ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 32 Выпуск 5

ISSN Online 2220-6426
Volume 32 Issue 5

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2020

ИСП РАН

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:

Google Scholar, ULRICHSWEB Global Serials Directory, Russian Science Citation Index, OCLC WorldCat, BASE, OpenDOAR, CYBERLENINKA, DOAJ Directory of Open Access Journals, eLIBRARY.RU, ROAR Registry of Open Access Repositories, Crossref, Math-Net.Ru

УДК004.45

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

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: proceedings@ispras.ru
Web: https://ispranproceedings.elpub.ru/

Содержание

Table of Contents

DOI: 10.15514/ISPRAS–2020–32(5)–1

Экспертная оценка результатов верификации инструментов верификации моделей программ

В.А. Гратинский, ORCID: 0000-0001-8598-1298 <gratinskiy@ispras.ru>
Е.М. Новиков, ORCID: 0000-0003-3586-3140 <novikov@ispras.ru>
И.С. Захаров, ORCID: 0000-0001-5713-5887 <ilja.zakharov@ispras.ru>
Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. При проверке программ на соответствие спецификациям требований инструменты верификации моделей программ могут выдавать различные результаты верификации. Среди них вердикты, свидетельства нарушений и отчеты о покрытии кода представляют собой наибольшую ценность с точки зрения экспертов, которые хотят обнаружить ошибки в программах и оценить полноту проверки. Для индустриального использования инструментов верификации моделей программ, когда проверяется множество различных требований к различным версиям и конфигурациям многих программ, экспертам необходимы удобные средства автоматизации оценки результатов верификации. В статье рассматриваются соответствующие подходы и их реализация в системе верификации Klever.

Ключевые слова: верификация моделей программ; результат верификации; свидетельство нарушения; покрытие кода; экспертная оценка

Для цитирования: Гратинский В.А., Новиков Е.М., Захаров И.С. Экспертная оценка результатов верификации инструментов верификации моделей программ. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 7-20. DOI: 10.15514/ISPRAS–2020–32(5)–1

Expert Assessment of Verification Tool Results

V.A. Gratinskiy, ORCID: 0000-0001-8598-1298 <gratinskiy@ispras.ru>
E.M. Novikov, ORCID: 0000-0003-3586-3140 <novikov@ispras.ru>
I.S. Zakharov, ORCID: 0000-0001-5713-5887 <ilja.zakharov@ispras.ru>
Ivannikov Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. Verification tools can produce various kinds of results while checking programs against requirement specifications. Experts, who seek for errors and estimate completeness of verification, mostly appreciate verdicts, violation witnesses and code coverage reports. They need convenient tools for automating the assessment of verification results to apply verification tools in practice when many program configurations and versions are checked against various requirements. In this paper, we propose new methods for expert evaluation of verification results, covering all those problems that are most significant in accordance with our experience in verifying large programs for compliance with a large number of requirements specifications. Some ideas are borrowed from the areas of testing and static analysis. However, specific methods and technical solutions are unique, since the verification results provided by verification tools are either not found in other areas or have special semantics. The paper presents our approaches and their implementation in the Klever software verification framework.

Keywords: software model checking; verification result; violation witness; code coverage; expert assessment

For citation: Gratinskiy V.A., Novikov E.M., Zakharov I.S. Expert Assessment of Verification Tool Results. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 7-20 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–1

1. Введение

Инструменты верификации моделей программ (далее для краткости – *инструменты верификации*) позволяют проверять программы на соответствие набору спецификаций требований автоматизированным и тщательным образом [1]. На вход инструменты верификации принимают *верификационные задачи*. Их формат подробно описан на сайте международных соревнований инструментов верификации (*SV-COMP*) [2, 3]. Каждая верификационная задача должна содержать подготовленный для верификации фрагмент исходного кода целевой программы среднего размера и спецификацию свойств безопасности/живучести, например, достижимость вызова специальной функции или безопасность работы с памятью. Для получения результатов верификации приемлемого качества при рассмотрении программ по частям, а также для проверки большинства реальных требований необходимо дополнить фрагменты программ достаточно точными моделями их окружения.

В этой статье рассматривается верификация больших, быстро развивающихся программ на языке Си, например, ядер операционных систем. Эти программы должны быть декомпозированы на множество фрагментов (десятки, сотни или тысячи), каждый из которых проверяется независимо. Кроме того, из-за некоторых ограничений инструментов верификации также необходимо проверять различные требования к каждому фрагменту программы независимым образом. В результате могут получиться большие наборы верификационных задач, для каждой из которых инструмент верификации выдает результат ее решения. Вопросы получения наборов верификационных задач и выполнения их решения рассматриваются в работе [4].

В соответствии с правилами *SV-COMP* инструмент верификации предоставляет следующие артефакты в качестве результатов решения верификационной задачи (ниже мы называем их *результатами верификации*).

- *Вердикт*, который отвечает на вопрос: «удовлетворяет ли программа спецификации?». Возможны следующие вердикты:
  - *TRUE* – программа корректна с точки зрения инструмента верификации (однако ошибки могут быть пропущены, например, из-за неточных моделей окружения);
  - *FALSE* – программа содержит ошибку (однако это может быть ложное сообщение об ошибке, например, из-за неточного анализа инструмента верификации);
  - *UNKNOWN* – данный вердикт выдается, когда для завершения проверки инструменту верификации не хватает отведенных вычислительных ресурсов или он завершается аварийно из-за внутренних ошибок.
- *Свидетельство нарушения* (*violation witness*) – машиночитаемый файл, содержащий фрагменты формального доказательства обнаруженного нарушения спецификации проверяемых свойств. Инструмент верификации должен выдавать его для вердикта *FALSE* так, чтобы инструменты валидации свидетельств могли бы подтвердить или опровергнуть корректность данного доказательства автоматически [5].
- *Свидетельство корректности* (*correctness witness*) – аналогично для вердикта *TRUE* [6]. Мы не рассматриваем свидетельства корректности в статье, так как они практически не могут помочь экспертам понять соответствующие доказательства корректности.

Кроме того, инструменты верификации могут выдавать дополнительные результаты верификации, которые могут быть полезны для экспертов. Например, CPAchecker выдает следующие файлы:

- *Отчет о покрытии кода*, демонстрирующий исходный код, который считается достижимым с точки зрения инструмента верификации во время проверки.
- *Лог*, который может включать сведения о проводимом анализе, а также сведения о внутренних ошибках инструмента верификации.

Исследователи и разработчики инструментов статического анализа и тестирования добились больших успехов в представлении результатов. В области верификации моделей программ это направление пока развито не так хорошо. Такие работы, как визуализация трасс ошибок в SDV [7] или свидетельств нарушения в рамках *SV-COMP* [8] не предоставляют пользователям достаточно полный набор средств анализа результатов верификации. В данной работе предложены новые методы экспертной оценки результатов верификации, охватывающие все те проблемы, которые являются наиболее существенными в соответствии с нашим опытом верификации больших программ на соответствие большому количеству спецификаций требований. Некоторые идеи заимствованы из областей тестирования и статического анализа. Однако конкретные методы и технические решения уникальны, поскольку результаты верификации, предоставляемые инструментами верификации, либо не встречаются в других областях, либо обладают особенной семантикой.

В следующих разделах подробно рассматриваются предлагаемые методы экспертной оценки результатов верификации и их реализация в системе верификации Klever [4].

2. Методы экспертной оценки результатов верификации

2.1 Вспомогательная терминология

Перед описанием предлагаемых методов необходимо определить несколько вспомогательных терминов:

- *База сборки* – директория, которая генерируется в результате работы Clade [9] и содержит следующие данные в структурированном виде:
  - файлы с исходным кодом целевой программы;
  - сведения о сущностях программы, таких как типы, функции и переменные, а также об их взаимосвязях;
  - информация о процессе сборки программы.
- *Верификационное задание* – набор файлов, включающий базу сборки, спецификации и конфигурационные файлы, необходимые для запуска верификации. Верификационные задания могут быть созданы на основе предустановленных верификационных заданий, соответствующих различным проектам. Пользователи могут создавать несколько версий для каждого верификационного задания, например, при разработке новой спецификации требований. Проекты, предустановленные верификационные задания, верификационные задания и их версии хранятся и отображаются структурированным образом подобно файлам на диске.
- *Верификационная задача* – набор файлов для запуска инструмента верификации. В целом это определение соответствует определению *SV-COMP*, за исключением того, что здесь и далее в состав верификационных задач также включаются конкретное название инструмента верификации, а также специфичные для него опции командной строки и ограничения вычислительных ресурсов. Верификационные задачи генерируются и решаются по мере решения верификационных заданий.
- *Верификационные отчеты* – отчеты о результатах верификации, а также вспомогательные отчеты, которые создаются в ходе решения верификационного задания. Верификационные отчеты могут включать в себя *атрибуты*, характеризующие проверяемые программы и требования, например, названия и версии проектов, названия

фрагментов программ и идентификаторы спецификаций требований.

- *Оригинальные файлы с исходным кодом* – файлы с исходным кодом целевой программы из базы сборки.
- *Дополнительные файлы с исходным кодом* или *модели* – файлы с исходным кодом, которые генерируются на основе спецификаций при решении верификационных заданий. Модели включаются в верификационные задачи наряду с оригинальными файлами с исходным кодом.
- *CIL файлы* – файлы, объединяющие оригинальные и дополнительные файлы с исходным кодом. Эти файлы получаются с помощью инструмента CIL [10]. Каждая верификационная задача включает в себя ровно один CIL файл, так как большинство инструментов верификации не поддерживает никакого другого представления целевой программы. CIL файлы ссылаются на оригинальные и дополнительные файлы с исходным кодом посредством директив препроцессора *#line*.

2.2 Предлагаемые методы

Первоочередная цель предлагаемых методов заключается в том, чтобы представить экспертам результаты верификации удобным и интуитивным образом с учетом того, что их основные задачи таковы.

- Понять, соответствуют ли выдаваемые предупреждения ошибкам (кроме того, необходимо понять первопричины этих ошибок) или ложным сообщениям об ошибках.
- Оценить полноту проверки. Для этого необходимо предоставить экспертам оценку достижимого исходного кода, чтобы они могли убедиться в том, что анализ не прекратился в начале или в середине программы.

Наши основные предложения перечислены ниже.

- Поддержать нескольких пользователей, каждый из которых может иметь определенный набор прав как глобально, так и для конкретных верификационных заданий. Это может быть полезно, потому что несколько человек могут проводить экспертизу одновременно, а некоторым экспертам нельзя доверять оценку важных результатов верификации. Администратор должен иметь возможность назначать глобальные роли новым пользователям. Пользователи, обладающие правами на верификационные задания, должны иметь возможность назначать права на экспертную оценку соответствующих результатов верификации непривилегированным пользователям.
- Связать свидетельства нарушений и покрытие кода с оригинальными файлами с исходным кодом и моделями; выделить те части свидетельств нарушений, которые соответствуют моделям и изменениям связанных с ошибками данных; скрыть части свидетельств нарушений, которые наименее релевантны обнаруженным предупреждениям. Например, мы предлагаем максимально скрывать от экспертов те части свидетельств нарушений, которые соответствуют реализациям моделей, поскольку соответствующие детали обычно не представляют интереса. Однако должна быть возможность анализировать все подробно, поскольку неточные модели могут вызывать ложные сообщения об ошибках и экспертам может потребоваться их изучить.
- Должна быть возможность создавать *экспертные оценки* для свидетельств нарушений. Для каждого предупреждения эксперт должен иметь возможность указать, соответствует ли оно ошибке или ложному сообщению об ошибке, добавить один или несколько тегов и произвольное текстовое описание. Для изменений экспертных отметок должна быть возможность оставлять комментарии, описывающие эти изменения.
- Автоматически ассоциировать экспертные оценки с аналогичными свидетельствами нарушений. Автоматическая оценка может сэкономить достаточно много времени за счет

отсутствия повторного анализа результатов верификации, полученных, например, для разных версий или конфигураций одной и той же программы. Эксперты должны иметь возможность подтвердить или отклонить автоматически ассоциированные оценки вручную, так как автоматическая процедура может работать некорректно.

- Предоставить экспертам статистические данные о вынесенных вердиктах, а также об ассоциированных экспертных оценках. Кроме того, эксперты должны иметь возможность изучить соответствующие списки, например, список всех предупреждений, которые были автоматически оценены как ошибки.
- Должна быть возможность сравнивать результаты верификации друг с другом. Это полезно для оценки как изменений в проверяемых программах, включая сравнение их различных конфигураций, так и изменений в компонентах и спецификациях Klever.
- Пользователи должны иметь возможность настраивать представление результатов верификации, чтобы анализировать их более удобным способом, например, добавлять или удалять некоторые столбцы в таблицах, сортировать и фильтровать результаты верификации по некоторым критериям.
- Позволить перемещать результаты верификации, а также экспертные оценки между различными машинами. Это необходимо, поскольку результаты верификации могут быть получены в разное время на разных машинах, включая те, которые не соединены по сети. Однако может оказаться более удобным хранить и анализировать все результаты верификации на одной машине.
- Сохранять как версии верификационных заданий, так и полученные для них результаты верификации. Например, это может быть полезно для сравнения их друг с другом через некоторое время.
- Сохранять историю изменений экспертных оценок для того, чтобы иметь возможность откатить некоторые изменения и проанализировать эту историю.
- Должна быть возможность проведения экспертной оценки результатов верификации по мере их получения, так как этот процесс может занять значительное количество времени, например, несколько дней даже на достаточно мощной машине.

Реализация предлагаемых методов должна быть достаточно эффективной и не создавать значительную нагрузку на диск и память. Для этого необходимо учесть следующее.

- Нельзя выполнять сложные вычисления непосредственно при представлении результатов верификации пользователям. Например, данные для представления перекрестных ссылок должны быть получены до того, как будут визуализироваться соответствующие файлы с исходным кодом.
- Необходимо кэшировать данные, которые нетривиально получить и которые невозможно вычислить заранее. Например, это касается сравнения результатов верификации.
- Нельзя дублировать оригинальные файлы с исходным кодом и вспомогательную информацию для них, такую как данные для подсветки синтаксиса и перекрестных ссылок. Благодаря этому можно существенно уменьшить занимаемое дисковое пространство (для больших программ, таких как ядро Linux, соответствующие данные могут занимать сотни мегабайт даже в архивах). Как правило, это требование не актуально для моделей, поскольку они довольно часто меняются. Кроме того, большинство моделей имеют несколько десятков или сотен строк.
- Должны существовать средства для удаления ненужных данных, таких как промежуточные версии верификационных заданий и их результаты верификации. Это может помочь сократить объем отображаемой информации и очистить диск.

Наконец, мы предлагаем поддержать специальные возможности для разработчиков.

- Показ логов работы компонентов Klever.
- Поддержка автоматизированной оценки верификационных отчетов с информацией о внутренних ошибках компонентов Klever и инструментов верификации.
- Загрузка архивов конкретных верификационных задач. Например, это может быть полезно для того, чтобы подготовить для разработчиков инструментов верификации примеры, когда инструменты верификации работают некорректным образом.

В следующих подразделах подробно рассматриваются наиболее важные и сложные подходы, предложенные выше.

2.3 Представление вердиктов

Мы предлагаем рассматривать вердикты *TRUE* и *FALSE*, соответствующие доказательствам корректности и обнаружению нарушений проверяемых спецификаций требований, как *Safe* и *Unsafe* соответственно, так как это должно быть более понятно экспертам. Если при решении одной верификационной задачи обнаружено более одного нарушения проверяемой спецификации требований, то должно быть создано соответствующее количество верификационных отчетов *Unsafes*, а каждый из них должен быть дополнен своим собственным свидетельством нарушения.

2.4 Представление свидетельств нарушений и их экспертная оценка

2.4.1 Формат свидетельств нарушений

Свидетельство нарушения – это специальный XML-файл в формате *graphML* [5, 6, 11]. Этот файл описывает наблюдающий автомат (*observer automaton*) для обхода программы от точки входа до найденной ошибки. Инструмент валидации свидетельств рассматривает наблюдающий автомат в сочетании с автоматом потока управления, извлекаемым из представления программы, для проверки корректности решения верификационной задачи. Свидетельство нарушения не является трассой ошибки, поскольку оно может описывать подмножество возможных путей выполнения и может не содержать нужных деталей, таких как части путей через некоторые функции, итерации цикла и т.д.

Далее в данном подразделе рассматриваются ключевые элементы формата свидетельств нарушений. Пути выполнения программы задаются с помощью вершин и ребер. Промежуточные вершины имеют входные и выходные ребра. Тупиковые вершины (*sink nodes*) не имеют выходных ребер. Пример тупиковой вершины – это вершина, соответствующая найденной ошибке.

Каждое ребро соответствует некоторой декларации или выражению в программе. Оно ссылается на них с помощью номера строки и смещения в CIL файле (теги *startline, endline, startoffset* и *endoffset*). Для указания входа и выходы из функции, условного выражения и цикла используются теги *enterFunction, returnFromFunction, control* и *enterLoopHead* соответственно. Для представления значений переменных, которые предполагает инструмент верификации, используется тег *assumption*.

На Листингах 1 и 2 приведены примеры программы на языке Си, содержащей ошибку, и соответствующего свидетельства нарушения соответственно. Для краткости для свидетельства нарушения не показаны некоторые теги, тупиковые вершины, а также ребра, ведущие к ним.

Данный пример демонстрирует ситуацию, когда свидетельство нарушения не содержит важный фрагмент пути выполнения через функцию *func* (ребро от вершины *A2* до вершины *A3*). Хотя такое свидетельство нарушения корректно и пригодно для валидации, оно не может использоваться в качестве полноценной трассы ошибки.

```
 1 int func(int arg) {
 2     return arg > 0 ? 1 : 0;
 3 }
 4 void entry_point(int arg) {
 5     usigned int cond = 0;
 6     if (arg)
 7         cond = func(arg);
 8     if (arg)
 9         assert(cond != 1);
10 }
```

Листинг 1. Пример программы на языке Си, содержащей ошибку
Listing 1. An example of a C program containing an error

```
<node id="A0">
    <data key="entry">true</data>
</node>
<edge source="A0" target="A1">
    <data key="startline">4</data>
</edge>
<node id="A1"/>
<edge source="A1" target="A2">
    <data key="startline">6</data>
    <data key="control">condition-true</data>
    <data key="assumption">arg == 1;</data>
</edge>
<node id="A2"/>
<edge source="A2" target="A3">
    <data key="startline">7</data>
</edge>
<node id="A3"/>
<edge source="A3" target="A4">
    <data key="startline">8</data>
    <data key="control">condition-true</data>
    <data key="assumption">cond == 1;</data>
</edge>
<node id="A4">
    <data key="violation">true</data>
</node>
```

Листинг 2. Пример свидетельства нарушения
Listing 2. An example of a violation witness

Для проверки большинства требований (свойств) верификационные задачи должны включать исходный код целевых программ и моделей окружения, которые сериализуют возможные выполнения каким-либо разумным образом, поскольку тщательная проверка многопоточных программ является чрезвычайно сложной задачей. Тем не менее, например, для поиска состояний гонок инструменты верификации должны получать многопоточные программы и модели окружения в качестве входных данных. В этом случае они добавляют для ребер свидетельств нарушений информацию о потоках, в которых они выполняются, в частности идентификаторы потоков.

2.4.2 Расширенный формат свидетельств нарушений

Мы предлагаем расширить формат свидетельств нарушения таким образом, чтобы иметь все части пути [12]. Кроме того, данный формат позволяет добавить важную информацию от алгоритмов проверки для ребер свидетельств нарушения. Благодаря этому, например, при проверке безопасной работы с памятью можно отметить те места, где выделяется память, которая впоследствии не освобождается (утечка памяти).

2.4.3 Преобразование свидетельств нарушений в трассы ошибок

В данной работе предлагается преобразовывать свидетельства нарушений в трассы ошибок, которые более удобны для визуализации и анализа [13]. Данное преобразование включает в себя следующие действия.

- Преобразование ссылок на CIL файлы для ребер свидетельств нарушений в соответствующие декларации и выражения, а также номера строк в оригинальных и дополнительных файлах с исходным кодом программы.
- Обработка специальных комментариев из моделей окружения и спецификаций требований для определения действий со специальными сущностями, таких как вызовы точек входа фрагментов программы и изменения состояния модели. В соответствии с этими данными все части свидетельств нарушения, которые кажутся несущественными для обнаруженных нарушений проверяемых требований, помечаются так, чтобы их можно было скрыть при последующей визуализации.
- Если инструмент верификации не предоставляет информацию о потоках для ребер, мы предлагаем добавить ее в соответствии с комментариями в моделях окружения. Такие комментарии указывают на создание и объединение потоков, а автоматически генерируемые идентификаторы потоков распространяются по соответствующим стекам вызовов.

2.4.4 Представление покрытия кода

Отчеты о покрытии кода являются важным артефактом при применении верификации на практике. Они отражают части программы, такие как строки кода, ветви и функции, которые фактически проверяются. Эта информация необходима для оценки качества модели окружения, поскольку ни свидетельства нарушений, ни свидетельства корректности не включают в себя данные о рассматриваемых путях. Отчеты о покрытии кода помогают понять, какие точки входа в программу должны дополнительно вызываться из моделей окружения.

Отчет о покрытии кода – это вспомогательный файл, описывающий исходный код верификационной задачи, который был достижим во время верификации. Этот файл предназначен только для визуализации пользователям. Насколько известно авторам статьи, CPAchecker – это единственный инструмент верификации, который может выдавать отчеты о покрытии кода в дополнение к свидетельствам [14]. Формат файла соответствует формату тестового покрытия *LCOV* [15].

Как и для свидетельств нарушений, мы предлагаем преобразовать отчеты о покрытии кода в более подходящую форму для их визуализации [16], что включает в себя следующие шаги.

- Связывание покрытия кода с соответствующими оригинальными файлами с исходным кодом и моделями вместо CIL файлов.
- Вычисление статистики для каждого отдельного файла с исходным кодом, а также для каждой поддиректории дерева исходного кода программы с учетом либо файлов с исходным кодом, рассматриваемых инструментом верификации, либо всех файлов с исходным кодом из базы сборки. Эти наборы файлов могут отличаться, так как инструмент верификации может анализировать не все файлы с исходным кодом,

например, из-за его внутренних ошибок.

- Объединение отчетов о покрытии кода, выданных для отдельных верификационных задач, с целью получить общий отчет о покрытии кода для программы независимо для каждой спецификации требований.

2.4.5 Сравнение результатов верификации

При сравнении результатов верификации двух версий одного или двух различных верификационных заданий предлагается либо получить данные по сравнению из кэша, либо вычислить их и заполнить кэш в соответствие со следующим алгоритмом:

- Для каждой из двух версий верификационного(ых) задания(й) определить список атрибутов, которые должны использоваться для сравнения, в соответствии со значениями специального флага атрибутов верификационных отчетов. Упорядочить полученные списки по именам атрибутов в алфавитном порядке. Типичный список атрибутов состоит из имени фрагмента программы и идентификатора спецификации требований, так как обычно они однозначно различают верификационные задачи верификационных заданий.
- Сообщить, что верификационные результаты не могут быть сравнены, если полученные списки атрибутов не совпадают. В противном случае перейти к следующим шагам.
- Получить вердикты для всех верификационных отчетов, соответствующих уникальным значениям атрибутов из списка, и вычислить суммарный вердикт для каждого такого списка верификационных отчетов.
- Для каждого уникального набора значений атрибутов сохранить в кэше следующее: набор значений атрибутов, суммарный вердикт первой и второй версии верификационного(ых) задания(й). Когда список отчетов одной из версии верификационного задания пуст, необходимо сохранить специальный суммарный вердикт *Unmatched*.
- Для каждой пары суммарных вердиктов рассчитать количество соответствующих верификационных отчетов.

На основе данных в кэше предлагается показать количество верификационных отчетов для всех возможных пар суммарных вердиктов. Кроме того, эксперты должны иметь возможность анализировать конкретные верификационные отчеты, соответствующие уникальным значениям атрибутов, а также ассоциированные с ними экспертные оценки, если таковые имеются.

3. Реализация предложенных методов

Предложенные методы экспертной оценки результатов верификации были реализованы в компоненте Klever Bridge системы верификации Klever [4, 17]. Оценить удобство и производительность Klever Bridge довольно сложно, поскольку существует множество различных вариантов использования, некоторые из которых предполагают интенсивное взаимодействие с пользователем. Хотя некоторые действия, например, анализ и экспертная оценка свидетельств нарушений, не являются тривиальными и широко известными, как правило, новые пользователи начинают выполнять данные действия самостоятельно после нескольких часов обучения.

Для демонстрации Klever Bridge мы перепроверили безопасность работы с памятью для всех загружаемых модулей ядра Linux 5.5 (архитектура *x86_64* и конфигурация *allmodconfig*) на новом инстансе Klever и загрузили на него экспертные оценки, созданные на старом инстансе Klever.

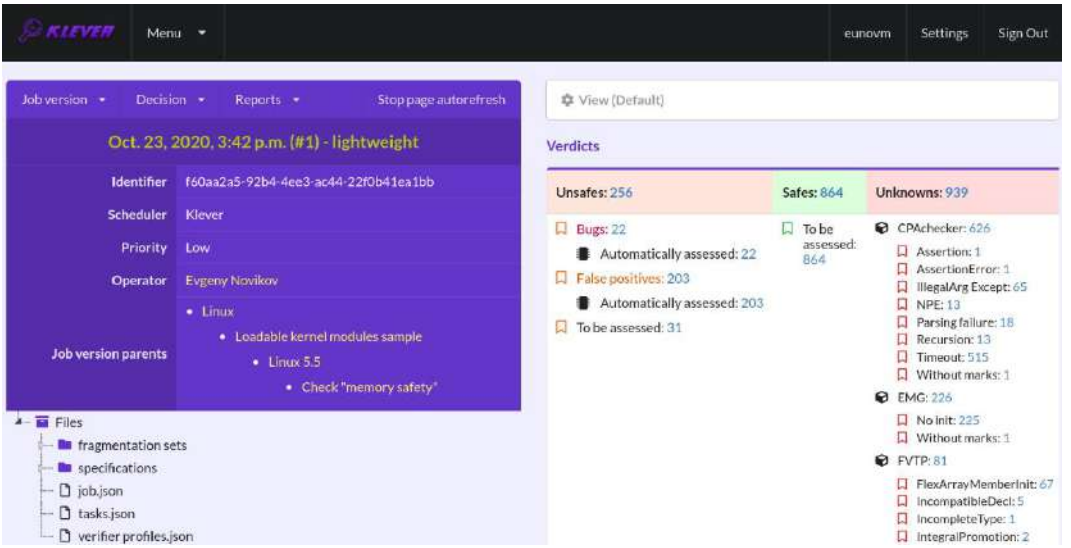


Рис. 1. Статистика по вердиктам и ассоциированным экспертным оценкам
Fig. 1. Statistics on verdicts and associated expert marks

Рис. 1 показывает статистику по вердиктам и ассоциированным экспертным оценкам. Можно увидеть, что большинство *Unsafes* были оценены автоматически. На рис. 2 представлен пример визуализации свидетельства нарушения, файлов с исходным кодом и покрытия кода для загружаемого модуля ядра (фрагмента программы) *drivers/media/platform/s5p-jpeg/s5p-jpeg.ko*. Выданное предупреждение соответствует ошибке в том случае, если функция *of_match_node()* может завершиться не успешно. В этом случае *jpeg_get_drv_data()* возвращает *NULL*, который присваивается указателю *jpeg→variant*, разыменовываемому позже без каких-либо проверок. Исправление данной ошибки были сообщено разработчикам ядра Linux [18].

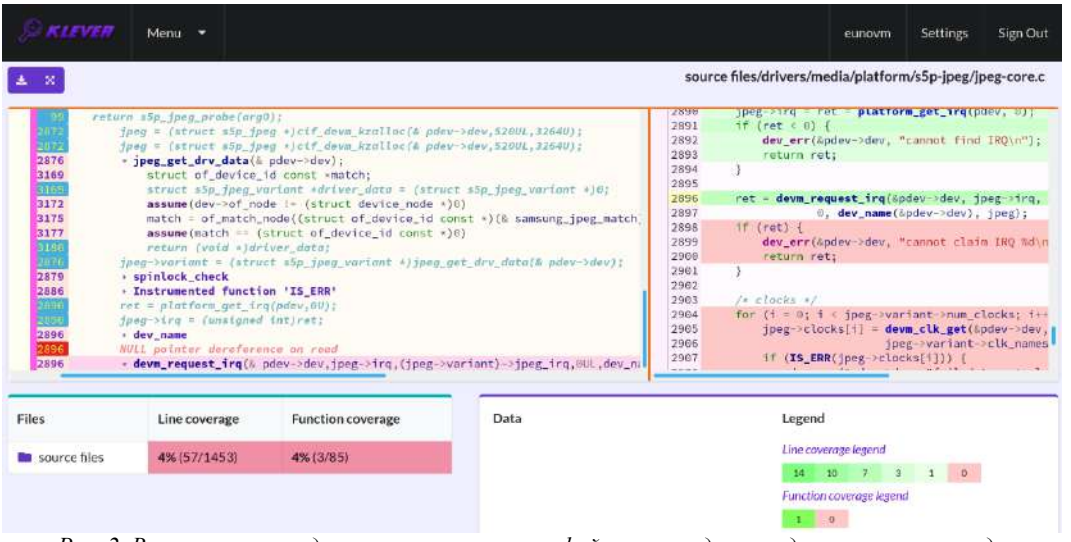


Рис. 2. Визуализация свидетельства нарушения, файлов с исходным кодом и покрытия кода
Fig. 2. Visualized violation witness, source files and code coverage

Автоматически ассоциированные экспертные оценки для данного свидетельства нарушения представлены на рис. 3. Вторая экспертная оценка в списке была создана для старой версии спецификаций, таким образом, она не похожа на новое свидетельство нарушения. Первая

экспертная оценка ассоциирована со свидетельством нарушения на 100%, и эксперт может ее подтвердить.

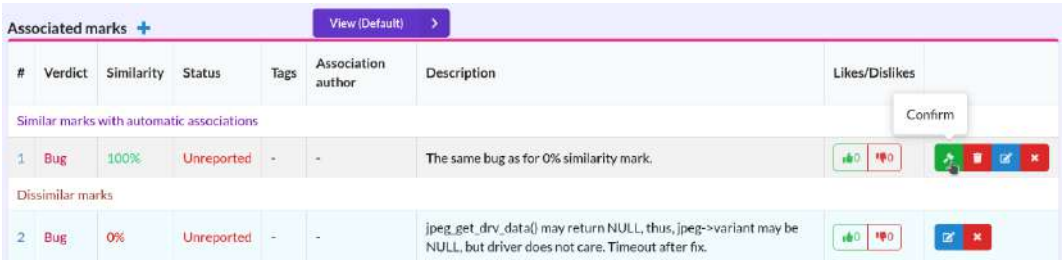


Рис. 3. Автоматически ассоциированные экспертные оценки
Fig. 3. Automatically associated expert marks

Рис. 4 демонстрирует сравнение тех результатов верификации, которые рассматривались выше, с результатами верификации, полученными для Linux 5.9-rc8. Большинство результатов верификации остались теми же самыми, однако есть и много различий. Например, под таблицей приведены детали для отличия, которое произошло после того, как мы отправили патч разработчикам ядра Linux.

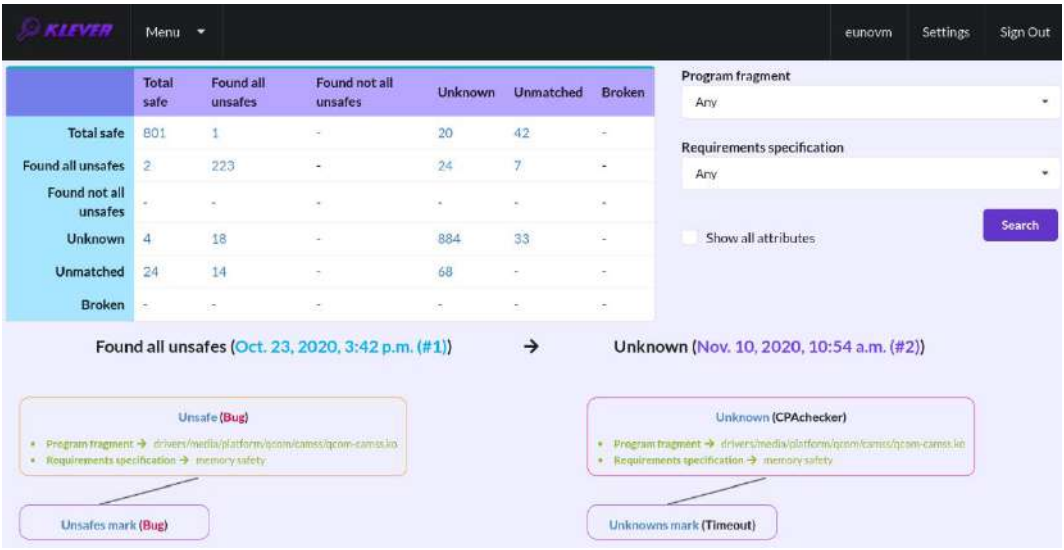


Рис. 4. Сравнение результатов верификации
Fig. 4. Comparison of verification results

Производительность Klever Bridge достаточно высокая: несколько пользователей могут выполнять большинство доступных операций с десятками больших верификационных заданий и сотнями экспертных оценок без существенных задержек. При этом нагрузка на память и на диск незначительна.

Расширенный формат свидетельств нарушений был реализован в инструменте верификации CPAchecker. Другие инструменты верификации не выдают свидетельства нарушений в данном формате, поэтому их визуализация и экспертная оценка могут быть не настолько наглядны и эффективны, как для CPAchecker.

4. Заключение

Опыт практического применения инструментов верификации моделей для больших, быстро развивающихся программ показывает, что недостаточно подготовить верификационные задачи, запустить инструменты верификации и получить результаты верификации, такие как

вердикты, свидетельства нарушений и отчеты о покрытии кода. Экспертам нужны удобные и эффективные инструменты для их хранения, визуализации и анализа.

В данной статье рассмотрены существующие подходы и предложены новые методы экспертной оценки результатов верификации. Предложенные методы были реализованы в системе верификации Klever, которая достаточно успешно используется при проверке различного программного обеспечения. На странице проекта [19] можно найти список возможных улучшений Klever в рассматриваемом направлении. Кроме того, у нас есть планы транслировать свидетельства нарушений других инструментов верификации с помощью CPAchecker, чтобы повысить наглядность их визуализации, а также поддержать их автоматизированную экспертную оценку.

Список литературы / References

[1]. R. Jhala and R. Majumdar. Software model checking. ACM Computing Surveys, vol. 41, issue 4, 2009, article 21, 54 p.
[2]. D. Beyer. Advances in Automatic Software Verification: SV-COMP 2020". Lecture Notes in Computer Science, vol. 12079, 2020, pp. 347-367.
[3]. Definitions and Rules of the 10th International Competition on Software Verification. URL: https://sv-comp.sosy-lab.org/2021/rules.php, accessed 27.11.2020.
[4]. E. Novikov and I. Zakharov. Towards automated static verification of GNU C programs. Lecture Notes in Computer Science, vol. 10742, 2018, pp. 402-416.
[5]. D. Beyer, M. Dangl, D. Dietsch, M. Heizmann, and A. Stahlbauer. Witness validation and stepwise testification across software verifiers. In Proc. of the 10th Joint Meeting on Foundations of Software Engineering, 2015, pp. 721-733.
[6]. D. Beyer, M. Dangl, D. Dietsch, and M. Heizmann. Correctness witnesses: exchanging verification results between verifiers. In Proc. of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2016, pp. 326-337.
[7]. Static Driver Verifier Report. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/devtest/static-driver-verifier-report, accessed 27.11.2020.
[8]. D. Beyer and M. Dangl. Verification-aided debugging: An interactive web-service for exploring error witnesses. Lecture Notes in Computer Science, vol. 9780, 2016, pp. 502-509.
[9]. Clade. URL: https://github.com/17451k/clade, accessed 27.11.2020.
[10]. G.C. Necula, S. McPeak, S.P. Rahul, W. Weimer. CIL: Intermediate Language and Tools for Analysis and Transformation of C Programs. Lecture Notes in Computer Science, vol. 2304, 2002, pp. 213-228.
[11]. Exchange Format for Violation Witnesses and Correctness Witnesses. URL: https://github.com/sosy-lab/sv-witnesses, accessed 27.11.2020.
[12]. Klever extended violation witness format. URL: https://klever.readthedocs.io/en/latest/dev.html#extended-violation-witness-format, accessed 27.11.2020.
[13]. Klever error trace format. URL: https://klever.readthedocs.io/en/latest/dev.html#error-trace-format, accessed 27.11.2020.
[14]. R. Castaño, V. Braberman, D. Garbervetsky, and S. Uchitel. Model checker execution reports. In Proc. of the 32nd IEEE/ACM International Conference on Automated Software Engineering, 2017, pp. 200-205.
[15]. LCOV. URL: https://github.com/linux-test-project/lcov, accessed 27.11.2020.
[16]. Klever code coverage format. URL: https://klever.readthedocs.io/en/latest/dev.html#code-coverage-format, accessed 27.11.2020.
[17]. Klever. URL: https://forge.ispras.ru/projects/klever, accessed 27.11.2020.
[18]. [PATCH] s5p-jpeg: hangle error condition in s5p_jpeg_probe. URL: https://lkml.org/lkml/2020/11/13/673, accessed 27.11.2020.
[19]. Klever Bridge issues. URL: https://tinyurl.com/yyq9ljk8, accessed 27.11.2020.

Информация об авторах / Information about authors

Владимир Анатольевич ГРАТИНСКИЙ – программист отдела технологий программирования. Сфера научных интересов: методы автоматизации анализа и оценки результатов верификации моделей программ с помощью графических интерфейсов.

Vladimir Anatolyevich GRATINSKIY – software developer at the Software Engineering Department. Research interests: methods for automating analysis and evaluation of results of software model checking using graphical interfaces.

