что в общем случае порядок наслоения монад друг на друга существенно влияет на результат, что оказалось ограничением для написания полиморфного кода.

Например, композиция множественных и частичных вычислений может продуцировать (в зависимости от порядка применения эффектов множественности List и частичности Option) либо множественность частичных результатов (List из Option'ов), либо частичный множественный результат (Option List'ов).

2.4.2 Трансформация монад

В качестве попытки разрешения этих и связанных с этим проблем, был предложен механизм *трансформаций монад* (*monad transformers*) [14].

Однако, этот подход не позволял полноценно описывать код, который продуцирует отдельный эффект или несколько эффектов, при этом композируемый с другими эффектами. Код, использующий этот принцип, должен был всегда четко определять, в какой последовательности применяются используемые эффекты, и не допускал произвольной вставки других эффектов между используемыми эффектами (если только не был написан в специальной и очень громоздкой манере).

Таким образом, раньше времени принималось решение о последовательности применяемых эффектов. Из-за этого невозможно было описать единственную функцию, которая могла бы композироваться с другими функциями тех же эффектов в другом порядке. Также возникали проблемы с возникновением несколько раз одного и того же эффекта в этой последовательности эффектов.

2.4.3 Расширяемые эффекты

Позже был предложен подход, описывающий специальную монаду, названную `Eff`, которая позволяла описывать ленивое вычисление с неупорядоченным множеством эффектов [15][16]. Это неупорядоченное множество эффектов (по историческим причинам иногда называемое *стеком эффектов*) является параметром типа `Eff` и тем самым присутствует в сигнатуре всех функций, возвращающих значение типа `Eff`.

При этом были сформулированы правила монадической композиции таких вычислений. Результирующее вычисление имеет объединение множеств эффектов композируемых вычислений. Легкость объединения (и дальнейшей композиции) таких вычислений обеспечивается путем использования *ограниченного полиморфизма* (*bounded polymorphism*) по этому множеству. В таком случае объединение множеств сводится к объединению ограничений на полиморфный параметр вычислений, что естественным образом поддерживается в языках, поддерживающих ограниченный полиморфизм.

Соответственно, для того, чтобы выполнить вычисление со всеми эффектами, требуется применить к значению типа `Eff` (имеющего множество эффектов как параметр типа) специальные интерпретаторы эффектов (в требуемом порядке), где каждый из интерпретаторов будет либо убирать из множества эффектов как минимум один эффект, либо преобразовывать один эффект в

другие. Это должно продолжаться до тех пор, пока множество эффектов не станет пустым. Полученное значение будет чистым.

Этот подход обеспечивает гибкие возможности по реализации полиморфного по эффектам кода, который не требует раннего упорядочивания эффектов и не имеет проблем с повторением одного и того же эффекта в наслаивании несколько раз (из-за рассмотрения множества эффектов взамен последовательности). Ограничением подхода является то, что все функции, имеющие какой-либо эффект, необходимо оформлять в виде функций, возвращающих объект специального типа `Eff` (с определенными параметрами типа), а также необходимость использования примитивов, специфичных для типа данных `Eff`.

Рассмотрим пример вычислений, которые оформлены для использования с монадой `Eff`.

```
def multipleIntegers[R: _list]: Eff[R, Int] = ???
def sqrt[R: _option](x: Int): Eff[R, Double] = ???

def composition[R: _list:_option]: Eff[R, Double] =
for {
i <- multipleIntegers
x <- sqrt(i)
} yield x
```

В этом примере мы имеем два вычисления (`multipleIntegers` и `sqrt`), которые полиморфны по эффекту с использованием монады `Eff` (параметр типа `R`), но при этом первая функция требует, чтобы `R` поддерживал эффект множественности, а вторая – эффект частичных вычислений. Соответственно, функция композиции требует оба этих эффекта.

Далее, мы можем проинтерпретировать функцию композиции как минимум в двух упорядочиваниях эффектов: сначала эффект множественности, затем эффект частичных вычислений или наоборот.

```
type LO = Fx.fx2[List, Option]
// множество из двух эффектов

composition[LO].runList.runOption.run
// результат типа Option[List[Double]]

composition[LO].runOption.runList.run
// результат типа List[Option[Double]]
```

Мы можем использовать эту функцию и в более обширном контексте, например, с эффектом ввода-вывода. Таким образом, мы можем использовать одни и те же функции в произвольных контекстах, которые предоставляют обработку требуемых конкретному вычислению эффектов.

```
type Ext = Fx.fx3[List, Option, IO]

composition[Ext].runList.runOption
```

```
// результат типа
//   Eff[Fx.fx1[IO], Option[List[Double]]]
```

2.4.4 Полиморфизм по эффекту

Своего рода обобщением предыдущих двух решений является вместо использования определенной структуры данных и полиморфизма по ее параметру использование *полиморфизма высокого порядка* (*higher-kinded polymorphism*) по самому типу эффекта вкупе с *ограниченным полиморфизмом* в каждой отдельной функции. По историческим причинам (и из-за истоков в теории категорий) этот подход получил название *tagless-final* стиль [17][18][19]. В одном частном, но широко распространенном случае подход называется «`F` с дыркой», что подчеркивает использование полиморфизма высокого порядка по эффекту.

В этом подходе функции, обладающие тем или иным эффектов и предназначаемые для композируемости, объявляются полиморфными по типу эффекта (то есть по типу высшего порядка, обычно обозначаемого `F`), возвращающими значение `F` с конкретным типом. При этом для описания того, какие именно эффекты могут быть произведены этой функцией, используется *ограниченный полиморфизм*, то есть полиморфизм, который требует конструктивных доказательств того, что `F`, по которому осуществляется полиморфизм, обладает достаточными возможностями обеспечивать заданный эффект.

Этот подход имеет лишь то принципиальное отличие от подхода с монадой `Eff`, что в общем случае для типа `F` не обязательно требуется, чтобы он являлся монадой. Если функции с эффектом `F` не требуется именно монадическая сущность эффекта, она не обязана требовать это. С другой стороны, если функции требуется, чтобы эффект `F` был монадой (например, сама функция является монадической композицией других функций), она должна потребовать монадичность `F` явно (в отличие от монады `Eff`, которая является монадой сама по себе). Это может иметь свои преимущества и позволяет функции четче декларировать, что именно ей требуется (ограниченный полиморфизм эксплуатируется не только для типов эффекта, но и для способа композиции).

2.4.5 Другие альтернативы

Стоит упомянуть язык Unison, функциональный язык в разработке, который использует наслоение функторов и монад на уровне языка, а не на уровне библиотек (как это делают Haskell и Scala) [20]. Фактически, в языке переопределена композиция функций, которая может быть реализована как композиция аппликативных функторов или монад (в зависимости от типа выражения). Из-за этого улучшается читаемость кода и, возможно, скорость компиляции. Однако, по всей видимости, принципиальные возможности

остаются на том же уровне. Поэтому в дальнейшем мы будем рассматривать реализацию наслоения монад при помощи библиотек к широко используемым языкам (в частности, Scala).

Альтернативным вариантом решения проблемы композируемости монад является использование *вместо* монад для композиции вычислений с эффектами абстракции, являющейся чуть менее мощной, чем монада, но при этом все еще применимой и при этом более легко композируемой. Эта абстракция получила название *стрелки* (*arrow*) [21][22]; она является более выразительно мощной, чем аппликативный функтор, но менее мощной, чем монада [23]. Чисто функциональное программирование с эффектами не в монадическом, а в стрелочном стиле требует несколько необычного (и порой непривычного) способа выражения программы, однако позволяет комбинировать различные эффекты легко (для тех эффектов, к которым применимо понятие стрелки) [24][25][26].

3. Монадический подход к моделированию

3.1 Базовая идея

Начнем с представления модельного времени при помощи использования монад. Основной идеей “монадического” подхода к работе с вычислениями в модельном времени состоит в рассмотрении использования модельного времени как эффекта (в широком смысле).

В простейшем случае мы можем представлять функцию, моделирующую вычисление, которое требует определенное модельное время, как функцию, возвращающую вычисленное значение вместе с величиной требуемого времени.

Для *композируемости* такого рода вычислений, представим эффект вычисления вместе с тратой модельного времени как монаду; будем называть ее `TimedValue`.

Например, рассмотрим простейшую функцию сложения двух чисел.

```
def sumTimed(x: Int, y: Int): TimedValue[Int] =
  (x + y) |: 2.microseconds
```

Данная функция описывает вычисление, состоящее из сложения двух чисел, которое (исходя из определения этой функции) занимает две микросекунды модельного времени на вычисление. Здесь оператор `|:` – синтаксически удобный способ создания значений монады `TimedValue`.

Эта монада похожа на монаду `Writer`, но она не имеет типового параметра накапливаемого значения (тут он зафиксирован, это тип `Time`). Как следствие, эта монада не требует дополнительных ограничений на способ композиции для времени. Для последовательной композиции вычислений времена вычислений складываются.

Таким образом, имея, например, две функции

```
  val inc: Int => TimedValue[Long]
  val sqrt: Long => TimedValue[Double]
```

мы можем скомпозировать их в одну функцию, занимающую сумму модельного времени обеих функций:

```
inc >=> sqrt: Int => TimedValue[Double]
```

3.2 Последовательная и параллельная композиция

Функция сложения выбрана именно для *последовательной композиции* вычислений в модельном времени, подходящая для использования в монаде, то есть для описания зависимых вычислений.

В общем случае, может использоваться произвольная *ассоциативная бинарная* функция композиции двух времен. Выбор этой функции в большой степени определяет семантику комбинации монадических значений `TimedValue`.

Например, эта функция могла бы добавлять определенную величину времени для моделирования задержек на собственно вызов функций (при необходимости и очень точном моделировании затрачиваемого времени). Однако, в данной работе мы не будем рассматривать такой случай.