Евгений Михайлович НОВИКОВ – кандидат физико-математических наук, старший научный сотрудник и руководитель программных проектов отдела технологий программирования. Его научные интересы включают верификацию моделей программ, спецификацию требований и операционные системы.

Evgeny Mikhailovich NOVIKOV – PhD, Senior Researcher and Software Project Manager at the Software Engineering Department. His research interests include software model checking, requirements specification and operating systems.

Илья Сергеевич ЗАХАРОВ – кандидат физико-математических наук, младший научный сотрудник отдела технологий программирования. Его научные интересы включают методы формальной верификации системного программного обеспечения, в частности операционных систем.

Ilja Sergeevich ZAKHAROV – Ph.D., Junior Researcher at the Software Engineering Department. His research interests include formal verification and model checking of system software and operating systems in particular.

DOI: 10.15514/ISPRAS–2020–32(5)–2

Обнаружение дефекта взаимной блокировки с помощью статического анализа

С.А. Поляков, ORCID: 0000-0002-8542-8035 <inly@ispras.ru>
А.Е. Бородин, ORCID: 0000-0003-3183-9821 <alexey.borodin@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. В статье описывается расширение статического анализа программ на основе резюме для поиска ошибок взаимных блокировок потоков. Анализ на основе резюме является популярным подходом для поиска ошибок в программах благодаря высокой производительности и масштабируемости. При этом реализация детекторов поиска взаимных блокировок в таком анализе является нетривиальной, так как в процессе внутрипроцедурного анализа функций отсутствует информация об удерживаемых блокировках выше по стеку вызовов. Для моделирования семантики многопоточных программ используется граф блокировок, который строится во время основного анализа. Граф блокировок является модификацией графа вызовов с добавлением информации об удерживаемых блокировках. После построения графа блокировок запускается детектор обнаружения взаимных блокировок. Как построение графа блокировок, так и алгоритм обнаружения взаимных блокировок, не требуют существенного процессорного времени. На выполненных замерах общее время анализа увеличилось на 4%. По результатам анализа 8 проектов с открытым исходном кодом на языках C/C++/Java общим размером более 14 млн. строк кода предложенный алгоритм показал высокий уровень истинных срабатываний. Описываемые алгоритмы были реализованы в инструменте Svace.

Ключевые слова: статический анализ; символьное выполнение; параллелизм; взаимные блокировки

Для цитирования: Поляков С. А., Бородин А. Е. Обнаружение дефекта взаимной блокировки с помощью статического анализа. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 21-34. DOI: 10.15514/ISPRAS–2020–32(5)–2

Deadlock Detection using Static Analysis

S.A. Polyakov, ORCID: 0000-0002-8542-8035 <inly@ispras.ru>
A.E. Borodin, ORCID: 0000-0003-3183-9821 <alexey.borodin@ispras.ru>

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

Abstract. The paper describes an extension to summary based static program analysis to find deadlock errors. Summary based analysis is a popular approach aimed at the detection of bugs in programs due to its high performance and scalability. At the same time, the implementation of deadlock detectors in such an analysis is nontrivial, because there is no information about the locks held higher in the call stack during the process of function intraprocedural analysis. A lock graph, which is built during the main analysis, is used to model the semantics of multithreaded programs. Lock graph is a modification of call graph which contains additional information about held locks. After the lock graph is built, the deadlock detector is launched. Both the construction of the lock graph and the deadlock detection algorithm do not require significant processor time. On the performed measurements, the total analysis time increased by 4%. Based on the results of the analysis of 8 open source projects in C/C++/Java with a total size of more than 14 million lines of code, the proposed algorithm showed a high level of true positives. The described algorithms were implemented in the Svace tool.

Keywords: static analysis; symbolic execution; concurrency; deadlock freedom

For citation: Polyakov S. A., Borodin A. E. Deadlock Detection using Static Analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 21-34 (in Russian). DOI: 10.15514/ISPRAS-2020-32(5)-2

1. Introduction

Современные вычислительные устройства в основном являются многоядерными и многопроцессорными. Они предоставляют возможность выполнять параллельные программы, эффективность которых выше аналогичных последовательных программ. Параллельная программа представляет из себя несколько взаимодействующих потоков управления, одновременно выполняющих операции. Разработка параллельных программ является нетривиальной задачей. При проектировании параллельных программ, в отличие от однопоточных, могут быть допущены специфические ошибки проектирования (дефекты): состояния гонки и взаимные блокировки. При наличии состояний гонки работа программы зависит от порядка выполнения частей кода, что приводит к искажениям данных, последствия которых могут проявляться в непредсказуемые моменты времени. Взаимные блокировки приводят к одноименной ошибке.

Взаимная блокировка потоков – ошибка, возникающая, когда несколько потоков находятся в состоянии бесконечного ожидания ресурсов, занятых самими этими потоками. В случае двух потоков: первый поток ждет, пока второй освободит ресурс, необходимый для завершения работы первого потока, в то время как второй поток, в свою очередь, ждет освобождения другого ресурса первым.

В статье описывается подход для обнаружения взаимных блокировок в многопоточных программах с помощью статического анализа. Алгоритмы из данной статьи могут быть реализованы в любом статическом анализаторе на основе символьного выполнения с объединением состояний и резюме с обходом графа вызовов «снизу-вверх». В описываемом подходе во время фазы обхода графа вызовов собирается необходимая информация, которая будет использована после обхода всех функций. На используемые алгоритмы накладываются несколько ограничений: найти как можно больше ошибок за ограниченное время при приемлемом уровне ложных срабатываний. При этом допустимы как пропуски ошибок, так и выдача ложных срабатываний. Недопустимы любые требования по подготовке анализируемой программы, либо ограничение используемых конструкций языка. В качестве базового статического анализатора используется инструмент Svace. В данной статье описывается подход для обнаружения взаимных блокировок двух потоков. Однако предложенный подход может быть расширен на случай нескольких потоков.

В качестве анализируемого языка будем использовать императивный язык, содержащий вызовы функций, в том числе по указателю, и операции ветвления. В качестве примитива синхронизации будем использовать мьютекс, который можно заблокировать функцией *lock*, и разблокировать функцией *unlock*. Мы не рассматриваем циклы, поскольку в большинстве случаев для анализа они будут эквивалентны условным выражениям. В реальном коде в теле цикла либо не будет блокировок вообще, либо за блокировкой будет следовать разблокировка мьютекса.

Для сбора необходимой информации используется абстракция «Граф блокировок», содержащая информацию о последовательности заблокированных мьютексов и вызовов функций. Упоминание и краткое описание графа блокировок приведено в статье [1], описывающей особенности анализа Java программ. Поиск ошибок там осуществляется только для стандартного средства управления потоками – конструкции *synchronize*. Использование конструкции *synchronize* аналогично использованию «сбалансированных» вызовов блокировки и разблокировки мьютексов, что упрощает анализ. Под «сбалансированностью» подразумевается следующее правило: цепочка заблокированных мьютексов может быть разблокирована только в порядке, обратном порядку их блокировки.

Данная статья содержит описание модифицированного алгоритма для анализа программ на языках C, C++ и Java, в том числе для «несбалансированных» вызовов блокировки и разблокировки мьютексов.

2. Базовый анализ

Предлагаемый подход к поиску ошибок взаимоблокировок является расширением статического анализа на основе резюме. В качестве базового анализатора в данной работе был использован анализатор Svace. Используемые алгоритмы могут быть реализованы в любом другом похожем анализе. В данном разделе опишем какими свойствами должен обладать базовый анализ, чтобы его можно было расширить.

Анализ на основе резюме является достаточно популярным, так как имеет ряд существенных преимуществ. При таком анализе функции анализируемой программы обходятся «снизу-вверх» по графу вызовов. После анализа функции создаётся ее резюме, описывающее поведение функции, которое дальнейшем используется при анализе вызова функции. При отсутствии циклов в графе вызовов каждая функция посещается только один раз. Многие существующие реализации игнорируют циклы в графе вызовов. Анализы, учитывающие циклы в графе вызовов, анализируют некоторые функции более одного раза, но как правило количество проходов не велико. Анализ на основе резюме имеет высокую произоводительность и масштабируемость. Скорость анализа достигается за счёт того, что каждая функция анализируется только один раз. Масштабируемость получается за счёт ограничения размера резюме. Кроме того, анализ на основе резюме легко распараллеливается, так как информация передается только от вызываемых функций к вызывающим, и поэтому значительное количество функций могут анализироваться независимо. Дополнительным преимуществом анализа на основе резюме является возможность анализировать библиотеки, а не только целые программы. Из"=за множества преимуществ анализа на основе резюме, его довольно часто используют в инструментах статического анализа. Например, следующие инструменты используют анализ на основе резюме: PREfix [2], Archer [3], Saturn [4], Calysto [5], SharpChecker [6], Coverity [7].

Базовый анализ функции должен обладать потоковой чувствительностью, т.е. учитывать порядок инструкций. Кроме этого, он должен отслеживать значения переменных. Мы будем рассматривать символьное выполнение с объединением абстрактных состояний в точках объединения путей. В базовом анализе потребуется реализация некоторых вспомогательных анализов.

3. Граф блокировок

3.1 Расширение анализа на основе резюме

Во время анализа функции мьютексы, которые были заблокированы выше по графу вызовов, не известны, поэтому анализ на основе резюме плохо подходит для поиска взаимных блокировок. Для поиска таких ошибок предлагается расширить базовый анализ на основе резюме с помощью использования графа блокировок. В предложенном анализе поиск ошибок осуществлялся в две фазы.

- Фаза обхода функций программы по графу вызовов «снизу-вверх». На этой фазе происходит построение графа блокировок. Во время обхода функции доступны резюме вызываемых функций, содержащие информацию о блокировке и разблокировке мьютексов.
- Анализ графа блокировок для обнаружения ошибок в программе. На данной фазе информация, содержащаяся в резюме уже не доступна, что сделано из соображений

производительности, чтобы не требовалось хранить резюме функций на протяжении всего анализа.

Таким образом, алгоритм поиска взаимных блокировок представляет собой надстройку над анализом на основе резюме. Вторая фаза анализа графа блокировок не занимает значительного времени и не требует много памяти. Построение графа блокировок на первой фазе незначительно увеличивает время первой фазы, и это увеличение пропорционально количеству инструкций в программе. Время анализа первой фазы зависит от реализации в немодифицируемом анализаторе.

Первая фаза может занимать существенное время в зависимости от алгоритмов, реализованных в оригинальном анализаторе.

3.2 Определение графа блокировок

Граф блокировок приближенно описывает параллельную программу и используется для обнаружения взаимных блокировок. Будем считать, что в программе присутствует взаимная блокировка двух потоков, если существует такой путь выполнения, на котором последовательно блокируются мьютексы *l*1 и *l*2, а также существует путь выполнения, на котором последовательно блокируются мьютексы *m*2 и *m*1, причем *m*1 является алиасом *l*1, а *m*2 является алиасом *l*2.

Рассмотрим пример из листинга 1. Внутри функции *foo* происходят последовательные блокировки мьютексов, а затем – разблокировки мьютексов в обратном порядке. При вызовах данной функции из *test*1 и *test*2 в качестве формальных параметров передаются глобальные мьютексы *g*1 и *g*2. В данном примере возможна взаимная блокировка потоков, если функции *test*1и *test*2 выполняются параллельно. Смоделируем ошибочное поведение программы. В одном потоке выполняется функция *test*1, глобальные переменные *g*1 и *g*2 передаются в функцию *foo* в качестве первого и второго параметров соответственно, *g*1 блокируется вызовом *pthread_mutex_lock* на 6 строке, предпринимается попытка заблокировать *g*2 на 7 строке. Параллельно в другом потоке выполняется функция *test*2, глобальные переменные *g*2 и *g*1 передаются в функцию *foo* в качестве первого и второго параметров соответственно, *g*2 блокируется на 6 строке, предпринимается попытка заблокировать *g*1 на 7 строке.

```
1:  #include <pthread.h>
2:
3:  pthread_mutex_t *g1, *g2;
4:
5:  void foo(pthread_mutex_t *p1, pthread_mutex_t *p2) {
6:      pthread_mutex_lock(p1);
7:      pthread_mutex_lock(p2);
8:      pthread_mutex_unlock(p2);
9:      pthread_mutex_unlock(p1);
10: }
11:
12: void test1() {
13:     foo(g1, g2);
14: }
15:
16: void test2() {
17:     foo(g2, g1);
18: }
19: // alias.c
```

Листинг 1. Пример взаимной блокировки
Listing 1. Deadlock example

Граф блокировок – ориентированный граф, содержащий информацию о вызовах функций и захвате мьютексов. Вершина в графе блокировок сопоставляется событию в выполнении программы одного из следующих типов: блокировка мьютекса; передача мьютекса или

содержащего его объекта в функцию, где в дальнейшем мьютекс будет заблокирован; вызов функции; начало выполнения функции. Все вершины аннотируются строками в исходном коде. Заметим, что в графе блокировок явно не содержится информация об освобождения мьютексов.

Вершина типа **VLock** описывает блокировку мьютекса, она содержит информацию о мьютексе. Вершина **VLock** означает, что мьютекс мог быть заблокирован. Вершина типа **VCall** описывает вызов функции. Вершина типа **VStart** описывает начало выполнения функции. Вершины последних двух типов содержат информацию о функции, с помощью которой ее можно идентифицировать. Вершина типа **VLockAlias** ⊂ **VLock** описывает передачу мьютекса в функцию. Вершина такого типа описывает захват блокировки внутри вызова функции, а также является «алиасом» вершин типа **VLockAlias** или **VLock**. Вершина типа **VLockAlias** содержит информацию о мьютексе и его алиасе.

На изображениях графов блокировок, приведенных в данной работе, вершины типа **VLock** изображены в виде октагона, вершины типа **VAliasLock** – октагона с двойным контуром, вершины типа **VCall** – прямоугольника, а вершины типа **VStart** – пентагона.

Ребра в графе блокировок могут быть следующих видов.

- **VLock**[$l1$] → **VLock**[$l2$] Ребро между вершиной, описывающей захват мьютекса $l1$, и вершиной, описывающей захват мьютекса $l2$. Наличие такого ребра означает, что $l1$ является последним захваченным мьютексом, предшествующей захвату $l2$. Данный факт будем обозначать $l1 \rightarrow l2$.
- **VLock**[l] → **VCall**[p] Ребро между вершиной, описывающей захват блокировки l, и вершиной, описывающей вызов функции p. Наличие такого ребра означает, что l является последней захваченной блокировкой, предшествующей вызову p. Данный факт будем обозначать $l \rightarrow p$.
- **VCall**[p] → **VStart**[p'] Ребро между вершиной p, описывающей вызов функции, и вершиной p', описывающей начало выполнения вызываемой функции. Фактически это ребро от вызова функции к её определению. Такое ребро существует только тогда, когда p и p' идентифицируют одну и ту же функцию.

  Отсутствие исходящих ребёр из **VCall**[p] означает отсутствие информации о вызванной функции. Несколько исходящих ребёр означают несколько альтернатив для разрешения вызова процедуры. Последнее возможно при вызове функции по указателю, когда возможен вызов нескольких функций. Таким образом вершина вызова функций имеет либо пустое множество исходящих ребёр, либо содержит рёбра на исчерпывающее множество возможных определений вызванных функций. Данный факт будем обозначать $p \rightarrow p'$.
- **VStart**[p] → **VLock**[l] Ребро между вершиной, описывающей начало выполнения функции p, и вершиной, описывающей блокировку мьютекса l. Наличие такого ребра означает, что в ГПУ существует путь выполнения, на котором все инструкции выполняются при заблокированном мьютексе l. Данный факт будем обозначать $p \rightarrow l$.
- **VStart**[$p1$] → **VCall**[$p2$] Ребро между вершиной, описывающей начало выполнения функции $p1$, и вершиной, описывающей вызов функции $p2$. Наличие такого ребра означает, что $p1$ вызывает $p2$, не удерживая никакую блокировку. Данный факт будем обозначать $p1 \rightarrow p2$.
- **VCall**[p] → **VLockAlias**[l] Ребро между вершиной, описывающей вызов функции p, и вершиной, описывающей блокировку мьютекса l внутри функции p. Данный факт будем обозначать $p \rightarrow l$.
- **VLockAlias**[l'] → **VLock**[l] Ребро между вершинами, описывающими блокировку мьютекса l, алиасом которого является l'. Наличие такого ребра означает, что мьютекс l' (или объект, его содержащий) был передан в качестве фактического параметра в некоторую функцию p, внутри которой мьютекс l был заблокирован, причем l (или

объект, его содержащий) является формальным параметром функции *p*, и *l'* является алиасом *l*. Данный факт будем обозначать *l'* → *l*.

- **VLockAlias**[*l*1] → **VLockAlias**[*l*2] Ребро между вершинами, описывающими мьютексы *l*1 и $l2$, которые являются алисами. Наличие такого ребра означает, что мьютекс *l*1 (или объект, его содержащий) был передан в качестве фактического параметра в некоторую функцию *p*1, внутри которой мьютекс *l*2 был передан в другую функцию *p*2, причем *l*2 (или объект, его содержащий) является формальным параметром функции *p*1 и *l*1 является алиасом *l*2. Внутри функции *p*2 мьютекс *l'*, являющийся алиасом *l*2, может как блокироваться, так и передаваться в качестве параметра дальше, но в конечном итоге окажется заблокированным. Данный факт будем обозначать *l*1 → *l*2.

Заметим, что только ребра типа **VCall**[*p*] → **VLockAlias**[*l*] могут быть добавлены в граф блокировок во время обхода графа вызовов программы. Ребра типа **VLockAlias**[*l'*] → **VLock**[*l*] и типа **VLockAlias**[*l*1] → **VLockAlias**[*l*2] создаются после обхода графа вызовов перед анализом графа блокировок. Для создания таких ребер используется информация, сохраненная в вершинах **VLockAlias**, а сами ребра создаются с целью оптимизации анализа графа блокировок.

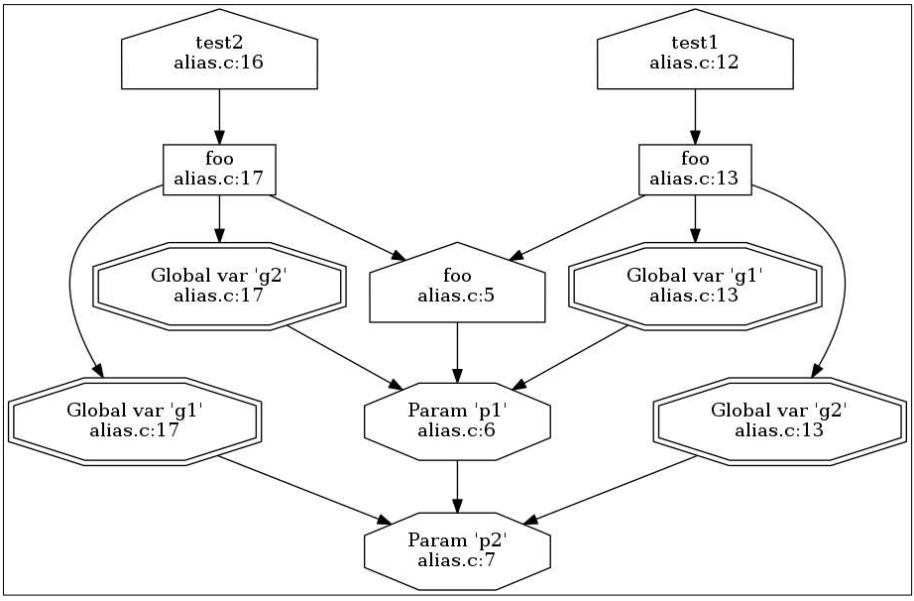


Рис. 1. Граф блокировок, соответствующий листингу 1
Fig. 1. Lock graph corresponding to listing 1

Граф блокировок, который соответствует листингу 1, изображен на рис. 1. В результате анализа графа блокировок будет выдано предупреждение о возможной взаимной блокировке на основании следующих фактов.

- Внутри функции *foo* мьютексы блокируются в следующем порядке: *p*1 → *p*2.
- В случае вызова функции *foo* из *test*1 мьютекс *g*1 соответсвует *p*1, а мьютекс *g*2 соответствует *p*2. Получаем пару заблокированных мьютексов в порядке: *g*1 → *g*2.
- В случае вызова функции *foo* из *test*2 мьютекс *g*2 соответсвует *p*1, а мьютекс *g*1 соответствует *p*2. Получаем пару заблокированных мьютексов в порядке: *g*2 → *g*1.
- В программе существует путь выполнения, на котором последовательно блокируются мьютексы *g*1 и *g*2, а также существует путь, на котором последовательно блокируются *g*2 и *g*1.

3.3 Вспомогательные анализы

Для построения графа блокировок необходимо несколько вспомогательных анализов. Анализы устанавливают некоторые свойства программы. Для описания свойств используются атрибуты. Атрибуты могут ассоциироваться с ребром в ГПУ, либо с идентификатором значения.[1]

Атрибут типа **LockStatus**, с одной стороны, описывает состояние мьютекса в некоторой точке программы, с другой стороны – эффект от вызова некоторой функции, то есть каким образом вызов функции влияет на состояние мьютекса. Возможные состояния описываются множеством значений данного атрибута.

- *default* – над мьютексом не выполнялись операции блокировки/разблокировки. Вызов функции не влияет на состояние мьютекса.
- *lock* – мьютекс заблокирован. Вызов функции блокирует мьютекс.
- *may_lock* – мьютекс возможно заблокирован, то есть мьютекс блокируется хотя бы на одном пути выполнения, но не на всех возможных путях. Вызов функции возможно блокирует мьютекс.
- *unlock* – мьютекс разблокирован, то есть над мьютексом была выполнена операция разблокировки. Вызов функции разблокирует мьютекс.
- *may_unlock* – мьютекс возможно разблокирован. Вызов функции возможно разблокирует мьютекс.
- *unlock_then_lock* – мьютекс был разблокирован, а затем заблокирован. Вызов функции разблокирует, а затем заблокирует мьютекс.
- *unlock_then_may_lock* – мьютекс был разблокирован, а затем возможно заблокирован. Вызов функции разблокирует, а затем возможно заблокирует мьютекс.
- *may_unlock_then_lock* – мьютекс возможно был разблокирован, а затем заблокирован. Вызов фукнции возможно разблокирует, а затем заблокирует мьютекс.
- *may_unlock_then_may_lock* – мьютекс возможно был разблокирован, а затем возможно заблокирован. Вызов фукнции возможно разблокирует, а затем возможно заблокирует мьютекс.

Значения данного атрибута ассоциируются с идентификаторами значений.

Введем функцию *val*, которая для каждой переменной программы возвращает ассоциированный с ней идентификатор значения.

```
1:  #include <pthread.h>
2:
3:  pthread_mutex_t *g;
4:
5:  void safe();
6:  void unsafe();
7:
8:  void may_lock_f(int lock) {
9:      if (lock) {
10:         pthread_mutex_lock(g);
11:     }
12: }
13:
14: void may_unlock_f(int unlock) {
15:     if (unlock) {
16:         pthread_mutex_unlock(g);
17:     }
18: }
```

[1] Под идентификатором значения будем понимать символьную переменную в терминах символьного выполнения.

```
19:
20: void test1(int safe) {
21:     may_lock_f(safe);
22:     unsafe();
23:     may_unloc_f(safe);
24: }
25:
26: void do_unsafe(void (*do_smth_ptr)()) {
27:     pthread_mutex_unlock(g);
28:     (*do_smth_ptr)();
29:     pthread_mutex_lock(g);
30: }
31:
32: void test2(int all_safe) {
33:     pthread_mutex_lock(g);
34:     safe();
35:     if (all_safe) {
36:         unsafe();
37:     } else {
38:         do_unsafe(&unsafe);
39:     }
40:     pthread_mutex_unlock(g);
41: }
```

Листинг 2. Пример распространения атрибута ***LockStatus***
Listing 2. Example of ***LockStatus*** *attribute flow*

Рассмотрим распространение данного атрибута на примере листинга 2. Функция *may_lock_f* блокирует мьютекс g в зависимости от условия, заданного параметром функции. Внутри данной функции значение g не меняется, и $val(g)$ в каждой вершине ГПУ будет возвращать один и тот же идентификатор значения. Обозначим его v_g^{ml}. В ГПУ образуется ветвление из-за условного перехода на строке 9. В первом базовом блоке не будет ни одной инструкции, и значение атрибута, ассоциированное с v_g^{ml}, не будет изменено, то есть останется равным *default*. Однако во втором блоке мьютекс g будет заблокирован после вызова функции *pthread_mutex_lock* на строке10, и значение атрибута станет *lock*. После слияния путей в ГПУ на строке 11, согласно описанной функции *joinval*[**LockStatus**], получим значение *may_lock*. Данное значение, ассоциированное с v_g^{ml}, попадет в резюме функции *may_lock_f*. Аналогично, в резюме функции *may_unlock_f* попадет значение *may_unlock*, ассоциированное с v_g^{mu}. Функция *test*1 вызывает функцию *unsafe*, блокируя мьютекс g при условии, что *safe* != 0. Данный факт реализован посредством вызова *may_lock_f* на строке 21. Внутри функции *test*1 значение g не меняется, обозначим идентификатор значения $val(g) = v_g^{t1}$. До вызова *may_lock_f* значение атрибута, ассоциированное с v_g^{t1}, равно *default*. После вызова данной функции значение изменяется на *may_lock*, поскольку в момент вызова v_g^{t1} соответствует v_g^{ml}, а с v_g^{ml} в резюме ассоциировано значение *may_lock*. Получаем значение *default* после вызова функции *may_unlock_f* на строке 23, поскольку в момент вызова v_g^{t1} соответствует v_g^{mu}, а с v_g^{mu}в резюме ассоциировано значение *may_unlock*.

Функция *do_unsafe* вызывает функцию по указателю, переданному в качестве параметра. Она спроектирована таким образом, что вначале разблокирует мьютекс g, затем вызовет требуемую функцию, а затем заново заблокирует g. В резюме *do_unsafe* попадет значение *unlock_then_lock*, ассоциированное с идентификатором v_g^{do}. Внутри функции *test*2 значение g не меняется, обозначим идентификатор значения $val(g) = v_g^{t2}$. После вызова функции *pthread_mutex_lock* на строке 33 значение атрибута, ассоциированное с v_g^{t2}, равно *lock*. Данное значение распространится в каждый базовый блок, полученный в результате условного перехода на строке 35. В первом базовом блоке состояние мьютекса не меняется, следовательно, не поменяется и значение атрибута. Во втором базовом блоке вызывается

do_unsafe. Учитывая резюме функции (с v_g^{do}, который соответствует v_g^{t2}, ассоциировано значение *unlock_then_lock*) и значение *lock* атрибута до вызова, получим новое значение *lock*, ассоциированное с v_g^{t2}. После вызова функции *pthread_mutex_unlock* на строке 40, значение атрибута вновь станет равным *default*.

Атрибут типа **LocksOnPath** содержит информацию о том, какие мьютексы заблокированы на ребре, входящем в конкретную вершину ГПУ. Значением атрибута данного типа является множество всех возможных последовательностей блокировок. Формально записывается $\{LockChain_1, LockChain_2, \ldots, LockChain_n\}$,

$LockChain \equiv [lock_1 \rightarrow lock_2 \rightarrow \ldots \rightarrow lock_m]$,

$lock \equiv (lId, vId, trace)$, где

$vId \in V$ – множество идентификаторов значений,

$lId \in L$ – множество идентификаторов мьютексов, *trace* – последовательность точек в программе;

Значением по умолчанию для атрибута **LocksOnPath** является множество из одного элемента – пустой последовательности блокировок {[]}, которая означает отсутствие заблокированных мьютексов на ребре ГПУ. Значения данного атрибута ассоциируются с ребрами в ГПУ.

Реализация функции *joinval*[**LocksOnPath**] (функции определения значения атрибута при слиянии путей в ГПУ) тривиальна – поскольку значением атрибута является множество цепочек блокировок, то определим данную функцию следующим образом:

$Joinval\ (LocksOnPath_l, LocksOnPath_r) = LocksOnPath_l \cup LocksOnPath_r$.

Данный атрибут является межпроцедурным, то есть его значения хранятся в резюме и в случае вызова функции, значение атрибута меняется и в вызывающей функции. Причем для создания нового значения используется как значение данного атрибута в резюме, так и значения атрибута **LockStatus** из резюме, которые соответствуют идентификаторам значений, сохраненных в тройке *lock*.

```
1:  #include <pthread.h>
2:
3:  pthread_mutex_t *g1, *g2;
4:
5:  void unlock_then_lock() {
6:      pthread_mutex_unlock(g1);
7:      pthread_mutex_lock(g1);
8:  }
9:
10: void test(int cond) {
11:     // p1
12:     pthread_mutex_lock(g1);
13:     // p2
14:     pthread_mutex_lock(g2);
15:     // p3
16:     unlock_then_lock();
17:     // p4
18:     if (cond) {
19:         pthread_mutex_unlock(g1);
20:         pthread_mutex_unlock(g2);
21:         // p5
22:     }
23:     // p6
24: }
```

Листинг 3. Пример распространения атрибута ***LocksOnPath***
Listing 3. Example of ***LocksOnPath*** *attribute flow*

Рассмотрим распространение данного атрибута на примере листинга 3. В точке *p*1 на ребре ГПУ, входящем в вершину инструкции вызова функции *pthread_mutex_lock* на строке 12, атрибут **LocksOnPath** имеет значение по умолчанию {[]}. В точке *p*2 атрибут принимает {[*lockId*(*val*(*g*1))]} после вызова *pthread_mutex_lock* на строке 12. На строке 14 функция *pthread_mutex_lock* вызывается при уже заблокированном мьютексе~$g1$, таким образом, в точке~$p3$ атрибут принимает значение {[*lockId*(*val*(*g*1)) → *lockId*(*val*(*g*2))]}. Заметим, что функция *unlock_then_lock*, вызываемая на строке 16, сначала разблокирует, а затем опять заблокирует мьютекс *g*1. Таким образом, значение атрибута изменится на {[*lockId*(*val*(*g*2)) → *lockId*(*val*(*g*1))]} в точке *p*4. Далее в ГПУ образуется ветвление из-за условного перехода на строке 18. В первом базовом блоке не будет ни одной инструкции, и значение атрибута не будет изменено. Однако во втором блоке оба мьютекса *g*1 и *g*2 будут разблокированы, и значение атрибута в точке *p*5 станет равным значению по умолчанию. В точке *p*6 после слияния путей в ГПУ, согласно описанной функции *joinval*[**LocksOnPath**], получим значение {[], [*lockId*(*val*(*g*2)) → *lockId*(*val*(*g*1))]}.

Атрибут типа **PathToLock** содержит информацию, необходимую для идентификации мьютекса, являющегося алиасом ранее заблокированного мьютекса.

3.4 Построение графа блокировок

Во время анализа графа вызовов для каждой функции запускается анализ графа потока управления (ГПУ). Для каждой функции во время анализа ГПУ строится локальный граф блокировок, который добавляется в глобальный граф блокировок по завершении анализа текущей функции. Локальные графы добаляются в глобальный путем создания ребер типа **VCall**[*p*] → **VStart**[*p'*].

Опишем построение локального графа блокировок.

Перед началом обхода графа потока управления анализируемой функции *f* в граф добавляется вершина **VStart**[*f*].

Встретив инструкцию вызова функции *p* добавляем в граф вершину **VCall**[*p*]. Пути из вершины **VStart**[*f*]. в нее строятся с использованием значения атрибута **LocksOnPath** на ребре ГПУ, входящем в вершину, соответствующую инструкции вызова функции *p*. Соединим добавленную вершину с вершиной **VStart**[*f*] в случае если атрибут **LocksOnPath** имеет значение по умолчанию. В ином случае ищем путь из вершины **VStart**[*f*], проходящий через вершины типа **VLock**, описывающие мьютексы, которые соответствуют одной из цепочек мьютексов, содержащейся в атрибуте. Из последней вершины, лежащей на пути, проводим ребро к добавленной ранее вершине **VCall**[*p*]. Повторяем данную процедуру для каждой цепочки мьютексов, содержащейся в значении атрибута **LocksOnPath**. Если в резюме вызываемой функции *p* атрибут *PathToLock* имеет значение, не равное значению по умолчанию, то в граф блокировок добавляется вершина типа **VLockAlias**. Проводится ребро из ранее созданной вершины **VCall**[*p*] в вершину **VLockAlias**.

Встретив инструкцию вызова функции, в резюме которой значение атрибута **LocksOnPath** не равно значению по умолчанию, для каждой цепочки мьютексов для каждого мьютекса в цепочке создаем вершину типа **VLock**. Способ ее добавления в граф блокировок аналогичен способу добавления вершины типа **VCall**.

3.5 Анализ графа блокировок

Для обнаружения дефекта взаимной блокировки потоков предлагается следующий алгоритм.

1. Обходим граф блокировок в глубину. В каждый момент времени отслеживается множество захваченных блокировок на пути к текущей вершине. Также отслеживается множество вызванных функций на пути к текущей вершине.
2. Встретив вершину захвата блокировки **VLock**, запускаем процедуру поиска её ближайших потомков такого же типа, а также процедуру поиска ее алиасов **VLockAlias**.

Данные процедуры также отслеживают множество захваченных блокировок и множество вызванных функций.

3. При добавлении новой пары ($l1$, $l2$) проверяется, была ли инвертированная пара ($l2$, $l1$) ранее добавлена в коллекцию.
   - Если такой пары ($l2$, $l1$) не было, то пара ($l1$, $l2$) добавляется в коллекцию.
   - Если такая пара была ранее добавлена в коллекцию, то, пропуская её добавление в коллекцию, выполняются несколько дополнительных проверок, и, если проверки будут пройдены, будет выдано сообщение о дефекте.

В качестве примера дополнительной проверки для подавления ложных срабатываний рассмотрим ситуацию, когда в программе используется так называемый guarding/gate lock – такая блокировка специального мьютекса, что несколько потоков не могут одновременно достигнуть области возможной взаимной блокировки потоков. Для предотвращения выдачи ложного предупреждения необходимо выполнить следующую проверку. Для каждой вершины, являющейся первой в своей паре, собрать множество доминирующих её вершин, имеющих тип **VLock**, либо **VAliasLock**. Если пересечение полученных множеств не пусто, необходимо отменить выдачу предупреждения.

4. Результаты

В табл. 1 приведены проекты, на которых был протестирован предложенный в работе алгоритм поиска взаимных блокировок. Также в таблице приведены данные о размерах проектов и указано время анализа каждого отдельного проекта.

Анализ проектов проводился на машине со следующими характеристиками: 4 восьмиядерных процессора Intel Xeon E5-2680, 128 Gb RAM. Работа модулей, предназначенных для поиска дефектов многопоточности, занимала не более 4% времени всего анализа.

Табл. 1. Оценка производительности
Table 1. Performance evaluation

Проект	Строк кода, тыс.	Функций, тыс.	Время анализа, мин.
Android 5.2 (Java)	4364	489	40
Android 5.2 (C/C++)	8561	1147	164
jenkins	121	26	5.36
tomcat	299	28	11.41
cassandra	297	49	9.22
flink	363	36	4.38
phoenix	234	32	10.55
storm	188	21	2.41

В табл. 2 приведены результаты работы алгоритма поиска взаимных блокировок. Указаны только те из проанализированных проектов, в исходном коде которых были найдены дефекты данного типа.

Алгоритм показал низкий уровень ложных срабатываний. Однако в некоторых случаях оказалось невозможно определить является ли выданное предупреждение истинным или ложным, ввиду сложной логики работы некоторых частей проектов и запутанности их исходного кода.

Табл. 2. Результаты работы алгоритма поиска взаимных блокировок
Table 2. Deadlock algorithm results

Проект	Истинные	Ложные	Сложно определить
Android 5.2 (Java)	34	11	21

Android 5.2 (C/C++)	72	47	63
jenkins	4	0	2
tomcat	2	1	2
cassandra	2	0	1
flink	2	9	9

Также был проанализирован исходный код самого анализатора Svace на наличие состояний взаимной блокировки потоков. Однако таковые не были найдены.

5. Похожие работы

Подходы к обнаружению ошибок синхронизации потоков в параллельных программах можно разделить на две группы: статические и динимаческие методы.

Инструменты, использующие динамический подход [8-11], анализируют только те пути выполнения программы, которые она проходит на тестовых запусках. Также агрессивная инструментация кода, которая сопровождает динамический анализ, исключает использование данного подхода на низкоуровневом коде: операционные системы, драйверы устройств – то есть в тех случаях, когда ошибки многопоточности наиболее опасны. Динамический анализ может влиять на взаимодействие потоков, из-за чего дефект может быть пропущен в ходе анализа. Преимущество же динамического подхода заключается в высокой точности анализа, так как в момент выполнения программы детектор имеет точные сведения о значениях переменных и существующих алиасах. Однако точность достигается путем больших накладных расходов – скорость работы программы, которую инспектирует динамический детектор, снижается в десятки раз, а потребление памяти растет.

Инструменты, использующие статический подход, не вмешиваются в работу программы и не замедляют скорость ее выполнения. Кроме того, они анализируют все пути выполнения программы, даже те, которые выполняются крайне редко. Главным недостатком статического подхода является невысокая точность анализа.

В работе [12] описывается система типов для языка программирования Java. Она разработана с целью предотвратить и взаимные блокировки, и состояния гонки. С помощью определенной в данное работе системы типов программисту необходимо описать используемые в программе механизмы синхронизации потоков, то есть аннотировать программу. Отсутствие в программе взаимных блокировок обеспечивается с помощью введения классов эквивалентности на множестве блокировок и определения частичного порядка на множестве введенных классов. Несмотря на то, что такая система типов достаточно эффективна, данный подход не используется в промышленном программировании по двум причинам. Во-первых, масштабное «ручное» аннотирование кода, необходимое в данном случае, требует значительных затрат человеческих ресурсов. Во-вторых, такая система типов требует либо специальный компилятор, либо инструмент, который будет транслировать корректно типизированную программу в Java байт-код.

В анализаторе Chord [13, 14] используется потоко-чувствительный анализ для обнаружения взаимных блокировок и состояний гонки в программах на языке Java. Алгоритм обнаружения взаимных блокировок использует комбинацию нескольких статических анализов, каждый из которых аппроксимирует различные необходимые условия существования дефектов взаимной блокировки в исходном коде программ. Авторы выделяют шесть задач, которые эффективно могут быть решены методами статического анализа. Первая задача – задача достижимости: может ли поток выполнения достигнуть точки захвата объекта блокировки B после того, как он уже завладел объектом блокировки A. Вторая задача – анализ указателей, определяющий, какие указатели и переменные в программе ссылаются на один и тот же участок памяти. Третья задача – анализ сохраненных значений: доступна ли блокировка, захваченная одним потоком выполнения, из других потоков. Четвертая задача – задача

определения следующего факта: могут ли два потока выполнения одновременно достичь точек, где первый поток попытается завладеть блокировкой, удерживаемой вторым потоком, а второй поток попытается завладеть блокировкой, удерживаемой первым потоком. Пятая и шестая задачи сформулированы с целью уменьшения числа ложных предупреждений, которые могут появиться по двум причинам. Если поток выполнения пытается завладеть блокировкой, которой он уже завладел ранее, то взаимная блокировка произойти не может, поскольку спецификация языка Java не запрещает этого. Также взаимная блокировка невозможна, если два потока владеют общей блокировкой (guarding/gate lock). Каждая из шести задач является в общем случае неразрешимой. Поэтому любое решение отдельной задачи является либо неточным, либо неполным. Предложенный в статье алгоритм консервативно аппроксимирует первые четыре задачи. Однако для решения последних двух задач используется неконсервативные алгоритмы, поэтому некоторые случаи взаимных блокировок могут быть пропущены.

В работах [15, 16] используется метод проверки модели. Идея проверки модели концептуально очень проста: исследовать все возможные пути выполнения для всех возможных значений переменных, чтобы определить, могут ли возникать те или иные виды нежелательного поведения. Конечно, сформулированная таким образом задача является алгоритмически неразрешимой. По этой причине создаются такие абстракции, как поток управления или значения данных для конкретной программы, и именно эта модель параллельной программы изучается.

6. Заключение

В данной работе описан граф блокировок для многопоточных программ, написанных на языках C/C++/Java. Во время анализа, в графе блокировок накапливается информация, необходимая для поиска дефектов, связанных с некорректной синхронизацией потоков. Данный граф представляет из себя модификацию графа вызовов – в него добавляются специальные вершины, содержащие информацию об инструкциях блокировки, полученную во время обхода графа потока управления.

Построение такого графа требует незначительное количество ресурсов – памяти и процессорного времени.

Основные результаты работы.

- Предложен граф блокировок для многопоточных программ, расширяющий граф вызовов программы вершинами с информацией об инструкциях блокировки.
- Предложен алгоритм, выявляющий взаимные блокировки потоков.
- Разработанные алгоритмы реализованы в статическом анализаторе Svace, демонстрируют высокий уровень истинных срабатываний (60-90%) и занимают не более 4% времени работы всего инструмента.

Главное преимущество предложенного в работе подхода заключается в том, что описанные алгоритмы возможно реализовать в любом анализаторе, использующем анализ «снизу-вверх» на основе резюме. Такой подход является популярным из-за высокой скорости и масштабируемости анализа. Однако статические анализаторы, нацеленные на обнаружение взаимных блокировок потоков, используют обход графа вызовов сверху-вниз, что упрощает построение и анализ структур, аналогичных графу блокировок. Использование обхода сверху-вниз негативно сказывается на скорости и масштабируемости анализа, а также возможностях обнаружения других типов дефектов, не связанных с неправильной синхронизацией потоков.

Другими преимуществами предложенного подхода являются отсутствие необходимости ручного аннотирования кода, возможность анализировать как полные программы, так и отдельные библиотеки.

Список литературы/References

[1] А.П. Меркулов, С.А. Поляков, А.А. Белеванцев. Анализ программ на языке Java в инструменте Svace. Труды ИСП РАН, том 29, вып. 3, 2017 г., стр. 57-74 / A.P. Merkulov, S.A. Polyakov, A.A. Belevantsev. Supporting Java programming in the Svace static analyzer. Trudy ISP RAN/Proc.ISP RAS, vol.29, issue 3, 2017, pp. 57-74 (in Russian). DOI: 10.15514/ISPRAS-2017-29(3)-5.

[2] W.R. Bush, J D. Pincus, and D.J. Sielaff. A static analyzer for finding dynamic programming errors. Software: Practice and Experience, vol. 30, no. 7, 2000, pp. 775-802.

[3] Y. Xie, A. Chou, and D. Engler. Archer: using symbolic, path-sensitive analysis to detect memory access errors. In Proc. of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering, 2003, pp. 327-336.

[4] I. Dillig, T. Dillig, and A. Aiken. Static error detection using semantic inconsistency inference. In Proc. of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2007, pp. 435-445.

[5] D. Babic and A.J. Hu. Calysto: scalable and precise extended static checking. In Proc. of the 30th international conference on Software engineering, 2008, pp. 211-220.

[6] В.К. Кошелев, И.А. Дудина, В.Н. Игнатьев, А.И. Борзилов. Чувствительный к путям поиск дефектов в программах на языке C# на примере разыменования нулевого указателя. Труды ИСП РАН, том 27, вып. 5, 2015 г., стр. 59-86 / V. Koshelev, I. Dudina, V. Ignatyev, A. Borzilov. Path-Sensitive Bug Detection Analysis of C# Program Illustrated by Null Pointer Dereference. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 5, 2015, pp.59-86 (in Russian). DOI: 10.15514/ISPRAS-2015-27(5)-5.

[7] McPeak, C.-H. Gros, and M. K. Ramanathan. Scalable and incremental software bug detection. In Proc. of the 2013 9th Joint Meeting on Foundations of Software Engineering, 2013, pp. 554-564.

[8] R. Agarwal and S. D. Stoller. Run-time detection of potential deadlocks for programs with locks, semaphores, and condition variables. In Proc. of the 2006 Workshop on Parallel and Distributed Systems: Testing and Debugging, 2006, pp. 51-60.

[9] K. Havelund. Using runtime analysis to guide model checking of java programs. Lecture Notes in Computer Science, vol. 1885, 2000, pp. 245-264.

[10] S. Savage, M. Burrows, G. Nelson, P. Sobalvarro, and T. Anderson. Eraser: a dynamic data race detector for multithreaded programs. ACM Transactions on Computer Systems (TOCS), vol. 15, no. 4, 1997, pp. 391-411.

[11] T. Elmas, S. Qadeer, and S. Tasiran. Goldilocks: a race and transaction-aware java runtime. ACM SIGPLAN Notices, vol. 42, no. 6, 2007, pp. 245-255.

[12] C. Boyapati and M. Rinard. A parameterized type system for race-free java programs. In Proc. of the 16th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, 2001, pp. 56-69.

[13] M. Naik, C.-S. Park, K. Sen, and D. Gay. Effective static deadlock detection. In Proc. of the IEEE 31st International Conference on Software Engineering, pp. 386-396.

[14] M. Naik, A. Aiken, and J. Whaley. Effective static race detection for java. In Proc. of the 27th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2006, pp. 308-319, 2006.

[15] S. D. Stoller. Model-checking multi-threaded distributed java programs. Lecture Notes in Computer Science, vol. 1885, 2000, pp. 224-244.

[16] T.A. Henzinger, R. Jhala, and R. Majumdar. Race checking by context inference. In Proc. of the ACM SIGPLAN 2004 conference on Programming Language Design and Implementation, 2004, pp. 1-13.

Информация об авторах / Information about authors

Сергей Андреевич ПОЛЯКОВ – сотрудник. Сфера научных интересов: статический анализ, параллелизм, JVM языки.

Sergey Andreevich POLYAKOV – employee. Research interests: static analysis, concurrency, JVM languages.

Алексей Евгеньевич БОРОДИН – кандидат физико-математических наук, научный сотрудник. Сфера научных интересов: статический анализ исходного кода программ для поиска ошибок.

Alexey Evgenevich BORODIN – PhD, researcher. Research interests: static analysis for finding errors in source code.

DOI: 10.15514/ISPRAS–2020–32(5)–3

Разработка компиляторов предметно-ориентированных языков для спецпроцессоров

П.Н. Советов, ORCID: 0000-0002-1039-2429 <peter.sovietov@gmail.com>
МИРЭА – Российский технологический университет,
119454 г. Москва, проспект Вернадского, д. 78

Аннотация. В составе современных вычислительных систем все чаще используются аппаратные спецпроцессоры, программируемые на предметно-ориентированных языках. Популярность набирает подход compiler-in-the-loop, предполагающий совместную разработку спецпроцессора и компилятора. При этом традиционный инструментарий (GCC и LLVM) оказывается недостаточным для быстрой разработки оптимизирующих компиляторов, порождающих целевой код нерегулярной архитектуры со статическим параллелизмом операций. В статье предлагается использовать методы решения NP-полных задач для реализации машинно-зависимых фаз компиляции. Фазы осуществляются на основе сведения к задаче SMT, что дает возможность избавиться при построении компилятора от эвристических и приближенных подходов, требующих трудоемкой программной реализации. В частности, с использованием SMT-решателя предлагается реализовать синтез правил машинно-зависимой оптимизации, выбор команд, планирование команд и распределение регистров. Обсуждаются вопросы апробации разработанных методов и алгоритмов на примере компилятора для спецпроцессора с набором команд, ускоряющим реализации алгоритмов низкоресурсных шифров из области Интернета вещей. Полученные результаты компиляции и программного моделирования для 8 криптоалгоритмов и 3 вариантов спецпроцессора (CISC, VLIW и вариант delayed load) демонстрируют практическую применимость предложенного подхода.

Ключевые слова: предметно-ориентированный язык; компилятор; спецпроцессор; SMT-решатель; выбор команд; планирование команд; распределение регистров

Для цитирования: Советов П.Н. Разработка компиляторов предметно-ориентированных языков для спецпроцессоров. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 35-56. DOI: 10.15514/ISPRAS–2020–32(5)–3

Accelerating the Development of DSL Compilers for Specialized Processors

P.N. Sovietov, ORCID: 0000-0002-1039-2429 <peter.sovietov@gmail.com>
MIREA – Russian technological university,
78 Vernadsky Avenue, Moscow 119454

Abstract. Specialized processors programmable in domain-specific languages are increasingly used in modern computing systems. The compiler-in-the-loop approach, based on the joint development of a specialized processor and a compiler, is gaining popularity. At the same time, the traditional tools, like GCC and LLVM, are insufficient for the agile development of optimizing compilers that generate target code of an exotic, irregular architecture with static parallelism of operations. The article proposes methods from the field of program synthesis for the implementation of machine-dependent compilation phases. The phases are based on a reduction to SMT problem which allows to get rid of heuristic and approximate approaches, that requires complex software implementation of a compiler. In particular, a synthesis of machine-dependent optimization rules, instruction selection and instruction scheduling combined with register allocation are implemented with help of SMT solver. Practical applications of the developed methods and algorithms are illustrated by the

example of a compiler for a specialized processor with an instruction set that accelerates the implementation of lightweight cryptography algorithms in the Internet of Things. The results of compilation and simulation of 8 cryptographic primitives for 3 variants of specialized processor (CISC-like, VLIW-like and a variant with delayed load instruction) show the vitality of the proposed approach.

Keywords: DSL; compiler, specialized processor; SMT solver; instruction selection; instruction scheduling; register allocation

For citation: Sovietov P.N. Accelerating the development of DSL compilers for specialized processors. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 35-56 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–3

1. Введение

В настоящее время все активнее применяются спецпроцессоры, отличающиеся от процессоров общего назначения специализацией своей вычислительной структуры и предметно-ориентированным набором команд.

Рост популярности спецпроцессоров обусловлен сложившейся ситуацией в микроэлектронике: замедлением роста числа размещаемых транзисторов на кристалле (вопреки предсказанию закона Мура), и ограничением бюджета энергопотребления для интегральных микросхем, изготавливаемых по современным технологическим процессам (прекращение действия закона Деннарда).

Специализация вычислений для узкой предметной области является перспективным подходом к созданию энергоэффективных и высокопроизводительных решений в таких областях, как Интернет вещей и периферийные вычисления (edge computing).

Для широкого класса спецпроцессоров характерно выполнение преимущественно вычислительных операций с минимальным числом ветвлений и обращений к внешней памяти, с набором сложных, многооперандных команд и параллелизмом операций, выполняемых на основе статического планирования.

Программирование спецпроцессоров, как считают лауреаты премии Тьюринга Д. Паттерсон и Дж. Хеннесси [1], должно осуществляться на высокоуровневых предметно-ориентированных языках (DSL), чтобы создание прикладных программ для спецпроцессоров было доступно не только системным программистам, но и экспертам в выбранной предметной области.

В табл. 1 приведены примеры компиляторов для спецпроцессоров. Эти компиляторы имеют дело не с языками общего назначения, а с DSL, каждый из которых специально разработан для заданной узкой предметной области.

К положительным сторонам использования DSL для программирования спецпроцессоров можно отнести:

- использование предметно-ориентированной нотации, а не низкоуровневых pragma-директив или intrinsic-функций;
- ограничение выразительности конструкций, взятых из языка общего назначения, для более эффективного анализа и порождения целевого кода компилятором.

Табл. 1. Компиляторы для спецпроцессоров
Table 1. Compilers for special processors

Спецпроцессор	Предметная область	DSL	Компилятор
Google Pixel Visual Core	Обработка изображений	Halide	Halide
Google TPU	Машинное обучение	TensorFlow	XLA
Barefoot Tofino	Программно-определяемые сети	P4	P4c

GreenArrays GA144	Системы управления с низким энергопотреблением	Chlorophyll	Chlorophyll

Разработка прикладных программ для спецпроцессоров во многих случаях имеет свою специфику:

- в программах преобладают вычислительные операции, поэтому для получения ожидаемого качества целевого кода достаточными являются локальные методы компиляции – методы уровня линейного участка;
- время компиляции программ может достигать от десятков минут до нескольких часов;
- спецпроцессор обладает небольшим объемом памяти кода – компилируемые программы отличаются небольшим размером;
- спецпроцессор обладает достаточным числом физических регистров – при компиляции программ не требуется обеспечивать выгрузку регистров в память.

Сегодня спецпроцессоры все чаще проектируются для использования в конкретной системе на кристалле. Это означает, что своевременно должен быть разработан оптимизирующий компилятор, поддерживающий предметно-ориентированные конструкции входного языка и учитывающий архитектурные особенности конкретного спецпроцессора.

Можно выделить два следующих сценария разработки компилятора.

- Традиционный подход. Создание компилятора после завершения проектирования процессора. При таком порядке разработки отсутствует обратная связь между разработчиком компилятора и архитектором спецпроцессора.
- Подход codesign [2]. Прототипирование компилятора в процессе проектирования спецпроцессора. Этот подход позволяет находить компромиссные решения, касающиеся архитектуры спецпроцессора, качества порождаемого целевого кода и выразительности DSL.

Подход codesign становится все актуальнее с ростом числа спецпроцессоров, которые проектируются специально для использования в конкретных системах на кристалле.

При этом современные средства разработки перенацеливаемых компиляторов, такие, как LLVM и GCC, обладают рядом недостатков:

- поддерживается ограниченный набор входных языков общего назначения, поэтому в случае DSL необходима самостоятельная разработка модуля переднего плана компилятора;
- модуль заднего плана компилятора ориентирован на порождение кода для небольшого набора популярных процессорных архитектур CISC и RISC, а не для спецпроцессоров;
- алгоритмы порождения кода являются эвристическими, не гарантирующими точного решения задач компиляции;
- компиляторы являются крупными системами, насчитывающими миллионы строк кода; без средств автоматизации создания фаз компиляции затруднены быстрая разработка и внесение изменений в проект компилятора.

В этой связи актуальна разработка методов и алгоритмов, ориентированных на оперативное создание оптимизирующих DSL-компиляторов для спецпроцессоров с использованием точных методов решения NP-полных задач, позволяющих рутинную часть генератора кода компилятора реализовать силами универсальных программ-решателей.

Дальнейшая часть статьи организована следующим образом. В разд. 2 описана предлагаемая модель генератора кода DSL-компилятора на основе сведения задач компиляции к задаче SMT, в разд. 3 рассматриваются результаты разработки компилятора JC, реализованного на основе предложенной модели генератора кода, обзор существующих решений приведен в разд. 4, и в разд. 5 делается заключение по проделанной работе.

2. Модель генератора кода

Модель генератора кода компилятора (рис. 1) предназначена для трансляции линейного участка программы, имеющего форму графа зависимостей (ГЗ), в оптимизированный целевой код спецпроцессора (СВМ – специализированная вычислительная машина). В данной модели SMT-решатель применяется для нахождения точного решения ряда задач:

- синтез правил машинно-зависимой оптимизации;
- выбор команд;
- планирование команд и распределение регистров.

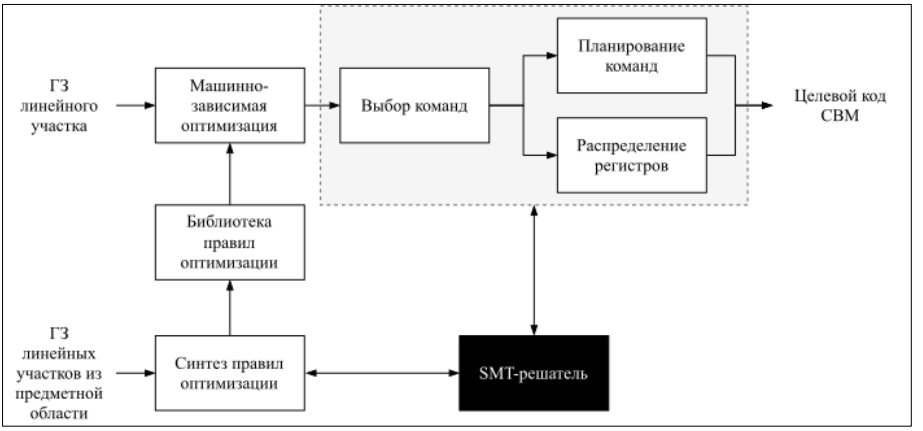


Рис. 1. Схема генератора кода DSL-компилятора
Fig. 1. Diagram of the DSL compiler code generator

Архитектурные особенности спецпроцессора поддерживаются путем введения ограничений в SMT-формулу. При формулировании ограничений используется теория линейной целочисленной арифметики (LIA) и теория битовых векторов без кванторов (QFBV). Это дает возможность выразительного описания задач компиляции, а также позволяет использовать для их решения быстродействующую программу-решатель [3].

Правила машинно-зависимой оптимизации кода синтезируются автоматически, на основе анализа набора программ из выбранной предметной области и с учетом архитектуры конкретного спецпроцессора.

Фаза выбора команд предшествует планированию команд и распределению регистров. Это позволяет провести анализ промежуточного представления на основе разработанных алгоритмов для сокращения числа порождаемых ограничений для планирования команд и распределения регистров. Шаблоны команд в фазе выбора команд представлены в декларативном виде на основе графа И/ИЛИ.

Планирование команд и распределение регистров выполняются совместно, что дает возможность решить проблему оптимального выбора порядка этих фаз компиляции.

2.1 Граф зависимостей по данным и состоянию памяти

В фазах модели генератора кода компилятора используется единое представление линейного участка программы – ориентированный ациклический граф зависимостей (ГЗ) по данным, дополненный П-зависимостями.

П-зависимость – зависимость выполнения одной операции от другой по состоянию разделяемой памяти. Установление П-зависимостей в виде ребер ГЗ задает частичный порядок выполнения обращений к разделяемой памяти. Это позволяет выявить возможный параллелизм обращений к памяти и избежать конфликтов чтения и записи в память.

Формально ГЗ представляется тройкой:

$$G = (O, \tau, i_{end}).$$

Множество операций $O = O_d \cup O_m \cup \{o_{start}, o_{end}\}$ содержит:

- множество O_d операций над данными;
- множество O_m операций обращения к разделяемой памяти;
- псевдооперацию o_{start}, которая определяет начало выполнения линейного участка и является источником зависимостей от входных данных ГЗ;
- псевдооперацию o_{end}, которая завершает выполнение линейного участка и является адресатом зависимостей для выходных данных ГЗ.

Функция $\tau: I \subset \mathbb{N} \to O$ сопоставляет индексу операцию.

Операции имеют уникальные индексы. Индекс i_{end} используется в качестве начального для топологической сортировки ГЗ.

Операция $o \in O$ определена следующим образом:

$$o = (o^{(c)}, o^{(x_1,\dots,x_n)}, o^{(M)}, o^{(v)}).$$

Компоненты операции:

- код операции $o^{(c)}$ с указанием типа результата;
- вектор индексов аргументов операции $o^{(x_1,\dots,x_n)} \in I^n$, где n – число аргументов;
- множество индексов источников П-зависимостей $o^{(M)} = \{i_1, \dots, i_n\}$, где $i_1, \dots, i_n \in I$ и n – число операций из O_m от которых зависит o;
- непосредственное значение $o^{(v)}$ для констант, формально тоже являющихся операциями и для которых $|o^{(x_1,\dots,x_n)}| = 0$.

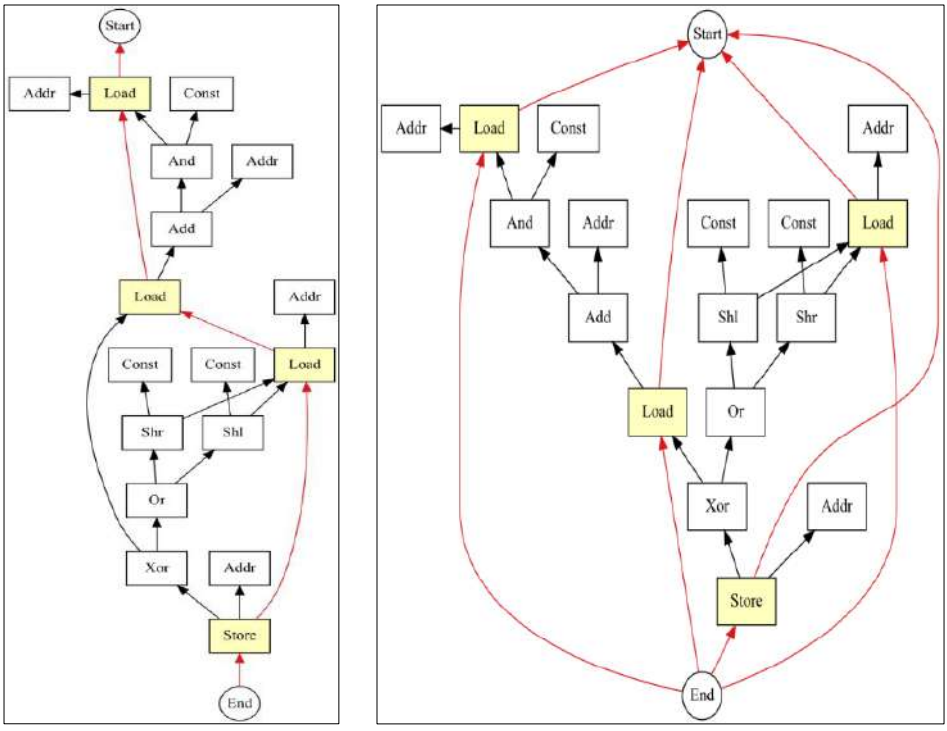


Рис. 2. Два варианта организации П-зависимостей линейного участка программы
Fig. 2. Two options for organizing P-dependencies of the linear section of the program

На рис. 2 показаны два варианта организации П-зависимостей (отмечены красным) в ГЗ:

- с последовательными обращениями к памяти;
- с одним из возможных вариантов распараллеливания П-зависимостей.

В некоторых случаях при работе с ГЗ требуется информация о множестве операций, которые используют результат, выработанный данной операцией. Для получения этих данных предназначен вспомогательный алгоритм определения адресатов результата каждой операции из ГЗ, на вход которого поступает ГЗ. Результат работы этого алгоритма – таблица U, в которой каждому индексу операции сопоставлено множество индексов адресатов результата этой операции.

Общей называется операция, результат которой разделяется между несколькими операциями-адресатами. Для общей операции с индексом i выполняется $|U[i]| > 1$.

Операция o_a *следует* за операцией o_b, а o_b *предшествует* o_a, если $(o_a, o_b) \in R = D \cup M$. Факт последования (предшествования) обозначается, как $o_a \succ o_b$ $(o_b \prec o_a)$. С учетом свойства транзитивности, если $(o_a, o_b) \in R$ и $(o_b, o_c) \in R$, тогда $o_a \succ o_c$ и $o_c \prec o_a$.

В модели вычислений ГЗ определено, что операция будет выполнена только после того, как выполнены все ее предшественники. Один из возможных вариантов порядка вычислений ГЗ определяется с помощью топологической сортировки операций.

Для получения информации о предшественниках каждой операции предназначен вспомогательный алгоритм определения множества предшественников каждой операции из ГЗ, на вход которого поступает ГЗ. Результатом работы этого алгоритма является таблица P, где каждому индексу операции сопоставлено множество индексов предшественников этой операции.

Конечным адресатом операции с индексом i является такая операция из множества $U[i]$, которая не является предшественником никакой другой операции из $U[i]$. У операции может быть несколько конечных адресатов.

ГЗ может быть получен из формы SSA линейного участка программы [4]. Операции обращения к памяти связываются П-зависимостями последовательно, в порядке появления этих операций в представлении SSA. Для распараллеливания полученных П-зависимостей могут использоваться результаты анализа указателей.

Переписывание термов ГЗ используется в фазах машинно-зависимой оптимизации и выбора команд, а также при синтезе правил машинно-зависимой оптимизации.

Входная операция или *вход* – аргумент операции терма, который сам не входит в число операций терма. Входы описываются переменными и трактуются, как виртуальные регистры. *Внутренняя операция* – операция терма, которая не является входной или корневой операцией.

Термы позволяют описать корневые подграфы, в которых общими операциями могут быть только корень и входы. Переменные термов содержат сопоставленные индексы входов.

Для реализации переписывания ГЗ используется алгоритм, на вход которого поступают: ГЗ и система переписывания R с правилами, использующими термы. Результат работы этого алгоритма – трансформированный граф. Стратегией переписывания в рассматриваемом алгоритме является применение правил из R к операциям из ГЗ при нисходящем обходе графа в глубину. Правила упорядочены и применяются последовательно.

2.2 Метод синтеза правил машинно-зависимой оптимизации

Автоматическое формирование библиотеки правил машинно-зависимой оптимизации производится с учетом анализа набора характерных для выбранной предметной области ГЗ и архитектуры конкретного спецпроцессора.

Пояснения к этапам метода синтеза правил машинно-зависимой оптимизации, представленного на рис. 3.

1. Канонизация порядка аргументов операций ГЗ осуществляется с помощью сортировки индексов аргументов $o^{(x_1,\ldots,x_n)}$ для каждой коммутативной операции o. При сортировке сравниваются коды операций. Рассматриваемый этап позволяет сократить число извлеченных из ГЗ шаблонов, отличающихся только порядком аргументов коммутативных операций.
2. Из ГЗ извлекается множество всех допустимых термов. Это множество содержит шаблоны-кандидаты для использования в левой части синтезируемых правил.
3. Извлеченные термы записываются в таблицу по ключу, которым является хеш-значение терма. Это позволяет исключить из рассмотрения дублирующиеся термы.
4. Окончательная фильтрация термов осуществляется на основе переписывания ГЗ. В качестве шаблонов рассматриваются только те термы, которые использовались при переписывании.
5. Для каждого из полученных шаблонов с помощью техники синтеза программ генерируется машинно-зависимая часть правила. Если синтез завершился успешно, то сформированное правило добавляется к множеству ранее полученных правил.

Рис. 3. Метод синтеза правил машинно-зависимой оптимизации
Fig. 3. Method for synthesing rules of machine-dependent optimization

Результатом синтеза правил машинно-зависимой оптимизации является множество оптимизирующих правил легализации, то есть правил для трансформации фрагментов ГЗ в форму, содержащую только поддерживаемые архитектурой спецпроцессора операции. При успешном синтезе правил по всем полученным шаблонам гарантируется, что синтезированные правила достаточны для осуществления легализации ГЗ.

Задача извлечения шаблонов из ГЗ поставлена следующим образом. Пусть множество извлеченных термов T содержит все подграфы из ГЗ, которые являются термами term:

$$T = \{g \mid g \subset G \wedge \mathrm{term}(g)\}.$$

Тогда S содержит такие подмножества элементов из T, которые покрывают (cover) ГЗ:

$$S = \{P \mid P \subseteq T \wedge \text{cover}(G, P)\}.$$

Множество шаблонов $P = \{p_1, \ldots, p_n\}$ *покрывает* ГЗ, если:

- существует система переписывания R, в которой для каждого элемента $p \in P$ существует отдельное правило переписывания; в правой части правила находится специальная псевдооперация, обозначающая факт переписывания.
- после $n \leq |V(G)|$ шагов переписывания в ГЗ остаются только псевдооперации, запрещенные для переписывания;
- каждое правило из R использовано при переписывании ГЗ не менее одного раза.

В общем виде задача извлечения шаблонов из ГЗ сводится к поиску наименьшего подмножества термов из T, которое покрывает ГЗ:

$$P^* \in \underset{P \in S}{\text{argmin}} |P|.$$

Для извлечения всех допустимых термов из ГЗ предназначен алгоритм 1, представленный на рис. 4. На вход этого алгоритма поступает ГЗ, а результатом работы алгоритма является множество допустимых термов T. Рассматриваемый алгоритм основан на восходящем обходе ГЗ в глубину с использованием запоминания промежуточных результатов.

```
1:  function EXTRACT(G)
2:      U ← get_uses(G)
3:      M ← ∅
4:      T ← ∅
5:      for all i ∈ toposort(G, i_end) do
6:          M[i] ← ∅
7:          o ← τ_G(i)
8:          if o это константа then
9:              M[i] ← {o}
10:         else
11:             I ← []
12:             for all j ∈ o^(x_1,...,x_n) do
13:                 o_j ← τ_G(j)
14:                 if o_j это константа then
15:                     I.append({M[j]})
16:                 else if |U[j]| = 1 ∧ |o_j^(M)| = 0 then        ▷ Не общая операция
17:                     I.append({IN, M[j]})                     ▷ и не обращение к памяти
18:                 else
19:                     I.append({IN})
20:             for all I' ∈ I[0] × ... × I[n − 1] do            ▷ n = |o^(x_1,...,x_n)|
21:                 for all x ∈ I'[0] × ... × I'[n − 1] do
22:                     t ← set_ins(G, i, make_term(o, x))
23:                     if терм t легален then
24:                         save_term(T, hash(t), t)
25:                     M[i] ← M[i] ∪ {t}
26:     return T
```

Рис 4. Алгоритм 1 извлечения всех допустимых термов из ГЗ
Fig 4. Algorithm 1 for extracting all valid terms from the dependency graph

Детали алгоритма 1:

- *строка 3*. Определение таблицы M для хранения промежуточных результатов. В M каждому индексу i операции сопоставлено множество ранее извлеченных подтермов, корень которых находится в i;
- *строка 5*. Цикл обхода индексов операций в порядке топологической сортировки ГЗ;

- *строка 8*. Если очередная операция оказалась константой, то она становится единственным подтермом в таблице M по индексу i;
- *строка 10*. Разбор общего случая операции с наличием аргументов;
- *строки 11–19*. Формирование списка вариантов аргументов I, элементы которого являются множествами. Эти множества, в зависимости от типа операции, могут содержать до двух значений: 1) значение IN – указание на входную операцию, 2) значение из таблицы M для индекса аргумента;
- *строка 20*. Декартово произведение производится для получения последовательности вариантов аргументов текущей корневой операции с учетом вхождения IN;
- *строка 21*. Второе декартово произведение производит все последовательности аргументов;
- *строка 22*. Формирование терма с использованием корневой операции o и аргументов из x. Входы нумеруются с учетом их возможного дублирования;
- *строка 23*. Для термов могут быть введены дополнительные ограничения, в том числе: максимальная глубина дерева и максимальное число аргументов;
- *строка 23*. Запись терма в хеш-таблицу;
- *строка 25*. Добавление очередного подтерма к множеству из таблицы M.

Для оценки вычислительной сложности алгоритма 1 может быть рассмотрен случай, когда ГЗ, за исключением псевдоопераций, представляет собой полное бинарное дерево высоты h. Тогда число извлеченных термов $t(h)$ до записи в хеш-таблицу определяется с помощью рекуррентной формулы:

$$t(h) = \begin{cases} t_1(h) + 2t(h-1), & h > 1 \\ 1, & h \le 1 \end{cases},$$
$$t_1(h) = \begin{cases} (t_1(h-1)+1)^2, & h > 1 \\ 1, & h \le 1 \end{cases}.$$

Значения функции $t(h)$ показывают, что уже при $h = 5$ число извлеченных термов становится слишком велико:

$$1, 6, 37, 750, 459829, 210067308558, \ldots$$

Однако в прикладных программах часто встречаются константы и общие операции. Их наличие позволяет ограничить рост подтермов в таблице M и избежать комбинаторного взрыва.

На завершающем этапе шаблоны фильтруются путем переписывания ГЗ с использованием элементов из множества T, при этом ведется подсчет числа успешных применений для каждого из правил переписывания. Правила переписывания упорядочиваются от наибольшего шаблона-терма по числу содержащихся в нем операций, к наименьшему. Результат фильтрации – шаблоны из множества T, которые использовались при переписывании хотя бы один раз.

Жадная тактика выбора правил не гарантирует нахождения наименьшего числа покрывающих ГЗ шаблонов, как задано в общей постановке задачи, но позволяет существенно сократить их итоговое число.

Синтезированное правило машинно-зависимой оптимизации имеет следующую форму:

- левая часть является шаблоном-термом и называется спецификацией f;
- правая часть – линейная программа p, которая состоит из элементарных операций.

Пусть множество P содержит все линейные программы для спецпроцессора, а множество S включает в себя только те программы из P, которые соответствуют спецификации f:

$$S = \{p \mid p \in P \wedge \forall \vec{x} \in X : p(\vec{x}) = f(\vec{x})\}.$$

Здесь предполагается, что для программы p задана трансляция в представление на языке спецификации.

Тогда **задача синтеза машинно-зависимой части правила оптимизации** состоит в нахождении программы с наименьшей длиной:

$$p^* \in \operatorname*{argmin}_{p \in S} |p|.$$

Для практической реализации синтеза линейной программы по спецификации используется SMT-решатель и метод CEGIS [5] (рис. 5).

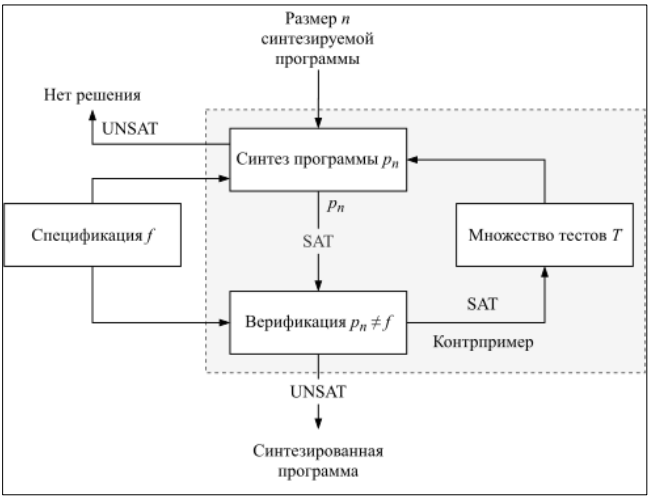


Рис. 5. Синтез линейной программы методом CEGIS
Fig. 5. Synthesis of a linear program by the CEGIS method

Ключевыми этапами метода CEGIS являются синтез программы-кандидата и ее верификация.

1. Синтез программы p_n, состоящей из n операций осуществляется по спецификации f. При этом используется множество тестов T, которое содержит примеры выполнения f на конкретных аргументах.
2. Если синтез p_n завершился успешно, то выполняется этап верификации. Верификация основана на поиске контрпримера – вектора аргументов, для которого $p_n \neq f$:
   a) если контрпример найден, то он добавляется к T и процесс синтеза p_n запускается; снова;
   b) если контрпример не найден, то p_n удовлетворяет спецификации и синтезированная программа выдается в качестве результата.

Для синтеза программы может быть выбрана теория QFBV SMT-решателя, что позволяет применить в поиске решения быстродействующую SAT-подсистему.

Успешно синтезированная программа далее преобразуется в форму подграфа ГЗ, входами которого являются виртуальные регистры-аргументы. Полученная правая часть завершает формирование правила.

2.3 Выбор команд

В фазе выбора команд подграфы операций ГЗ трансформируются в команды спецпроцессора. Выбор команд состоит из следующих этапов:

1. Порождение всех вариантов выбора команд с помощью представления набора команд спецпроцессора на основе графа И/ИЛИ.
2. Нахождение варианта выбора команд, удовлетворяющего ограничениям SMT-формулы.

Задача выбора команд поставлена следующим образом. Пусть функция rewrite^+ возвращает множество C_i вариантов выбора команды из множества шаблонов команд R. Выбор осуществляется для подграфа ГЗ, корень которого находится по индексу i операции. Тогда может быть определена функция m, которая отображает индекс i операции на множество вариантов выбора команды C_i, с включением варианта отсутствия выбора:

$$m = \{(i, C_i \cup \{\emptyset\}) \mid i \in \operatorname{dom} \tau_G \wedge C_i = \text{rewrite}^+(G, i, R)\}.$$

Далее, пусть определен вектор команд $\vec{c}$, индексы элементов которого взаимно однозначно отображаются на индексы операций ГЗ. Элемент вектора c_i принимает одно из следующих значений:

- выбранная команда из C_i;
- значение $\emptyset$ для предотвращения наложения графов-замещений. В этом случае операция по индексу i является внутренней.