В качестве другого (весьма важного) примера альтернативной функции можно выбрать функцию максимума. При таком выборе, получится параллельная (с точки зрения модельного времени) комбинация вычислений `TimedValue`. Однако, с такой композицией может возникнуть проблема *физического смысла* результата. В частности, не всегда доступна параллельная композиция вычислений, занимающих (модельное) время, из-за нехватки (модельных) ресурсов даже в случае независимости этих вычислений по данным.

Для гранулированного подхода используются различные абстракции (например, `Parallel` [27]), которые позволяют не давать композировать функции, тратящие модельное время параллельно в случае нехватки модельных ресурсов для этого. Это достигается за счет сложного параметрического полиморфизма, фактически эквивалентного по выразительной мощности системе Хорновских дизъюнктов [28].

3.3 Монады времени в стеке монад

Как и другие монады, монаду вычислений с модельным временем можно использовать в стеке с другими монадами.

Это позволяет использовать один и тот же монадический код в различных ситуациях без переписывания и без изменения семантики с сохранением типобезопасности. При этом, различная семантика итоговой работы функций достигается за счет различных комбинаций функций-интерпретаторов при использовании этого монадического кода.

Правда, такой подход накладывает определенные требования на оформление монадического кода. Так, если до этого функции `A => B`, которые

моделируют вычисление, затрачивающее время, мы представляли как `A => TimedValue[B]`, для того, чтобы использовать такие функции в наслоении с другими монадами и эффектами, нам придется представлять из как полиморфные функции по типу эффекта с ограничениями на этот тип эффекта.

Например, функция вычисления квадратного корня вместо того, чтобы быть объявленной как

```
def sqrt(x: Int): TimedValue[Double] =
  math.sqrt(x) |: 1.millisecond
```

должна быть объявлена *полиморфной по типу эффекта*. В стиле *`F` с дыркой* это будет выглядеть как

```
def sqrt[F[_]: TimedValueAlgebra](x: Int): F[Double] =
  math.sqrt(x) ||: 1.millisecond
```

В стиле монады `Eff`, эта же функция выглядела бы как

```
def sqrt[R: _timedValue](x: Int): Eff[R, Double] =
  math.sqrt(x) ||: 1.millisecond
```

или, используя «стрелочный» синтаксис вместо указания типа `Eff`,

```
def sqrt[R: _timedValue](x: Int): R ||> Double =
  math.sqrt(x) ||: 1.millisecond
```

Принципиально, в рамках данной работы, оба способа полиморфного задания эквивалентны. Поэтому, для простоты, в этом разделе будем рассматривать только второй способ изложения (при помощи монады `Eff`).

В данном примере типовой параметр `R` отвечает за произвольный эффект, который ограничен тем, что в нем обязан присутствовать эффект `_timedValue`. При этом, `_timedValue` технически объявлен как `TimedValue |= ?`, то есть означает, что эффект, моделируемый монадой `TimedValue` присутствует в стеке монад. Синтаксис оператора `||:` позволяет удобно создавать значения соответствующего типа (то есть, `Eff` с произвольным эффектом `R`, который, в свою очередь, ограничен наличием `TimedValue` в стеке монад).

Перечислим виды значений, с которыми приходится работать в задачах моделирования.

- **Единичное значение**

  Положим, у нас есть простое целочисленное значение, запакованное в произвольный стек монад (то есть, нет никаких ограничений на R):

  ```
  def simpleVal[R]: R ||> Int
  ```

  Мы можем применить функцию вычисления квадратного корня к этому значению:

  ```
  simpleVal >>= sqrt
  ```

  Результатом будет значение типа `R ||> Double`, где на `R` будет наложено ограничение наличия `TimedValue` в стеке монад. Мы можем

запустить вычисление этого выражения и получим значение типа `Double` на выходе:

```
type S1 = Fx.fx1[TimedValue]
(simpleVal[S1] >>= sqrt[S1]).runTimed.run
// результат типа TimedValue[Double]
```

- **Множественное значение**

Представим себе, что на входе есть множественное значение, для которого мы хотим выполнить наше timed-вычисление. Такое множественное значение может быть промоделировано следующим образом:

```
def severalVals[R: _list]: R ||> Int
```

Разница с рассмотренным выше `simpleVal` состоит в том, что эффект `R` содержит требование множественности `_list` (которое раскрывается в `List |= ?`, наподобие тому, как `_timedValue` раскрывается в `TimedValue |= ?`).

Мы можем таким же образом скомпозировать это множественное значение с функцией вычисления квадратного корня (напомним, что эта функция тратит модельное время):

```
severalVals >>= sqrt
```

Мы так же, как и в предыдущий раз, можем запустить вычисление этого выражения. Однако, до момента запуска результирующее значение имеет неоднозначность порядка запуска. Компилятор не знает хотим ли мы получить результат вычисления списка, занимающий модельное время, или же список результатов, каждый занимающий свое модельное время на вычисление. Это определяется типом результата: в первом случае от `TimedValue[List[Double]]`, во втором он `List[TimedValue[Double]]`. Мощность подхода с использованием стека монад состоит в том, что мы можем получить оба варианта результатов, и при этом отдельные участки кода (в данном случае, функции `severalVals` и `sqrt`) будут одинаковыми в обоих случаях, то есть собственно описание алгоритма вычислений полностью переиспользуется:

```
type S2 = Fx.fx2[TimedValue, List]

(severalVals[S2]>>= sqrt[S2]).runList.runTimed.run
// результат типа TimedValue[List[Double]]

(severalVals[S2] >>= sqrt[S2]).runTimed.runList.run
// результат типа List[TimedValue[Double]]
```

Разница есть только в порядке вызова интерпретаторов runList и runTimed.

- **Вероятностное значение**

  Аналогичным образом мы можем переиспользовать функцию sqrt для работы с вероятностным значением. Имея variadicVal, возвращающее конечное дискретное распределение целых чисел, мы можем запустить следующий код и получить дискретное распределение чисел Double, занимающих модельное время:

```
type S3 =
  Fx.fx2[TimedValue, DiscreteFiniteDistribution]

(variadicVal[S3] >>= sqrt[S3]).runDFD.runTimed.run
// результат типа
//   DiscreteFiniteDistribution[TimedValue[Double]]
```

- **Более сложные смешения**

  В общем случае мы можем использовать стек произвольной глубины. В качестве примера, мы можем рассмотреть ситуацию, когда timed-вычисление зависит от источников разных монадических функций. Для простоты, мы будем использовать описанные выше функции `severalVals` и `variadicVal`, а в timed-функцию `sqrt` будем передавать сумму от тех двух функций.

```
def f[R: _timedValue:_list:_dFD]: R ||> Double = for {
  v1 <- severalVals
  v2 <- variadicVal
  s <- sqrt(v1 + v2)
} yield s
```

  В общем случае существует шесть вариантов выполнения этой функции и получения конкретного значения. Все эти варианты будут давать результат различных типов (и, как следствие, вычисление будет иметь различную семантику). При этом для получения этих значений будет использоваться один и тот же код функции `f`.

  Рассмотрим несколько примеров таких вычислений. В первом случае мы будем получать список вероятностных распределений timed-величин типа `Double`. Во втором случае мы будем получать вероятностное распределение timed-величин типа `List[Double]`.

```
type S4 =
  Fx.fx3[TimedValue, List, DiscreteFiniteDistribution]

f[S4].runTimed.runDFD.runList.run
// результат типа List[
//   DiscreteFiniteDistribution[TimedValue[Double]]]

f[S4].runList.runTimed.runDFD.run
// результат типа DiscreteFiniteDistribution[
//   TimedValue[List[Double]]]
```

Таким образом, мы можем использовать вычисления, описывающие вычисления, занимающие модельное время, вместе с другого рода эффектами.

При этом легко осуществляется разделение описания действий и выполнения этих действий. Вместе с этим, способ выполнения определяет конечную семантику всей операции, которая задается упорядочиванием эффектов из стека монад. Это позволяет переиспользовать один и тот же код поведений в различных ситуациях и дает возможность не принимать определенных проектных решений (в частности, по упорядочиванию эффектов) раньше времени.

3.4 Обобщения затрачиваемого времени

Значение типа `TimedValue` содержит единственное значение времени, обозначающее длительность – модельное время, затрачиваемое для вычисления хранимого значения. В общем же случае может потребоваться иметь не простое значение длительности, а более сложные варианты. В частности, рассмотрим следующие примеры таких потребностей.

- **Косвенное время**

  При описании вычисления не всегда известно собственно затрачиваемое время на выполнение. Время может зависеть от используемого процессора, затрат на коммуникацию, работы кэша и пр. При детальном моделировании времени, это может существенно сказаться на результате вычислений.

  В частности, вместо указания значений затрачиваемого модельного времени, может быть потребность в указании затрачиваемых модельных тиков процессора или в указании числа различных инструкций определенного вида, количества обращений к памяти и подобном.

  То есть, в конечном итоге, время указывается *косвенно*, в терминах процессора или аппаратуры.

  Можно вводить специализированные аналоги монады `TimedValue` для такого рода косвенного задания времени. Однако могут возникнуть проблемы, связанные с возможной потребностью в разнообразии такого рода монад.

  Также возникают проблемы с композицией этих монад с монадой `TimedValue`, так как для корректной композиции они должны знать конкретные параметры используемых процессоров. Это может приводить к необходимости изменения кода, описывающего вычисления при внешних по отношению к нему изменениях (смене типа процессора).

- **Не единственное значение времени**

  Не всегда может быть известно точное (пусть даже и модельное) время выполнения тех или иных функций. Единственное значение времени, использующееся в монаде `TimedValue` можно, конечно, интерпретировать как ограничение сверху. Правда, в таком случае мы

лишаемся возможности оценивать затрачиваемое модельное время снизу, что может быть нужным иногда [29].

Проблемы с нижней границей можно пробовать решать введением интервала времен вместо отдельного времени в `TimedValue`. Соответственным образом нужно будет изменить правила композиции монады (со сложением интервалов вместо сложения отдельных величин и соответствующей композицией интервалов при параллельной композиции).