Допустимые варианты выбора команд определяются множеством векторов команд, каждый из которых покрывает ГЗ:

$$S = \{(c_1, \dots, c_n) \mid c_1 \in m(i_1) \wedge \dots \wedge c_n \in m(i_n) \wedge \text{cover}(G, \{c_1, \dots, c_n\})\},$$
$$\{i_1, \dots, i_n\} = \operatorname{dom} \tau_G.$$

Тогда **задача выбора команд** состоит в нахождении вектора команд из S с минимальным числом выбранных команд:

$$\vec{c}^{\,*} \in \underset{\vec{c} \in S}{\operatorname{argmin}} |c_1 \cup \dots \cup c_n|.$$

Варианты выбора команд порождаются на основе анализа ГЗ с использованием шаблонов команд, представленных в виде графа И/ИЛИ.

Результатом анализа ГЗ является множество $C = \{C_1, \dots, C_{|V(G)|}\}$, каждый элемент C_i которого содержит множество вариантов выбора команды для подграфа ГЗ с корневой операцией под индексом i.

Для описания набора команд могут потребоваться тысячи шаблонов команд. Последовательная проверка всех шаблонов команд для каждого индекса операции из ГЗ является во многих случаях вычислительно трудоемкой задачей. Для более компактного представления системы команд, а также для ускорения порождения вариантов выбора команд предлагается использовать граф И/ИЛИ.

Граф И/ИЛИ – ориентированный ациклический граф, в котором тип каждого узла является элементом множества {И, ИЛИ}. Узел типа И зависит от всех своих непосредственных потомков, а узел типа ИЛИ зависит только от одного из своих непосредственных потомков.

При построении графа И/ИЛИ учитывается иерархическая структура системы команд. Построение осуществляется в нисходящем порядке: от абстрактных типов команд к конкретным полям команды. В процессе создания графа И/ИЛИ происходит объединение общих подвыражений шаблонов команд.

Получение единственного результата сопоставления подграфа ГЗ с набором команд происходит на основе обхода графа И/ИЛИ в глубину, в нисходящем порядке. Общие префиксы шаблонов команд проверяются только один раз. Для того, чтобы избежать повторного сопоставления с общими подвыражениями промежуточные результаты, полученные для конкретных индексов операций из ГЗ, сохраняются в специальной таблице.

Порождение всех вариантов выбора команды выполняется с помощью перебора с возвратом в графе И/ИЛИ, что позволяет, благодаря свойству префиксности и использованию общих узлов, избежать сопоставления с заведомо неподходящими фрагментами шаблонов команд.

В шаблоне команды могут использоваться ограничения на использование таких разделяемых ресурсов, как поле литерала или ФУ. Для предотвращения конфликтов доступа к разделяемым ресурсам в граф И/ИЛИ добавляются специальные узлы типа И – узлы-ограничения. Ограничения представляют собой записи о состоянии разделяемых ресурсов в

таблице ограничений. При использовании ограничений совместно с запоминанием необходимо хранить информацию о состоянии разделяемых ресурсов совместно с промежуточными результатами сопоставления.

На завершающем этапе выбора команд формируется система ограничений для SMT-решателя. Используются SMT-переменные следующих типов:

- переменная $c_k \in \{0,1\}$ определяет факт выбора команды в двоичном векторе всех вариантов команд $\vec{c} = (c_1, \dots, c_{|\zeta|})$, где $\zeta = C_1 \cup \dots \cup C_{|V(G)|}$;
- переменная $n_i \in \{1, \dots, |\zeta|\}$ связывает индекс i операции ГЗ с выбранной командой, обозначаемой индексом элемента c_k;
- переменная $a_i \in \{0,1\}$ определяет, является ли индекс i операции ГЗ одним из входов команды. Если i является индексом входной операции, то этот индекс является также индексом результата какой-то иной команды.

Выбор команды c_k определен с использованием функции assigned, которая возвращает множество назначенных этой команде индексов операций:

$$\text{assign}(c_k) :: = \bigwedge_{n_i \in \text{assigned}(c_k)} n_i = k.$$

Если команда выбрана, то для всех индексов i операций, которые являются входами этой команды (функция ins), в переменных a_i установлено значение 1:

$$\mathbf{io}(\boldsymbol{c_k}) :: = \bigwedge_{\boldsymbol{a_i} \in \mathbf{ins}(\boldsymbol{c_k})} \boldsymbol{a_i} = \mathbf{1}.$$

Может быть выбрана только одна команда из множества C_i:

$$\text{unique}(C_i, c_k) :: = c_k = 1 \wedge \bigwedge_{c_{k'} \in C_i \setminus \{c_k\}} c_{k'} = 0.$$

Ограничение cover использует рассмотренные ранее условия выбора команды из C_i:

$$\text{cover}(C_i) :: = \bigvee_{c_k \in C_i} (\text{assign}(c_k) \wedge \text{io}(c_k) \wedge \text{unique}(C_i, c_k)).$$

Функция overlap определяет по индексу i в ГЗ множество команд, которые содержат операцию с индексом i среди своих внутренних операций. Если ни одна команда из C_i не выбрана, то используется одна из сторонних команд:

$$\text{other}(C_i) :: = \bigwedge_{c_k \in C_i} c_k = 0 \wedge \bigvee_{c_{k'} \in \text{overlap}(i) \setminus C_i} c_{k'} = 1.$$

Допустимое решение задачи выбора команд определяется следующим образом:

$$\text{select}(C) :: = \bigwedge_{i=1}^{|C|} (\text{cover}(C_i) \wedge a_i = 1 \vee \text{other}(C_i) \wedge a_i = 0).$$

С использованием select формулируется **минимальное по числу $\boldsymbol{n}$ выбранных команд решение**:

$$m^* \in \underset{\substack{m \in \text{SMT}(\phi(C,n)) \\ \forall n \in \mathbb{N}}}{\text{argmin}} m(n),$$

$$\text{где } \phi(C, n) :: = \text{select}(C) \wedge \sum_{c_i \in \vec{c}} c_i < \text{n}.$$

Здесь множество полученных с помощью SMT-решателя моделей, для которых формула φ выполнима, представляет собой допустимое множество, в котором определяется модель m^* с минимальным значением параметра n.

Современные SMT-решатели имеют встроенные средства решения оптимизационных задач [6]. Однако для детального управления поиском решения, а также с точки зрения времени нахождения результата, оптимизация, управляемая бинарным поиском [7], может оказаться более предпочтительной.

При решении задачи минимизации на основе бинарного поиска SMT-решатель вызывается итеративно с добавлением новых ограничений, сужающих интервал поиска. На шаге итерации i SMT-формула имеет вид:

$$\varphi(n) \wedge n < N_i \,.$$

Если SMT-решатель возвращает на очередном шаге итерации значение SAT, то для новой верхней границы поиска выбирается значение n из полученной модели, которое может оказаться меньшим, чем вычисленная средняя точка интервала поиска. Поэтому решение в некоторых случаях может быть найдено за меньшее число шагов, чем в традиционном алгоритме бинарного поиска.

2.4 Совместное планирование команд с выбором регистров

На вход фазы совместного планирования команд и распределения регистров поступает ГЗ, операции которого являются командами спецпроцессора. Выходное представление рассматриваемой фазы – целевой код, состоящий из упорядоченных команд. Аргументы и результат команд целевого кода являются номерами уже не виртуальных, а физических регистров.

Целевой код порождается для модели архитектуры спецпроцессора, которая содержит несколько параллельно работающих функциональных узлов (ФУ) различных типов. Операции ГЗ, полученные в результате выбора команд, выступают в качестве машинных операций в составе широкой команды архитектуры VLIW. Также в модели архитектуры спецпроцессора могут быть поддержаны операции с многотактной задержкой выработки результата.

Фаза совместного планирования команд и распределения регистров включает в себя анализ ГЗ и формирование системы ограничений для SMT-решателя. К операциям o ГЗ добавляются новые компоненты – SMT-переменные:

- позиция операции в целевом коде $o^{(t)}$;
- номер ФУ $o^{(f)}$ для операции в составе широкой команды;
- номера регистров-аргументов $o^{(r_1,\dots,r_n)}$;
- номер регистра-результата $o^{(r_y)}$.

В системе ограничений используются параметры числа команд целевого кода n и числа регистров R. Система ограничений состоит из ограничений планирования команд $\mathrm{schedule}$ и ограничений распределения регистров $\mathrm{alloc\_regs}$.

Задача совместного планирования команд и распределения регистров формулируется для SMT-решателя и состоит в получении целевого кода минимальной длины с учетом заданных ограничений:

$$m^* \in \underset{\substack{m \in \mathrm{SMT}(\phi(G,R,n)) \\ \forall n \in \mathbb{N}}}{\operatorname{argmin}} m(n),$$

$$\text{где } \phi(G,R,n) :: = \mathrm{schedule}(G,n) \wedge \mathrm{alloc\_regs}(G,R).$$

Дополнительно, с помощью минимизации параметра R определяется решение с наименьшим числом использованных регистров.

Для нахождения решения используется SMT-решатель в режиме бинарного поиска. Полученные от SMT-решателя значения переменных применяются для формирования результата компиляции в виде целевого кода спецпроцессора.

В фазе **планирования команд** используется алгоритм получения информации об отношении предшествования для заданной пары операций. Это позволяет сократить число сформированных ограничений.

SMT-формула schedule содержит ограничения:

- на очередность расположения команд с учетом задержки (order);
- на размещение операций в различных ФУ (alloc_fu);
- на расположение операций в широкой команде (place);
- на размер целевого кода (limit).

Определение schedule:

$$\text{schedule}(G, n) ::= \text{order}(G) \wedge \text{alloc\_fu}(G) \wedge \text{place}(G) \wedge \text{limit}(G, n).$$

Операции получают номера позиций в целевом коде с учетом зависимостей и времени задержки:

$$\text{order}(G) ::= \bigwedge_{\substack{o \in V(G) \\ \forall p \in o^{(x_1, \ldots, x_n) \cup o(M)}}} o^{(t)} > o_p^{(t)} + \text{delay}(o^{(c)}).$$

Функция delay возвращает задержку в тактах для заданной операции. Учитываются отношения только с непосредственными предшественниками каждой операции. Это позволяет сократить количество порожденных неравенств, предполагая, что в SMT-решателе учитывается транзитивность отношений между операциями.

Операции размещаются на одном из допустимых ФУ:

$$\text{alloc\_fu}(G) ::= \bigwedge_{o \in V(G)} \bigvee_{i \in \text{fu}(o^{(c)})} o^{(f)} = i.$$

Функция fu возвращает множество номеров ФУ, на которые может быть назначена операция.

Размещение нескольких операций в широкой команде возможно при условии, что эти операции назначены на различные ФУ:

$$\text{place}(G) ::= \bigwedge_{\substack{i, j \in \text{dom}\, \tau_G \wedge i > j \\ \forall o_i, o_j : o_i \not\prec o_j \wedge o_j \not\prec o_i}} (o_i^{(t)} \neq o_j^{(t)} \vee o_i^{(f)} \neq o_j^{(f)}).$$

Неравенства порождаются только для тех пар операций, между которыми не определено отношение предшествования.

Размер целевого кода не должен превышать n команд:

$$\text{limit}(G, n) ::= o_{start}^{(t)} = 0 \wedge \bigwedge_{o \in V(G) \setminus \{o_{start}\}} 0 < o^{(t)} < n.$$

Распределение регистров осуществляется с учетом анализа ГЗ. Определяется множество конечных адресатов для каждой из операций и для некоторых пар операций устанавливается факт отсутствия пересечения их областей жизни результатов. Это позволяет сократить число сформированных ограничений.

SMT-формула alloc_regs содержит следующие ограничения:

- на предел области жизни результата операции (use);
- на связи между регистровыми аргументами и результатом операции (io);
- на взаимное расположение областей жизни результата операций (live).

Определение alloc_regs:

$$\text{alloc\_regs}(G,R)::=\ \text{use}(G)\wedge\text{io}(G,R)\wedge\text{live}(G).$$

Для получения информации о конечных адресатах каждой операции предназначен вспомогательный алгоритм, на вход которого поступает ГЗ. Результат работы этого алгоритма – таблица L, где каждому индексу операции сопоставлено множество индексов конечных адресатов этой операции.

Предел области жизни результата операции равен максимальной позиции в целевом коде среди операций с индексами, полученными из множества L:

$$\text{use}(G)::=\bigwedge_{o\in V(G)}\text{maxset}\,(o^{(u)},\{\tau_G(j)^{(t)}\mid j\in L[i]\}).$$

Вспомогательное определение maxset связывает значение переменной y с максимальным значением из множества X:

$$\text{maxset}(y,X)::=\bigvee_{x\in X}x=y\wedge\bigwedge_{x\in X}x\le y.$$

Результат операции – номер регистра в заданном диапазоне. Каждый номер регистра-аргумента операции равен номеру регистра-результата, полученного от источника этого аргумента:

$$\text{io}(G,R)::=\bigwedge_{o\in V(G)}(0\le o^{(r_y)}<R\wedge\bigwedge_{i=1}^{|o^{(x_1,\dots,x_n)}|}o^{(r_i)}=\tau_G(o^{(x_i)})^{(r_y)}).$$

На этапе анализа ГЗ для некоторых пар операций можно определить, что их области жизни результата не пересекаются. Область жизни результата операции o_i следует за областью жизни результата операции o_j, если все конечные адресаты o_j являются предшественниками o_i:

$$\text{live\_after}(i,j)::=\ \forall u\in L[j]\colon u\in P[i].$$

Для тех пар операций, пересечение областей жизни результата которых не удалось установить с помощью live_after, добавляется ограничение – номера регистров-результатов различны или же области жизни результата не пересекаются:

$$\text{live}(G)::=\bigwedge_{\substack{o_i,o_j\in V(G)\wedge i>j\\ \neg\text{live\_after}(i,j)\wedge\neg\text{live\_after}(j,i)}}(o_i^{(r_y)}\neq o_j^{(r_y)}\vee o_i^{(t)}\ge o_j^{(u)}\vee o_j^{(t)}\ge o_i^{(u)}).$$

3. Компилятор JC

Для апробации разработанного математического и алгоритмического обеспечения разработан компилятор JC, транслирующий текст на подмножестве предметно-ориентированного языка Jasmin [8] в машинное представление спецпроцессора [9], набор команд которого ориентирован на выполнение задач низкоресурсной симметричной криптографии [10]. Для тестирования компилятора использованы программные реализации 8 низкоресурсных симметричных шифров из области Интернета вещей. Структурная схема компилятора JC показана рис. 6.

Компилятор JC поддерживает конструкцию for только для времени компиляции. В фазе AST-преобразований циклы полностью разворачиваются, в результате программа представляет собой линейный участок.

Для тестирования компилятора JC использовалась следующая конфигурация: компьютер Intel Core i7-3770 CPU 3.40 ГГц, 32 ГБайт ОЗУ, SMT-решатель Z3 версии 4.8.7.

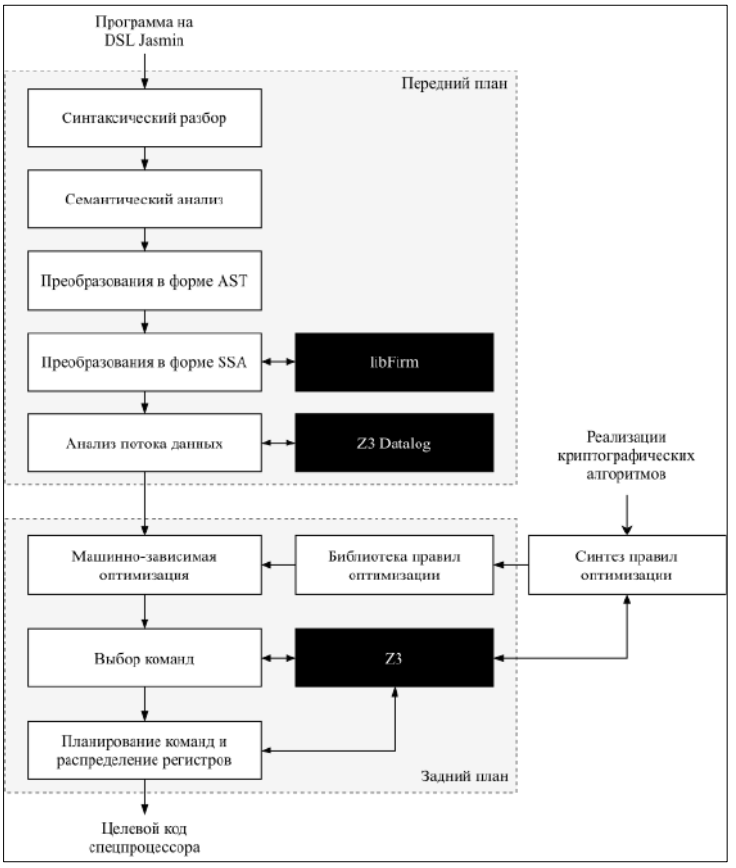


Рис. 6. Структурная схема компилятора JC
Fig. 6. Block diagram of the JC compiler

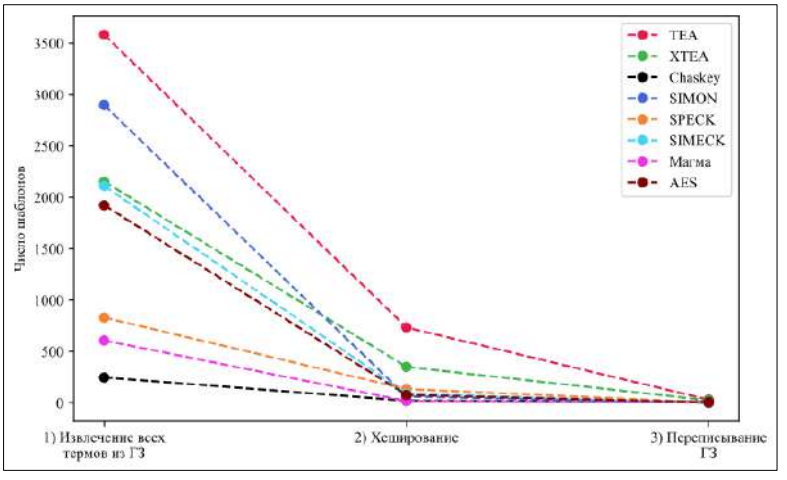


Рис. 7. Этапы фильтрации шаблонов
Fig. 7. Stages of filtering templates

На рис. 7 показаны этапы фильтрации шаблонов, извлеченных из 8 реализаций шифров. Во всех случаях максимальная высота шаблона не превысила 4, а максимальное число входов –

6, то есть используемый алгоритм извлечения шаблонов на практике продемонстрировал полиномиальное время вычислений.

Реализован алгоритм синтеза правил машинно-зависимой оптимизации для выбранного спецпроцессора на основе техники CEGIS. Правила были успешно синтезированы для 8 реализаций шифров. Полученная библиотека правил использовалась для проведения автоматической легализации ГЗ. Результат сравнивался с полученным с помощью отдельного набора составленных вручную правил легализации для операций побитового сдвига. Синтезированные правила позволили сократить число выбранных команд: для теста Магма на 28% и для теста AES на 30%.

Для описания набора команд спецпроцессора использован граф И/ИЛИ, что позволило заменить 501555 исходных шаблонов команд представлением, близким к структуре АЛУ спецпроцессора и занимающим в текстовой форме 54 строки.

В фазе совместного планирования команд и распределения регистров использование предварительного анализа ГЗ позволило сократить число сформированных ограничений в 3.3–27.6 раз для варианта компиляции без распараллеливания П-зависимостей и в 1.9–24.2 раз для варианта с распараллеливанием П-зависимостей.

Результаты компиляции и выполнения 8 тестовых программ в моделируемых тактах (табл. 2) получены с использованием трех вариантов программной модели спецпроцессора с древовидным АЛУ [9]:

- базовый; с последовательным выполнением однотактных команд;
- Delayed Load; с дополнительным тактом на получение результата чтения из памяти;
- VLIW; с двумя параллельно работающими ФУ, каждый из которых содержит АЛУ. При этом только один из ФУ поддерживает операции обращения к памяти.

Данные для процессора RISC-V приведены для сравнения и получены с использованием компилятора GCC 9.2.0 (опция -O3) и программной модели RISC-V.

Табл. 2. Результаты компиляции и выполнения программ
Table 2. Results of compilation and execution of programs

Программа	**Результат выполнения, тактов**					
		Вариант спецпроцессора (компилятор JC)				
	RISC-V (компилятор GCC)	**Базовый**	**Delayed Load**	**VLIW**	**С распараллеливанием П-зависимостей**	
					Delayed Load	**VLIW**
TEA	588	391	393	261	391	259
XTEA	517	318	323	222	318	198
Chaskey	233	78	83	45	78	43
SIMON	599	213	214	130	Таймаут	Таймаут
SPECK	470	177	180	95	177	92
SIMECK	491	180	219	134	Таймаут	Таймаут
Магма	708	236	336	198	Таймаут	Таймаут
AES	1224	336	420	212	Таймаут	Таймаут

Режим компиляции с распараллеливанием П-зависимостей подразумевает использование компилятором JC результатов анализа указателей при построении П-зависимостей. Это позволяет перемещать позиции операций обращения к памяти в целевом коде.

Процесс поиска в VLIW-варианте для 8 реализаций шифров показан на рис. 8. Время поиска каждого успешного решения не превысило 270 с.

Как видно из табл. 2, компиляция программ SIMON, SIMECK, Магма и AES в режиме с распараллеливанием П-зависимостей завершилась таймаутом SMT-решателя. Эти

результаты связаны со спецификой работы SMT-решателя и не могут быть объяснены только числом сформированных ограничений, поскольку размер SMT-формулы для успешно скомпилированной реализации XTEA превышает размеры формул, полученных для SIMON, SIMECK и Магма.

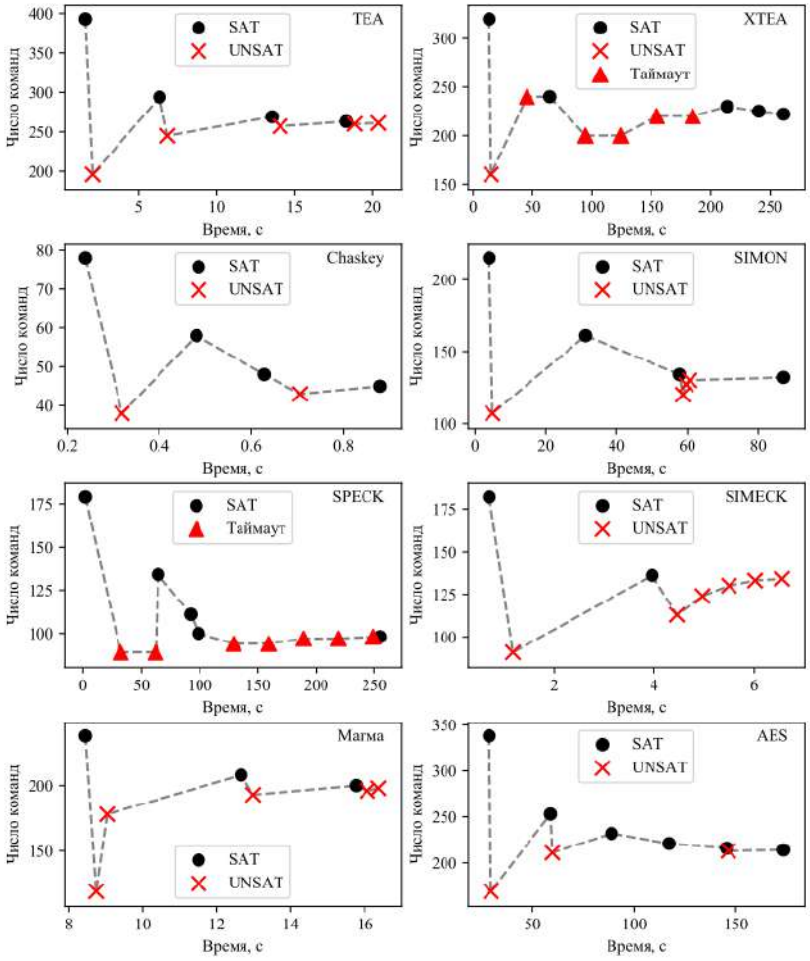


Рис. 8. Процесс поиска решений для VLIW-варианта спецпроцессора
Fig. 8. The process of finding solutions for the VLIW-variant of the special processor

Сокращение объема целевого кода, полученное с помощью компилятора JC, по сравнению с GCC/RISC-V, составило: 1.5–3.6 раз для базового варианта спецпроцессора, а также для варианта Delayed Load и 2.2–5.8 раз для варианта VLIW.

4. Обзор существующих решений

Машинно-зависимая оптимизация в компиляторах LLVM, GCC и libFirm реализована в виде эвристических правил на языке C/C++. Для автоматического порождения правил машинно-зависимой оптимизации используются методы из области синтеза программ [5].

Простейший из таких методов основан на технике супероптимизации [11], в которой используется полный перебор в пространстве линейных программ ограниченной длины. Более эффективные системы синтеза правил машинно-зависимой оптимизации основаны на применении SAT- и SMT-решателей [12–14]. В существующих решениях порождаются все

варианты шаблонов, что является избыточным подходом, в котором не учитывается ориентация на предметную область.

Точные подходы к решению задач выбора команд, планирования команд и распределения регистров, известные из литературы, приведены в табл. 3. Среди вариантов комбинированных подходов для порождения целевого кода наиболее перспективным представляется совмещение планирования команд и распределения регистров. Наилучший результат среди известных для реализации такой комбинированной фазы – оптимальное решение для линейного участка из 647 операций [15]. В практических ситуациях, когда цикл в программе для спецпроцессора полностью разворачивается, число операций может легко превысить этот предел.

Табл. 3. Точные методы реализации фаз генератора кода
Table 3. Precise methods of implementing the phases of the code generator

Выбор команд	Планирование команд	Распределение регистров
NOLTIS [16], PBQP [17], CP [18]	SAT [19], ЦЛП [20], CP [21]	Раскраска графа [22], метод головоломок [23], ЦЛП [24], PBQP [25]
ЦЛП [26]		
	ЦЛП [27], CP [28]	
ЦЛП [29]		

5. Заключение

Разработка прикладных программ для спецпроцессоров имеет свои особенности, которые могут быть учтены при создании компилятора. Для реализации генератора кода могут быть использованы методы решения NP-полных задач. В этом случае рутинная часть машинно-зависимой стороны компилятора может быть реализована силами сторонней программы-решателя. Основная идея работы – сведение задач во всех основных фазах генератора кода к задаче SMT. Такой подход представляется перспективным, поскольку развитие алгоритмических методов для реализации эффективных SAT/SMT-решателей в последние десятилетия происходит очень интенсивно.

Практическая применимость разработанных методов и алгоритмов автоматизации создания генератора кода DSL-компиляторов для спецпроцессоров подтверждена экспериментальной оценкой реализованного компилятора JC на наборе прикладных программ.

Список литературы / References

[1]. Hennessy J.L., Patterson D.A. A new golden age for computer architecture. Communications of the ACM, vol. 62, no. 2), 2019, pp. 48-60.
[2]. Sampson A., Bornholt J., Ceze, L. Hardware-software co-design: not just a cliché. Leibniz International Proceedings in Informatics, vol. 32, 2015, pp. 262–273.
[3]. Roselli, S.F., Bengtsson K., Åkesson K. SMT solvers for job-shop scheduling problems: Models comparison and performance evaluation. In Proc of the IEEE 14th International Conference on Automation Science and Engineering (CASE), 2018, pp. 547-552.
[4]. Braun M., Buchwald S., Hack S., Leißa R., Mallon C., Zwinkau A. Simple and efficient construction of static single assignment form. Lecture Notes in Computer Science, vol. 7791, 2018, pp. 102-122).
[5]. Gulwani S., Jha S., Tiwari A., Venkatesan R. Synthesis of loop-free programs. ACM SIGPLAN Notices, vol. 46, no. 6, 2011, pp. 62-73.
[6]. Bjørner N., Phan A.D., Fleckenstein L. νz-an optimizing SMT solver. Lecture Notes in Computer Science, vol. 9035, 2015, pp. 194-199).
[7]. Sebastiani R., Tomasi S. Optimization in SMT with $LA(\mathbb{Q})$ Cost Functions. Lecture Notes in Computer Science, vol. 7364, 2012, pp. 484-498.
[8]. Almeida J.B., Barbosa M. et al. Jasmin: High-assurance and high-speed cryptography. In Proc. of the ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1807-1823.

[9]. Елизаров С.Г., Марков Д.С., Советов. Арифметико-логическое устройство и способ преобразования данных с использованием такого устройства. Патент на изобретение № 2681702, Российская Федерация, МПК G06F 15/76, 2019 г. / Elizarov S.G., Markov D.S., Sovetov P.N. Councils. Arithmetic logic device and method for converting data using such a device. Patent for invention No. 2681702, Russian Federation, MPK G06F 15/76, 2019.
[10]. Biryukov A., Perrin L.P. State of the art in lightweight symmetric cryptography. Cryptology ePrint Archive: Report 2017/511, 2017, 55 p.
[11]. Massalin H. Superoptimizer: a look at the smallest program. ACM SIGARCH Computer Architecture News, vol. 15, no. 5, 1987, pp. 122-126.
[12]. Phothilimthana P.M., Jelvis T., Shah R., Totla N., Chasins S., Bodik R. Chlorophyll: Synthesis-aided compiler for low-power spatial architectures. ACM SIGPLAN Notices, vol. 49, no. 6, 2014, pp. 396-407.
[13]. Buchwald S. Optgen: A generator for local optimizations. Lecture Notes in Computer Science, vol. 9031, 2015, pp. 171-189.
[14]. Sasnauskas R., Chen Y. et al. Souper: A synthesizing superoptimizer. arXiv preprint arXiv:1711.04422, 2017.
[15]. Kjellin, M. Adapting a constraint-based compiler tool to a new VLIW architecture. Master's thesis, Uppsala University, Sweden, 2018, 74 p.
[16]. Koes D.R., Goldstein S.C. Near-optimal instruction selection on dags. In Proc. of the 6th Annual IEEE/ACM International Symposium on Code Generation and Optimization, 2008, pp. 45-54.
[17]. Buchwald S., Zwinkau A. Instruction selection by graph transformation. In Proc. of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems, 2010, pp. 31-40.
[18]. Blindell G.H., Carlsson M., Lozano R.C., Schulte C. Complete and practical universal instruction selection. ACM Transactions on Embedded Computing Systems (TECS), 2017, vol. 16, no. 5s, pp. 1-18.
[19]. Frolov N., Själander M., Larsson-Edefors P., McKee S.A. Declarative, SAT-solver-based Scheduling for an Embedded Architecture with a Flexible Datapath. In Proc. of the 11th Swedish System-on-Chip Conference, 2011, 4 p.
[20]. Wilken K., Liu J., Heffernan M. Optimal instruction scheduling using integer programming. ACM SIGPLAN Notices, vol. 35, no. 5, 2000, pp. 121-133.
[21]. Malik A.M., McInnes J., Van Beek P. Optimal basic block instruction scheduling for multiple-issue processors using constraint programming. International Journal on Artificial Intelligence Tools, vol. 17, no. 01, 2008, pp. 37-54.
[22]. Ершов А. П. Сведение задачи распределения памяти при составлении программ к задаче раскраски вершин графов. Доклады АН СССР, том 142, no. 4, 1962 г., стр. 785-787 / Ershov A.P. Reduction of the problem of memory allocation in programming to the problem of coloring the vertices of graphs. Soviet Mathematics, vol. 3, 1962, pp. 163-165.
[23]. Quintão Pereira F.M., Palsberg J. Register allocation by puzzle solving. In Proc. of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2008, pp. 216-226.
[24]. Fu C., Wilken K. A faster optimal register allocator. In Proc. of the 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002, pp. 245-256.
[25]. Buchwald S., Zwinkau A., Bersch T. SSA-based register allocation with PBQP. Lecture Notes in Computer Science, vol. 6601. 2011, pp. 42-61.
[26]. Вьюкова Н.И., Галатенко В.А., Самборский С.В. Генерация кода методом точного совместного решения задач выбора и планирования команд. Программная инженерия, no. 6, 2014 г., стр. 8-15 / Vyukova N.I., Galatenko V.A., Samborskij S.V. Code Generation by Exact Joint Solution of the Instruction Selection and Scheduling Tasks. Software Endineering, no. 6, 2014, pp. 8-15 (in Russian).
[27]. Chang C.M., Chen C.M., King C.T. Using integer linear programming for instruction scheduling and register allocation in multi-issue processors. Computers and Mathematics with Applications, 1997, vol. 34, no. 9, pp. 1-14.
[28]. Lozano R.C., Carlsson M., Blindell G.H., Schulte C. Combinatorial register allocation and instruction scheduling. ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 41, no. 3, 2019, pp. 1-53.
[29]. Eriksson M., Kessler C. Integrated code generation for loops. ACM Transactions on Embedded Computing Systems (TECS), vol. 11, no. 1, 2012, pp. 1-24.

Информация об авторах / Information about authors

Пётр Николаевич СОВЕТОВ – старший преподаватель кафедры корпоративных информационных систем. Сфера научных интересов: технологии компиляции и реализации языков программирования, предметно-ориентированные языки, совместное проектирование программно-аппаратных систем, решатели NP-полных задач.

Pyotr Nikolaevich SOVIETOV – senior lecturer in Department of Corporate Information Systems. Research interests: technologies of compilation and implementation of programming languages, domain-specific languages, codesign of software and hardware systems, solvers of NP-complete problems.

DOI: 10.15514/ISPRAS–2020–32(5)–4

Практика и перспективы применения семейства эмуляторов архитектур мейнфреймов IBM

А.В. Шмид, ORCID: 0000-0002-4672-1458 <ashmid@ec-leasing.ru>
ЗАО «ЕС-лизинг»,
117587, Россия, г. Москва, Варшавское шоссе, д. 125, стр.1
Национальный исследовательский университет «Высшая школа экономики»,
Россия, 101000, г. Москва, ул. Мясницкая, д. 20

Аннотация. В статье представлено описание семейства эмуляторов архитектур мейнфреймов IBM, история их разработки, их функциональные особенности и возможности применения, а также опыт многолетнего развития (начиная с 1994 года) состава эмуляторов, и сфер их применения. Решена относительно простая по современным представлениям задача создания виртуальной машины в операционной системе VSE/ESA для переноса в эту целевую среду унаследованных платформозависимых приложений. Задача решена сначала для ЕС ЭВМ в РФ, а потом для машин IBM 9221 в ФРГ и других западных странах. Перенос осуществлялся в среду современной по тем временам OS/390, а также IBM AIX. Обеспечена возможность виртуального исполнения любых существующих операционных систем мейнфреймов IBM в средах основных серверных ОС: Linux, Windows, AIX, Z/OS, ZLinux. Разработано решение по объединению образовавшихся виртуальных вычислительных узлов любых типов в гетерогенные территориально-распределенные вычислительные сети, обеспечивающие в частности множественное взаимное резервирование узлов в сети.

Ключевые слова: интерпретативное исполнение; эмулятор; хост система; гостевая система; IBM

Для цитирования: Шмид А.В. Практика и перспективы применения семейства эмуляторов архитектур мейнфреймов IBM. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 57-66. DOI: 10.15514/ISPRAS–2020–32(5)–4

Practice and Prospects for Using the Emulator Family of IBM Mainframe Architecture

A.V. Shmid, ORCID: 0000-0002-4672-1458 <ashmid@ec-leasing.ru>
EC-leasing Co.,
Warshavskoye Shosse, Moscow, 117587, Russia
National Research University Higher School of Economics,
20, Myasnitskaya st., Moscow, 101000, Russia

Abstract. This article describes the family of emulators for IBM mainframe architectures, their development history, functional features and capability, as well as the experience of many years (since 1994) of emulators development and their implementation area. There was sold the relatively simple task (for modern standards) of creating a virtual machine in the VSE/ESA operating system for transferring legacy platform-dependent applications to this target environment. The problem was solved at first for EU computers in Russia, and then for IBM 9221 in Germany and in the other western countries. The transfer was made to the OS/390 environment, and to IBM AIX, quite modern at that time. The virtual execution of any existing IBM mainframe operating systems in the main server OS environments: Linux, Windows, AIX, Z/OS, ZLinux had been provided. There was developed the solution for combining any types of formed virtual computing nodes into heterogeneous geographically distributed computing networks that provide, in particular, multiple mutual redundancy of nodes in the network.

Keywords: interpretive execution; emulator; host system; guest system, IBM

For citation: Shmid A.V. Practice and prospects for using the emulator family IBM mainframe architecture. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 57-66 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–4

1. Введение и постановка задачи

В настоящее время мейнфреймы IBM традиционно применяются в ряде критически важных автоматизированных систем управления (АСУ) РФ. Достаточно назвать два критически важных как для населения, так и для экономики примера: ЦБ РФ и Пенсионный фонд.

И эти системы, как и любые другие АСУ на любых других платформах, подвергаются ряду внешних и внутренних угроз. Но именно в силу значения хотя бы этих примеров для экономики и населения РФ любые работы, позволяющие устранить для систем этого класса хотя бы некоторые из существующих угроз, несомненно, актуальны.

Наша история борьбы с различными угрозами для машин этого класса началась очень давно.

К середине 90-х годов в РФ был прекращен выпуск вычислительных машин ЕС ЭВМ и перспективы возможности дальнейшей эксплуатации многочисленных унаследованных АСУ ЕС ЭВМ оказались под угрозой.

Одновременно появилась возможность поставки в РФ по приемлемым ценам архитектурно близких ЕС ЭВМ вычислительных машин IBM на то время новейшей архитектуры вместе с операционной системой (ОС) VSE/ESA как платформы, замещающей ЕС ЭВМ.

Проблема заключалась в том, что унаследованные АСУ были платформозависимы от ОС ЕС ЭВМ, и их перенос в среду исполнения ОС IBM потребовал бы значительной непредсказуемой по трудоемкости индивидуальной переработки каждой АСУ по отдельности.

Была поставлена задача: найти универсальное массовое решение, не требующее индивидуальной работы с каждой из АСУ ЕС ЭВМ при ее переносе в среду VSE/ESA. Проще говоря, по постановке задачи требовалось переносить АСУ «как есть и за один день». Наличие такого решения позволило бы сохранить на длительную перспективу огромный программный задел АСУ ЕС ЭВМ.

Стало очевидным, что в силу такой постановки задачи почти единственно возможным решением возникшей проблемы являлся бы перенос АСУ вместе с ее ОС ЕС в виртуальную машину ОС VSE/ESA, эмулирующую для ОС ЕС архитектуру ЕС ЭВМ в полном объеме.

Альтернатива переноса АСУ вместе с ОС ЕС в виртуальную машину ОС VM IBM была отвергнута по ценовым и техническим соображениям.

Эмулятор с заданной функциональностью для VSE/ESA был нами разработан и успешно прошел тестирование в лаборатории IBM в Беблингене, ФРГ.

Вскоре выяснилось, что подобные проблемы продления сроков эксплуатации платформозависимых унаследованных приложений имелись как в ФРГ, так и в других странах. Так, оказалось, что в результате прекращения выпуска вычислительных машин IBM 9221 их операционная система DPPX с критически важными для ФРГ на последующий длительный период приложениями (полный аналог ситуации РФ) осталась без вычислительных машин, без дисков и без сетевых адаптеров, с которыми DPPX должна работать.

С учетом дополнительных требований и на основе эмулятора для VSE/ESA был разработан эмулятор архитектур S/370, S/390 IBM для IBM OS/390, который позволял исполнять в гостевом режиме все существовавшие на тот момент ОС архитектур IBM (включая DPPX) с необходимым набором эмулируемых (в том числе уже снятых с производства) внешних устройств IBM. На этот эмулятор (ISX – Interpretative Space Executive) в 2003 году нами был получен патент США [1].

Проблема замены стоящих в поле множества машин IBM 9221 на современное оборудование IBM с OS/390 в ФРГ была успешно решена, и этот опыт был транслирован IBM и в другие страны.

Позже, по желанию заказчика (IBM) была также разработана версия подобного эмулятора для операционной системы AIX машин IBM P-series.

В итоге нами было реализовано несколько сотен лицензий эмуляторов для переноса унаследованных приложений вместе со средой их исполнения на платформы AIX и OS/390. Затем были также разработаны версии этого эмулятора для ОС Windows и Linux.

Когда появилась информация о Z-архитектуре IBM, эмуляция S/390 была расширена до уровня Z-архитектуры.

Со временем появилось целое семейство различных эмуляторов, отличительной особенностью которых явилась детальная эмуляция как принципов операций архитектур мейнфреймов от IBM S/370 до IBM Z [2-8], так и принципов работы всех видов внешних и сетевых устройств [9-14] в соответствии с технической документацией фирмы изготовителя [2-8]. Причем эмуляция могла быть исполнена на всех программных платформах основных вендоров.

В результате была обеспечена принципиальная возможность полного, либо частичного быстрого перевода прикладных систем, ранее разработанных для мейнфреймов IBM, на оборудование других вендоров, уже установленное в компании.

Как следствие, появились новые постановки задач по оптимизации таких общесистемных характеристик теперь принципиально гетерогенных систем как совокупная стоимость владения, доступность, катастрофоустойчивость.

2. Семейство эмуляторов ISX – системные возможности

К настоящему времени разработано семейство из 10 версий эмулятора ISX, в совокупности обеспечивающее возможности гостевого исполнения всего ассортимента ОС IBM для мейнфреймов IBM (как ранее выпускавшихся, так и современных – Guest OS на рис. 1) на оборудовании современных архитектур различных вендоров – Real Machine на рис. 1, в средах основных современных ОС – Host OS на рис 1.

Причем с точки зрения системного программирования для любой из Host OS (рис. 1) ISX вместе с исполняемой им гостевой ОС представляет собой единицу работы (Workunit на рис. 1), участвующую в борьбе за ресурсы этой ОС по ее правилам.

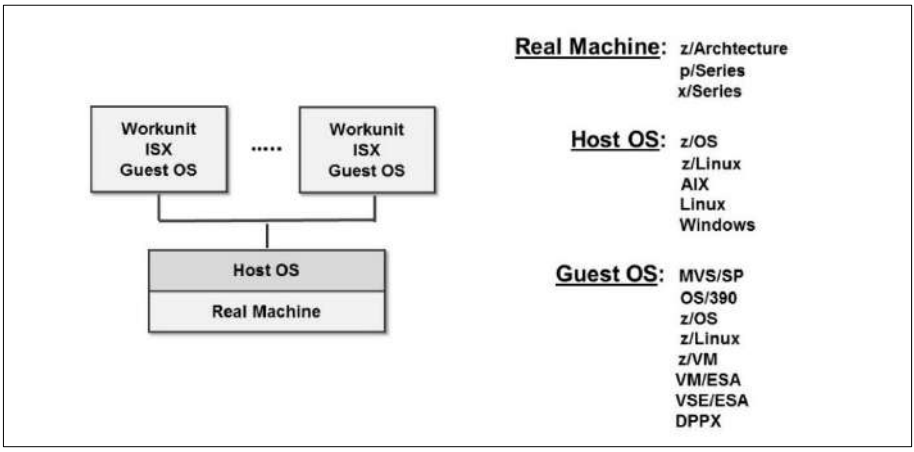


Рис. 1. Программа - исполнитель интерпретативного пространства
Fig. 1. Hardware Interpreter – Interpretive Space eXecutive (ISX)

Также с точки зрения системного программирования, отличия группы эмуляторов семейства, исполняемых на оборудовании z/Architecture, от оборудования прочих архитектур (рис. 1) состоит в том, что в случае машин IBM Z эмуляция ведется с использованием режима интерпретативного исполнения самой архитектуры: команды SIE – Start Interpretative Execution. Поэтому в этом случае производительность эмуляционного решения практически не снижается

В то же время для прочих архитектур в интересах эмулятора прописана вся система команд архитектуры любой из архитектур мейнфреймов IBM. По этой причине на производительность эмулятора ISX на любой платформе, кроме IBM/Z, очевидно влияют и издержки на эмуляцию архитектуры.

Кроме того, во втором случае исполнение гостевой системы практически ведется в контролируемой среде. Эмулятор не принимает к исполнению запросов гостевой ОС на исполнение команд, не прописанных в нем разработчиками эмулятора.

Эмуляторы ISX обеспечивают также эмуляцию как ранее производимого, так и современного оборудования мейнфреймов IBM (процессоров, каналов, устройств ввода-вывода, сетевых устройств машин IBM архитектур S/ 370, S/390, Z – табл. 1), обеспечивающую возможность исполнения Guest OS в средах любых Host OS (в том числе иных вендоров) на оборудовании этих вендоров (рис. 1).

Табл. 1. Эмуляция устройств ввода-вывода
Table 1. I/O Device Emulation

Unit record devices	3215, 2540R, 2540P, 1403
FBA devices	9332, 9335, 9336
CKD devices:	3380, 3390
Magnetic tape devices	3480, 3490
Display units	3274 Models 2, 3, 4, 5
Tokin-Ring adapter	IEEE 802.5
Channel-to-channel adapter	
LCS adapter	8232
OSA Express adapter	

Различные аспекты эмуляции как архитектур, так и внешних устройств общеизвестны и рассмотрены в ряде работ, например, [1], [15-19].

Основной же особенностью рассматриваемого семейства эмуляторов (как следует из рис. 1 и табл. 1) является его функциональная полнота, обеспечивающая перенос приложений с мейнфреймов на платформы большинства основных вендоров.

По мере разрастания семейства эмуляторов по охвату вендорных платформ и приобретения опыта работы с самыми разнообразными клиентами, становилось очевидной необходимость перехода к решению и более сложной задачи.

А именно, требовалось разработать также и инструменты взаимодействия между собой множества виртуальных вычислительных узлов ISX (в простейшем случае двух узлов: основного и резервного). В общем же случае нужно было поставить и решить технологическую задачу организации сетевого взаимодействия множества виртуальных вычислительных территориально-распределенных узлов ISX, работающих на платформах различных вендоров.

Эта технологическая задача также была решена, что и предопределило направление дальнейшего развития проекта и его перспективы.

3. Перспективы: от локальных резервированных ЦОД к построению гетерогенных территориально-распределенных резервированных вычислительных сред узлов ISX

В этом разделе перечисляются новые системные возможности семейства эмуляторов ISX, в совокупности обеспечивающие возможность построения гетерогенных территориально-распределенных резервированных вычислительных сред узлов ISX. Приводятся примеры их применения.

3.1 Системные возможности эмуляторов ISX и команды оператора по управлению узлами ISX

Концептуально управление множеством узлов ISX производится операторами узлов с применением команд управления узлом, доступных оператору.

Команды оператора позволяют задать окружение гостевой операционной системы (количество процессоров, размер памяти, используемые устройства ввода-вывода и т.д.). Гостевая система может быть загружена для исполнения командой IPL.

ISX обеспечивает команды для эмуляции аппаратных прерываний в гостевую систему, команды останова и возобновления работы гостевой системы, команды наблюдения за работой гостевой системы, команды снятия и анализа дампов гостевой системы, команды управления миграцией гостевых систем, команды обслуживания копий дисков гостевых систем. В табл. 2 представлен полный список команд оператора по управлению эмулятором ISX.

Табл. 2. Команды ISX
Table 2. ISX commands

BEGIN	продолжить исполнение гостевой системы
CAC	ввести данные для гостевой консоли
CACHE	сохранить историю исполнения гостевых инструкций
DEFINE	определить устройство ввода-вывода для гостевой системы
DETACH	удалить устройство ввода-вывода из гостевой конфигурации
DISPLAY	показать данные гостевой системы
DUMP	сохранить состояние гостевой системы
EXEC	выполнить последовательность команд эмулятора из командного файла
EXTERNAL	сгенерировать прерывание от ключа внешних прерываний
HELP	получить информацию о командах эмулятора
HMC	ввести данные для пультовой консоли
LOG	определить ведение журнала входных/выходных сообщений
IPL	выполнить начальную загрузку гостевой системы
PAUSE	приостановить процесс исполнения команд эмулятора
RECEIVE	принять данные состояния гостевой системы
RESTART	сгенерировать рестарт-прерывание
SHOW	показать состояние гостевой системы
SEND	послать данные состояния гостевой системы
SET	установить параметры работы эмулятора

SHUTDOWN	завершить работу эмулятора
STOP	остановить исполнение гостевой системы
STORE	изменить данные гостевой системы
SYNCH	синхронизировать базу и копию гостевого диска
TAKE	переопределить параметры устройства ввода-вывода
TRACE	получить трассу событий в гостевой системе

3.2 Локальные и удаленные диски гостевых систем

ISX поддерживает локальные и удаленные CKD DASD устройства. При определении дисков для гостевых систем оператором можно указать, в каком узле сети – собственном или удаленном – располагаются данные диска. Гостевая система «не замечает» разницы между локальным и удаленным диском, кроме времени исполнения операций ввода/вывода

Для работы с удаленным диском могут использоваться механизмы сжатия данных при пересылке данных по сети и буферизации в памяти для уменьшения нагрузки на сеть и ускорения операций ввода-вывода.

3.3 Ведение копий дисков гостевых систем

ISX поддерживает ведение локальных и удаленных копий CKD DASD устройств. Если для диска (базы) определена копия, то данные диска будут прочитываться из базы, пока она работоспособна, и будут записываться и в базу, и в копию. Если база перестает быть работоспособной, то копия начинает исполнять роль базы.

Сигнал о завершении операции ввода-вывода, полученный от узла ISX, в котором размещена копия, подтверждает синхронизацию дорожки диска. Если копия теряет связь с эмулятором ISX, который исполняет гостевую систему, то поддерживается только базовый том. Если базовый том теряет связь с эмулятором ISX, который исполняет гостевую систему, то поддерживается только копия. При восстановлении связи копия и база автоматически синхронизируются.

Первоначально копии дисков создаются при помощи команды оператора ISX SYNCH. Эта команда включает процесс создания копии диска независимо от того, исполняется ли гостевая система или нет.

3.4 Миграция гостевых систем по узлам ISX

ISX поддерживает специальные возможности для передачи образа гостевой системы в удаленный узел в сети TCP/IP. Для этого используются команды оператора SEND и RECEIVE. Если дисковые устройства гостевой системы имеют копии в удаленном узле, может быть разработан сценарий полной миграции гостевой системы из одного узла в другой и обратно или в третий узел. Пересылаются состояние процессоров, памяти, устройств ввода-вывода на момент исполнения команды SEND по указываемому в команде сетевому адресу. Гостевая система в текущем узле прекращает свою работу и возобновляет свою работу в новом узле сети TCP/IP после исполнения команды RECEIVE в этом узле.

3.5 Характерные примеры построения сетей гостевых ОС

На рис. 2 приведена типичная схема реализации критически важных приложений на мейнфрейме IBM для большой организации.

Как правило, на мейнфрейме заводится множество (несколько десятков) логических разделов (LPAR), в которых размещаются операционные системы Z/OS и Z/Linux.

На рис. 2 во всех LPAR все ОС находятся в гостевом режиме под ISX, что практически не влияет на производительность, поскольку виртуализация для гостевой ОС осуществляется интерпретативными командами самого мейнфрейма.

При этом, однако, виртуализация узла исполнения под ISX на основной машине Z придает этому узлу новые возможности сетевого взаимодействия с другими виртуальными узлами, в том числе на платформах любых других вендоров.

Например, новые возможности (по командам операторов виртуальных узлов): такие, например, как заведение любого количества виртуальных дисков в удаленных узлах, ведение синхронных копий, перевод на обработку образа z/OS в любую другую географическую точку, в том числе на платформу другого вендора.

Поскольку виртуальные внешние по отношению к z Machine узлы (рис. 2) могут находиться удаленно в различных географически точках, то может быть организовано множественное распределенное синхронное резервирование системы в целом или отдельно по каждому из LPAR, показанных на рис. 2, в том числе на виртуальных узлах другого удаленно расположенного мейнфрейма.

Таким образом, это решение может быть рассмотрено как импортозамещающая технология множественного территориально-распределенного резервирования критически важных приложений для мейнфреймов.

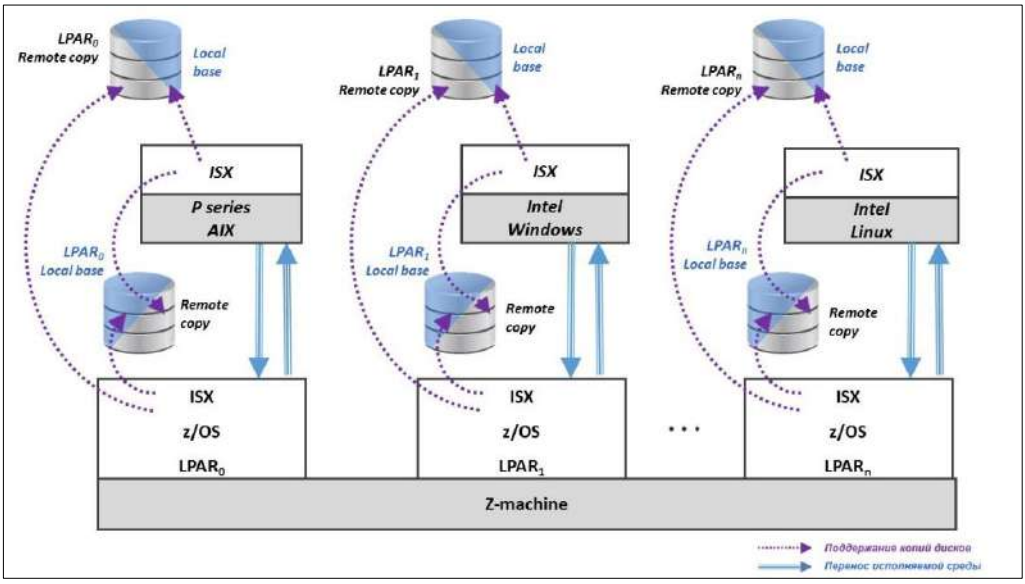


Рис. 2. Схема реализации критически важных приложений на мейнфрейме
Fig. 2. Implementation diagram for critical applications on the mainframe

Наконец, на рис. 3 показано, что в силу полной эмуляции всех архитектур IBM мейнфрейм для всех HOST машин, за исключением самой IBM Z, появление в будущем машин любой неизвестной архитектуры, работающей под управлением Linux, позволит безболезненно включить это вновь появившееся оборудование в проекты с предложенной технологией. Можно условно квалифицировать эту структуру как гетерогенную территориально-распределенную вычислительную сеть, в которой наращивание мощностей и функциональности, любые ремонты принципиально могут вестись без прерывания работы сети в целом – временным переводом обработки гостевой ОС со всеми приложениями с модифицируемого либо ремонтируемого узла в любой другой узел по команде оператора.

Также без остановки сети в целом динамически могут быть проведены изменения схемы резервирования узлов.

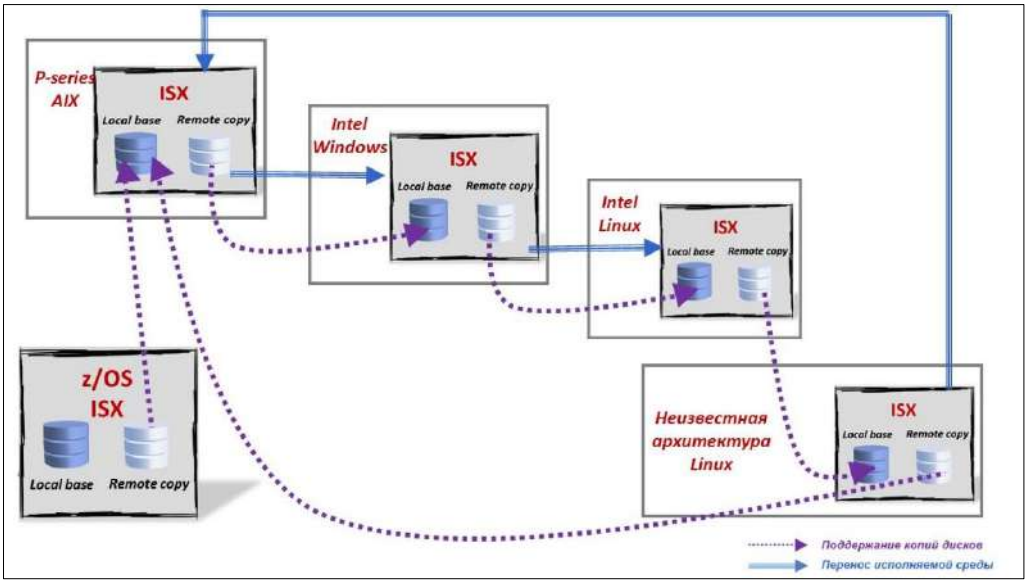


Рис. 3. Пример гетерогенной территориально-распределенной вычислительной среды
Fig. 3. Sample of a heterogeneous geographically distributed computing environment

Проблемы проектирования гетерогенных сетей рассмотрены в множестве работ, например, [20-27].

Концептуально наиболее близкое к ISX решение, обеспечивающее гостевое исполнение OS zLinux (только) в среде zLinux Z машин, а также в средах некоторых других платформ, описано в [28].

Однако присущие ему ограничения как по HOST, так и по GUEST OS не обеспечивают как решения задачи по переносу унаследованных приложений, так в общем случае и задачи по построению гетерогенных сетей в связи с отсутствием ряда необходимых для таких решений функциональных возможностей, например, функционально полного синхронного ведения копий дисков (см. табл. 2).

Особенностью же предлагаемого технического решения ISX, как семейства взаимодействующих эмуляторов, является тот факт, что его можно рассматривать в качестве «конструктора» с функциональной полнотой, необходимой и достаточной для самостоятельного создания набора виртуальных узлов ISX на платформах большинства вендоров, последующего объединения виртуальных узлов ISX в гетерогенную территориально-распределенную вычислительную среду, управляемую операторами среды с применением уже существующих инструментов управления созданными виртуальными узлами ISX (табл. 2).

4. Выводы

1. Разработанная технология переноса унаследованных приложений архитектур S/370, S/390, Z (мейнфреймов), допускающая их перенос на любую из ныне существующих серверных платформ, обеспечивает долгосрочную стратегическую устойчивость этих (критически важных) приложений РФ, в том числе в условиях импортозамещения.
2. Представляемая технология реализует альтернативный вариант множественного территориально-распределенного резервирования критически важных систем на

полностью отечественных программных продуктах, повышая живучесть таких проектов с перспективой простого переноса уже существующих проектов на полностью отечественное оборудование по достижению этим оборудованием целевой для проекта производительности.

В случае ее применения возникает также возможность динамического перемещения обработки оператором (пока) в наименее загруженные узлы вычислительной сети, несомненно приводящая к снижению совокупной стоимости владения проекта в целом.

3. Можно также констатировать, что проблемы переноса платформозависимых старых унаследованных приложений названных архитектур на какую-либо платформу без каких-либо изменений все-таки существуют и поныне.

   Только в РФ можно назвать более 100 крупных предприятий, которые сталкиваются с подобными проблемами.

Список литературы/References

[1] Alexander V. Shmid, Viacheslav V. Naumov. Virtual machines in OS/390 for execution of any guest system. US 6,530,078 B1 USA, 4 Mart 2003.
[2] GA22-7000-10. IBM System/370. Principles of Operation. 1987.
[3] SA22-7201-04. IBM Enterprise System Architecture/390. Principles of operation. 1997.
[4] SA22-7832-07. IBM z/Architecture. Principles of Operation. 2009.
[5] SA22-7095-1. IBM System/370 Extended Architecture Interpretive Execution. 1985.
[6] GA26-1660-1. IBM3310 Direct Access Storage Reference Manual.
[7] GA32-0274-05. IBM 3990/9390 Storage Control Reference.
[8] GA32-0127-03. IBM 3490 Magnetic Tape Subsystem Hardware Reference.
[9] GA23-0059-07. 3270 Information Display System. Data Stream Programmer's Reference.
[10] SA24-4236-03. Enterprise System/9000 Models 120,130,150,170, and 200. Work Station Subsystem Description and Reference.
[11] SA33-1600-03. Enterprice System/9000. 9221 processors. IBM Token-Ring and IEE802.3 Local Area Network. Programming Information.
[12] SA22-7203-00. Enterprise Systems Architecture/390. ESCON Channel-to_Channel Adapter.
[13] SA22-T091-01. Channel-to-Channel Adapter for the System/360 and System/370 I/O Interface.
[14] GC23-3870-01. IBM System/390. Planning for the System/390. Open Systems Adap.
[15] S. Matic. Emulation of hypercube architecture on nearest-neighbor mesh-connected processing elements. IEEE Transactions on Computers, vol. 39, № 5, 1990, pp. 698-700.
[16] M. Guttenbrunner, A. Rauber. Evaluating Emulation and Migration: Birds of a Feather? Lecture Notes in Computer Science, vol. 7634, 2012, pp. 158-167.
[17] D. Seto, M. Watanabe. A dynamic optically reconfigurable gate array – perfect emulation. IEEE Journal of Quantum Electronics, vol. 44, № 5, 2008, pp. 493-500.
[18] R. Helaihel, K. Olukotun. Emulation and prototyping of digital systems. NATO ASI Series, vol. 310, Hardware/Software co-design, 1996, pp. 339-366.
[19] A. Suzuki, S Oikawa. Implementing a simple trap and emulate VMM for the ARM architecture. In Proc. of the 17th International Conference on Embedded and Real-Time Computing Systems and Applications, vol. 1, 2011, pp. 371-379.
[20] M. Fang et al. High Performance X86 Emulation IO Architecture on Heterogeneous Processor Platform. In Proc. of the International Conference on Computational and Information Sciences, 2010, pp. 944-947.
[21] R.W. Heath, M. Kountouris. Modeling heterogeneous network interference. In Proc. of the Information Theory and Applications Workshop, 2012, pp. 17-22.
[22] B.H. Jung, N.O. Song, D.K. Sung. A network-assisted user-centric WiFi-offloading model for maximizing per-user throughput in a heterogeneous network. IEEE Transactions on Vehicular Technology, vol. 63, № 4, 2013, pp. 1940-1945.
[23] B. Ahlgren et al. Ambient networks: bridging heterogeneous network domains. In Proc. of the 16th International Symposium on Personal, Indoor and Mobile Radio Communication, vol. 2, 2005, pp. 937-941.
[24] Q. Li et al. Intracell cooperation and resource allocation in a heterogeneous network with relays. IEEE Transactions on Vehicular Technology, vol. 62, № 4, 2012, pp. 1770-1784.

[25] J. Liu et al. Uplink power control and interference coordination for heterogeneous network. In Proc. of the 23rd International Symposium on Personal, Indoor and Mobile Radio Communications, 2012, pp. 519-523.
[26] T. Alpcan, J.P. Singh, T. Başar. Robust rate control for heterogeneous network access in multihomed environments. IEEE Transactions on Mobile Computing, vol. 8, № 1, 2008, pp. 41-51.
[27] D. Kutscher, J. Ott. Service maps for heterogeneous network environments. In Proc. of the 7th International Conference on Mobile Data Management (MDM'06), 2006, 27 p.
[28] QEMU s390x Guest Support. URL: https://wiki.qemu.org/Documentation/Platforms/S390X, accessed 11.11.2020.

Информация об авторе / Information about author

Александр Викторович ШМИД – д.т.н., проф., генеральный директор ЗАО «ЕС-лизинг», заведующий базовой кафедрой ЗАО «ЕС-лизинг» «Информационо-аналитические системы» МИЭМ НИУ ВШЭ. Сфера научных интересов: big data.

Alexander Viktorovich SHMID – Doctor of Technical Sciences, Prof., Chief Executive Officer «EC-leasing» Co, Head of the Department of Information and Analytical Systems: Joint Department with EC-leasing, HSE. Research interests: big data.

DOI: 10.15514/ISPRAS–2020–32(5)–5

Динамическая компиляция пользовательских функций на языке PL/pgSQL

[1,2] *В.М. Джиджоев, ORCID: 0000-0002-0967-6766 <dzhivl@ispras.ru>*
[1] *Р.А. Бучацкий, ORCID: 0000-0001-8522-1811 <ruben@ispras.ru>*
[1] *М.В. Пантилимонов, ORCID: 0000-0003-2277-7155 <pantlimon@ispras.ru>*
[1,2] *А.Н. Томилин, ORCID: 0000-0002-4040-4179 <tom@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*

Аннотация. Многие современные СУБД предоставляют процедурное расширение декларативного языка программирования SQL, которое позволяет выполнять сложные вычисления с условной логикой на стороне сервера. Данные расширения стимулируют модулярность и переиспользование кода, упрощают процесс программирования логики некоторых приложений, а также позволяют избавиться от накладных расходов на передачу данных по сети и повысить производительность вычислений. В большинстве случаев для исполнения SQL-запросов и процедурных расширений используется подход интерпретации, который сопряжен с существенными накладными расходами по неявному вызову функций и выполнению лишних обобщенных проверок. Ситуация ухудшается тем, что для выполнения SQL-запросов и процедурных расширений, как правило, применяются два разных движка-исполнителя, в рамках которых необходимо осуществлять дополнительные операции по переключению контекста для сохранения промежуточных вычислений и контроля используемых ресурсов. В совокупности, совместная интерпретация SQL-запросов и процедурных расширений может в значительной степени снижать общую производительность СУБД. Одно из решений этой проблемы – динамическая компиляция. В рамках данной работы рассматривается метод динамической компиляции процедурного расширения PL/pgSQL в СУБД PostgreSQL с использованием компиляторной инфраструктуры LLVM. Работа выполняется в рамках динамического компилятора SQL-запросов, реализованного в виде расширения к СУБД PostgreSQL. Предлагаемый метод позволяет избавиться от накладных расходов, связанных с интерпретацией. Результаты тестирования на некоторых синтетических тестах показывают ускорение выполнения SQL-запросов в несколько раз.

Ключевые слова: динамическая компиляция; JIT-компиляция; выполнение запросов; UDF; СУБД; PostgreSQL; PL/pgSQL; LLVM

Для цитирования: Джиджоев В.М., Бучацкий Р.А., Пантилимонов М.В., Томилин А.Н. Динамическая компиляция пользовательских функций на языке PL/pgSQL. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 67-80. DOI: 10.15514/ISPRAS–2020–32(5)–5

Благодарности: Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект № 20-07-00877 А

Dynamic Compilation of User-Defined Functions in PL/pgSQL Language

[1,2] *V.M. Dzhidzhoev, ORCID: 0000-0002-0967-6766* <dzhivl@ispras.ru>
[1] *R.A. Buchatskiy, ORCID: 0000-0001-8522-1811 <ruben@ispras.ru>*
[1] *M.V. Pantilimonov, ORCID: 0000-0003-2277-7155 <pantlimon@ispras.ru>*
[1,2] *A.N. Tomilin, ORCID: 0000-0002-4040-4179 <tom@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. Many modern RDBMS provide procedural extensions for SQL programming language, which allow users to perform server-side complex computations. Use of such extensions improves modularity and code reuse, simplifies programming of application logic, and helps developers to avoid network overhead and enhance performance. Interpretation is mostly used to execute SQL queries and procedural extensions code, resulting in significant computational overhead because of indirect function calls and performing of generic checks. Moreover, most RDBMS use different engines for SQL queries execution and procedural extensions code execution, and it is necessary to perform additional computations to switch between different engines. Thus, interpretation of SQL queries combined with interpretation of procedural extensions code may drastically degrade performance of RDBMS. One solution is to use a dynamic compilation technique. In this paper, we describe the technique of dynamic compilation of PL/pgSQL procedural language for the PostgreSQL database system using LLVM compiler infrastructure. Dynamic compiler of PL/pgSQL procedural language is developed as part of PostgreSQL queries dynamic compiler. Proposed technique helps to get rid of computational overhead caused by interpretation usage. Synthetic performance tests show that the developed solution speeds up SQL queries execution by several times.

Keywords: dynamic compilation; JIT-compilation; query execution; UDF; DBMS; PostgreSQL; PL/pgSQL, LLVM

For citation: Dzhidzhoev V.M., Buchatskiy R.A., Pantilimonov M.V., Tomilin A.N. Dynamic compilation of user-defined functions in PL/pgSQL language. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 67-80 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–5

Acknowledgements. This work was supported by the Russian Foundation for Basic Research, project № 20-07-00877 A

1. Введение

При разработке приложений, использующих СУБД, в последние годы стало актуальным использование пользовательских функций (UDF) – фрагментов кода, хранимых в базе данных, которые можно вызывать из SQL-запросов. Одним из главных преимуществ использования пользовательских функций является механизм их исполнения – код пользовательских функций исполняется на стороне сервера СУБД. Такой механизм позволяет сократить обмен данными между сервером СУБД и приложением-клиентом, и, таким образом, увеличить эффективность использования СУБД. При этом, заметную долю времени выполнения SQL-запросов, содержащих вызовы пользовательских функций, занимает интерпретация кода пользовательских функций.

В данной работе предлагается подход, позволяющий уменьшить время выполнения таких запросов в СУБД PostgreSQL [1] с помощью динамической компиляции пользовательских функций. Альтернативным направлением оптимизаций времени выполнения пользовательских функций являются работы [2-4] посвященные автоматической трансформации процедурной логики в реляционные алгебраические выражения, которые затем встраиваются в тело SQL-запроса. При таком подходе планировщик запросов может использовать все доступные ему оптимизации для улучшения времени исполнения,

осуществлять распараллеливание вычислений, применять динамическую компиляцию, реализованную в SQL интерпретаторе.

Основным языком для написания хранимых процедур, пользовательских функций и обработчиков триггеров в PostgreSQL является PL/pgSQL [5]. PL/pgSQL – это процедурное расширение языка SQL в рамках структурной парадигмы программирования. Программа на языке PL/pgSQL состоит из операторов языка SQL, операторов ветвлений и циклов, может использовать переменные для передачи значений между операторами SQL, вызывать хранимые процедуры и функции.

Выполнение пользовательской функции, написанной на языке PL/pgSQL, в PostgreSQL осуществляется путём интерпретации абстрактного синтаксического дерева, соответствующего коду функции. В случае, когда при выполнении SQL-запроса пользовательская функция вызывается много раз, обход синтаксического дерева является повторным вычислением. Предлагаемый подход состоит в том, что все пользовательские функции языка PL/pgSQL, которые упоминаются в поступившем в PostgreSQL запросе, компилируются в машинный код путём однократного обхода их синтаксических деревьев. При исполнении поступившего запроса, вызовы интерпретатора функций PL/pgSQL заменяются на вызовы скомпилированного кода соответствующих функций. Таким образом, предлагаемый подход позволяет избавиться от повторных вычислений, связанных с многократным обходом абстрактных синтаксических деревьев пользовательских функций PL/pgSQL, и, предположительно, сократить время выполнения запросов, содержащих вызовы таких пользовательских функций.

Работа выполняется с использованием компиляторной инфраструктуры LLVM [6] в динамическом компиляторе [7-10] запросов PostgreSQL, разрабатываемом в ИСП РАН [11].

2. PL/pgSQL – процедурное расширение SQL

PL/pgSQL – процедурное расширение языка SQL для СУБД PostgreSQL. PL/pgSQL позволяет пользователям достаточно просто и безопасно определять функции и триггеры в PostgreSQL. С помощью него можно выполнять не только SQL-запросы, но и сложные вычисления. Функции PL/pgSQL могут использоваться везде, где допустимо использование встроенных функций PostgreSQL. PL/pgSQL в отличии от традиционного SQL, используемого в PostgreSQL и большинства других СУБД в качестве языка запросов, позволяет группировать блоки вычислений и последовательности запросов внутри сервера базы данных, что значительно увеличивает скорость обработки данных, решая такие проблемы как: излишнее взаимодействие клиента и сервера, передача промежуточных данных между исполнителями, увеличивающие время выполнения запросов, а также многократное выполнение одних и тех же запросов.

Все операторы PostgreSQL группируются в блоки операторов. Тело функции, написанной на PL/pgSQL так же является блоком операторов. Каждый блок операторов помечен некоторым идентификатором – меткой, содержит последовательность операторов, которые выполняются последовательно при выполнении блока, и может содержать объявления переменных, доступных для использования операторами блока. Блок операторов является оператором, то есть допустимо вложение блоков. Структура блока имеет следующий вид:

```
[ <<метка>> ]
[ DECLARE
    объявления ]
BEGIN
    операторы
END [ метка ];
```

Каждое объявление и каждый оператор в блоке должны завершаться символом ";"(точка с запятой). Блок, вложенный в другой блок, должен иметь точку с запятой после END, однако финальный END, завершающий тело функции, не требует точки с запятой.
Для определения функций, написанных на языки PL/pgSQL, используется команда CREATE FUNCTION. Например, функция gt, которая принимает на вход два аргумента типа integer и возвращает истину, если значение первого аргумента больше, чем значение второго, и ложь в противном случае, на языки PL/pgSQL выглядит так:

```
CREATE FUNCTION gt (x
integer, y integer)
RETURNS BOOLEAN AS $$
BEGIN
   IF x > y THEN
      RETURN TRUE;
   ELSE
      RETURN FALSE;
   END IF;
END;
$$ LANGUAGE plpgsql;
```

PL/pgSQL – типизированный язык, можно создавать переменные скалярных типов данных (тех же, что используются при написании выражений в SQL-запросах), типов кортежей с фиксированными типами элементов, типов кортежей с неизвестными во время компиляции типами элементов, типов массивов, типов структур.
Основными операторами PL/pgSQL являются:

- оператор присваивания переменной значения некоторого выражения;
- оператор выполнения фиксированного SQL-запроса;
- оператор выполнения SQL-запроса, формируемого в процессе выполнения функции;
- оператор ветвления;
- оператор цикла с логическим условием;
- оператор цикла по элементам массива;
- оператор цикла по результатам запроса;
- оператор возврата значения из функции.

2.1 Устройство интерпретатора PL/pgSQL

При вызове пользовательской функции, написанной на PL/pgSQL, осуществляется синтаксический и семантический анализ исходного кода функции. Результатом анализа является абстрактное синтаксическое дерево кода функции.

Каждая вершина абстрактного синтаксического дерева кода функции PL/pgSQL соответствует некоторому оператору в коде пользовательской функции. В зависимости от типа оператора, вершина хранит информацию, необходимую для выполнения соответствующего оператора – сам тип оператора, подвыражения, рёбра к связанным операторам.

Для выполнения пользовательской функции интерпретатор PL/pgSQL обходит абстрактное синтаксическое дерево функции. Каждому типу оператора PL/pgSQL соответствует

некоторая функция языка C в интерпретаторе, которая принимает на вход вершину абстрактного синтаксического дерева и выполняет вычисление, которое описано в вершине.

Пример абстрактного синтаксического дерева, соответствующего коду пользовательской функции на языке PL/pgSQL, приведен на рис 1. На этом рисунке также приведен граф вызовов функций интерпретатора, выполняющих вычисление пользовательской функции, представленной абстрактным синтаксическим деревом.

Так как функции интерпретатора, вычисляющие операторы PL/pgSQL, написаны в общем виде, без учёта специфики конкретного оператора PL/pgSQL (специфицирован только тип оператора, но не значения операндов, их типы и связанные операторы), при многократном вызове любого оператора осуществляются вычисления, результат которых определяется не данными, получаемыми во время вычисления оператора, а данными, известными после синтаксического и семантического анализа программы. Предлагаемое решение предполагает, что при кодогенерации известны вызываемые функции и операторы программы, известны типы переменных, выражений, и сгенерированный код операторов не будет содержать повторных вычислений, результат которых определяется информацией, известной во время кодогенерации.