Но интервал может стать лишь грубым приближением действительно необходимого значения времени. Аналогично различным входным параметрам вычислений при построении стека монадических вычислений, может потребоваться выражать, например, конкретный список отдельных времен (а не непрерывный интервал) или даже вероятностное распределение затрачиваемых времен.

Если обобщить эти мысли, получится, что монада `TimedValue` должна содержать не *отдельное единичное значение времени*, а *обобщенное значение, содержащее время*.

```
case class TimedValue[A, F[_]](a: A, time: F[Time])
```

При этом, например, для композиции может требоваться, чтобы данный `F[_]` был, например, функтором. Для параллельной композиции, соответственно, может требоваться, чтобы `F[_]` был бы аппликативным функтором. Аналогично, для последовательной композиции может быть аналогичное требование для монадичности этого параметра типа. Могут существовать и другие дополнительные ограничения на `F[_]`.

Подобные ограничения на `F[_]` могут быть отнесены на как можно более поздний момент по коду (то есть, в функциях, которые действительно используют эти свойства (функторности, аппликативности, монадичности или другие)).

Можно рассматривать также обобщение (с одной стороны усиливающее, с другой ограничивающее), когда хранимое время – это не обернутое значение, а произвольный стек монад, в конечном счете продуцирующий значение времени (в соответствии с семантикой стека).

```
case class TimedValue[A, R](a: A, time: R ||> Time)
```

Более того, это обобщение позволяет естественным образом описывать случай, когда время задано косвенно (например, через затрачиваемые тики процессора и пр.). В таком случае эти данные должны позволять получить конечное время через промежуточные монады, которые в конечном итоге показываются в стеке монад.

```
for {
  a <- timedVal ||: (1 to 5).milliseconds
  b <- tickedVal(a) ||: (100 to 200).processorTicks
  c <- instVal ||: (500.arithOps + 20.controlOps)
```

```
} yield b + c
```

Стоит отметить, что даже при использовании непрямого представления времени стеки монад позволяют задавать интервалы, множественности или даже вероятностные распределения затрачиваемого времени через тики процессора и другие способы. При этом при смене типа процессора и других характеристик (и, как следствие, возможного изменения затрачиваемого модельного времени), код, описывающий вычисления, не будет меняться.

3.5 Контроль времени на уровне типов

Рассмотрим другой вариант изменения базовой идеи. Мощная система типов позволяет более точно контролировать затрачиваемое модельное время и соответствие имеющегося и требуемого модельного времени.

В частности, есть возможность указать, что определенный исполнитель timed-функций может принять как аргумент только функции, затрачивающие не более чем заданный объем времени. Это может быть полезно для статической проверки в случае использования со строго-периодическими операционными системами, которые находят широчайшее применение в ответственных системах реального времени.

Само затрачиваемое модельное время можно присоединить к выражению при помощи литеральных типов. Так, например, представим себе функцию, возвращающую значение, которое будет иметь тип, соответствующий результату, вычисляемому за 10 миллисекунд модельного времени. Если мы попытаемся передать это выражение в функцию, которая может выполнить выражения, занимающее не более, например, восьми миллисекунд модельного времени, мы получим ошибку времени компиляции.

Для того, чтобы это сделать, нужно, чтобы тип, соответствующий затрачиваемому модельному времени, был приписан к timed-выражению. Это можно сделать как минимум двумя способами.

- **Типовой параметр**

  Одним из способов является добавление типово́го параметра в монадический тип, который описывает выражение, вычисляемое за модельное время, то есть в тип `TimedValue`.

  Таким образом, этот тип начинает иметь сигнатуру не `TimedValue[A]`, а `TimedValue[A, TimeBound]`. Соответственно, все операции над этим типом должны в своих сигнатурах учитывать этот типовой параметр.

  Это, на самом деле, представляет собой некую проблему для некоторых методов, потому что они перестают отвечать своим абстракциям. Например, таким является функция `flatMap`, определяющая монадическую композицию. Это ограничивает применимость такого метода.

- **Тип-поле**

  Другой способ состоит в том, чтобы привязать typelevel-значение к выражению со временем – это *зависимые типы*. Имеется несколько ограниченный вариант, который называется *path-dependent type* [30]. Подход представляет собой добавление поля в структуру данных `TimedValue`, которое является *типом*, а не значением.

Реализовывать само ограничение по времени можно тоже двумя способами. В обоих случаях типовым параметром является *конструктивное доказательство* (или *свидетельство*, *evidence*) того, что содержимое значения времени является ограниченным (например, не превышает заданную величину).

4. Заключение

Монадический подход к моделированию времени фактически является применением хорошо известных техник функционального программирования к области архитектурного моделирования, в частности, к области моделирования поведения сложных компьютерных систем.

В частности, строгая типизированность и широкие возможности мощной системы типов позволяют выражать многие аспекты модели системы напрямую в коде, которые будут проверены на стадии компиляции кода, сгенерированного на основе модели системы. Это позволяет обеспечивать ранние проверки композируемости и соответствия отдельных поведений при изменениях модели и отдельных спецификаций поведения компонентов модели. Эти проверки производятся статически, до запуска моделирования всей системы.

Появляются также широкие возможности по композиции разнородных поведений или их аспектов за счет использования ограниченного параметрического полиморфизма функций. За счет использования полиморфизма высокого порядка по эффектам появляется возможность наслаивания дополнительных эффектов к имеющемуся коду без его переписывания (и даже без перекомпиляции). Это позволяет расширять и уточнять семантику отдельных поведений и их композиций без потери корректности.

Помимо прочего, данный подход обладает другими преимуществами функционального программирования, в том числе, возможностью использования повышенного уровня абстракции в коде (как следствие, увеличения скорости разработки и простоты понимания), широкими возможностями автоматической оптимизации кода во время компиляции и автоматического распараллеливания во время выполнения, а также многими другими возможностями. Конечно, при этом стоит учитывать непривычность такого рода подхода для многих профессионалов в прикладной области, что

необходимо иметь в виду в будущем и что не относится напрямую к идеям, изложенным в этой работе.

В рамках проекта РФФИ 17-01-00504 в 2019 году планируется разработать экспериментальный комплекс инструментальных средств для моделирования систем управления. Комплекс будет строиться на платформе Scala, имеющей все необходимые для этого средства функционального программирования.

Совместное использование механизмов функционального программирования дает возможность описывать вычисления с минимальными требованиями на инфраструктуру такие, что они гарантированно корректно композируются с другими вычислениями в произвольных подходящих инфраструктурах (при этом их невозможно использовать в неподходящих инфраструктурах, потому что проверки осуществляются на уровне компилятора). Это позволяет переиспользовать (даже без перекомпиляции) имеющийся код не только для решения конкретной задачи, для которой этот код был разработан, но и для решения задач расширяющего класса.

Как следствие, наращивание набора инструментальных средств и видов анализа в перспективной среде моделирования будет требовать лишь добавления или изменения кода, ответственного за конкретные частные аспекты (в частности, специфичные для задачи виды входных данных и анализ специфичных результатов) и не будет затрагивать имеющийся (полиморфный) код. При этом использование мощных механизмов функционального программирования требует определенной культуры и следования определенному стилю программирования, что в контексте разработки и аттестации систем ответственного назначения является уместной платой за получение преимуществ в плане повышения доли повторно используемых компонентов (re-use), композируемости и достоверной консистентности системы.

Список литературы

[1] Denis Buzdalov, Alexey Khoroshilov. A Discrete-Event Simulator for Early Validation of Avionics Systems. In Proc. of the First International Workshop on Architecture Centric Virtual Integration co-located with the 17th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2014), 2014, CEUR Workshop Proceedings, vol. 1233,

[2] Denis Buzdalov. Simulation of AADL models with software-in-the-loop execution. ACM SIGAda Ada Letters, vol. 36, issue 2, December 2016, pp. 49-53

[3] John Hughes. Why functional programming matters. PMG-40, Chalmers University of Technology, Goteborg, Sweden, 1984. Режим доступа: http://www.cse.chalmers.se/~rjmh/Papers/whyfp.pdf, дата обращения 18.12.2018

[4] Simon Peyton Jones. Functional programming languages as a software engineering tool. In Embedded Systems. Lecture Notes in Computer Science, vol 284, 1986, pp 153-173

[5] Simon Peyton Jones, Philip Wadler. Imperative functional programming. In Proc. of the 20th ACM SIGPLAN-SIGACT Symposium on Principles of programming languages, 1993, pp. 71-84.

[6] The IO Monad for Scala. Режим доступа: https://typelevel.org/cats-effect/, дата обращения 18.12.2018
[7] Monix library, Task monad. Режим доступа: https://monix.io/docs/3x/eval/task.html https://typelevel.org/cats-effect/, дата обращения 18.12.2018
[8] ZIO, Scala-library for asynchronous and concurrent programming, IO monad. Режим доступа: https://scalaz.github.io/scalaz-zio/datatypes/io.html, дата обращения 18.12.2018
[9] Eugenio Moggi. Notions of computation and monads. Information and Computation, vol. 93, issue 1, July 1991, pp. 55-92
[10] Philip Wadler. Comprehending Monads. Mathematical Structures in Computer Science, vol 2, issue 4, Cambridge University Press, 1992, pp. 461–493
[11] Conor McBride, Ross Paterson. Applicative programming with effects. Journal of Functional Programming, vol. 18, issue 1, 2008, pp. 1-13
[12] Oleg Kiselyov. Having an Effect. Presentation at the Seminar at the Institute of Information Science, Academia Sinica, Taipei, Taiwan, December 2, 2016. Режим доступа: http://okmij.org/ftp/Computation/having-effect.html, дата обращения 18.12.2018
[13] Guy L. Steele Jr. Building interpreters by composing monads. In Proc. of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of programming languages (POPL '94), 1994, pp. 472–492
[14] Sheng Liang, Paul Hudak, Mark Jonest. Monad Transformers and Modular Interpreters. In Proc. of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '95), 1995, pp. 333-343
[15] Oleg Kiselyov, Amr Sabry, Cameron Swords. Extensible Effects: An Alternative to Monad Transformers. In Proc. of the 2013 ACM SIGPLAN Symposium on Haskell, 2013, pp. 59-70
[16] Oleg Kiselyov, Hiromi Ishii. Freer Monads, More Extensible Effects. In Proc. of the 2015 ACM SIGPLAN symposium on Haskell, 2015, pp. 94-105
[17] Oleg Kiselyov. Typed Tagless Final Interpreters. Lecture Notes in Computer Science, vol 7470, 2012, pp. 130-174
[18] Noel Welsh. Uniting Church and State: FP and OO Together. Scala Days conference, Copenhagen, 2017. Режим доступа: https://www.youtube.com/watch?v=IO5MD62dQbI, дата обращения 18.12.2018
[19] Олег Нижников. Современное ФП с Tagless Final. Конференция Joker 2018, Санкт-Петербург. Режим доступа: https://www.youtube.com/watch?v=sWEtnq0ReZA, дата обращения 18.12.2018
[20] Rúnar Bjarnason. Introduction to the Unison programming language. Lambda World 2018 conference, Living Computer Museum, Seattle, 18 september 2018. Режим доступа: https://www.youtube.com/watch?v=rp_Eild1aq8, дата обращения 18.12.2018
[21] John Hughes.Generalising monads to arrows. Science of Computer Programming, vol. 37, issues 1–3, May 2000, pp. 67–111
[22] John Hughes. Programming with Arrows. In Advanced Functional Programming, 2004, pp. 73-129
[23] Sam Lindley, Philip Wadler, Jeremy Yallop. Idioms are Oblivious, Arrows are Meticulous, Monads are Promiscuous. Electronic Notes in Theoretical Computer Science, vol. 229, issue 5, 2011, pp. 97–117