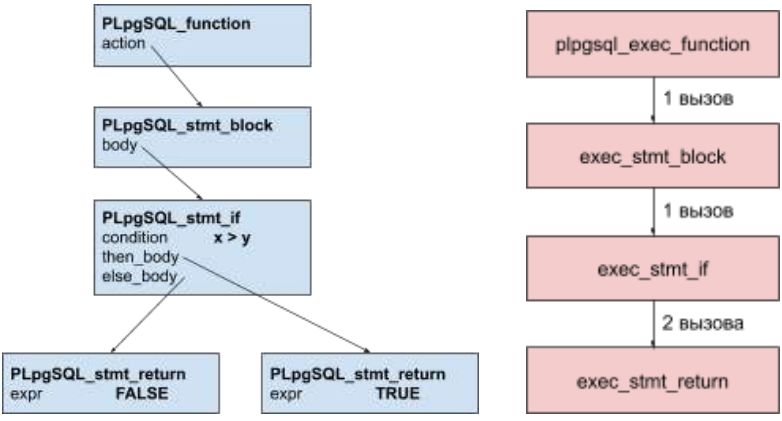


Рис. 1. Абстрактное синтаксическое дерево (слева) пользовательской функции gt, тело которой представлено в главе 2, и граф вызовов функций интерпретатора PL/pgSQL, осуществляющих вычисление функции gt (справа)

Fig. 1. Abstract syntax tree (left) of user defined function gt, which body is represented in chapter 2, and call graph of PL/pgSQL interpreter that executes gt (right)

3. Динамический компилятор запросов PostgreSQL, разрабатываемый в ИСП РАН

В ИСП РАН разрабатывается расширение [8] к СУБД PostgreSQL, реализующее динамическую компиляцию запросов с использованием компиляторной инфраструктуры LLVM.

Для исполнения запросов в PostgreSQL используется модель итераторов (Volcano-модель) [11]. Использование данной модели сопряжено с существенными накладными расходами, связанными с неявными вызовами функций и сопутствующими ошибками предсказания переходов, невозможноси выполнения оптимизаций и необходимостью сохранения состояния операторов между вызовами.

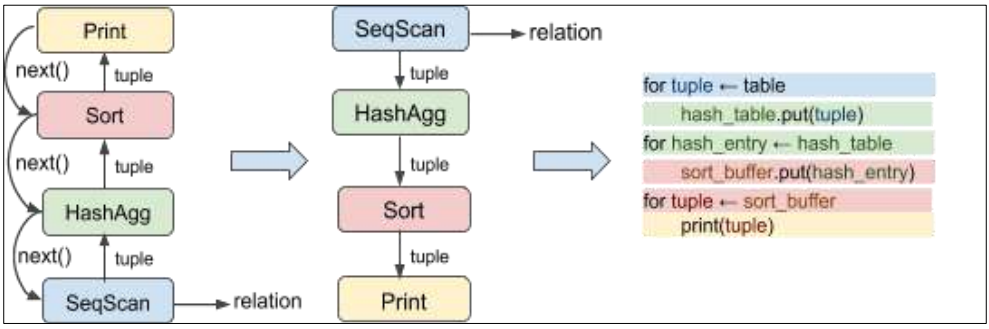


Рис. 2. Переход от модели итераторов (слева) к push-модели (посередине и справа) выполнения запроса

Fig. 2. Query plan transformation from iterator model into push-based model

В разработанном динамическом компиляторе запросов реализован переход от модели итераторов к модели явных циклов – push-модели (рис. 2) и алгоритм генерации кода в этой модели, основанный на сопоставленных каждому оператору функциях `consume` и `finalize`. Во время обход плана запроса выполняется генерации кода с использованием LLVM C API, при котором каждая вершина-оператор получает на вход функции `consume` и `finalize` от родительского оператора и генерирует код, в котором функция `consume` вызывается для каждого очередного возвращаемого родительскому оператору кортежа, а функция `finalize` – после возврата последнего кортежа. Генерации кода для внутренних операторов состоит в генерации функций `consume` и `finalize` (содержащих вызовы родительских функций `consume` и `finalize`) по одной на дочерний оператор и вызова генераторов дочерних операторов. В листовых операторах происходит генерация основных циклов обхода таблицы или индекса и вызов переданных функций `consume` и `finalize`.

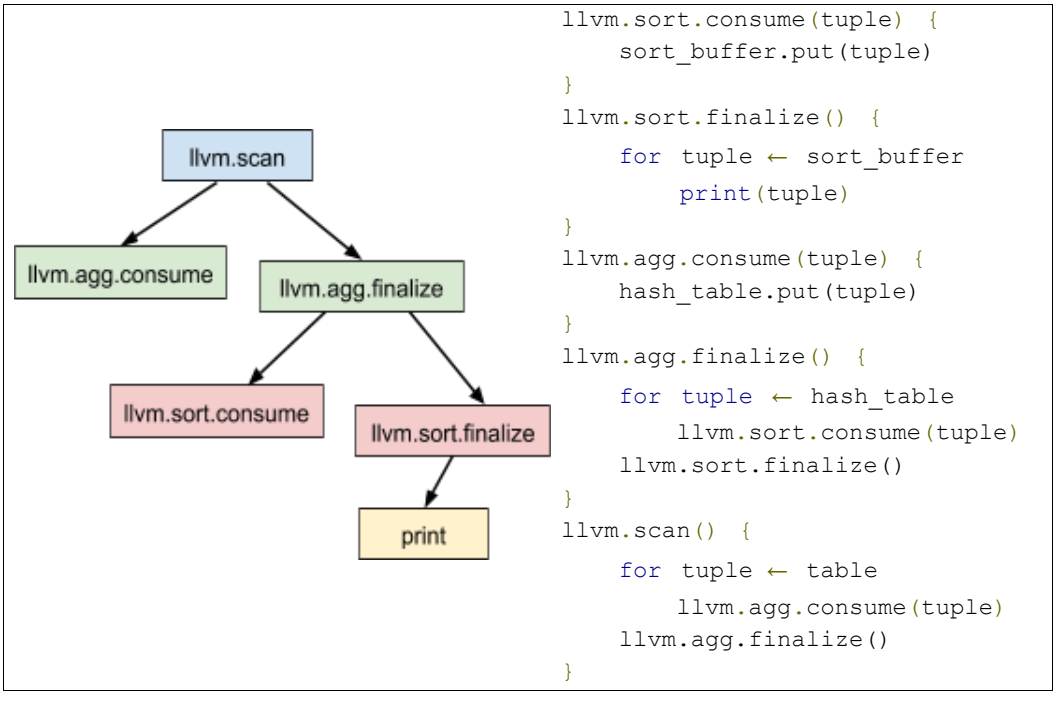


Рис. 3. Сгенерированный код в push-модели

Fig. 3. Generated code in push-based model

На рис. 3 показан результат генерации кода в push-модели. После встраивания получается код как на рис. 3 справа.

Также, в динамическом компиляторе запросов был разработан метод кэширования [10] и переиспользования сгенерированного динамическим компилятором машинного кода, позволяющий избавиться от накладных расходов на его создание (оптимизацию и компиляцию) при повторном использовании.

На бенчмарке TPC-H динамический компилятор запросов позволяет получить ускорение до 5.5 раз.

4. Динамическая компиляция пользовательских функций на языке PL/pgSQL

Предлагаемый в настоящей статье подход был реализован в виде дополнения к динамическому компилятору запросов к СУБД PostgreSQL, разрабатываемому в ИСП РАН (см. разд. 3). Предварительно был произведен анализ накладных расходов при интерпретации пользовательских функций на языке PL/pgSQL с целью идентификации наиболее горячих участков кода, которые можно оптимизировать с использованием рассматриваемого метода.

4.1 Анализ накладных расходов, связанных с вызовом PL/pgSQL функций

Для измерения расходов на интерпретацию пользовательских функций PL/pgSQL было проведено профилирование выполнения SQL-запросов, содержащих функции на языке PL/pgSQL, с помощью промышленного профилировщика perf [13], по результатам которого выяснилось, что на некоторых запросах доля времени вычисления пользовательских функций достигает более 75%. На рис. 4 представлен результат профилирования простого запроса "*select count(*) from tbl where gt(a,b)=true"*, где gt – это пользовательская функция на языке PL/pgSQL, представленная в разд. 2.

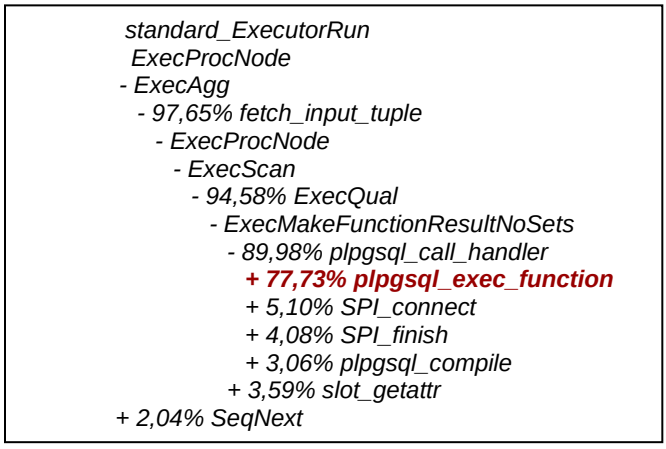


Рис. 4. Результат профилирования запроса "select count(*) from tbl where gt(a,b)=true;"
Fig. 4. Query "select count(*) from tbl where gt(a,b)=true;" profiling result

СУБД PostgreSQL выделяет память [14] в так называемых контекстах памяти (memory context), и тем самым реализует удобный способ управления выделением памяти в различных местах, с разными сроками жизни. При уничтожении контекста памяти освобождается вся выделенная в рамках данного контекст память.

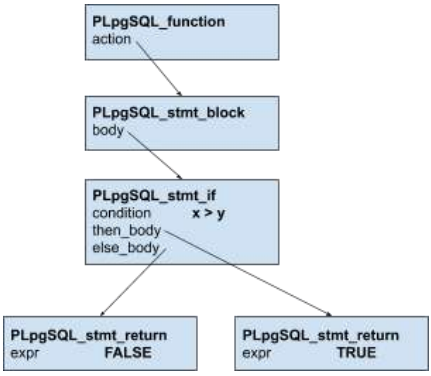
В интерпретаторе PL/pgSQL перед выполнением пользовательской функции вызывается функция SPI_connect, которая создает новый контекст памяти и делает его текущим, а после

выполнения вызывается функция SPI_finish для уничтожения контекста, созданного функцией SPI_connect, и восстановления предыдущего контекста памяти. Как видно из рис. 4, переключение контекста накладывает дополнительные расходы на время выполнения пользовательских функций, в дополнение к расходам, связанным с интерпретацией кода пользовательской функции (`plpgsql_exec_function`). Накладные расходы на переключение контекста могут быть уменьшены за счёт сокращения количества динамических аллокаций путем анализа графа вызова UDF функций, однако в данной работе эта оптимизация не рассматривается.

4.2 Реализация динамической компиляции пользовательских функций в компиляторе запросов СУБД PostgreSQL

Компиляция пользовательских функций производится при компиляции запроса, поступившего в СУБД. В случае, когда поступивший в СУБД SQL-запрос содержит вызов пользовательской функции PL/pgSQL, компилятор пользовательских функций производит построение и анализ абстрактного синтаксического дерева кода вызываемой пользовательской функции. Осуществляется обход абстрактного синтаксического дерева с целью проверки возможности компиляции пользовательской функции динамическим компилятором. В случае прохождения проверки, происходит компиляция пользовательской функции PL/pgSQL, иначе вызов обрабатывается интерпретатором PL/pgSQL, встроенным в PostgreSQL.

Компилятор пользовательских функций генерирует код на языке LLVM IR с использованием LLVM C API. Генерация кода пользовательской функции PL/pgSQL осуществляется путём обхода её абстрактного синтаксического дерева. Каждый узел этого дерева является представлением некоторого оператора PL/pgSQL, и для каждого узла при обходе дерева вызывается функция кодогенерации, соответствующая типу оператора, представленного данным узлом. Функция кодогенерации оператора каждого типа написана на основе функции интерпретации оператора соответствующего типа, содержащейся в интерпретаторе PL/pgSQL.




```
define  i64 @"llvm_gt"(%struct.FunctionCallInfoData*)
{entry:
  %args  = getelementptr
%struct.FunctionCallInfoData,
%struct.FunctionCallInfoData* %0, i32  0, i32  6
  %arg_ptr  = getelementptr  [100  x  i64],
[100  x  i64]* %args, i32  0, i32  0
  %arg  = load  i64, i64* %arg_ptr
          %arg_ptr2  = getelementptr  [100  x  i64],
          [100  x  i64]*%args, i32  0, i32  1
  %arg2  = load  i64, i64* %arg_ptr2
  %if_cond  = icmp gt  %arg, %arg2
  br  i1  %if_cond, label %then_body, label %else_body
then_body:
  ret  i64  1
else_body:
  ret  i64  0
}
```

Рис. 5. Слева – абстрактное синтаксическое дерево функции gt, справа – сгенерированный для этой функции LLVM IR

Fig. 5. On the left: abstract syntax tree of the gt function, on the right: generated LLVM IR for it.

При этом, если функция интерпретации оператора содержит вычисление, результат которого не зависит от значений времени выполнения пользовательской функции, функция кодогенерации оператора выполняет это вычисление во время кодогенерации, и скомпилированный код строится с учётом уже известного результата такого вычисления. Для тех вычислений в функции интерпретации оператора, результат которых нельзя определить во время кодогенерации, функция кодогенерации оператора генерирует код, осуществляющий данное вычисление во время выполнения пользовательской функции. В результате обхода абстрактного синтаксического дерева пользовательской функции генерируется функция на языке LLVM IR, которая динамически компилируется один раз с помощью LLVM и вызывается при выполнении SQL-запроса для вычисления пользовательской функции.

На рис. 5 приведён пример сгенерированного динамическим компилятором пользовательских функций LLVM IR для абстрактного синтаксического дерева функции gt, приведённой в разд. 2.

При динамической компиляции кода запроса и кода пользовательской функции на языке LLVM IR, компилятор LLVM осуществляет множество оптимизаций, одна из которых – встраивание кода вызываемой функции вместо её вызова. Такая оптимизация позволяет улучшить производительность скомпилированного кода за счёт избавления от вызова и проведения внутрипроцедурных оптимизаций кода. Пример того, как код пользовательской функции, сгенерированный динамическим компилятором PL/pgSQL, встраивается в код выполнения запроса, сгенерированный динамическим компилятором запросов к СУБД PostgreSQL, приведён на рис. 6.

1

```
llvm.ExecQual(tuple) {
   return
(*plpgsql_exec_function)(tuple)
}

llvm.scan() {
    for tuple ← table
        if llvm.ExecQual(tuple):
            print(tuple)
}
```

2

```
llvm_gt(tuple) {
    return tuple.a > tuple.b
}

llvm.ExecQual(tuple) {
    return llvm_gt(tuple)
}

llvm.scan() {
    for tuple ← table
        if llvm.ExecQual(tuple):
            print(tuple)
}
```

3

```
llvm.scan() {
    for tuple ← table
       if tuple.a > tuple.b:
            print(tuple)
}
```

Рис. 6. Слева – код запроса, сгенерированный динамическим компилятором запросов, с вызовом интерпретатора PL/pgSQL; справа – код, сгенерированный динамическим компилятором запросов и динамическим компилятором пользовательских функций: сверху – до оптимизации, снизу – после оптимизации встраивания функций

Fig. 6. On the left – code generated by JIT compiler with PL/pgSQL interpreter call; on the right – code generated by JIT compiler with PL/pgSQL compilation feature turned on: at the top before optimizations, at the bottom after inlining optimization.

Одной из ключевых задач при реализации компилятора пользовательских функций являлась реализация компиляции выражений, содержащихся в коде на языке PL/pgSQL. Сложность данной задачи заключалась в том, что типов операций, которые могут содержаться в выражении на языке PL/pgSQL, очень много. Для упрощения реализации и чтобы избежать дублирования кода, компиляция выражений была реализована следующим образом: если выражение на языке PL/pgSQL поддерживается в динамическом компиляторе [6], то выполняется соответствующая кодогенерация, иначе для его вычисления генерируется вызов интерпретатора PostgreSQL.

5. Результаты

Реализованный в рамках данной работы динамический компилятор поддерживает компиляцию пользовательских функций, написанных на подмножестве языка PL/pgSQL. Пользовательские функции, входящие в поддерживаемое подмножество языка, могут использовать переменные всех скалярных типов данных, поддерживаемых СУБД PostgreSQL, содержать операторы присваивания, ветвления, возврата значения из функции, простые циклы, выполнения произвольных фиксированных SQL-запросов.

Для тестирования производительности динамического компилятора были выбраны пользовательские функции, указанные в табл. 1. Было выполнено измерение времени выполнения запросов, содержащих вызов тестовых пользовательских функций, с использованием интерпретатора функций на языке PL/pgSQL и с использованием разработанного динамического компилятора функций на языке PL/pgSQL. Замер времени выполнения осуществлялся на компьютере с 16-ядерным процессором Intel Xeon CPU E5520 с ограничением тактовой частоты 2.27 ГГц под управлением операционной системы openSUSE Leap 15.1.

Сравнение производительности интерпретатора и компилятора выполнялось с использованием СУБД PostgreSQL версии 9.6.3. Размер таблицы table, из которой осуществлялась выборка данных, составляет 5 Гб; таблица содержит пять столбцов и 40000000 кортежей. Столбцы a и b, использующиеся в тестовых запросах, имеют тип integer.

Запросы, на которых производились замеры времени выполнения и результаты замеров указаны в табл. 2. В столбце «PG» указано время выполнения запроса с использованием интерпретатора запросов PostgreSQL и интерпретатора функций PL/pgSQL. В столбце «SQL JIT» указано время выполнения запроса с использованием динамического компилятора SQL-запросов и интерпретатора пользовательских функций на языке PL/pgSQL. В столбце «PL/pgSQL JIT» указано время выполнения запроса с использованием динамического компилятора SQL-запросов и разработанного динамического компилятора пользовательских функций на языке PL/pgSQL.

Табл. 1. Пользовательские функции для тестирования производительности динамического компилятора
Table 1. User defined functions used for performance testing of dynamic compiler

Имя пользовательской функции	Код функции на языке PL/pgSQL

gt	`CREATE FUNCTION gt (x integer, y integer)`
	`RETURNS boolean AS $$`
	`BEGIN`
	`  IF x > y THEN`
	`  RETURN TRUE;`
	`  ELSE`
	`  RETURN FALSE;`
	`  END IF;`

	`END;` `$$ LANGUAGE plpgsql;`

inc	`CREATE FUNCTION inc (x integer)`
	`RETURNS integer AS $$`
	`BEGIN`
	`  RETURN x + 1;`
	`END;`
	`$$ LANGUAGE plpgsql;`

sumN	`CREATE FUNCTION sumN (n integer)`
	`RETURNS INTEGER AS $$`
	`DECLARE`
	`sum integer := 0;`
	`idx integer := 0;`
	`BEGIN`
	`  LOOP`
	`  IF idx > n THEN`
	`  RETURN sum;`
	`  END IF;`
	`  sum = sum + idx;`
	`  idx = idx + 1;`
	`  END LOOP;`
	`END;`
	`$$ LANGUAGE plpgsql;`

is_prime	`CREATE FUNCTION is_prime (n integer)`
	`RETURNS BOOLEAN AS $$`
	`DECLARE`
	`idx integer := 2;`
	`BEGIN`
	`  LOOP`
	`  IF idx > n/2 THEN`
	`  RETURN TRUE;`
	`  END IF;`
	`  IF mod(n, idx) = 0 THEN`
	`  RETURN FALSE;`
	`  END IF;`
	`  idx = idx + 1;`
	`  END LOOP;`
	`END;`
	`$$ LANGUAGE plpgsql;`

Табл. 2. Сравнение времени выполнения запросов (в секундах) с использованием стандартного интерпретатора запросов PostgreSQL, динамического компилятора SQL-запросов и динамического компилятора SQL-запросов со включённым динамическим компилятором пользовательских функций на языке PL/pgSQL
Table 2. Comparison of execution times (in seconds) of PostgreSQL interpreter, SQL JIT compiler and SQL JIT compiler with enabled PL/pgSQL UDF compilation feature

№	Тестовый запрос	SQL JIT	PL/pgSQL JIT	PG	PG: SQL JIT	PG: PL/pgSQL JIT
1	`select is_prime(100000007);`	57.04	2.09	57.10	1.00	**27.32**
2	`select count(*) from table where gt(a,b)=true;`	77.99	16.01	95.22	1.22	**5.95**
3	`select sum(inc(a)) from table where gt(a,b)=true;`	137.99	25.54	167.72	1.22	**6.57**
4	`select sumN(a) from table where gt(a,b)=false;`	50.35	15.40	83.79	1.66	**5.44**

Можно отметить, что используемые для тестирования запросы являются синтетическими и в полной мере не могут отражать реальные задачи, но в то же время хорошо демонстрируют высокую производительность сгенерированного с помощью LLVM машинного кода, который сравним по эффективности с кодом на языке C. Например, в тестовом запросе №1 структура выбрана таким образом, чтобы минимизировать влияние используемого метода выполнения плана запроса на общее время выполнение запроса. Получается, что основной вклад в общее время выполнения данного запроса вносит время выполнения пользовательской функции. Исходя из структуры данного запроса и результатов замеров времени его выполнения, можно сделать вывод, что независимо от метода выполнения плана запроса, выполнение пользовательских функций PL/pgSQL методом динамической компиляции занимает на порядок меньше времени по сравнению с выполнением методом интерпретации.

Полученные результаты показывают, что динамическая компиляция пользовательских функций на языке PL/pgSQL в совокупности с динамической компиляцией запросов позволяет ускорить выполнение SQL-запросов в несколько раз, а на некоторых функциях, выполняющих целочисленные вычисления, на порядок.

6. Заключение

В данной работе демонстрируются промежуточные результаты исследования и реализации метода динамической компиляция процедурного расширения PL/pgSQL в СУБД PostgreSQL. Метод позволяет значительно увеличить производительность СУБД на SQL-запросах, использующих PL/pgSQL функции.

Динамическая компиляция подмножества языка PL/pgSQL реализована в динамическом компиляторе запросов СУБД PostgreSQL с использованием компиляторной инфраструктуры LLVM. Результаты тестирования динамической компиляция запросов с процедурными

расширениями на синтетических примерах демонстрируют ускорение времени выполнения запросов в несколько раз.

В будущем планируется обеспечить более широкую поддержку языка PL/pgSQL в динамическом компиляторе запросов – реализовать компиляцию функций, содержащих сложные операторы языка, рекурсивные вызовы и использующих переменные типов массивов и структур. Для увеличения производительности разработанного решения возможна оптимизации использования контекстов памяти, уменьшающая накладные расходы на переключение контекстов, а также применение кэширования динамически сгенерированного кода пользовательских функций. Особый интерес вызывает сравнительное тестирование производительности разработанного решения с производительностью других процедурных расширений СУБД PostgreSQL: PL/V8 [15], PL/Python [16], PL/Perl [17].

Список литературы / References

[1]. PostgreSQL official site. Available at: https://www.postgresql.org/, accessed: 20.10.2020.

[2]. Karthik Ramachandra, Kwanghyun Park, K. Venkatesh Emani, Alan Halverson, César A. Galindo-Legaria, Conor Cunningham Froid. Optimization of Imperative Programs in a Relational Database. Proceedings of the VLDB Endowment, vol. 11, no. 4, 2017, pp. 432-444.

[3]. Christian Duta, Denis Hirn, and Torsten Grust. Compiling PL/SQL Away. In Proc. of the 10th Annual Conference on Innovative Data Systems Research (CIDR'20), 2020, 8 p.

[4]. Denis Hirn and Torsten Grust. PL/SQL Without the PL. In Proc. of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), 2020, pp. 2677–2680.

[5]. PL/pgSQL – SQL Procedural Language. Available at: https://www.postgresql.org/docs/9.6/plpgsql-overview.html, accessed: 20.10.2020

[6]. The LLVM Compiler Infrastructure. Available at: http://llvm.org/, accessed: 20.10.2020.

[7]. Шарыгин Е.Ю., Бучацкий Р.А., Скворцов Л.В., Жуйков Р.А., Мельник Д.М. Динамическая компиляция выражений в SQL-запросах для СУБД PostgreSQL. Труды ИСП РАН, том 28, вып. 4, 2016 г., стр. 217-240. DOI: 10.15514/ISPRAS-2016-28(4)-13 / Sharygin E.Y., Buchatskiy R.A., Skvortsov L.V., Zhuykov R.A., Melnik D.M. Dynamic compilation of expressions in SQL queries for PostgreSQL. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 4, 2016. pp. 217-240 (in Russian).

[8]. Бучацкий Р.А., Шарыгин Е.Ю., Скворцов Л.В., Жуйков Р.А., Мельник Д.М., Баев Р.В. Динамическая компиляция SQL-запросов для СУБД PostgreSQL. Труды ИСП РАН, том 28, вып. 6, 2016, стр. 37-48. DOI: 10.15514/ISPRAS-2016-28(6)-3 / Buchatskiy R.A., Sharygin E.Y., Skvortsov L.V., Zhuykov R.A., Melnik D.M., Baev R.V. Dynamic compilation of SQL queries for PostgreSQL. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 37-48 (in Russian).

[9]. E. Sharygin, R. Buchatskiy, R. Zhuykov, and A. Sher. Runtime specialization of postgresql query executor. Lecture Notes in Computer Science, vol. 11, 2018, pp. 375–386.

[10]. Пантилимонов М.В., Бучацкий Р.А., Жуйков Р.А. Кэширование машинного кода в динамическом компиляторе SQL-запросов для СУБД PostgreSQL. Труды ИСП РАН, том 32, вып. 1, 2020, стр. 205-220. DOI: 10.15514/ISPRAS-2020-32(1)-11 / Pantilimonov M.V., Buchatskiy R.A., Zhuykov R.A. Machine code caching in PostgreSQL query JIT-compiler. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 205-202 (in Russian).

[11]. ISP RAS website. Available at: https://www.ispras.ru/en/, accessed: 20.10.2020.

[12]. Graefe G. Volcano – an extensible and parallel query evaluation system. IEEE Transactions on Knowledge and Data Engineering, vol. 6, no. 1, 1994, pp. 120-135.

[13]. Perf profiler website. Available at: https://perf.wiki.kernel.org/index.php/Main_Page, accessed: 20.10.2020.

[14]. PostgreSQL SPI Memory Management. Available at: https://www.postgresql.org/docs/9.6/spi-memory.html, accessed: 20.10.2020.

[15]. PLV8 – A Procedural Language in Javascript powered by V8. Available at: https://github.com/plv8/plv8, accessed: 20.10.2020.

[16]. PL/Python – Python Procedural Language. Available at: https://www.postgresql.org/docs/9.6/plpython.html, accessed: 20.10.2020.

[17]. PL/Perl – Perl Procedural Language, Available at: https://www.postgresql.org/docs/9.6/plperl.html, accessed: 20.10.2020.

Информация об авторах / Information about authors

Владислав Муратович ДЖИДЖОЕВ – лаборант отдела компиляторных технологий. Научные интересы: компиляторные технологии, оптимизации.

Vladislav Muratovich DZHIDZHOYEV – laboratory assistant at Compiler Technologies Department. Research interests: compiler technologies, optimization.

Рубен Артурович БУЧАЦКИЙ – младший научный сотрудник отдела компиляторных технологий. Научные интересы: компиляторные технологии, оптимизации.

Ruben Arturovich BUCHATSKIY – Researcher at Compiler Technology Department. Research interests: compiler technologies, optimizations.

Михаил Вячеславович ПАНТИЛИМОНОВ – стажер-исследователь отдела компиляторных технологий. Научные интересы: компиляторные технологии, СУБД.

Michael Vyacheslavovich PANTILIMONOV – Researcher in Compiler Technology Department. Research interests: compiler technologies, DBMS.

Александр Николаевич ТОМИЛИН – доктор физико-математических наук, профессор, главный научный сотрудник ИСП РАН, профессор МГУ. Научные интересы: операционные системы, архитектура и структура вычислительных систем, компиляторные технологии.

Alexander Nikolaevich TOMILIN – Doctor of Physical and Mathematical Sciences, Professor, Chief Researcher at ISP RAS, Professor at MSU. Research interests: Operating Systems, Architecture and structure of computing systems, compiler technologies.

DOI: 10.15514/ISPRAS–2020–32(5)–6

Синтез модели машинного обучения для обнаружения компьютерных атак на основе набора данных CICIDS2017

М.Н. Горюнов, ORCID: 0000-0003-0284-690X <max.gor@mail.ru>
А.Г. Мацкевич, ORCID: 0000-0001-9557-3765 <mag3d.78@gmail.com>
Д.А. Рыболовлев, ORCID: 0000-0003-4524-655X <dmitrij-rybolovlev@yandex.ru>
Академия ФСО России,
302015, Россия, г. Орел, ул. Приборостроительная, д. 35

Аннотация. В работе рассмотрены вопросы построения и практической реализации модели обнаружения компьютерных атак на основе методов машинного обучения. Среди доступных публичных наборов данных выбран один из наиболее актуальных – CICIDS2017. Для рассматриваемого набора данных подробно разработаны процедуры предварительной обработки данных и сэмплирования. При проведении экспериментов для сокращения времени вычислений в обучающей выборке оставлен единственный класс компьютерных атак – веб-атаки (brute force, XSS, SQL injection). Последовательно описана процедура формирования признакового пространства, позволившая существенно снизить его размерность – с 85 до 10 наиболее значимых признаков. Произведена оценка качества десяти наиболее распространенных моделей машинного обучения на полученной предобработанной подвыборке данных. Среди моделей (алгоритмов), которые продемонстрировали наилучшие результаты (k-nearest neighbors, decision tree, random forest, AdaBoost, logistic regression), с учетом минимального времени выполнения обоснован выбор модели «случайный лес». На этапе настройки и обучения выбранной модели осуществлен квазиоптимальный подбор гиперпараметров, что позволило добиться повышения качества модели в сравнении с ранее опубликованными результатами исследований. Произведена апробация синтезированной модели обнаружения атак на реальном сетевом трафике, показавшая ее состоятельность только при условии обучения на данных, собираемых в конкретной защищаемой сети, в виду зависимости ряда значимых признаков от физической структуры сети и настроек используемого оборудования. Сделан вывод о возможности применения методов машинного обучения для обнаружения компьютерных атак с учетом указанных ограничений.

Ключевые слова: информационная безопасность; система обнаружения атак; машинное обучение; дерево решений; случайный лес; сетевой трафик; компьютерная атака

Для цитирования: Горюнов М.Н., Мацкевич А.Г., Рыболовлев Д.А. Синтез модели машинного обучения для обнаружения компьютерных атак на основе набора данных CICIDS2017. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 81-94. DOI: 10.15514/ISPRAS–2020–32(5)–6

Synthesis of a Machine Learning Model for Detecting Computer Attacks Based on the CICIDS2017 Dataset

M.N. Goryunov, ORCID: 0000-0003-0284-690X <max.gor@mail.ru>
A.G. Matskevich, ORCID: 0000-0001-9557-3765 <mag3d.78@gmail.com>
D.A. Rybolovlev, ORCID: 0000-0003-4524-655X <dmitrij-rybolovlev@yandex.ru>
The Academy of Federal Security Guard Service of the Russian Federation,
35, Priborostroitelnaya st., Oryol, 302015, Russia

Abstract. The paper deals with the construction and practical implementation of the model of computer attack detection based on machine learning methods. Among available public datasets one of the most relevant was chosen - CICIDS2017. For this dataset, the procedures of data preprocessing and sampling were developed in detail. In order to reduce computation time, the only class of computer attacks (brute force, XSS, SQL injection) was left in the training set. The procedure of feature space construction is described sequentially, which allowed to significantly reduce its dimensions - from 85 to 10 most important features. The quality assessment of ten most common machine learning models on the obtained pre-processed dataset was made. Among the models (algorithms) that demonstrated the best results (k-nearest neighbors, decision tree, random forest, AdaBoost, logistic regression), taking into account the minimum time of execution, the choice of random forest model was justified. A quasi-optimal selection of hyper parameters was carried out, which made it possible to improve the quality of the model in comparison with the previously published research results. The synthesized model of attack detection was tested on real network traffic. The model has shown its validity only under the condition of training on data collected in a specific network, since important features depend on the physical structure of the network and the settings of the equipment used. The conclusion was made that it is possible to use machine learning methods to detect computer attacks taking into account these limitations.

Keywords: information security; intrusion detection system; machine learning; decision tree; random forest; network traffic; computer attack

For citation: Goryunov M.N., Matskevich A.G., Rybolovlev D.A. Synthesis of a machine learning model for detecting computer attacks based on the CICIDS2017 dataset. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 81-94 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–6

1. Введение

Бурное развитие информационных технологий в современном мире, расширение спектра и увеличение количества предоставляемых потребителям инфокоммуникационных услуг неизбежно сопровождается ростом числа угроз и факторов, приводящих к нарушению функционирования информационных систем и компьютерных сетей. По этой причине государственные институты и коммерческие компании уделяют повышенное внимание проблемам информационной безопасности и, как следствие, вопросам развития применяемых методов и средств обнаружения компьютерных атак [1].

В настоящее время основным методом выявления компьютерных атак, применяемым во всех современных средствах защиты информации, является сигнатурный анализ. Однако данный подход не позволяет обнаруживать новые виды деструктивных воздействий [2], что делает актуальным задачу разработки эвристических методов, способных детектировать ранее неизвестные типы атак [3].

Проведенный анализ ряда опубликованных на данный момент исследований [3-6] подтверждает возможность применения технологий машинного обучения для решения задач обнаружения компьютерных атак. Данное обстоятельство обусловливает целесообразность проведения прикладных исследований в указанной области, направленных на выработку конкретных предложений по построению моделей обнаружения и перспектив их практической реализации.

Цель исследования состоит в разработке модели машинного обучения для построения системы обнаружения компьютерных атак. Ее достижение предполагает решение

следующих основных задач: выбор обучающего набора данных, оценка значимости признаков и формирование признакового пространства, обоснование выбора модели машинного обучения и подбор квазиоптимальных параметров модели, оценка качества и апробация модели в реальных условиях. Новизна работы заключается в разработке макета системы обнаружения атак на основе современной модели машинного обучения и экспериментальной проверке применимости предлагаемых решений.

2. Постановка задачи и релевантные работы

Вопросы применения методов машинного обучения для обнаружения компьютерных атак активно обсуждаются в последние годы, при этом важным аспектом исследуемой предметной области является оценка возможности практической реализации разрабатываемых алгоритмов. По указанной тематике опубликовано достаточное количество работ, которые могут служить основой дальнейших исследований.

В статье [4] для решения задачи классификации и фильтрации сетевого трафика предложено использование модели типа «случайный лес» (random forest). В ходе экспериментов получены оценки эффективности работы предложенного алгоритма классификации в условиях наличия и отсутствия фонового сетевого трафика. В результатах исследования отмечено, что используемая модель машинного обучения демонстрирует высокую эффективность в отложенном режиме анализа (offline), но при обработке в реальном времени точность классификации снижается по причине высокой временной сложности и необходимости снижения сложности модели для соблюдения требований оперативности. Вместе с тем в работе не уточняются итоговые настройки используемой модели и не подтверждается их оптимальность.

В работе [5] рассматривается применение технологий нейронных сетей для обнаружения ботнет-атак. Предлагаемая модель (многослойный персептрон), обученная на публичном наборе данных CSE-CIC-IDS2018, демонстрирует на тестовых данных высокое качество обнаружения – близкое к единице значение F1-меры. Однако авторы не раскрывают особенностей формирования тестового набора данных и не оценивают возможность переобучения модели.

В исследовании [6] проводится экспериментальное сравнение семи различных моделей машинного обучения, используемых для обнаружения компьютерных атак. Рассмотрены алгоритмы: Naive Bayes, QDA, Random Forest, ID3, AdaBoost, MLP и K Nearest Neighbors. Обучение и тестирование моделей выполнено на публичном наборе данных CICIDS2017, при этом предварительно был проведен анализ значимости признаков и выполнено сокращение размерности признакового пространства. В работе получены высокие показатели точности обнаружения атак, однако предлагаемые решения не апробировались на реальном сетевом трафике. Кроме того, рассматриваемые в статье модели машинного обучения использовались с параметрами по умолчанию, что не позволяет сделать вывод о возможности оптимизации параметров и повышении точности обнаружения.

В отмеченных выше работах подробно описаны варианты реализации технологий машинного обучения, приведены оценки точности обнаружения атак в приложениях информационной безопасности. Однако опубликованные результаты носят недостаточно полный и системный характер с точки зрения оценки практической применимости, оставляя без ответов вопросы оптимальной настройки параметров моделей, апробации и применения предварительно обученных моделей на сетях с отличными характеристиками, встраивания разрабатываемых программных модулей в действующие системы и комплексы, формализации требований к производительности аппаратно-программной платформы и др.

Решаемая в данном исследовании задача заключается в практической реализации макета системы обнаружения атак на основе модели машинного обучения, оптимальном

(квазиоптимальном) выборе гиперпараметров модели и апробации предлагаемых решений на реальной сетевой инфраструктуре.

Макет разработан на языке Python с использованием свободно распространяемого пакета scikit-learn, исходный код проекта доступен для выполнения в среде Google Colaboratory: https://colab.research.google.com/github/infosecdemos/ml-2020/blob/master/ml-ids/web-attack-detection.ipynb.

3. Формирование признакового пространства

Для обучения системы обнаружения атак среди доступных публичных наборов данных (DARPA1998, KDD1999, ISCX2012, ADFA2013 и др.) выбран один из наиболее актуальных – «Intrusion Detection Evaluation Dataset» CICIDS2017 [8, 9]. Набор данных CICIDS2017 подготовлен Канадским институтом кибербезопасности по результатам анализа сетевого трафика в изолированной среде, в которой моделировались действия 25 легальных пользователей, а также вредоносные действия нарушителей [10]. Набор объединяет более 50 Гб «сырых» данных в формате PCAP и включает 8 предобработанных файлов в формате CSV, содержащих размеченные сессии с выделенными признаками в разные дни наблюдения. Краткое описание файлов и количественный состав набора данных представлены в табл. 1, 2.

Табл. 1. Описание файлов набора данных CICIDS2017
Table 1. Description of CICIDS2017 dataset files

№	Название файла	Содержащиеся атаки
1	Monday-WorkingHours.pcap_ISCX.csv	Benign (обычный трафик)
2	Tuesday-WorkingHours.pcap_ISCX.csv	Benign, FTP-Patator, SSH-Patator
3	Wednesday-workingHours.pcap_ISCX.csv	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed
4	Thursday-WorkingHours-Morning-WebAttacks.pcap_ ISCX.csv	Benign, Web Attack – Brute Force, Web Attack – Sql Injection, Web Attack – XSS
5	Thursday-WorkingHours-Afternoon-Infilteration.pcap_ ISCX.csv	Benign, Infiltration
6	Friday-WorkingHours-Morning.pcap_ISCX.csv	Benign, Bot
7	Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Benign, PortScan
8	Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Benign, DDoS

Табл. 2. Количественный состав набора данных CICIDS2017
Table 2. Number of attacks in the CICIDS2017 dataset

№	Тип записи	Количество записей
1	BENIGN	2359087
2	DoS Hulk	231072
3	PortScan	158930
4	DDoS	41835
5	DoS GoldenEye	10293
6	FTP-Patator	7938
7	SSH-Patator	5897
8	DoS slowloris	5796
9	DoS Slowhttptest	5499
10	Bot	1966
11	Infiltration	36
12	Heartbleed	11
13	Web Attack – Brute Force	1507
14	Web Attack – XSS	652
15	Web Attack – SQL Injection	21

3.1 Предварительная обработка данных

При проведении экспериментов для сокращения времени вычислений в обучающей выборке оставлен единственный класс компьютерных атак – веб-атаки (Brute Force, XSS, SQL Injection). Для этого использовался набор данных WebAttacks, подготовленный на основе обработки файла Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv. Набор WebAttacks включает 458968 записей, из которых 2180 относятся к веб-атакам, остальные – к нормальному трафику.

Каждая запись в наборе данных WebAttacks представляет собой сетевую сессию и характеризуется 84 признаками.

Этап предварительной обработки подвыборки веб-атак WebAttacks набора данных CICIDS2017 включает следующую последовательность действий:

1) исключение признака «Fwd Header Length.1» (признаки «Fwd Header Length» и «Fwd Header Length.1» являются идентичными);
2) удаление записей с null значениями идентификатора сессии «Flow ID» (из 458968 записей после удаления осталось 170366 записей);
3) замена нечисловых значений признаков «Flow Bytes/s», «Flow Packets/s» значениями -1;
4) замена неопределенных значений (NaN) и бесконечных значений значениями -1;
5) приведение строковых значений признаков «Flow ID», «Source IP», «Destination IP», «Timestamp» к числовым значениям методом label encoding;
6) кодирование ответов в обучающей выборке в соответствии с правилом: 0 – «нет атаки», 1 – «есть атака».

3.2 Сэмплирование

Предобработанный набор данных WebAttacks является несбалансированным: при общем количестве записей 170366 класс «нет атаки» объединяет 168186 экземпляров, класс «есть атака» – 2180 экземпляров [9]. Для устранения дисбаланса классов применяется метод случайного сэмплирования (субдискретизация, undersampling), заключающийся в удалении случайно выбранных экземпляров класса «нет атаки». Целевое соотношение количества экземпляров классов «нет атаки» и «есть атака» составляет 70% / 30%.

3.3 Оценка значимости и отбор признаков

При разработке модели машинного обучения важным является решение о том, какие признаки следует использовать в качестве входных данных для обучающего алгоритма [11]. Отбор признаков при формировании признакового пространства является обязательной процедурой как на подготовительном этапе (предшествующем обучению), так и на этапе оценки полученных результатов и последующей корректировки обучающей выборки и/или гиперпараметров модели [12].

Предварительно из признакового пространства исключены признаки «Flow ID», «Source IP», «Source Port», «Destination IP», «Destination Port», «Protocol», «Timestamp» в предположении, что признаки «формы» (соответствующие статистикам сетевого трафика) являются более значимыми для общего случая. Кроме того, исключаемые признаки адресации могут быть относительно легко подделаны злоумышленником и не должны учитываться при обучении [6].

Анализ значимости признаков выполнен с помощью встроенного механизма метода sklearn.ensemble.RandomForestClassifier (атрибут feature_importances_), реализующего энтропийный подход к оценке важности признаков.

Первые результаты оценки значимости показали сильную взаимосвязь признаков «Init_Win_bytes_backward», «Init_Win_bytes_forward» с метками класса в обучающей

выборке, что может свидетельствовать о допущенных погрешностях при формировании набора данных. Указанные признаки исключены из признакового пространства.

Итоговые результаты анализа значимости представлены на рис. 1, список ограничен первыми двадцатью признаками.

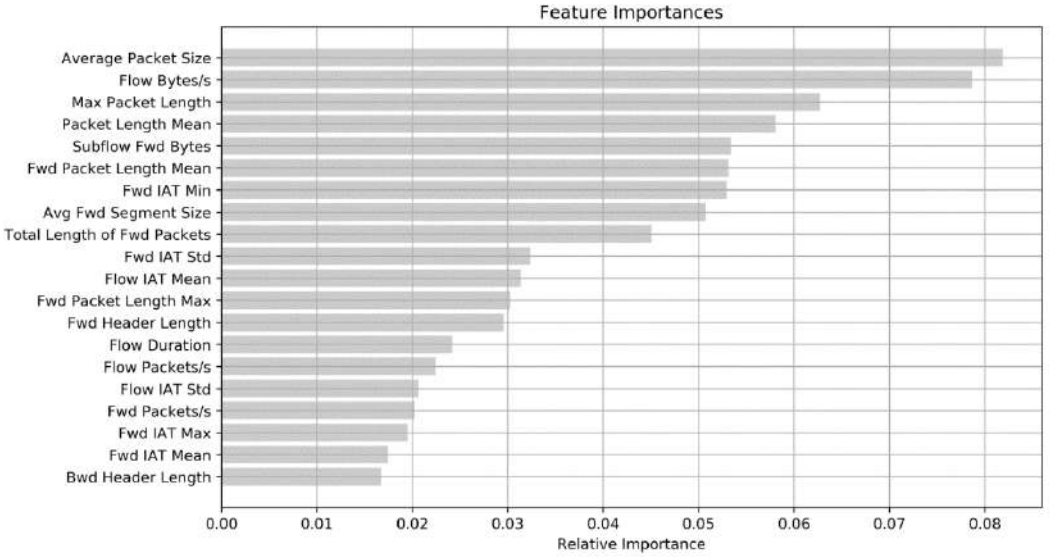


Рис. 1. Результаты анализа важности признаков
Fig. 1. Feature importance analysis results

3.4 Сокращение признакового пространства

На рис. 2 представлена корреляционная матрица с линейными коэффициентами корреляции (коэффициентами корреляции Пирсона), рассчитанными для всех пар двадцати наиболее значимых признаков. Насыщенность цвета заливки пропорциональна значению коэффициента корреляции.

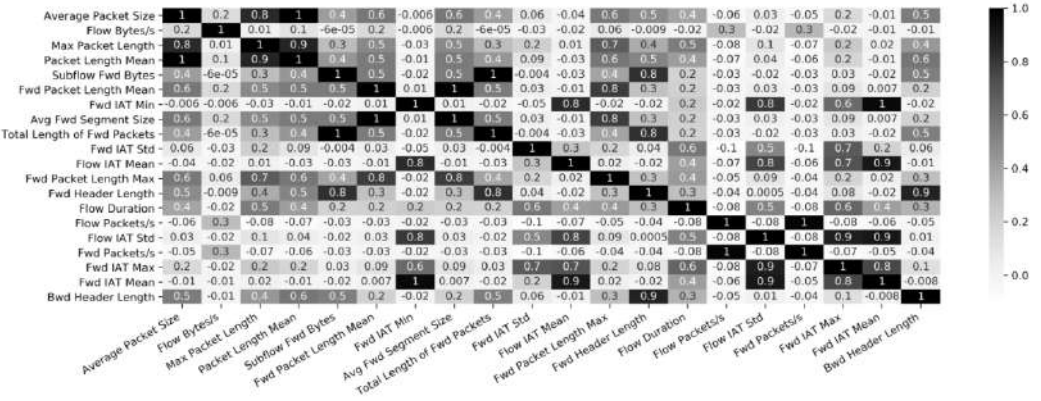


Рис. 2. Результаты корреляционного анализа двадцати наиболее значимых признаков
Fig. 2. The results of the correlation analysis of the twenty most significant features

Проведенный корреляционный анализ показал сильную зависимость между парами признаков:

1) «Average Packet Size» и «Packet Length Mean»;
2) «Subflow Fwd Bytes» и «Total Length of Fwd Packets»;

3) «Fwd Packet Length Mean» и «Avg Fwd Segment Size»;
4) «Flow Duration» и «Fwd IAT Total»;
5) «Flow Packets/s» и «Fwd Packets/s»;
6) «Flow IAT Max» и «Fwd IAT Max».

По результатам корреляционного анализа из признакового пространства исключены следующие признаки: «Packet Length Mean», «Subflow Fwd Bytes», «Avg Fwd Segment Size», «Fwd IAT Total», «Fwd Packets/s», «Fwd IAT Max».

После исключения признаков с наименьшей значимостью признаковое пространство сокращено до объединения 10 признаков:

1) «Average Packet Size», средняя длина поля данных пакета TCP/IP (далее – длина пакета);
2) «Flow Bytes/s», скорость потока данных;
3) «Max Packet Length», максимальная длина пакета;
4) «Fwd Packet Length Mean», средняя длина переданных в прямом направлении пакетов;
5) «Fwd IAT Min», минимальное значение межпакетного интервала (IAT, inter-arrival time) в прямом направлении;
6) «Total Length of Fwd Packets», суммарная длина переданных в прямом направлении пакетов;
7) «Fwd IAT Std», среднеквадратическое отклонение значения межпакетного интервала в прямом направлении пакетов;
8) «Flow IAT Mean», среднее значение межпакетного интервала;
9) «Fwd Packet Length Max», максимальная длина переданного в прямом направлении пакета;
10) «Fwd Header Length», суммарная длина заголовков пакетов, переданных в прямом направлении.

4. Выбор, настройка, обучение модели

4.1 Выбор модели

На этапе выбора модели для решения рассматриваемой задачи классификации была произведена оценка качества наиболее распространенных моделей машинного обучения на сбалансированной и предобработанной подвыборке веб-атак WebAttacks набора данных CICIDS2017.

Качество ответов классификаторов (моделей) сравнивалось с использованием следующих метрик:

- доля правильных ответов (accuracy);
- точность (precision, насколько можно доверять классификатору);
- полнота (recall, как много объектов класса «есть атака» определяет классификатор);
- F1-мера (F1-measure, гармоническое среднее между точностью и полнотой).

При определении значений метрик качества используются элементы матрицы ошибок (confusion matrix), соответствующие количеству правильных и неправильных ответов по результатам тестирования классификатора (табл. 3).

Табл.3. Описание матрицы ошибок классификатора
Table 3. Classifier error matrix description

	Ответ классификатора: класс 0, «нет атаки»	**Ответ классификатора: класс 1, «есть атака»**
Правильный ответ: класс 0, «нет атаки»	TN	FP
Правильный ответ: класс 1, «есть атака»	FN	TP

TP (True Positive) обозначает истинно-положительный ответ, TN (True Negative) – истинно-отрицательный ответ, FP (False Positive) – ложно-положительный ответ (ложное срабатывание, ошибка первого рода), FN (False Negative) – ложно-отрицательный ответ (пропуск атаки, ошибка второго рода). С учетом приведенных обозначений используемые метрики качества определяются следующими выражениями:

Accuracy = TP + TN / (TP + FP + FN + TN);

Precision = TP / (TP + FP);

Recall = TP / (TP + FN);

F1 = 2 * Precision * Recall / (Precision + Recall).

Для сравнения были выбраны следующие модели (алгоритмы) машинного обучения (в скобках указывается сокращенное обозначение и соответствующая реализация модели из состава пакета scikit-learn):

1) метод k ближайших соседей (KNN, sklearn.neighbors.KNeighborsClassifier);
2) метод опорных векторов (SVM, sklearn.svm.SVC);
3) дерево решений (CART, алгоритм обучения CART, sklearn.tree.DecisionTreeClassifier);
4) случайный лес (RF, sklearn.ensemble.RandomForestClassifier);
5) модель адаптивного бустинга над решающим деревом (AdaBoost, sklearn.ensemble.AdaBoostClassifier);
6) логистическая регрессия (LR, sklearn.linear_model.LogisticRegression);
7) байесовский классификатор (NB, sklearn.naive_bayes.GaussianNB);
8) линейный дискриминантный анализ (LDA, sklearn.discriminant_analysis.LinearDiscriminantAnalysis);
9) квадратичный дискриминантный анализ (QDA, sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis);
10) Многослойный персептрон (MLP, sklearn.neural_network.MLPClassifier).

Оценка качества классификаторов производилась на сбалансированной и предобработанной подвыборке веб-атак WebAttacks набора данных CICIDS2017 (соотношение нормального и аномального трафика 70% / 30%, 20 наиболее значимых признаков). В табл. 4 приведены полученные значения метрик качества, усредненные по результатам 5 итераций кросс-валидации.

Табл. 4. Результаты оценки качества классификаторов
Table 4. Results of evaluation of quality of classifiers

Модель (алгоритм)	**Accuracy**	**Precision**	**Recall**	**F1**	**Время выполнения, с**
KNN	0.971	0.942	0.961	0.969	4.57
SVM	0.705	0.669	0.036	0.602	176.04
CART	0.975	0.973	0.946	0.969	1.53
RF	0.971	0.978	0.943	0.970	1.14
AdaBoost	0.978	0.962	0.965	0.973	23.40
LR	0.955	0.939	0.914	0.963	15.80
Naive Bayes	0.722	0.520	0.956	0.754	0.47
LDA	0.939	0.921	0.872	0.941	2.23
QDA	0.872	0.978	0.597	0.949	1.28
MLP	0.904	0.921	0.912	0.776	93.83

Наилучшие результаты демонстрируют модели (алгоритмы) KNN, CART, RF, AdaBoost, LR. Принимая во внимание минимальное время выполнения, применение модели «случайный лес» (RF) для решения поставленной задачи является обоснованным выбором.

4.2 Настройка и обучение модели

В качестве используемого классификатора в исследовании выбрана модель типа «случайный лес» RandomForestClassifier свободно распространяемого пакета scikit-learn, реализующая построение ансамбля деревьев решений (decision tree) и характеризуемая потенциально высокой обобщающей способностью при решении рассматриваемого класса задач [7].

Для проведения квазиоптимального подбора гиперпараметров модели и оценки обобщающей способности используются четыре метрики качества: доля правильных ответов, точность, полнота, F1-мера.

Среди настраиваемых гиперпараметров выбраны следующие: количество деревьев в лесу (n_estimators), минимальное число объектов в одном листе дерева (min_samples_leaf), максимальная глубина дерева (max_depth), максимальное количество признаков для одного дерева (max_features).

Пример результатов подбора одного гиперпараметра (max_depth) при фиксированных значениях других гиперпараметров (n_estimators, min_samples_leaf, max_features) представлен на рис. 3 в виде зависимости метрики качества (F1-меры) от значения настраиваемого параметра (max_depth).

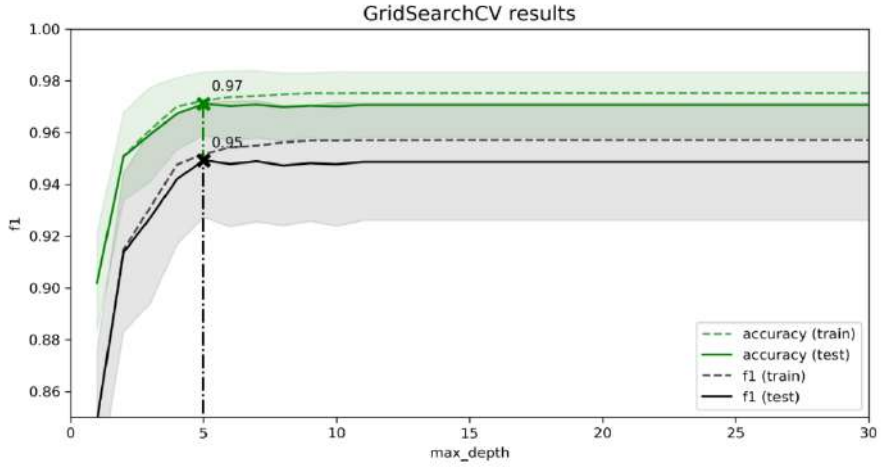


Рис. 3. Графики зависимости метрики качества (F-мера) модели «случайный лес» от максимальной глубины дерева в лесу (max_depth)

Fig. 3. Graphs of the dependence of the F-measure on the max_depth hyper parameter

Проведенный экспертный анализ дополнен результатами встроенного метода оптимизации параметров GridSearchCV библиотеки scikit-learn, итоговые значения параметров модели «случайный лес» представлены ниже:

```
RandomForestClassifier (bootstrap=True,
          class_weight=None, criterion='gini',
          max_depth=17, max_features=10, max_leaf_nodes=None,
          min_impurity_decrease=0.0, min_impurity_split=None,
          min_samples_leaf=3, min_samples_split=2,
          min_weight_fraction_leaf=0.0, n_estimators=50,
          n_jobs=None, oob_score=False, random_state=1, verbose=0,
          warm_start=False)
```

5. Тестирование и апробация модели

Настроенная и обученная модель RandomForestClassifier на валидационной выборке позволила получить оценку полноты (recall) 0.961 и F1-меры 0.971 (запуск № 1 в протоколе

эксперимента, табл. 5). Достигнутый результат свидетельствует о возможности повышения точности модели за счет квазиоптимального подбора гиперпараметров (в сравнении с результатами исследования [6], recall 0.94 и F1-мера 0.94).

Для апробации модели на реальной сетевой инфраструктуре разработан сетевой анализатор – сниффер (язык программирования C#, среда разработки Microsoft Visual Studio 2019). Анализатор позволяет перехватить передаваемый сетевой трафик и с использованием алгоритмов реконструкции TCP сессий свободно распространяемых программных продуктов WireShark и TCP Session Reconstruction Tool выделить отдельные сессии. Для каждой сохраненной сессии сниффер на основе алгоритма CICFLOWMETER [13] выделяет признаки и таким образом формирует набор данных.

В качестве атакуемого веб-приложения использовалась разработанная консоль администратора безопасности (язык программирования PHP) с единственным включенным модулем авторизации, функционирующая под управлением веб-сервера Apache.

Нормальный трафик соответствовал запросам легальных пользователей на подключение к консоли администратора и авторизацию. Вредоносный (аномальный) трафик моделировался программным средством OWASP ZAP и включал три типа атак: Brute Force, XSS, SQL Injection. Соотношение нормального и аномального трафика в реальном тестовом наборе данных составило 70% / 30%.

Табл. 5. Протокол эксперимента
Table 5. Experiment protocol

Эксперимент / Характеристика	Запуск №1	Запуск №2	Запуск №3
Этап обучения модели			
Используемый набор данных	Сбалансированная и предобработанная подвыборка веб-атак WebAttacks набора данных CICIDS2017. 7267 записей, из них 5087 экземпляров класса «нет атаки» и 2180 экземпляров класса «есть атака».		Сформированный набор данных, соответствующих реальному сетевому трафику
Обучающая выборка	70% записей используемого набора данных		70% записей используемого набора данных
Модель машинного обучения	RandomForestClassifier(max_depth=17, max_features=10, min_samples_leaf=3, n_estimators=50)		RandomForestClassifier (max_depth=None, max_features='auto', min_samples_leaf=1, n_estimators=250)
Признаковое пространство	1. Average Packet Size 2. Flow Bytes/s 3. Max Packet Length 4. Fwd Packet Length Mean 5. Fwd IAT Min 6. Total Length of Fwd Packets 7. Fwd IAT Std 8. Flow IAT Mean 9. Fwd Packet Length Max 10. Fwd Header Length		Flow Packets/s Flow IAT Max Bwd Packet Length Min Flow Duration Flow IAT Mean Flow IAT Std Average Packet Size Fwd Packet Length Max Total Packets Fwd Header Length
Этап тестирования модели			
Тестовая выборка	30% записей используемого набора данных. Тестовая и обучающая выборка не имеют пересечений.	100% записей сформированного набора данных, соответствующих реальному сетевому трафику	30% записей используемого набора данных. Тестовая и обучающая выборка не имеют пересечений.

Значения метрик качества			
Accuracy	0.983	0.456	0.858
Precision	0.982	0.812	0.812
Recall	0.961	0.033	0.966
F1	0.971	0.064	0.882

Проведенные эксперименты на сформированном наборе данных (запуски № 2, № 3 в протоколе эксперимента, табл. 5) показали невозможность применения модели, обученной на наборе данных CICIDS2017, по следующим причинам.

1. Анализ обучающей выборки (набор данных CICIDS2017) показывает, что характер моделируемых компьютерных атак в исследовании [10] отличается от реального. Так, атаки типа Brute Force присутствуют в сессиях с максимальными скоростями до 10 Кбит/с, что не соответствует случаям применения автоматизированных средств перебора паролей.
2. Среди десяти признаков с наибольшей значимостью четыре признака – «Flow Bytes/s» (скорость потока данных), «Fwd IAT Min» (минимальное значение межпакетного интервала в прямом направлении), «Flow IAT Std» (среднеквадратическое отклонение значения межпакетного интервала), «Flow IAT Mean» (среднее значение межпакетного интервала) – непосредственно зависят от физической структуры сети, в которой производится сбор сетевого трафика, а также настроек сетевого оборудования. В обучающем наборе данных сессии с признаками веб-атак записаны с низкими значениями скорости потока и высокими значениями межпакетных интервалов, что не соответствует характеристикам реальной сетевой инфраструктуры (сеть Ethernet 100 Мбит/с).

Полученные результаты свидетельствуют о необходимости обучения предлагаемой модели машинного обучения на наборе данных, полученном на основе анализа сетевого трафика в защищаемой сети. В противном случае, при использовании предобученной модели обязательным является соответствие физической структуры защищаемой сети и сети, в которой обучалась модель, а также настроек сетевого оборудования.

Оценка вычислительной сложности производилась косвенным способом: разработанный в среде Jupyter Notebook макет системы обнаружения веб-атак запускался на персональном компьютере (процессор Intel Core i5-2300 CPU @ 2300 ГГц, ОЗУ 8 Гб) в режиме обнаружения. Тестовый набор данных содержал около 70000 записанных сессий, время обнаружения составило 0,74669 с. Таким образом скорость обнаружения веб-атак оценивается величиной порядка 100000 сессий в секунду.

6. Заключение

Для оценки применимости современных методов машинного обучения в системах обнаружения компьютерных атак в исследовании проведен эксперимент с настройкой модели «случайный лес», обучением на публичном наборе данных CICIDS2017 и тестированием в реальных условиях.

Настройка параметров выбранного классификатора RandomForestClassifier свободно распространяемого пакета scikit-learn позволила на валидационной выборке получить оценку полноты (recall) 0.961 и F1-меры 0.971 для набора данных CICIDS2017, а также 0.966 и 0.882 соответственно для сформированного в исследовании набора данных.

Невозможность применения модели на тестовой выборке, полученной на основе анализа сетевого трафика в реальной компьютерной сети, объясняется тем, что при формировании обучающей выборки характер моделируемых компьютерных атак отличался от реального. Кроме того, среди десяти признаков с наибольшей значимостью четыре признака (скорость потока данных, минимальное значение межпакетного интервала в прямом направлении, среднеквадратическое отклонение значения межпакетного интервала, среднее значение

межпакетного интервала) непосредственно зависят от физической структуры сети, в которой производился сбор сетевого трафика, а также настроек сетевого оборудования. Отличия в физической структуре сетей и настройках оборудования приводят к возникновению ошибок классификатора и снижению точности модели.

Применение модели «случайный лес» становится возможным при условии предварительного обучения модели на наборе данных, полученном на основе анализа сетевого трафика в защищаемой сети (аналоге с соответствующими характеристиками) и содержащем признаки классифицируемых компьютерных атак. При этом на этапе сбора и подготовки обучающей выборки необходимо избегать разбалансированности распределения нормальных и аномальных записей, что может стать причиной переобучения модели и/или резкого увеличения числа ложных срабатываний классификатора [3].

Реализация предлагаемых решений в системах реального (близкого к реальному) времени предполагает эффективную обработку и анализ высокоскоростных потоков данных в условиях признакового пространства большой мощности и возможна лишь при наличии высокопроизводительной программно-аппаратной платформы. Снижение требований к производительности возможно за счет применения «многоуровневых» классификаторов, объединяющих быстрые низкоэффективные модели на этапе предварительной обработки и эффективные вычислительно сложные модели на более высоких уровнях [6].

Указанные обстоятельства вместе с известными результатами исследований предметной области позволяют сделать вывод о возможности применения методов машинного обучения для поиска аномалий и обнаружения компьютерных атак.

В заключении необходимо отметить, что перспективным направлением дальнейших исследований является разработка алгоритмов обнаружения компьютерных атак, основанных на использовании независимых от физической структуры сети и настроек используемого оборудования признаков, а также использовании технологий глубокого обучения нейронных сетей (deep learning), которые демонстрируют более высокие результаты по сравнению с другими методами при решении широкого круга задач (распознавания речи, классификации изображений, автоматического перевода и др.) [14].

Список литературы / References

[1]. Lee K.-F. AI Superpowers: China, Silicon Valley, and the New World Order. Houghton Mifflin Harcourt, 2018, 272 p.

[2]. Talabis M, McPherson R., Miyamoto I., Martin J. Information Security Analytics. Elsevier, 2015, 166 p.

[3]. Sumeet D., Xian D. Data Mining and Machine Learning in Cybersecurity. Auerbach Publications, 2011, 223 p.

[4]. Шелухин О.И., Ванюшина А.В., Габисова М.Е. Фильтрация нежелательных приложений интернет-трафика с использованием алгоритма классификации Random Forest. Вопросы кибербезопасности, № 2 (26), 2018 г., стр. 44-51. / Sheluhin O., Vanyushina A., Gabisova M. The Filtering of Unwanted Applications in Internet Traffic Using Random Forest Classification Algorithm. Voprosy kiberbezopasnosti, № 2 (26), 2018, pp. 44-51 (in Russian).

[5]. Kanimozhi V., Jacob T.P. Artificial Intelligence based Network Intrusion Detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing. ICT Express, vol. 5, issue 3, 2019, pp. 211-214.

[6]. Kostas K. Anomaly Detection in Networks Using Machine Learning. Master thesis. School of Computer Science and Electronic Engineering, University of Essex, 2018, 70 p.

[7]. Scikit-learn documentation. Random forest classifier. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier, accessed 16.08.2020.

[8]. Intrusion Detection Evaluation Dataset (CICIDS2017). Available at: https://www.unb.ca/cic/datasets/ids-2017.html, accessed 16.08.2020.

[9]. Panigrahi R., Borah S. A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. International Journal of Engineering & Technology, vol 7, no 3.24, 2018, pp. 479-482..

[10]. Sharafaldin I., Lashkari A.H., Ghorbani Ali A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proc. of the 4th International Conference on Information Systems Security and Privacy (ICISSP), 2018, pp. 108-116.
[11]. Leskovec J., Rajaraman A., Ullman J. Mining Of Massive Datasets. Cambridge University Press, 2014. 476 p.
[12]. Domingos P. A Few Useful Things to Know about Machine Learning. Communications of the ACM, vol. 55, № 10, 2012. pp. 78-87.
[13]. Lashkari H. Characterization of Tor Traffic Using Time Based Features. In Proc. of the 3rd International Conference on Information System Security and Privacy, 2017, pp. 253-262.
[14]. McAfee A., Brynjolfsson E. Machine, Platform, Crowd. W.W. Norton & Company, 2017. 416 p.

Информация об авторах / Information about authors

Максим Николаевич ГОРЮНОВ – кандидат технических наук. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, системы анализа защищенности, машинное обучение.

Maxim Nikolaevich GORYUNOV – Ph.D. Research interests: information security, intrusion detection systems, security analysis systems, machine learning.

Андрей Георгиевич МАЦКЕВИЧ – кандидат технических наук, доцент. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, системы антивирусной защиты, машинное обучение, криптографические методы защиты информации.

Andrey Georgievich MATSKEVICH – Ph.D., associated professor. Research interests: information security, intrusion detection systems, anti-virus protection systems, machine learning, cryptographic methods for protecting information.

Дмитрий Александрович РЫБОЛОВЛЕВ – кандидат технических наук. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, машинное обучение, криптографические методы защиты информации.

Dmitry Aleksandrovich RYBOLOVLEV – Ph.D. Research interests: information security, intrusion detection systems, machine learning, cryptographic methods for protecting information.

DOI: 10.15514/ISPRAS–2020–32(5)–7

Реализация маркирования в подсистеме печати ОС семейства Windows на основе виртуального XPS-принтера

[1] *С.В. Козлов, ORCID: 0000-0003-1269-1681 <kozlov_sv@mail.ru>*
[1] *С.А. Копылов, ORCID: 0000-0003-2841-5243 <gremlin.kop@mail.ru>*
[2] *Б.В. Кондратьев, ORCID: 0000-0001-6348-117X <gae@mil.ru>*
[3] *Д.О. Обыденков, ORCID: 0000-0002-9296-6333 <obydenkov@ispras.ru>*
[1] *Академия Федеральной службы охраны Российской Федерации, 302015, Россия, г. Орёл, ул. Приборостроительная, д. 35*
[2] *Министерство обороны Российской Федерации, 119160, г. Москва, ул. Знаменка, д.19*
[3] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*

Аннотация. В статье представлен подход к маркированию электронных документов, выводимых на печать посредством реализации виртуального XPS-принтера в операционных системах семейства Windows. Разработанный подход позволяет осуществлять маркирование электронных документов в процессе печати вне зависимости от формата представления исходного документа и требований, предъявляемых к процессу печати. В ходе разработки и реализации подхода к маркированию осуществлен сравнительный анализ технических решений в области маркирования электронных документов, определены их достоинства и недостатки. Определены требования и ограничения, накладываемые на подход к маркированию. Обоснован выбор технологии виртуальных принтеров для реализации маркирования документов в процессе вывода их на печать. В ходе реализации подхода маркирования, основанного на технологии виртуальных принтеров, приведена структура организации и взаимодействия процесса маркирования с компонентами службы печати операционных систем семейства Windows. Разработана архитектура драйвера виртуального принтера, реализующего маркирование документов. Описан процесс практической реализации внедрения маркера в электронный документ посредством разработанного виртуального принтера. В ходе практической реализации подхода маркирования представлено описание особенностей взаимодействия разработанного фильтра печати с подсистемой печати, параметров обработки метаданных и особенностей организации многопоточной реализации сервера, выполняющего маркирование. Рассмотрены особенности реализации разработанного подхода к маркированию в отдельных операционных системах семейства Windows. Определены ограничения и допущения для каждой из рассмотренных операционных систем. Сформулированы требования к процессу маркирования и направления дальнейших исследований.

Ключевые слова: защита от утечки информации; маркирование документов; виртуальный принтер печати, подсистема печати; операционные системы семейства Windows

Для цитирования: Козлов С.В., Копылов С.А., Кондратьев Б.В., Обыденков Д.О. Реализация маркирования в подсистеме печати ОС семейства Windows на основе виртуального XPS-принтера. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 95-110. DOI: 10.15514/ISPRAS–2020–32(5)–7

Implementing Watermarking Based on a Virtual XPS Printer for Windows Operating Systems

[1] *S.V. Kozlov, ORCID: 0000-0003-1269-1681 <kozlov_sv@mail.ru>*
[1] *S.A. Kopylov, ORCID: 0000-0003-2841-5243 <gremlin.kop@mail.ru>*
[2] *B.V. Kondrat'ev, ORCID: 0000-0001-6348-117X <gae@mil.ru>*
[3] *D.O. Obydenkov, ORCID: 0000-0002-9296-6333 <obydenkov@ispras.ru>*

[1] *The Academy of Federal Security Guard Service of the Russian Federation, 35, Priborostroitelnaya st., Oryol, 302015, Russia*
[2] *Ministry of Defence of the Russian Federation, 19, Znamenka st., Moscow, 119160, Russia*
[3] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*

Abstract. The article presents an approach to electronic documents printed marking by implementing a virtual XPS printer in Windows operating systems. The developed approach allows marking electronic documents during printing, regardless of the document presentation format and requirements for the printing process. During the marking approach development and implementation, a comparative analysis of technical solutions in the field of marking electronic documents was carried out, advantages and disadvantages were determined. Requirements and limitations imposed on the marking approach are defined. The virtual printer technology choice for the marking documents implementation in the printing process is substantiated. In the course of the marking approach implementing based on virtual printer technology, the structure of the organization and interaction of the marking process with the components of the print service of Windows family operating systems is given. The architecture of a virtual XPS printer driver has been developed. The process of practical implementation of the marker embedding into an electronic document using the developed virtual printer is described. In the process of the marking approach practical implementation, a interaction features description of the developed print filter with the printing subsystem, the parameters of metadata processing and the organization features of the marking server multithreaded implementation is presented. The implementation features of the developed marking approach in individual operating systems of the Windows family are considered. Limitations and assumptions are determined for each of the considered operating systems. Marking process requirements and further research directions are formulated.

Keywords: information leakage protection; document marking; virtual printer; printing subsystem; operating systems of the Windows family

For citation: Kozlov S.V., Kopylov S.A., Kondrat'ev B.V., Obydenkov D.O. Implementing watermarking based on a virtual XPS printer for Windows operating systems. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 95-110 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–7

1. Введение

С развитием информационно-телекоммуникационных сетей в последние десятилетия в значительной степени увеличилось количество цифровой информации, содержащей как персональные данные пользователей, так и конфиденциальную информацию. Помимо роста числа и разнородности цифровых данных отмечается возросшее количество инцидентов нарушений информационной безопасности, связанных с утечкой защищаемой информации. Так, по данным аналитического центра InfoWatch в 2019 году в мире обнародовано и зарегистрировано 1276 случаев утечки конфиденциальной информации, а также информации, содержащей различные виды тайн, что на 23% превышает количество инцидентов, зарегистрированных за аналогичный период 2018 года (1039 утечек). В результате данных утечек скомпрометировано более 8,74 миллиарда записей персональных данных. При этом, более чем в 55 процентах случаев нарушение информационной безопасности осуществлялось внутренними нарушителями [1]

В распределении утечек по типам информации, подвергшейся компрометации, на персональные данные приходится 71,2% от общего числа, на информацию, содержащую

коммерческую тайну – 22% и сведения, составляющие государственную тайну – 6,8%. Одним из наиболее распространенных типов данных, подвергшихся компрометации, является текстовая информация [2]. При этом почти половина (47,7%) от общего числа утечек реализована через напечатанные на бумаге электронные текстовые документы.

Высокий процент утечек напечатанных текстовых документов обусловлен тем, что существенная часть документооборота совершается в неэлектронной форме. Кроме того, во многих компаниях и государственных организациях далеко не всегда соблюдаются правила обращения с бумажными носителями. При этом утечка напечатанных на бумаге текстовых документов реализуется посредством сканирования или фотографирования бумажного текстового документа с последующим выносом на машинном носителе информации или отправкой за пределы контролируемого периметра сети сформированного изображения, содержащего исходный текстовый документ.

Для защиты печатных копий электронных текстовых документов от утечки широкое распространение получили методы маркирования документов, основанные на технологии цифровых отпечатков и цифровых водяных знаков (ЦВЗ), позволяющие осуществить обнаружение факта утечки информации, а также идентифицировать ее источник [3-6].

2. Обзор существующих решений в области маркирования документов

Технология цифровых отпечатков основана на извлечении из анализируемого объекта (текстового документа, изображения, данных мультимедиа и т. д.) уникальных свойств («отпечатков»), характеризующих исходный документ [7]. Извлеченные характеристики составляют базу данных «отпечатков» (сигнатур), по которой осуществляется контентный анализ передаваемой информации [8, 9].

К практическим решениям в области маркирования, основанным на технологии цифровых отпечатков, относится программный продукт Trace Doc [10]. В процессе маркирования на основе методов машинного обучения и теории распознавания образов для электронного текстового документа, выводимого на печать, создается уникальная копия. Формула создания этой копии, а также метка времени и информация, идентифицирующая пользователя, сохраняются в базу данных. В случае утечки выполняется поиск формулы, позволяющей получить из исходного документа копию, попавшую в публичный доступ (по ошибке или намеренно). Достоинством рассмотренного решения является возможность обнаружения факта утечки и идентификации источника утечки по фотографии или изображению, полученному посредством сканирования напечатанного текстового документа. К недостаткам Trace Doc относится необходимость создания базы данных исходных электронных текстовых документов.

В отличие от технологии цифровых отпечатков в подходе к маркированию, основанном на технологии ЦВЗ, в исходный текстовый документ при выводе на печать внедряется дополнительная информация (маркер) [11].

В качестве маркера может выступать идентификационная информация, характеризующая владельца данных, сами данные, а также другая метаинформация. К существующим решениям в области маркирования электронных текстовых документов, выводимых на печать, относятся такие решения, как EveryTag [12], SafeCopy [13].

Программное средство EveryTag позволяет осуществить внедрение ЦВЗ посредством сдвига строк и слов исходного электронного текстового документа. При этом ЦВЗ содержит уникальную информацию, идентифицирующую пользователя, осуществляющего создание и обработку текстового документа. К недостаткам разработанного программного продукта относится необходимость создания и хранения защищенной базы данных подписанных электронных копий. В случае отсутствия в базе подписанной копии установить факт утечки и идентифицировать нарушителя не представляется возможным.

SafeCopy так же как и EveryTag позволяет маркировать электронные текстовые документы посредством изменения форматирования текста электронного текстового документа. При этом для внедрения ЦВЗ, содержащего идентификатор пользователя и метку времени, необходимо, чтобы страница документа содержала не менее 9 строк текста (40% страницы должно быть текстом). Как и в предыдущем случае требуется создание защищенной базы данных, содержащей подписанные копии электронных текстовых документов.

Требования по наличию подписанной копии электронного текстового документа в процессе идентификации источника утечки конфиденциальных электронных текстовых документов не позволяет использовать рассмотренные программные решения для защиты от утечки, обусловленной преобразованием «печать-сканирование» или «печать-фотографирование», и накладывает дополнительные ограничения на процесс документооборота.