[24] Yuriy Polyulya, Functional programming with arrows. Scala Days conference, 2015, Amsterdam. Режим доступа: https://www.youtube.com/watch?v=ZfAgvAIoUEY, дата обращения 18.12.2018
[25] Julien Richard Foy, Do it with (free?) arrows! Typelevel Summit Copenhagen, June 2017. Режим доступа: https://www.youtube.com/watch?v=PWBTOhMemxQ, дата обращения 18.12.2018
[26] John A De Goes. Blazing Fast, Pure Effects without Monads. LambdaConf conference, Boulder, CO, USA, June 2018. Режим доступа: https://www.youtube.com/watch?v=L8AEj6IRNEE, дата обращения 18.12.2018
[27] Parallel typeclass, cats library. Режим доступа: https://typelevel.org/cats/typeclasses/parallel.html, дата обращения 18.12.2018
[28] George Leontiev, There's a prolog in your scala. ScalaIO conference, Paris, France, October 2014. Режим доступа: https://www.youtube.com/watch?v=iYCR2wzfdUs, дата обращения 18.12.2018
[29] A.M. Troitskiy, D.V. Buzdalov. A static approach to estimation of execution time of components in AADL models. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 2, 2016, pp. 157-172, doi: 10.15514/ISPRAS-2016-28(2)-10
[30] Nada Amin, Tiark Rompf, Martin Odersky. Foundations of Path-Dependent Types. In Proc. of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications, 2014, pp. 233-249

On representation of simulation time in functional programming style

[1] D.V. Buzdalov <buzdalov@ispras.ru>
[1, 3, 4] A.K. Petrenko <petrenko@ispras.ru>
[1, 2, 3, 4] A.V. Khoroshilov <khoroshilov@ispras.ru>

[1] V.P.Ivannikov Institute for system programming, Russian Academy of Sciences, 109004, Russia, Moscow, Solzhenitsyn, d. 25
[2] Moscow Institute of physics and technology (State University), 141700, Moscow region, Dolgoprudny, Institutskiy per., 9
[3] Moscow State University named after M. V. Lomonosov, Moscow, 119991, GSP-1, Lenin hills, d. 1
[4] National research University "Higher school of economics", 101000, Russia, Moscow, Myasnitskaya, d. 20

Abstract. Functional programming plays the big role in the modern computer science and its importance is growing. This is not accidential: this approach helps to create better and more reliable software that is easy to reason about (both manually and automatically). However, these techniques are hardly used in the field of tools helping designing and modeling mission-critical systems. In this paper, we are trying to apply some nice techniques of functional programming to create a modeling system, in particular a simulation system for analysis of temporal behavioural properties of mission-critical systems. As a first step, we designed a representation of simulation time in terms of abstractions used in functional programming and tried to study its compositionability properties.

Keywords: architecture modeling, mission-critical systems, behavioural modeling, simulation time, functional programming, monads.

DOI: 10.15514/ISPRAS-2018-30(6)-20

For citation: Buzdalov D.V., Petrenko A.K., Khoroshilov A.V. On representation of simulation time for functional programming. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 341-366 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-20

References

[1] Denis Buzdalov, Alexey Khoroshilov. A Discrete-Event Simulator for Early Validation of Avionics Systems. In Proc. of the First International Workshop on Architecture Centric Virtual Integration co-located with the 17th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2014), 2014, CEUR Workshop Proceedings, vol. 1233,

[2] Denis Buzdalov. Simulation of AADL models with software-in-the-loop execution. ACM SIGAda Ada Letters, vol. 36, issue 2, December 2016, pp. 49-53

[3] John Hughes. Why functional programming matters. PMG-40, Chalmers University of Technology, Goteborg, Sweden, 1984. Available at: http://www.cse.chalmers.se/~rjmh/Papers/whyfp.pdf, accessed 18.12.2018

[4] Simon Peyton Jones. Functional programming languages as a software engineering tool. In Embedded Systems. Lecture Notes in Computer Science, vol 284, 1986, pp 153-173

[5] Simon Peyton Jones, Philip Wadler. Imperative functional programming. In Proc. of the 20th ACM SIGPLAN-SIGACT Symposium on Principles of programming languages, 1993, pp. 71-84.

[6] The IO Monad for Scala. Available at: https://typelevel.org/cats-effect/, accessed 18.12.2018

[7] Monix library, Task monad. Available at: https://monix.io/docs/3x/eval/task.html https://typelevel.org/cats-effect/, accessed 18.12.2018

[8] ZIO, Scala-library for asynchronous and concurrent programming, IO monad. Available at: https://scalaz.github.io/scalaz-zio/datatypes/io.html, accessed 18.12.2018

[9] Eugenio Moggi. Notions of computation and monads. Information and Computation, vol. 93, issue 1, July 1991, pp. 55-92

[10] Philip Wadler. Comprehending Monads. Mathematical Structures in Computer Science, vol 2, issue 4, Cambridge University Press, 1992, pp. 461–493

[11] Conor McBride, Ross Paterson. Applicative programming with effects. Journal of Functional Programming, vol. 18, issue 1, 2008, pp. 1-13

[12] Oleg Kiselyov. Having an Effect. Presentation at the Seminar at the Institute of Information Science, Academia Sinica, Taipei, Taiwan, December 2, 2016. Available at: http://okmij.org/ftp/Computation/having-effect.html, accessed 18.12.2018

[13] Guy L. Steele Jr. Building interpreters by composing monads. In Proc. of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of programming languages (POPL ’94), 1994, pp. 472–492

[14] Sheng Liang, Paul Hudak, Mark Jonest. Monad Transformers and Modular Interpreters. In Proc. of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL ’95), 1995, pp. 333-343

[15] Oleg Kiselyov, Amr Sabry, Cameron Swords. Extensible Effects: An Alternative to Monad Transformers. In Proc. of the 2013 ACM SIGPLAN Symposium on Haskell, 2013, pp. 59-70
[16] Oleg Kiselyov, Hiromi Ishii. Freer Monads, More Extensible Effects. In Proc. of the 2015 ACM SIGPLAN symposium on Haskell, 2015, pp. 94-105
[17] Oleg Kiselyov. Typed Tagless Final Interpreters. Lecture Notes in Computer Science, vol 7470, 2012, pp. 130-174
[18] Noel Welsh. Uniting Church and State: FP and OO Together. Scala Days conference, Copenhagen, 2017. Available at: https://www.youtube.com/watch?v=IO5MD62dQbI, accessed 18.12.2018
[19] Oleg Nizhnikov. Modern functional programming with Tagless Final. Joker 2018 Conference, Saint-Petersburg (in Russian). Available at: https://www.youtube.com/watch?v=sWEtnq0ReZA, accessed 18.12.2018
[20] Rúnar Bjarnason. Introduction to the Unison programming language. Lambda World 2018 conference, Living Computer Museum, Seattle, 18 september 2018. Available at: https://www.youtube.com/watch?v=rp_Eild1aq8, accessed 18.12.2018
[21] John Hughes.Generalising monads to arrows. Science of Computer Programming, vol. 37, issues 1–3, May 2000, pp. 67–111
[22] John Hughes. Programming with Arrows. In Advanced Functional Programming, 2004, pp. 73-129
[23] Sam Lindley, Philip Wadler, Jeremy Yallop. Idioms are Oblivious, Arrows are Meticulous, Monads are Promiscuous. Electronic Notes in Theoretical Computer Science, vol. 229, issue 5, 2011, pp. 97–117
[24] Yuriy Polyulya, Functional programming with arrows. Scala Days conference, 2015, Amsterdam. Available at: https://www.youtube.com/watch?v=ZfAgvAIoUEY, accessed18.12.2018
[25] Julien Richard Foy, Do it with (free?) arrows! Typelevel Summit Copenhagen, June 2017. Available at: https://www.youtube.com/watch?v=PWBTOhMemxQ, accessed 18.12.2018
[26] John A De Goes. Blazing Fast, Pure Effects without Monads. LambdaConf conference, Boulder, CO, USA, June 2018. Available at: https://www.youtube.com/watch?v=L8AEj6IRNEE, accessed 18.12.2018
[27] Parallel typeclass, cats library. Available at: https://typelevel.org/cats/typeclasses/parallel.html, accessed 18.12.2018
[28] George Leontiev, There's a prolog in your scala. ScalaIO conference, Paris, France, October 2014. Available at: https://www.youtube.com/watch?v=iYCR2wzfdUs, accessed18.12.2018
[29] A.M. Troitskiy, D.V. Buzdalov. A static approach to estimation of execution time of components in AADL models. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 2, 2016, pp. 157-172, doi: 10.15514/ISPRAS-2016-28(2)-10
[30] Nada Amin, Tiark Rompf, Martin Odersky. Foundations of Path-Dependent Types. In Proc. of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications, 2014, pp. 233-249

Компонентная верификация операционных систем[1]

[1,2,3] *В.В. Кулямин <kuliamin@ispras.ru>*
[1,2,3] *А.К. Петренко <petrenko@ispras.ru>*
[1,2,3,4] *А.В. Хорошилов <khoroshilov@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*
[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1.*
[3] *НИУ Высшая школа экономики, 101000, Россия, Москва, ул. Мясницкая, д. 20.*
[4] *Московский физико-технический институт (гос. университет), 141700, Россия, Московская обл., г. Долгопрудный, Институтский пер., д. 9.*

Аннотация. В работе рассматриваются полученные недавно результаты на пути к полномасштабной верификации промышленно используемых операционных систем (ОС). Таковыми считаются не системы, разработанные в целях демонстрации определенной исследовательской идеи, а ОС, активно используемые в каких-то областях экономики и управленческой деятельности и развиваемые на протяжении значительного времени. Предлагается декомпозиция заявленной цели верификации промышленной ОС в целом на задачи верификации различных компонентов ОС и их различных свойств, методы решения которых в дальнейшем можно интегрировать в метод достижения общей цели. На данном этапе пока рассматриваются различные подходы к решению выделенных отдельных задач. Статья является экспликацией опыта верификации различных компонентов нескольких ОС и интеграции используемых для этого технологий, полученного в рамках проектов, проводимых в ИСП РАН.

Ключевые слова: операционная система; верификация; тестирование; статический анализ; дедуктивная верификация; мониторинг

DOI: 10.15514/ISPRAS-2018-30(6)-21

Для цитирования: Кулямин В.В., Петренко А.К., Хорошилов А.В. Компонентная верификация операционных систем. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 367-382. DOI: 10.15514/ISPRAS-2018-30(6)-21

[1] Работа поддержана грантом Российского фонда фундаментальных исследований № 18-01-00378

1. Введение

Современные промышленные операционные системы (ОС), используемые для запуска множества разнообразных приложений, очень сложны. Они не только основаны на исходном коде очень большого объема (миллионы и десятки миллионов строк), но также вынуждены иметь огромное число разнородных функциональных возможностей, должны работать на разнообразном аппаратном обеспечении и с большим количеством устройств от совершенно не связанных друг с другом производителей. ОС также предоставляют разработчикам приложений множество различных, постепенно стандартизуемых интерфейсов, которые должны не только корректно работать в огромном разнообразии ситуаций, но и эффективно использовать устройства и аппаратные возможности компьютеров, продолжая надежную работу даже в случае сбоев. В данной работе мы рассматриваем ОС, активно используемые в какой-то отрасли экономики или же ОС общего назначения, поддерживаемые и развиваемые в течении достаточно долгого времени (от пяти лет).

ОС как таковая выполняет две основные задачи.

- Организует выполнение набора приложений на определенном аппаратном обеспечении и множестве устройств так, чтобы приложения, разделяя аппаратные ресурсы, не мешали работе друг друга.
- Предоставляет разработчикам приложений интерфейсы для эффективного и удобного использования аппаратных ресурсов в таком режиме, а также для передачи данных и обеспечения взаимодействия приложений при необходимости.

Важнейшей частью ОС является ядро, работающее в привилегированном режиме процессора и поэтому имеющее неограниченных доступ ко всем ресурсам системы. Ядро управляет доступом приложений к аппаратным ресурсам, устанавливая правила такого доступа и предотвращая их нарушения. Некоторые функции ОС, вообще говоря, не требующие привилегированного режима работы, часто включаются в ядро для повышения производительности.

Приложения взаимодействуют с ядром через системные вызовы, обращения к интерфейсным операциям ядра, переключающим систему в привилегированный режим. Иногда также используются дополнительные способы взаимодействия с ядром, например, специальные виртуальные файловые системы в ОС Linux (*procfs, sysfs, debugfs*). Чтобы сделать удобное для работы разработчиков приложений окружение, ОС обычно предоставляет системные библиотеки и наборы системных утилит, реализующих наиболее часто используемые функции, требующие обращения к ядру. Для решения задач, требующих активности со стороны ядра, таких как работа телекоммуникационных протоколов, управление специализированными устройствами и пр., предоставляются системные службы, которые могут работ

как в привилегированном, так и в обычном пользовательском режиме. На рисунке 1 показана типовая структура ОС общего назначения.

Рис. 1. Типовая структура ОС общего назначения
Fig. 1. Typical structure of general purpose OS

Приведенный выше очень краткий обзор структуры ОС дает некоторое представление о ее сложности. Верификация промышленной ОС также является очень сложной задачей, достаточно упомянуть следующие аспекты.

- Многообразие функциональных возможностей современных ОС создает огромное количество сценариев взаимодействия отдельных функций и множество специфических экстремальных ситуаций, в которых необходим гораздо более тщательный, чем обычно, анализ требуемого поведения.
- Поддержка многозадачности в современных ОС делает проверку корректности поведения гораздо более запутанной.
- Основные функции ОС должны работать, несмотря на сбои в аппаратном и программном обеспечении (ПО); соответственно, необходима верификация устойчивости к сбоям, причем на многих уровнях.
- Современные ОС обычно поддерживают взаимодействие по сети и предоставляют рабочее место для многих пользователей. Это возможно лишь при выполнении некоторых политик защиты данных и задач, накладывающих ограничения на доступ к данным и приложениям и на

взаимодействие между ними. Такие ограничения также должны выполняться несмотря на сбои в ПО и злонамеренные атаки извне и со стороны самих пользователей.

- Поддержка разнородного аппаратного обеспечения и устройств обычно основана на широкой конфигурируемости ОС. Однако верификация всех возможных конфигураций не реализуема на практике, поскольку их число немыслимо огромно.
- Сами по себе объем исходного кода ОС и количество реализуемых им функций огромны. Размер исходного кода ядра Linux версии 4.1 составляет более 20 миллионов строк [1], в том числе около 11.5 миллионов строк приходится на код драйверов, которые разрабатываются и поддерживаются многочисленными сторонними разработчиками. Размер исходного кода Windows XP оценивается в 45 миллионов строк [2]. Общее число функций в системных библиотеках дистрибутива Debian 7.0 около 720 тысяч [3], хотя системных вызовов среди них всего около 350.

Приведенные числовые характеристики показывают, что аккуратная верификация промышленной операционной системы является на настоящий момент недостижимой на практике целью. Однако, определенные методы проверки корректности, эффективности, защищенности и отказоустойчивости современных ОС все равно необходимы. Единственным работоспособным подходом пока является использование всего разнообразия имеющихся методов верификации и анализа для проверки отдельных свойств отдельных компонентов или групп компонентов ОС, начиная с наиболее критичных. Результатами такого подхода обычно являются многочисленные выявляемые ошибки, что, хотя бы, позволяет разработчикам ОС исправлять их, повышая общее качество систем. Несмотря на невозможность сейчас гарантировать корректность, надежность и защищенность разрабатываемых ОС, развитие эффективности и масштабируемости применяемых методов, расширение верифицированной части кода систем и их свойств, позволят постепенно достичь определенных гарантий качества в целом, если не забывать об этой цели в рутине выполнения всех указанных работ.

В данной статье приведен краткий обзор различных подходов к верификации компонентов и свойств промышленных ОС, применяемых в проектах, проводившихся в ИСП РАН на протяжении последних 20 лет. Эти работы интегрируют разнообразные техники верификации, применяя их к компонентам ОС Linux и нескольких специализированных ОС реального времени. Основные используемые методы таковы.

- **Тестирование и динамический анализ**
  Наиболее широко используется тестирование, в виде различных по трудоемкости, обеспечиваемым гарантиям и объему проверяемых свойств методов. Применяемые методы тестирования работоспособности проверяют только основные ограничения в рамках базовых сценариев

работы библиотечных функций и опираются только на покрытие таких функций в качестве критерия полноты. Гораздо более аккуратно и трудоемко тестирование соответствия формальным спецификациям, в рамках которого проверяется строгое выполнение зафиксированных в стандартах требований как в различных сценариях нормального функционирования, так и при возникновении разнообразных исключительных ситуаций, и при различных сценариях взаимодействия разных функций. Вместе с тестированием применяется и верификационный мониторинг, нацеленный на проверку определенных свойств без необходимости разрабатывать специальные тесты.

- **Статический анализ.**
  Статический анализ используется также в виде различных методов. Легковесный статический анализ может обнаруживать лишь небольшое число типов ошибок, но делать это весьма эффективно. Более тяжеловесный, использующий формальные модели требований, позволяет находить весьма сложные виды ошибок, но обычно требует достаточно много времени и усилий на подготовку исходных данных, выполнение самого анализа и инспекцию полученных им результатов. Для верификации компонентов ОС мы чаще использовали более сложные, но и более строгие техники статического анализа.
- **Дедуктивная верификация.**
  Дедуктивную верификацию можно использовать на практике для проверки наиболее важных свойств корректности и защищенности. Известно несколько примеров успешной дедуктивной верификации ядра ОС [4-6], но во всех этих случаях размер верифицированного кода существенно меньше размера ядра типичной промышленной ОС. Тем не менее, дедуктивная верификация тоже может быть использована для получения значимых результатов при проверке промышленных ОС.

Ниже мы попытались систематизировать опыт нескольких десятков проектов ИСП РАН, в рамках которых использовались различные техники верификации для разнообразных компонентов операционных систем.

2. Тестирование и мониторинг

Технология тестирования компонентов ОС разрабатывается в ИСП РАН еще со времени его основания в 1994 г. Первая такая технология под названием KVEST [7] использовалась для автоматизированной разработки на основе формальных спецификаций программных контрактов тестовых наборов для ОС реального времени, разработанных и поддерживавшихся Nortel Networks.

2.1 Формальные подходы

Дальнейшее развитие этих методов легло в основу технологии UniTESK [8]. Ее основные элементы могут быть сформулированы следующим образом.

- Функциональные требования к поведению библиотечных функций формулируются в виде программных контрактов, состоящих их предусловий и постусловий функций и инвариантов используемых типов данных. Программные контракты, хотя и представляют собой формальную модель поведения тестируемых компонентов, оформляются с помощью специализированных библиотек или на расширении языков программирования, используемых при разработке этих компонентов; для библиотек ОС это язык С.
- Критерий полноты тестирования задается в виде критерия покрытия ветвлений в программных контрактах. При необходимости выделить специальные ситуации, не возникающие в контрактах явно, они формулируются в виде дополнительных ветвлений, используемых далее инструментами для оценки полноты тестового покрытия.
- Сценарий теста описывается как расширенный конечный автомат, выполнение всех переходов в котором гарантирует достижение выбранного критерия полноты (покрытие ветвлений в контрактах всех задействованных в описании переходов функций). Расширенный автомат редуцируется к конечному за счет задаваемых разработчиком теста ограничений на достижимые состояния и использования конечного множества возможных значений параметров.
- Тестирование выполняется в виде автоматического построения обхода достижимых состояний и переходов автомата, заданного сценарием теста. При этом каждый вызов тестируемой функции сопровождается обращением к ее тестовому оракулу, сгенерированному из контракта и проверяющему корректность результатов ее работы.
- Тестирование параллелизма основывается на семантике чередования [9]. Оно выполняется с помощью сбора всех наблюдаемых событий (вызовов функций, возвращений результата функциями, а также, возможно, других событий, создаваемых тестируемой системой) и построения из них последовательности, в рамках которой все контракты отдельных событий выполняются. Ошибка обнаруживается, если такую последовательность не удается построить.

UniTESK применялся в рамках проекта OLVER [10] для построения тестового набора проверки на соответствие части Core стандарта Linux Standard Base (LSB), описывающей системные библиотеки в объеме, примерно соответствующем стандарту POSIX. Полученный тестовый набор содержит тесты для 1532 функций LSB. Этот же подход использовался при создании тестового набора для проверки соответствия стандарту ARINC-653(I) [11], описывающему 54 функции.

Другой метод построения тестов, не использующий формальные спецификации, но основанный на формальном анализе требований, был использован для построения тестов для математических функций,

работающих с числами с плавающей точкой в рамках системных библиотек, описываемых стандартом POSIX [12]. Метод основан на использовании в качестве источников тестовых данных специальных значений типов с плавающей точкой; значений, мантисса которых удовлетворяет некоторому набору паттернов; границ интервалов специфического поведения тестируемой функции (в рамках которых сохраняется характер монотонности, знак, известные асимптотики и пр.); а также чисел, для которых вычисление точно округленного значения функции наиболее трудоемко, т.е. требует значительно более высокой точности по сравнению с обычными числами. На настоящий момент разработаны тестовые наборы для 120 функций.

2.2 Неформальные методы

Несколько методов построения тестов, используемых в проектах ИСП РАН, не применяют формальные спецификации, но нацелены на обеспечение строгой прослеживаемости требований.

Первый такой метод [13] использует ручное создание параметризованных тестов. Он применялся для построения тестов для более чем 4000 функций из системных библиотек Linux и позволил выявить около 40 ошибок в них.

Другой метод [14] основан на автоматической генерации простейших тестов работоспособности (проверяющих только базовую функциональность) на основе заданных вручную и хранимых в базе данных процедур стандартной инициализации значений типов параметров и проверки простейших свойств корректности для типов результатов. Этот подход обеспечивает лишь самое простейшее тестирование, но позволяет охватить большое число библиотечных функций с небольшими трудозатратами. Он применялся для тестирования библиотек Linux, содержащих около 20000 функций.

Помимо описанных методов построения тестов был разработан метод конфигурационного тестирования – выбора представительного набора конфигураций ОС, основанный на использовании так называемых покрывающих наборов и позволяющий проверить взаимодействие различных конфигурируемых элементов ее функциональности друг с другом [15].

2.3 Динамический анализ и тестирование устойчивости к сбоям

Для использования методов мониторинга при проверке свойств ядра Linux в ИСП РАН разработана среда KEDR [16]. Она позволяет перехватывать обращения одного из модулей ядра к другим модулям и проверять некоторые свойства их корректности в динамике. На основе этой среды были реализованы следующие техники.

- KEDR Leak Check, предназначенный для обнаружения утечек памяти при работе модулей ядра. Он несколько более удобен, чем аналогичный

инструмент kmemleak [17], входящий в дистрибутивы Linux, но не применим для проверки работы сердцевины ядра.

- Kernel Strider [18], предназначенный для обнаружения гонок по данным, т.е., ситуаций, в ходе которых параллельные нити читают и пишут в одну область памяти в неопределенном порядке. Этот инструмент собирает информацию о работе заданного модуля ядра, которая затем может быть проанализирована с помощью ThreadSanitizer [19], инструмента обнаружения гонок от Google.
- KEDR Fault Simulation [20], предназначенный для тестирования отказоустойчивости. Этот инструмент позволяет записать обычное выполнение теста определенного модуля, а затем на основе этой записи создать набор тестов, в каждом из которых при одном из вызовов функций других модулей происходит сбой. С помощью этого инструмента было выявлено несколько ошибок при обработке сбоев в рамках реализаций файловых систем, таких как ext4.

Другим примером инструмента мониторинга, обнаруживающего гонки по данным, является RaceHound [21], реализующий для Linux те же идеи, которые были использованы в инструменте DataCollider [22] для ОС Windows. Этот инструмент позволяет проанализировать выбранный набор инструкций в рамках одной нити, выявить в динамике область памяти, с которой работают заданные инструкции, установить на запись в них аппаратную точку останова и вставить дополнительные интервалы ожидания при доступе к памяти из других нитей. Если это приводит к доступу к помеченной области, такая ситуация фиксируется как гонка.

3. Статический анализ

Для повышения производительности большая часть кода промышленных ОС работает в привилегированном режиме, что может привести при некорректной работе этого кода к повреждению важных данных системы. Особенно эта проблема актуальна для кода драйверов, который разрабатывается сторонними программистами, обычно не знакомыми близко с правилами написания корректного кода, действующими для ядра. В результате больше половины ошибок, обнаруживаемых в ядре Linux, связано с кодов драйверов [23]. Практически то же соотношение наблюдается и для Windows [24].

Чтобы уменьшить число ошибок, совершаемых при написании кода ядра, нужны специализированные инструменты, способные проверять правила корректного использования интерфейсных функций ядра. В Microsoft Research для этой цели разработан Static Driver Verifier [25] (в первых версиях называвшийся SLAM), способный решать эту задачу для Windows. Аналогичный инструмент для Linux, названный Linux Driver Verifier [26,27]

(LDV), разработан в ИСП РАН. Поддерживаемый им метод верификации описывается следующим образом.

- Правила корректного использования интерфейсных функций ядра формулируются в виде программных контрактов в нотации, расширяющей язык C. Они интерпретируются как аспектные вставки, которые вставляются во все места вызова соответствующих функций из проверяемого модуля. При нарушении этих правил, вставляемый код создает специфическую ошибочную ситуацию.
- Для функций модуля создается модель использования, задающая все возможные сценарии обращений к ним извне. Это важно, поскольку многие модули ядра и драйвера не вызываются явно, обращения к ним организует само ядро по определенным правилам.
- Код проверяемого модуля обрабатывается инструментом, генерирующим аспектные вставки и создание ошибок, а также дополняется моделью использования.
- Основную проверку выполняет инструмент статической верификации (обычно используются BLAST [28] или CPAChecker [29]), который пытается найти сценарий работы полученного кода, при котором ошибочная ситуация становится достижимой. Если это удается, значит обнаружена ошибка, которая состоит в нарушении определенного правила корректного использования функций ядра при некотором сценарии его работы. Достижимость анализируется при помощи метода, управляемого контрпримерами уточнения абстракций (CEGAR [30]), который строит все более точные модели работы кода, пока либо не обнаружит достижимость ошибки в рамках и модели, и реального кода, либо в рамках очередной модели не покажет ее недостижимость.

LDV обнаруживает 5-8 ошибок практически в каждом очередном выпуске ядра Linux, общее число найденных и исправленных разработчиками ошибок уже более 350. При этом с помощью LDV регулярно проверяется код примерно 4000 модулей ядра.

Еще один инструмент статического анализа CPALocator [31] предназначен для выявления гонок в коде ядра и разработан на основе CPAChecker.

4. Дедуктивная верификация

Дедуктивная верификация обычно считается одним из самых аккуратных и строгих методов верификации. В то же время для продуктивного применения она требует значительных затрат труда высококвалифицированных специалистов. Систематический обзор попыток использования дедуктивной верификации кода ОС можно найти в [32].

В рамках одного из проектов ИСП РАН дедуктивная верификация использовалась для проверки свойств защищенности специализированной ОС,

основанной на ядре Linux и предназначенной для применения в государственных учреждениях для работы с секретными данными [33,34]. Эта ОС реализует достаточно сложную модель политик безопасности (МРОСЛ ДП-модель), интегрирующую механизм контроля доступа, основанного на ролях, контроля целостности и многоуровневую защиту. Эти механизмы реализованы с помощью инфраструктуры Linux Security Modules (LSM) [35], обеспечивающую перехват всех операций доступа к данным или процессам в ядре Linux.

В рамках проекта МРОСЛ ДП-модель была формализована на языке Event-B и верифицирована с помощью среды интерактивного дедуктивного анализа Rodin. Все свойства безопасности модели (например, что процесс с низким уровнем доступа не может получить доступ на чтение к высококонфиденциальным данным, или что процесс не может получить доступ к данным, не имея привязки к роли, обладающей правом на такой доступ) были записаны в виде инвариантов и доказаны. Далее, интерфейсные функции LSM были специфицированы с помощью контрактов, построенных по аналогии с контрактами похожих операций МРОСЛ ДП-модели, и для них также были сформулированы и доказаны аналогичные свойства безопасности.

К сожалению, интерфейс операций модели и интерфейс LSM сильно отличаются, и для построения контрактов операций LSM потребовалось создать отдельную формальную модель, свойства безопасности которой являются переформулировкой свойств безопасности исходной модели в терминах типов данных, с которыми работают операции LSM. Полученные во второй модели контракты операций LSM были переписаны на расширении языка C, называемом ACSL и используемом в инструменте дедуктивной верификации кода Frama C/Jessie [36]. Используемая реализация LSM в дальнейшем должна быть верифицирована на соответствие полученным контрактам. Эта работа пока не закончена, поскольку верификация кода функций ядра потребовала существенной доработки инструментов дедуктивной верификации.

Хотя верификация кода еще не доведена до конца, множество ошибок было выявлено как в самой исходной модели политик безопасности, так и в коде, при попытках проведения верификации отдельных функций. Исправление этих ошибок, хотя и не дает полных гарантий защищенности системы, существенно повышает доверие к ней.

5. Заключение

В статье рассмотрены задачи верификации и анализа современных промышленных операционных систем и проведен систематический обзор техник верификации, используемых для проверки отдельных компонентов и свойств промышленных ОС в проектах ИСП РАН. Хотя конечная цель полномасштабной верификации такой ОС пока недостижима, опыт использования рассмотренных методов показывает, что за последние годы в

сообществе исследователей и разработчиков достигнуто существенное продвижение на пути к ней.

Использование методов и инструментов различных типов, использующих тестирование, мониторинг, статический и дедуктивный анализ, может взаимно обогатить их и позволить получать более аккуратные и полные результаты за счет заимствования специфических техник моделирования или анализа определенных свойств. Это хорошо видно на примере использования одинаковых моделей памяти в рамках инструментов статического анализа и дедуктивной верификации [37].

Список литературы

[1]. Why is the Linux kernel 15+ million lines of code? Режим доступа: https://unix.stackexchange.com/questions/223746/why-is-the-linux-kernel-15-million-lines-ofcode/223770, дата обращения 10.12.2018.

[2]. How Many Lines of Code in Windows XP? Режим доступа: https://www.facebook.com/windows/posts/155741344475532, дата обращения 10.12.2018.

[3]. Герлиц Е.А., Кулямин В.В., Максимов А.В., Петренко А.К., Хорошилов А.В., Цыварев А.В. Тестирование операционных систем. Труды ИСП РАН, том 26, вып. 1, 2014 г., стр. 73-108. DOI: 10.15514/ISPRAS-2014-26(1)-3.

[4]. Bevier W.R. Kit: a Study in Operating System Verification. IEEE Transactions on Software Engineering, vol. 15, no. 11, 1989, pp. 1382-1396.

[5]. Alkassar E., Paul W.J., Starostin A., Tsyban A. Pervasive Verification of an OS Microkernel. Lecture Notes in Computer Science, vol. 6217, 2010, pp. 71-85.

[6]. Klein G., Andronick J., Elphinstone K., Murray T., Sewell T., Kolanski R., Heiser G. Comprehensive Formal Verification of an OS Microkernel. ACM Transactions on Computer Systems, vol. 32, no. 1, 2014.

[7]. Burdonov I., Kossatchev A., Petrenko A., Galter D. KVEST: Automated Generation of Test Suites from Formal Specifications. Lecture Notes in Computer Science, vol. 1708, 1999, pp. 608-621.

[8]. Bourdonov I.B., Kossatchev A.S., Kuliamin V.V., Petrenko A.K. UniTesK Test Suite Architecture. Lecture Notes in Computer Science, vol. 2391, 2002, pp. 77-88.

[9]. Kuliamin V.V., Petrenko A.K., Pakoulin N.V., Kossatchev A.S., Bourdonov I.B. Integration of Functional and Timed Testing of Real-Time and Concurrent Systems. Lecture Notes in Computer Science, vol. 2890, 2003, pp. 450-461.

[10]. Grinevich A., Khoroshilov A., Kuliamin V., Markovtsev D., Petrenko A., Rubanov V. Formal Methods in Industrial Software Standards Enforcement. Lecture Notes in Computer Science, vol. 4378, 2006, pp. 456-466.

[11]. Maksimov A. Requirements-based conformance testing of ARINC 653 real-time operating systems. In Proc. of the International Conference on Data Systems in Aerospace (DASIA 2010), 2010.

[12]. Kuliamin V. Standardization and Testing of Mathematical Functions. Lecture Notes in Computer Science, vol. 5947, 2009, pp. 257-268.

[13]. Khoroshilov A., Rubanov V., Shatokhin E. Automated Formal Testing of C API Using T2C Framework. In Proc. of the International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2008), 2008, pp.56-70.

[14]. Zybin R.S., Kuliamin V.V., Ponomarenko A.V., Rubanov V.V., Chernov E.S. Automation of broad sanity test generation. Programming and Computer Software, vol. 34, no. 6, 2008, pp. 351-363.

[15]. Кулямин В.В. Комбинаторная генерация программных конфигураций ОС. Труды ИСП РАН, том 23, 2012 г., стр. 359-370. DOI: 10.15514/ISPRAS-2012-23-20.

[16]. Shatokhin E. Using Dynamic Analysis to Hunt Down Problems in Kernel Modules. Presentation at LinuxCon Europe, 2011. Режим доступа: http://linuxtesting.org/2011-LinuxConEurope-Shatokhin-KEDR.pdf, дата обращения 10.12.2018.

[17]. kmemleak description. Режим доступа: https://www.kernel.org/doc/Documentation/kmemleak.txt, дата обращения 10.12.2018.

[18]. Kernel Strider. Режим доступа: https://code.google.com/p/kernel-strider/, дата обращения 10.12.2018.

[19]. Serebryany K., Iskhodzhanov T. ThreadSanitizer: data race detection in practice. In Proc. of the Workshop on Binary Instrumentation and Applications (WBIA 2009), 2009, pp. 62-71.

[20]. Цыварев А.В., Хорошилов А.В. Использование симуляции сбоев при тестировании компонентов ядра ОС Linux. Труды ИСП РАН, том 27, вып. 5, 2015 г., стр. 157-174. DOI: 10.15514/ISPRAS-2015-27(5)-9.

[21]. Race Hound tool. Режим доступа: http://forge.ispras.ru/projects/race-hound/, дата обращения 10.12.2018.

[22]. Erickson J., Musuvathi M., Burckhardt S., Olynyk K. Effective data-race detection for the kernel. Proc. of the USENIX Conference on Operating systems design and implementation, 2010, pp. 151-162.

[23]. Мутилин В.С., Новиков Е.М., Хорошилов А.В. Анализ типовых ошибок в драйверах операционной системы Linux. Труды ИСП РАН, том 22, 2012 г., стр. 349-374. DOI: 10.15514/ISPRAS-2012-22-19.

[24]. Ball T., Levin V., Rajamani S.K. A decade of software model checking with SLAM. Communications of the ACM, vol. 54, issue 7, 2011, pp. 68-76.

[25]. Ball T., Bounimova E., Cook B., Levin V., Lichtenberg J., McGarvey C., Ondrusek B., Rajamani S.K., Ustuner A. Thorough static analysis of device drivers. In Proc. of the ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys), 2006, pp. 73-85.

[26]. Мутилин В.С., Новиков Е.М., Страх А.В., Хорошилов А.В., Швед П.Е. Архитектура Linux Driver Verification. Труды ИСП РАН, том 20, 2011 г., стр. 163-187.

[27]. Захаров И.С., Мандрыкин М.У., Мутилин В.С., Новиков Е.М., Петренко А.К., Хорошилов А.В. Конфигурируемая система статической верификации модулей ядра операционных систем. Труды ИСП РАН, том 26, вып. 2, 2014 г., стр. 5-42. DOI: 10.15514/ISPRAS-2014-26(2)-1.

[28]. Beyer D., Henzinger T., Jhala R., Majumdar R. The software model checker BLAST: Applications to software engineering. International Journal on Software Tools for Technology Transfer (STTT), vol. 5, 2007, pp. 505-525.

[29]. Beyer D., Keremoglu M.E. CPAchecker: A tool for configurable software verification. Lecture Notes in Computer Science, vol. 6806, 2011, pp. 184-190.

[30]. Clarke E., Grumberg O, Jha S., Lu Y., Veith H. Counterexample-Guided Abstraction Refinement. Lecture Notes in Computer Science, vol. 1855, 2000, pp. 154-169.

[31]. Андрианов П.С., Мутилин В.С., Хорошилов А.В. Конфигурируемый метод поиска состояний гонок в операционных системах с использованием предикатных абстракций. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 65-86. DOI: 10.15514/ISPRAS-2016-28(6)-5.

[32]. Klein G. Operating system verification – An overview. Sadhana, vol. 34, no. 1, pp. 27-69.

[33]. Devyanin P.N., Khoroshilov A.V., Kuliamin V.V., Petrenko A.K., Shchepetkov I.V. Formal Verification of OS Security Model with Alloy and Event-B. In Proc. of the International Conference on Abstract State Machines, 2014, pp. 309-313.

[34]. Devyanin P.N., Khoroshilov A.V., Kuliamin V.V., Petrenko A.K., Shchepetkov I.V. Comparison of specification decomposition methods in Event-B. Programming and Computer Software, vol. 42, no. 4, 2016, pp. 198-205.

[35]. Wright C., Cowan C., Morris J., Smalley S., Kroah-Hartman G. Linux Security Module Framework. In Proc. of the Ottawa Linux Symposium, 2002, pp. 6-16.

[36]. Marhé C., Moy Y. The Jessie Plugin for Deductive Verification in Frama-C. Режим доступа: http://krakatoa.lri.fr/jessie.pdf, дата обращения 10.12.2018.

[37]. Мандрыкин М.У., Мутилин В.С. Обзор подходов к моделированию памяти в инструментах статической верификации. Труды ИСП РАН, том 29, вып. 1, 2017 г., стр. 195-230. DOI: 10.15514/ISPRAS-2017-29(1)-12.

Component-based verification of operating systems

[1,2,3] *V.V. Kuliamin <kuliamin@ispras.ru>*
[1,2,3] *A.K. Petrenko <petrenko@ispras.ru>*
[1,2,3,4] *A.V. Khoroshilov <khoroshilov@ispras.ru>*

[1] *V.P.Ivannikov Institute for system programming, Russian Academy of Sciences A. Solzhenitsyn str., 25, Moscow, 109004, Russia.*
[2] *Lomonosov Moscow State University, Leninskie gory, 1, Moscow, 119991, Russia.*
[3] *FCS NRU Higher School of Economics, Myasnitskaya str., 20, Moscow, 101000, Russia.*
[4] *Moscow Institute of Physics and Technology, Institutskiy per., 9, Dolgoprudny, Moscow reg., 141700, Russia.*

Abstract. The paper presents recent results on the way towards accurate and complete verification of industrial operating systems (OS). We consider here OSes, either of general purpose or actively used in some industrial domain, elaborated and maintained for a significant time, and not touching research-related OSes usually developed as a proof-of-concept. In spite of the fact that the stated goal of accurate and complete verification of industrial OS is still unreachable, we consider its decomposition into tasks of verification of various functional OS components and various their properties. The paper shows that many of these tasks can be solved with the help of various modern verification techniques and their combinations. Proposed methods can be lately integrated into an approach to the final goal. The paper summarizes the experience of various OS component and features verification from the projects conducted in ISP RAS in the last years.

Keywords: operating system; software verification; testing; static analysis; deductive verification; monitoring.

DOI: 10.15514/ISPRAS-2018-30(6)-21

For citation: Kuliamin V.V., Petrenko A.K., Khoroshilov A.V. Component-based verification operating systems. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 367-382 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-21

References

[1] Why is the Linux kernel 15+ million lines of code? Available at: https://unix.stackexchange.com/questions/223746/why-is-the-linux-kernel-15-million-lines-ofcode/223770, accessed 10.12.2018.

[2] How Many Lines of Code in Windows XP? Available at: https://www.facebook.com/windows/posts/155741344475532, accessed 10.12.2018.

[3] Gerlits E.A., Kuliamin V.V., Maksimov A.V., Petrenko A.K., Khoroshilov A.V., Tsyvarev A.V. Testing of Operating Systems. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 1, 2014, pp. 73-108 (in Russian). DOI: 10.15514/ISPRAS-2014-26(1)-3.

[4] Bevier W.R. Kit: a Study in Operating System Verification. IEEE Transactions on Software Engineering, vol. 15, no. 11, 1989, pp. 1382-1396.

[5] Alkassar E., Paul W.J., Starostin A., Tsyban A. Pervasive Verification of an OS Microkernel. Lecture Notes in Computer Science, vol. 6217, 2010, pp. 71-85.

[6] Klein G., Andronick J., Elphinstone K., Murray T., Sewell T., Kolanski R., Heiser G. Comprehensive Formal Verification of an OS Microkernel. ACM Transactions on Computer Systems, vol. 32, no. 1, 2014.

[7] Burdonov I., Kossatchev A., Petrenko A., Galter D. KVEST: Automated Generation of Test Suites from Formal Specifications. Lecture Notes in Computer Science, vol. 1708, 1999, pp. 608-621.

[8] Bourdonov I.B., Kossatchev A.S., Kuliamin V.V., Petrenko A.K. UniTesK Test Suite Architecture. Lecture Notes in Computer Science, vol. 2391, 2002, pp. 77-88.

[9] Kuliamin V.V., Petrenko A.K., Pakoulin N.V., Kossatchev A.S., Bourdonov I.B. Integration of Functional and Timed Testing of Real-Time and Concurrent Systems. Lecture Notes in Computer Science, vol. 2890, 2003, pp. 450-461.

[10] Grinevich A., Khoroshilov A., Kuliamin V., Markovtsev D., Petrenko A., Rubanov V. Formal Methods in Industrial Software Standards Enforcement. Lecture Notes in Computer Science, vol. 4378, 2006, pp. 456-466.

[11] Maksimov A. Requirements-based conformance testing of ARINC 653 real-time operating systems. In Proc. of the International Conference on Data Systems in Aerospace (DASIA 2010), 2010.

[12] Kuliamin V. Standardization and Testing of Mathematical Functions. Lecture Notes in Computer Science, vol. 5947, 2009, pp. 257-268.

[13] Khoroshilov A., Rubanov V., Shatokhin E. Automated Formal Testing of C API Using T2C Framework. In Proc. of the International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2008), 2008, pp.56-70.

[14] Zybin R.S., Kuliamin V.V., Ponomarenko A.V., Rubanov V.V., Chernov E.S. Automation of broad sanity test generation. Programming and Computer Software, vol. 34, no. 6, 2008, pp. 351-363.

[15] Kuliamin V.V. Combinatoric generation of operating system software configurations. Trudy ISP RAN/Proc. ISP RAS, vol. 23, 2012, pp. 359-370 (in Russian). DOI: 10.15514/ISPRAS-2012-23-20.

[16] Shatokhin E. Using Dynamic Analysis to Hunt Down Problems in Kernel Modules. Presentation at LinuxCon Europe, 2011. Available at: http://linuxtesting.org/2011-LinuxConEurope-Shatokhin-KEDR.pdf, accessed 10.12.2018.

[17] kmemleak description. Available at: https://www.kernel.org/doc/Documentation/kmemleak.txt, accessed 10.12.2018.

[18] Kernel Strider. Available at: https://code.google.com/p/kernel-strider/, accessed 10.12.2018.

[19] Serebryany K., Iskhodzhanov T. ThreadSanitizer: data race detection in practice. In Proc. of the Workshop on Binary Instrumentation and Applications (WBIA 2009), 2009, pp. 62-71.

[20] Tsyvaerv A., Khoroshilov A. Using Fault Injection for Testing Linux Kernel Components. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 5, 2015, pp. 157-174 (in Russian). DOI: 10.15514/ISPRAS-2015-27(5)-9.

[21] Race Hound tool. Available at: http://forge.ispras.ru/projects/race-hound/, accessed 10.12.2018.

[22] Erickson J., Musuvathi M., Burckhardt S., Olynyk K. Effective data-race detection for the kernel. Proc. of the USENIX Conference on Operating systems design and implementation, 2010, pp. 151-162.

[23] Mutilin V.S., Novikov E.M., Khoroshilov A.V. Analysis of typical faults in Linux operating system drivers. Trudy ISP RAN/Proc. ISP RAS, vol. 22, 2012, pp. 349-374 (in Russian). DOI: 10.15514/ISPRAS-2012-22-19.

[24] Ball T., Levin V., Rajamani S.K. A decade of software model checking with SLAM. Communications of the ACM, vol. 54, issue 7, 2011, pp. 68-76.

[25] Ball T., Bounimova E., Cook B., Levin V., Lichtenberg J., McGarvey C., Ondrusek B., Rajamani S.K., Ustuner A. Thorough static analysis of device drivers. In Proc. of the ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys), 2006, pp. 73-85.

[26] Mutilin V.S., Novikov E.M., Strakh A.V., Khoroshilov A.V., Shved P.E. Architectire of Linux Driver Verification. Trudy ISP RAN/Proc. ISP RAS, vol. 20, 2011, pp. 163-187 (in Russian).

[27] Zakharov I.S., Mandrykin M.U., Mutilin V.S., Novikov E.M., Petrenko A.K., Khoroshilov A.V. Configurable Toolset for Static Verication of Operating Systems Kernel Modules. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 2, 2014, pp. 5-42 (in Russian). DOI: 10.15514/ISPRAS-2014-26(2)-1.

[28] Beyer D., Henzinger T., Jhala R., Majumdar R. The software model checker BLAST: Applications to software engineering. International Journal on Software Tools for Technology Transfer (STTT), vol. 5, 2007, pp. 505-525.

[29] Beyer D., Keremoglu M.E. CPAchecker: A tool for configurable software verification. Lecture Notes in Computer Science, vol. 6806, 2011, pp. 184-190.

[30] Clarke E., Grumberg O, Jha S., Lu Y., Veith H. Counterexample-Guided Abstraction Refinement. Lecture Notes in Computer Science, vol. 1855, 2000, pp. 154-169.

[31] Andrianov P.S., Mutilin V.S., Khoroshilov A.V. Adjustable method with predicate abstraction for detection of race conditions in operating systems. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 65-86 (in Russian). DOI: 10.15514/ISPRAS-2016-28(6)-5.

[32] Klein G. Operating system verification – An overview. Sadhana, vol. 34, no. 1, pp. 27-69.

[33] Devyanin P.N., Khoroshilov A.V., Kuliamin V.V., Petrenko A.K., Shchepetkov I.V. Formal Verification of OS Security Model with Alloy and Event-B. In Proc. of the International Conference on Abstract State Machines, 2014, pp. 309-313.

[34] Devyanin P.N., Khoroshilov A.V., Kuliamin V.V., Petrenko A.K., Shchepetkov I.V. Comparison of specification decomposition methods in Event-B. Programming and Computer Software, vol. 42, no. 4, 2016, pp. 198-205.

[35] Wright C., Cowan C., Morris J., Smalley S., Kroah-Hartman G. Linux Security Module Framework. In Proc. of the Ottawa Linux Symposium, 2002, pp. 6-16.

[36] Marhé C., Moy Y. The Jessie Plugin for Deductive Verification in Frama-C. Available at: http://krakatoa.lri.fr/jessie.pdf, accessed 10.12.2018.

[37] Mandrykin M.U., Mutilin V.S. Survey of memory modeling methods in static verification tools. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 1, 2017, pp. 195-230 (in Russian). DOI: 10.15514/ISPRAS-2017-29(1)-12.