Для обеспечения защищенности электронных текстовых документов необходимо разрабатывать программные решения, способные осуществлять внедрение ЦВЗ в электронный текстовый документ в процессе вывода на печать, который может быть извлечен только из подписанного напечатанного текстового документа или соответствующего изображения без необходимости наличия исходного документа.

При реализации таких алгоритмов возникает необходимость внедрения подсистемы маркирования в операционные системы. В данной статье предлагается подход к реализации маркирования на основе виртуального XPS-принтера для операционных систем семейства Windows.

3. Реализация маркирования в подсистеме печати операционных систем семейства Windows на основе виртуального XPS-принтера

Для маркирования документов в процессе вывода на печать применяются следующие подходы:

- встраивание маркера непосредственно в принтере;
- встраивание маркера в драйвере принтера;
- встраивание маркера в очереди печати (print spooler);
- реализация виртуального принтера с функцией маркирования.

Встраивание маркера непосредственно в принтере реализуется, как правило, аппаратно или в микропрограмме принтера. Эта возможность закладывается производителями принтеров. При этом у пользователя отсутствует или крайне ограничен выбор параметров маркирования.

Встраивание маркера в драйвере принтера во многом аналогично первому подходу, но функция маркирования выполняется на компьютере программным обеспечением драйвера принтера. Такой способ предоставляет пользователю больше возможностей для установки параметров маркирования, однако, изменить алгоритмы маркирования чаще всего невозможно, так как они реализованы производителем в исполняемых файлах драйвера.

Если драйвер принтера разработан в соответствии с архитектурой Microsoft Universal printer driver (Unidrv или V3 Printer Driver), которая поддерживает подключаемые модули и имеет необходимые для этого COM-интерфейсы, то сторонние разработчики могут разработать для такого драйвера подключаемый модуль маркирования. Однако производители принтеров и драйверов к ним чаще всего не предоставляют такую возможность исходя из коммерческих интересов или используют ее только в своих целях.

Первые два подхода могут использоваться в комбинации. Их объединяет одно – отсутствие открытого API для взаимодействия с драйвером и влияния на рендеринг страниц, и как следствие, невозможность реализации собственных алгоритмов маркирования сторонними разработчиками.

Два других подхода базируются на открытом API системы печати Windows, поэтому являются более перспективными с точки зрения реализуемости поставленной задачи.

Встраивание маркера в очереди печати средствами Print Spooler API – это наиболее старый API системы печати, доступный в user mode, реализованный еще в Windows 2000. Он предоставляет доступ к модификации растрового изображения каждой страницы, выводимой на печать. Из недостатков такого подхода можно отметить следующие:

- низкоуровневый API, имеющий некоторые проблемы в безопасности исполняемого кода;
- сложность в конфигурировании параметров маркирования;
- сложность или невозможность в получении доступа к метаданным печати: идентификационным данным компьютера, пользователя, сессии, процесса, из которого запущена печать.

Реализация маркирования в виртуальном принтере является наиболее мощным и гибким подходом в маркировании документов в Windows. Этот подход не имеет недостатков, характерных для Print Spooler API. Он реализован в подсистеме контроля печати в линейке средств защиты от несанкционированного доступа SecretNet [14].

Для реализации виртуального принтера может быть использована одна из следующих архитектур:

- GDI;
- PostScript Printer Driver;
- Microsoft Universal Printer Driver (Unidrv или V3 Printer Driver);
- V4 Printer Driver.

Архитектура Unidrv обладает хорошо проработанным API, позволяющим гибко управлять процессом печати. Существенным является также и то, что драйверы Unidrv работают в пространстве пользователя (User-mode drivers), а не ядра.

V4 Printer Driver предоставляет более современный и более гибкий API, но он поддерживается только с Windows 8.

Для универсального представления документов в Windows целесообразно использовать формат XPS (XML Paper Specification). Это открытый графический формат разметки документа на основе XML. Среди его преимуществ можно отметить следующее [15]:

- разметка документа проще и легче, чем у PDF;
- используется векторная непоследовательная разметка;
- интегрирован в .NET Framework;
- поддерживает многопоточную работу;
- поддерживает шифрование и цифровые сертификаты.

Архитектура XPS-драйвера принтера полностью совместима с Unidrv. При реализации алгоритма маркирования такой драйвер позволяет получить документ в виде универсальной DOM-структуры. Это дает возможность независимо обрабатывать отдельные части документа, включая текст, изображения, шрифты, ресурсы, метаданные задания, документа и отдельных страниц; добавлять в DOM-структуру видимые и невидимые маркеры, а также другие метаданные.

3.1 Маркирование документа в XPS-фильтре

Поддержка XPS-драйверов доступна в Windows XP; при этом требуется установить либо Microsoft XPS Essentials Pack, либо .NET Framework 3.0. Начиная с Windows Vista, компоненты XPS встроены в операционную систему.

Код, реализующий алгоритм встраивания маркера, в данной работе был реализован как XPS-фильтр, встраиваемый в конвейер драйвера виртуального принтера. Общая архитектура драйвера XPS-принтера представлена на рис. 1.

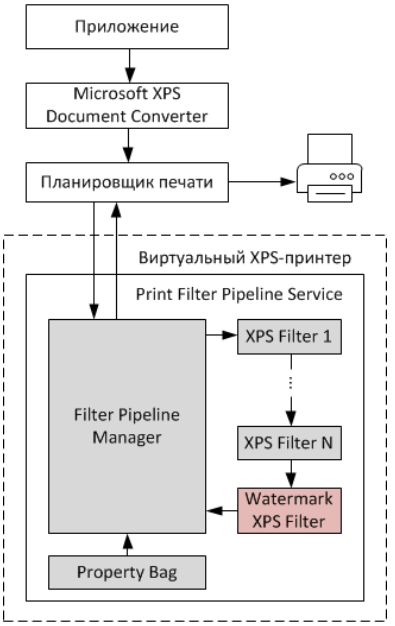


Рис. 1. Общая архитектура драйвера XPS-принтера
Fig. 1. General architecture of the XPS printer driver

Драйвер виртуального XPS-принтера представляет собой конвейер из одного или нескольких XPS-фильтров [16]. Каждый фильтр работает независимо от других. Их взаимодействие между собой осуществляется посредством Filter Pipeline Manager. XPS-фильтр представляет собой COM-объект, что позволяет обеспечить многопоточность, надежность и безопасность всего драйвера.

```
<Filters>
  <Filter dll = "Filter_1.dll"
    clsid = "{5552021F-9D7A-4514-93A6-18FA8EE43601}"
    name = "XpsRenderFilter">
    <Input guid = "{b8cf8530-5562-47c4-ab67-b1f69ecf961e}"
       comment="IID_IXpsDocumentProvider"/>
   <Output guid = "{4368d8a2-4181-4a9f-b295-3d9a38bb9ba0}"
       comment="IID_IXpsDocumentConsumer"/>
 </Filter>
 <Filter dll = "Filter_2.dll"
     clsid = "{C7B17E67-33FF-45DE-8E5B-47A4AFBF370B}"
    name = "XpsRenderFilter">
   <Input guid = "{b8cf8530-5562-47c4-ab67-b1f69ecf961e}"
       comment="IID_IXpsDocumentProvider"/>
  <Output guid = "{4368d8a2-4181-4a9f-b295-3d9a38bb9ba0}"
       comment="IID_IXpsDocumentConsumer"/>
 </Filter>
 <Filter dll = "XpsWatermarkFilter.dll"
     clsid = "{925A12DE-A638-466B-927A-15CBF786C827}"
     name = " XpsWatermarkFilter ">
    <Input guid = "{b8cf8530-5562-47c4-ab67-b1f69ecf961e}"
       comment="IID_IXpsDocumentProvider"/>
    <Output guid = "{4368d8a2-4181-4a9f-b295-3d9a38bb9ba0}"
       comment="IID_IXpsDocumentConsumer"/>
   </Filter>
</Filters>
```

Рис. 2. Пример конфигурационного файла XPS-фильтров
Fig. 2. Example of XPS filters configuration file

Каждый фильтр может обеспечивать определенную функциональность в процессе маркирования документа: нормализацию, выделение областей, маркирование текста,

маркирование изображений, обработку шрифтов и т.д. Обработанный конвейером фильтров документ возвращается в планировщик печати и отправляется в порт физического принтера. Физический принтер, на который выводится готовый документ, указывается в параметрах печати и может быть сконфигурирован в UI-плагине драйвера.

Очередность фильтров в конвейере определяется конфигурационным файлом, пример которого представлен в листинге на рис. 2.

Здесь же определяются стандартизованные COM-интерфейсы для входа и выхода каждого фильтра. Параметры печати передаются фильтрам посредством специального COM-объекта PropertyBag.

Далее будет рассмотрена реализация драйвера с одним XPS-фильтром, осуществляющим маркирование каждой страницы печатаемого документа. Структура фильтра маркирования представлена на рис. 3.

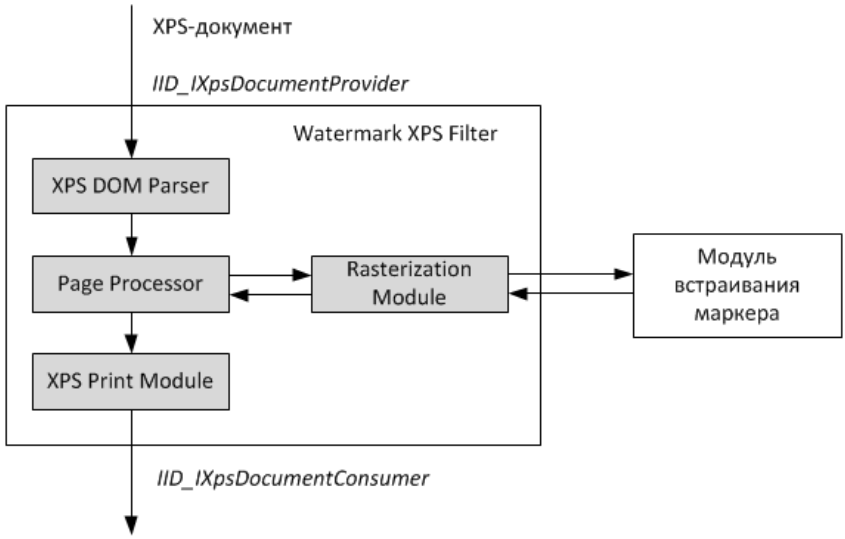


Рис. 3. Структура XPS-фильтра маркирования
Fig. 3. XPS marking filter structure

Сначала XPS DOM Parser разбирает документ или цепочку документов на страницы. Page Processor извлекает страницу, растеризует ее полностью (или часть) и передает модулю встраивания маркера. После встраивания маркера страница уже в виде растра помещается в DOM-модель документа, при этом старая страница заменяется на новую – растеризованную и маркированную.

3.2 Реализация модуля встраивания маркера

Модуль встраивания может быть реализован одним из перечисленных ниже способов.

1) Как многопоточный или однопоточный COM-объект, устанавливаемый вместе с драйвером фильтра XPS. Растровое изображение страницы для обработки передается через разделяемую память процесса виртуального принтера.
2) Как многопоточную или однопоточную динамическую библиотеку с обычным stdcall-интерфейсом. Растровое изображение страницы для обработки передается также через разделяемую память.

Особенностью двух этих способов является то, что исполняемый код модуля работает в изолированной среде процесса печати, что продиктовано соображениями безопасности [17]. Опытным путем было установлено, что в этой изолированной среде запрещен файловый ввод-вывод, а также запрещено создание дочерних процессов. Это накладывает

определенные ограничения на возможность использования внешних программных компонентов и интерпретируемых языков, таких как Python.

3) Как отдельный процесс. Этот способ предоставляет больше возможностей для реализации маркирования. В частности, он может быть реализован как набор скриптов на языке Python и работать в отдельном процессе интерпретатора.

Вместе с тем, при реализации такого способа маркирования возникают две проблемы, обусловленные ограничениями контекста процесса печати:

- невозможность из XPS-фильтра запустить внешний процесс;
- необходимость передачи растеризованной страницы модулю маркирования и приема обратно маркированного изображения.

Внешний процесс можно запустить автоматически в пространстве пользователя в фоновом режиме. Передать данные из драйвера во внешний процесс через файл невозможно. Для этих целей могут быть использованы следующие механизмы межпроцессного взаимодействия:

- именованные каналы;
- проецирование в память области из системного файла подкачки;
- сокеты.

Для синхронизации с внешним процессом драйвер может использовать именованный канал или объекты ядра для межпроцессной синхронизации.

4) Как TCP-сервер. Этот способ является разновидностью предыдущего, если для взаимодействия с внешним процессом использовать сокет. Преимуществом данного способа является то, что сервер маркирования может быть размещен на отдельном сетевом узле, что позволит легко масштабировать систему маркирования и оптимизировать вычислительные ресурсы сети организации.

Авторами был реализован четвертый способ. При этом модуль маркирования был реализован как скомпилированный бинарный Python-скрипт, запускаемый HTTP-сервером, написанным на C#. Структура модуля маркирования представлена на рис. 4.

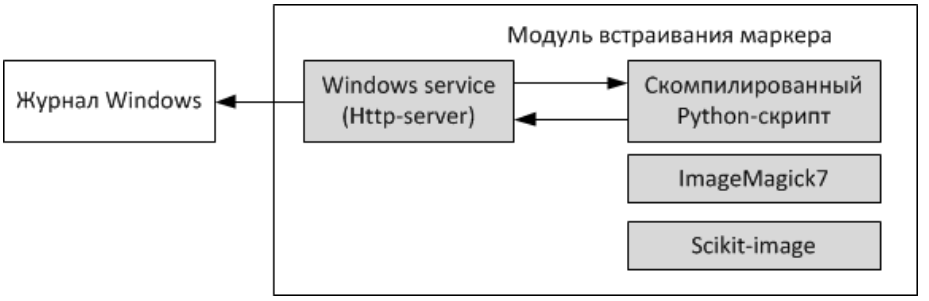


Рис. 4. Структура модуля маркирования
Fig. 4. Marking module structure

HTTP-сервер запускает отдельный процесс маркирования в синхронном режиме. Обмен данными осуществляется посредством временных файлов, создаваемых в кэше файловой системы.

3.3 Обработка задания на печать

В этом подразделе детально описан процесс обработки документа, поступающего на печать в XPS-фильтр.

Данные для печати передаются в фильтр системой печати через интерфейс IID_IXpsDocumentProvider (рис. 3). DOM-структура данных для печати включает в себя четыре вложенных уровня (рис. 5) [15]:

- задание для печати (XpsDocument);
- последовательность документов (FixedDocumentSequence);
- документ (FixedDocument);
- страница (FixedPage).

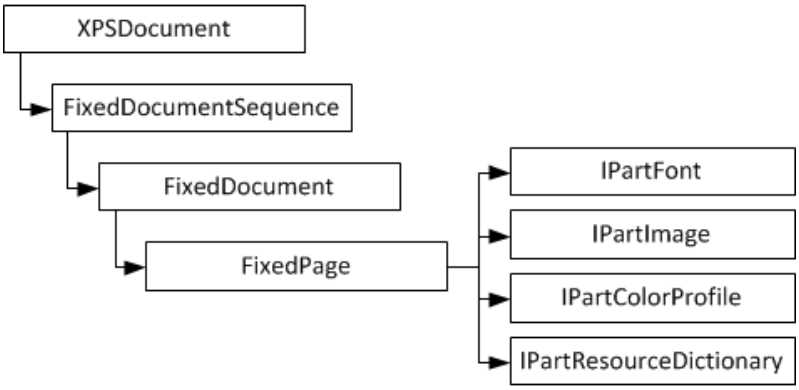


Рис. 5. DOM-структура данных для печати
Fig. 5. DOM data structure for printing

Фильтр запрашивает последовательно объект каждого уровня DOM-структуры, начиная с верхнего уровня. Таким образом, фильтр выполняет обход всей структуры данных в глубину.

Для объектов XpsDocument, FixedDocumentSequence, FixedDocument фильтр обрабатывает только метаданные и передает их на сервер маркирования в виде JSON-пакетов. Объекты FixedPage обрабатываются отдельно модулем PageProcessor (рис. 3).

Обработка FixedPage заключается в выполнении следующих действий:

- извлечение структуры страницы, включая текст, ссылки на ресурсы и данные верстки;
- извлечение внедренных объектов: шрифтов, изображений, цветовых профилей, словарей;
- упаковка извлеченных данных в пакет и передача его на HTTP-сервер;
- извлечение данных из пакета, растеризация и маркирование изображения страницы внешним модулем на HTTP-сервере;
- вывод маркированной страницы на физический принтер.

Опытным путем было установлено, что при печати в XPS-принтер из некоторых приложений (например, Microsoft Word), подсистема Microsoft XPS Document Converter внедряет в XPS-документ шрифты ODTTF в обфусцированном виде с целью защиты правообладателя. Однако, алгоритм обфускации шрифтов был раскрыт исследователями [18]. Для рассматриваемого модуля маркирования был разработан алгоритм деобфускации шрифтов и реализован в HTTP-сервере.

Сокращенный код деобфускации шрифтов ODTTF приведен на рис. 6.

```
int[] guidMapping = new int[] {15,14,13,12,11,10,9,8,6,7,4,5,0,1,2,3};
for (int i = 0; i < 16; i++) {
    buf[i] = (byte)(buf[i] ^ guid[guidMapping[i]]);
    buf[i + 16] = (byte)(buf[i + 16] ^ guid[guidMapping[i]]);
}
```

Рис. 6. Код деобфускации шрифтов ODTTF
Fig. 6. ODTTF font deobfuscation code

3.4 Многопоточная реализация сервера маркирования

В процессе маркирования документов наиболее ресурсоемкой операцией является внедрение маркера в растеризованную страницу. Поэтому печать документов, содержащих множество страниц большого формата (A3 и более), может занимать продолжительное время, зависящее от сложности алгоритма маркирования. Поскольку маркирование осуществляется в отдельном фоновом процессе Windows, это может потенциально порождать коллизии при одновременной печати нескольких документов одним или разными пользователями через один и тот же виртуальный XPS-принтер. Чем больше объем печатаемых документов, тем выше вероятность коллизии.

Для предотвращения коллизий и улучшения производительности системы маркирования предложена модифицированная схема с многопоточным сервером (рис. 7).

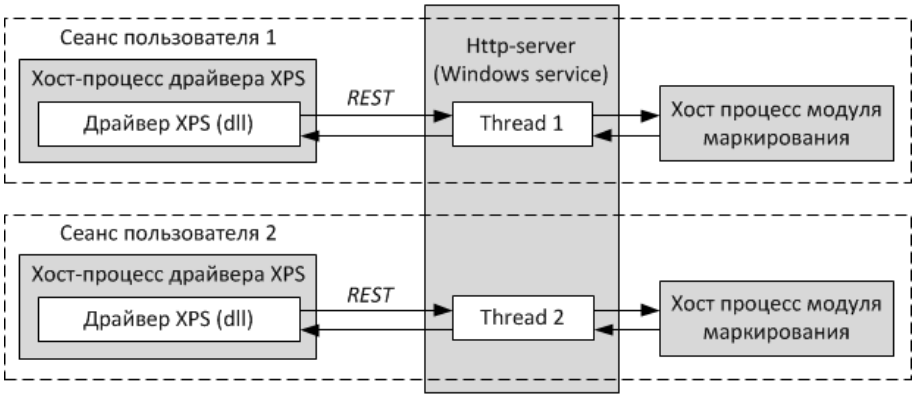


Рис. 7. Схема маркирования с многопоточным сервером
Fig. 7. Marking scheme with multithreaded server

Поскольку для каждого задания, отправляемого пользователем на печать, запускается код драйвера в отдельном хост-процессе, то для маркирования страниц HTTP-сервер создает отдельный поток для каждого подключения драйвера.

Маркирование осуществляется внешним процессом, каждый экземпляр которого запускается из потока сервера. Протокол обмена данными с модулем маркирования простой и позволяет синхронизироваться только с дескриптором процесса.

Протокол взаимодействия драйвера с сервером требует, чтобы состояние сессии сохранялось между HTTP-запросами. Это нужно, в частности, для реализации процедуры отмены печати или для отслеживания прогресса печати. Сохранение состояния сессии может быть реализовано двумя способами [19, 20]:

- с помощью менеджера сессий;
- на основе механизма REST.

Менеджер сессий усложняет архитектуру сервера, поскольку он не только должен хранить состояние каждой сессии, но и отслеживать ее время жизни, находить и удалять "мертвые" сессии, осуществлять «сборку мусора».

Механизм REST (Representation State Transfer) позволяет сохранять состояние сессии в GET или POST запросах, тем самым упрощая архитектуру сервера. Такой способ был реализован в системе, разработанной авторами данной работы.

Предлагаемая многопоточная схема обладает высокой надежностью, поскольку все сеансы печати изолированы друг от друга, и сбой при маркировании или печати в одном сеансе не повлияет на все остальные

3.5 Обработка метаданных XPS-документа

Для маркирования и последующей печати маркированных документов необходимы метаданные, которые должны передаваться драйверу XPS-принтера помимо DOM-структуры документа:

- для растеризации необходимо разрешение страниц (в DPI);
- для маркирования необходимы данные для генерации маркера, позволяющего идентифицировать пользователя, который запустил печать;
- для печати необходимы данные о физическом принтере, размере и ориентации страницы, цветовой схеме, разрешении и другие параметры.

Все необходимы данные могут быть получены драйвером через стандартные COM-интерфейсы из структуры PropertyBag (рис. 1).

Идентификационные данные для генерации маркера могут быть получены из системного токена пользователя, запустившего печать. Дескриптор токена, в свою очередь, передается драйверу через PropertyBag. Упрощенный код извлечения идентификационных данных представлен ниже (рис. 8):

```
VARIANT varUserSecurityToken;
VariantInit(&varUserSecurityToken);
m_pIPropertyBag->GetProperty(XPS_FP_USER_TOKEN,
  &varUserSecurityToken));

HANDLE user_handle = varUserSecurityToken.byref;
DWORD tuLen = 0;
GetTokenInformation(user_handle, TokenUser, NULL, 0, &tuLen);
PTOKEN_USER tu = (PTOKEN_USER)LocalAlloc(LPTR, tuLen);
GetTokenInformation(user_handle, TokenUser, tu, tuLen, &tuLen);

const size_t max_len = 256;
wchar_t nameUser[max_len] = { 0 };
wchar_t domainName[max_len] = { 0 };
DWORD nameUserLen = 256;
DWORD domainNameLen = 256;
SID_NAME_USE snu;
LookupAccountSidW(NULL, tu->User.Sid, nameUser, &nameUserLen,
  domainName, &domainNameLen, &snu);
```

Рис. 8. Код извлечения идентификационных данных для генерации маркера
Fig. 8. Identification data extraction code for mark generation

Параметры, используемые для растеризации и печати, извлекаются из структуры PrintTicket, доступ к которой может быть получен через PropertyBag, аналогично дескриптору токена пользователя.

DOM-структура PrintTicket соответствует спецификации Print Schema Specification [21]. В соответствие с ней параметры печати задаются на трех уровнях:

- задание (Job);
- документ (Document);
- страница (Page).

При обработке параметров печати драйвер выполняет слияние объектов PrintTicket с разных уровней по схеме JobPrintTicket -> DocumentPrintTicket -> PagePrintTicket.

Слияние осуществляется в соответствии с принципом наследования. Т.е. все параметры, установленные для задания, наследуются или переопределяются на уровне документа и страницы. Для слияния параметров используется системная функция PTMergeAndValidatePrintTicket.

Различные приложения, из которых осуществляется печать, могут по-разному формировать объекты PrintTicket для всех трех уровней. Так, например, Microsoft Word формирует параметры печати для всех трех уровней, а Notepad формирует параметры печати только для задания (PrintTicket для страницы не содержит данных).

Для управления параметрами печати в XPS-драйвере целесообразно использовать интерфейсы и компоненты Unidrv. В их составе имеется набор типовых параметров печати, таких как стандартные размеры бумаги, ориентация страниц, разрешение, качество печати и т.д. Кроме того, Unidrv предоставляет типовой пользовательский интерфейс для задания пользователем параметров печати.

Для настройки параметров печати, не предусмотренных Unidrv, DOM-структура PrintTicket может быть расширена, так как это предусмотрено спецификацией Print Schema Specification. Unidrv предоставляет COM-интерфейсы для расширения и модификации страниц свойств интерфейса настройки параметров печати драйвера XPS.

Для рассматриваемой системы маркирования было разработано расширение стандартной схемы параметров печати (листинг, рис. 9.) и выполнена модификация интерфейса настройки параметров Unidrv путем добавления новой страницы свойств.

```
<psf:Feature name="psk:PageWatermarking">
    <psf:Option>
        <psf:ScoredProperty name="psk:PageRedirectPrinterName">
        <psf:ParameterRef
           name="psk:PageWatermarkingPageRedirectPrinterName"/>
        </psf:ScoredProperty>
    </psf:Option>
</psf:Feature>
<psf:ParameterInit
  name="psk:PageWatermarkingPageRedirectPrinterName">
    <psf:Value xsi:type = "xsd:string">
        hp LaserJet 1010 HB
    </psf:Value>
</psf:ParameterInit>
```

Рис.9. Фрагмент расширенной схемы параметров печати
Fig. 9. Fragment of the extended print settings scheme

После того, как параметры печати попадут в драйвер, они преобразуются в бинарную упакованную структуру DEVMOD для последующей обработки конвейером печати. Это осуществляется с помощью системной функции PTConvertPrintTicketToDevMode.

4. Требования к процессу маркирования

Ключевыми требованиями к процессу маркирования являются:

- возможность извлечения параметров встраиваемого маркера и встроенной информации при отсутствии исходного документа;
- обеспечение выполнения требований по робастности относительно заданных искажений;
- обеспечение должной емкости встраивания для отдельных документов (изображений документов);
- обеспечение заданного уровня достоверности извлечения встроенной информации (с учетом возможного применения помехоустойчивого кодирования при встраивании информации);
- инвариантность изменяемых в процессе маркирования характеристик документа к виду его представления или преобразования на основе которого был получен его электронный эквивалент.

Далее будут рассмотрены основные задачи решаемые в процессе внедрения и извлечения маркера и требования к ним. Изменяемые характеристики документа при этом могут быть различными, но должны выполняться указанные выше требования.

4.1 Формирование и внедрение маркера

Одним из возможных вариантов внедрения маркера является кодирование информации за счет изменения яркости фона отдельных областей маркируемого документа при встраивании битов маркера. Формирование маркера при этом состоит из следующих этапов:

- кодирование встраиваемой информации, содержащей сведения об авторе, дате и времени

формирования документа в двоичный вид;

- преобразование полученной информации в последовательность встраивания по следующему правилу: символу "1" соответствует наличию изменяемой яркости фона определенной области документа, символу "0" – отсутствие изменений в яркости фона документа.

Полученная последовательность встраивания внедряется в электронный документ в процессе печати посредством разработанного подхода, основанного на технологии виртуального XPS-принтера.

Указанный подход к формированию маркера позволяет осуществлять маркирование не только электронных текстовых документов, но электронных таблиц и электронных документов, содержащих графические объекты, диаграммы и изображения, а также электронных документов, подготовленных для демонстрации и визуального восприятия.

В результате маркирования формируется печатный документ, содержащий информацию, однозначно идентифицирующую владельца данных. В процессе жизненного цикла распечатанный документ может быть подвергнут преобразованию формата в электронный вид посредством операции сканирования или фотографирования. Для извлечения встроенного маркера из сформированного изображения предложен подход к извлечению встроенного маркера, основанного на изменении яркости фона документа.

4.2 Извлечение встроенного маркера

В процессе преобразования формата напечатанного документа в изображение могут быть внесены следующие искажения:

- поворот изображения;
- масштабирование изображения;
- искажение перспективы изображения (горизонтальный наклон и вертикальное отклонение);
- наличие областей изображения, содержащих неравномерный фон.

Для устранения указанных искажений, обусловленных процессом фотографирования (сканирования) предлагается подход к извлечению маркера из изображений, состоящий из следующих этапов:

- коррекция перспективы (поворота) изображения;
- определение контуров (границ) документа;
- определение областей с измененными характеристиками;
- декодирование значений яркости в двоичный вид;
- идентификация автора документа и его атрибутов.

В ходе экспериментальной оценки предложенного подхода к маркированию, основанного на изменении областей фона изображения посредством реализации виртуального XPS-принтера в операционных системах семейства Windows, была осуществлена серия экспериментов, направленных на определение возможности извлечения встроенной информации из изображений электронного документа.

Полученные значения точности извлечения встроенного маркера при осуществлении преобразования "печать-сканирование" или "печать-фотографирование" (порядка 0,95 для отдельных документов и маркера размером 16 бит) позволили сделать вывод о наличии потенциальной возможности реализации системы маркирования на основе виртуального XPS-принтера для защиты авторских прав владельцев электронных документов. Однако реализация данной системы требует проведения существенно большего количества

экспериментальных исследований для обоснования выбираемых параметров и требований к процессу маркирования, что является направлением дальнейших исследований.

5. Выводы и направление дальнейших исследований

Предложенный авторами подход позволяет реализовать надежное внедрение маркера при печати документов в операционных системах семейства Windows через виртуальный XPS-принтер, установленный в качестве принтера по умолчанию для всех пользователей, а также, в случае наличия соответствующих алгоритмов маркирования, преодолеть недостатки существующих средств, требующих наличия оригинального документа для проведения расследования инцидента информационной безопасности.

Дальнейшего исследования требуют следующие вопросы:

- выбор и обоснование характеристик электронного документа, используемых для внедрения маркера с учетом предъявляемых требований;
- оптимизация протокола взаимодействия драйвера и HTTP-сервера;
- оценка ресурсозатрат по времени и производительности в зависимости от используемого типа электронно-вычислительных машин и средств вычислительной техники;
- улучшение производительности модуля встраивания маркера путем использования специализированных процессоров или ПЛИС;
- разработка подхода к внедрению маркера в подсистеме печати операционных систем семейства UNIX;
- разработка сетевой архитектуры системы маркирования, вынесение наиболее ресурсоемких операций на отдельную вычислительную платформу.

Список литературы / References

[1]. Глобальное исследование утечек конфиденциальной информации в первом полугодии 2019 года. InfoWatch. 2019, 30 стр. / Global Confidential Information Leak Survey in the first half of 2019. InfoWatch. 2019, 30 p. (in Russian).

[2]. Исследование утечек информации ограниченного доступа в госсекторе. Мир – Россия. InfoWatch. 2019, 24 стр. / Research of information leaks of limited access in the public sector. World – Russia. InfoWatch. 2019, 24 p. (in Russian).

[3]. M. Jain, S. K. Lenka. A Review on Data Leakage Prevention using Image Steganography / // International Journal of Computer Science Engineering, vol. 5, issue 2, 2016, pp. 56-59.

[4]. Lopez G., Richardson N., Carvajal J. Methodology for Data Loss Prevention Technology Evaluation for Protecting Sensitive Information. Revista Politécnica, vol. 36, issue. 3, 2015, pp. 1-69.

[5]. Alneyadi S., Sithirasenan E., Muthukkumarasamy V. A survey on data leakage prevention systems. Journal of Network and Computer Applications, vol. 62, 2016, pp. 137-152.

[6]. Jadhav P., Chawan P. M. Data Leak Prevention system: A Survey. International Research Journal of Engineering and Technology, vol. 6, issue 10, 2019, pp. 197-199.

[7]. Milano D. Content control: Digital watermarking and fingerprinting. White Paper. Rhozet, a business unit of Harmonic Inc., 2012, 11 p.

[8]. Data loss prevention in Exchange Server. Available at: https://docs.microsoft.com/en-us/Exchange/policy-and-compliance/data-loss-prevention/data-loss-prevention?redirectedfrom=MSDN&view=exchserver-2019, accessed 14.09.2020.

[9]. Graham R. How The Intercept Outed Reality Winner. Available at: https://blog.erratasec.com/2017/06/how-intercept-outedreality-winner.html, accessed 14.09.2020.

[10]. Trace Doc. Available at: https://secretgroup.ru/trace-doc, accessed 14.09.2020.

[11]. Kozachok A. V., Kopylov S. A., Shelupanov A. A., Evsutin O. O. Text marking approach for data leakage prevention. Journal of Computer Virology and Hacking Techniques. 2019, vol. 15, issue. 3, pp. 219-232. DOI: 10.1007/s11416-019-00336-9.

[12]. Unique Interface. EveryTag. Available at: https://everytag.ru/ui, accessed 14.09.2020.

[13]. Safe Copy. Available at: https://www.niisokb.ru/products/safecopy, accessed 14.09.2020.

[14]. Secret Net Studio. Available at: https://www.securitycode.ru/products/secret-net-studio/, accessed 14.09.2020.
[15]. Open XML Paper Specification (OpenXPS). Standard ECMA-388. 2009, 496 p.
[16]. XPSDrv Render Module. Available at: https://docs.microsoft.com/ru-ru/windows-hardware/drivers/print/xpsdrv-render-module, accessed 14.09.2020.
[17]. Understanding Printer Driver Isolation. Available at: https://sourcedaddy.com/windows-7/understanding-printer-driver-isolation.html, accessed 14.09.2020.
[18]. Celil U., Bekir K. Breaking Font Parsers. Available at: http://www.powerofcommunity.net/poc2015/celil.pdf, accessed 14.09.2020.
[19]. Архитектура REST / REST architecture. Available at: https://habr.com/ru/post/38730, accessed 14.09.2020 (in Russian).
[20]. Введение в REST API – RESTful веб-сервисы / Introduction to REST API – RESTful web services. Available at: https://habr.com/ru/post/483202, accessed 14.09.2020 (in Russian).
[21]. Print Schema. Available at: https://docs.microsoft.com/ru-ru/windows/win32/printdocs/printschema, accessed 14.09.2020.

Информация об авторах / Information about authors

Сергей Викторович КОЗЛОВ – кандидат технических наук. Сфера научных интересов: информационная безопасность, защита от несанкционированного доступа, особенности построения и функционирования операционных систем, средства и методы программирования.

Sergey Viktorovich KOZLOV – Candidate of Technical Sciences. Research interests: information security, protection from unauthorized access, construction and functioning features of operating systems, programming tools and methods.

Сергей Александрович КОПЫЛОВ, научные интересы включают методы машинного обучения, обработка цифровых изображений, текстовая стеганография.

Sergey Alexandrovich KOPYLOV, research interests include machine learning methods, digital image processing, text steganography.

Борис Владимирович КОНДРАТЬЕВ; сфера научных интересов: безопасность информации, защита информации от несанкционированного доступа и утечки по техническим каналам, построение информационных систем в защищённом исполнении, сертификация программного обеспечения по требованиям безопасности информации.

Boris Vladimirovich KONDRAT’EV, research interests: information security, protection of information from unauthorized access and leakage through technical channels, building information systems in a secure design, certification of software for information security requirements.

Дмитрий Олегович ОБЫДЕНКОВ – аспирант. Его научные интересы включают методы сокрытия и защищённой передачи информации, компьютерные сети, технологии анализа сетевого трафика.

Dmitry Olegovich OBYDENKOV is a graduate student. His scientific interests include methods for information hiding and secure transmission, computer networks, technologies of network traffic analysis.

DOI: 10.15514/ISPRAS–2020–32(5)–8

Application of HDBSCAN Method for Clustering scRNA-seq Data

[1,2] *M.A. Akimenkova, ORCID: 0000-0002-9064-5253 <m.akimenkova@ispras.ru>*
[2] *A. A. Maznina, ORCID: 0000-0002-5780-1330 <aamaznina@gmail.com>*
[1] *A. Y. Naumov, ORCID: 0000-0003-4851-7677 <vandedok@ispras.ru>*
[1,2] *E.A. Karpulevich, ORCID: 0000-0002-6771-2163 <karpulevich@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, 9, Institutskiy per., Dolgoprudniy, 141701, Russia*

Abstract. One of the main tasks in the analysis of single cell RNA sequencing (scRNA-seq) data is the identification of cell types and subtypes, which is usually based on some method of clustering. There is a number of generally accepted approaches to solving the clustering problem, one of which is implemented in the Seurat package. In addition, the quality of clustering is influenced by the use of preprocessing algorithms, such as imputation, dimensionality reduction, feature selection, etc. In the article, the HDBSCAN hierarchical clustering method is used to cluster scRNA-seq data. For a more complete comparison Experiments and comparisons were made on two labeled datasets: Zeisel (3005 cells) and Romanov (2881 cells). To compare the quality of clustering, two external metrics were used: Adjusted Rand index and V-measure. The experiments demonstrated a higher quality of clustering by the HDBSCAN method on the Zeisel dataset and a poorer quality on the Romanov dataset.

Keywords: HDBSCAN; scRNA-seq clustering; denoising autoencoder

For citation: Akimenkova M.A., Maznina A.A., Naumov A.Y., Karpulevich E.A. Application of HDBSCAN method for clustering scRNA-seq data. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 111-120. DOI: 10.15514/ISPRAS-2020-32(5)-8

Применение метода HDBSCAN для кластеризации данных scRNA-seq

[1,2] *М.А. Акименкова, ORCID: 0000-0002-9064-5253 <m.akimenkova@ispras.ru>*
[2] *А.А. Мазнина, ORCID: 0000-0002-5780-1330 <aamaznina@gmail.com>*
[1] *А.Ю. Наумов, ORCID: 0000-0003-4851-7677 <vandedok@ispras.ru>*
[1,2] *Е.А. Карпулевич, ORCID: 0000-0002-6771-2163 <karpulevich@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, Россия, 141701, г. Долгопрудный, Институтский пер., д. 9*

Аннотация. Одной из основных задач при анализе данных РНК-секвенирования единичных клеток является идентификация типов и подтипов клеток, которая обычно основана на каком-либо методе кластеризации. Существует ряд общепринятых подходов к решению проблемы кластеризации, один из которых реализован в пакете Seurat. На качество кластеризации, помимо прочего, влияет использование алгоритмов предварительной обработки, таких как импутация, уменьшение размерности, отбор признаков и т. д. В статье для кластеризации данных scRNA-seq используется метод иерархической

кластеризации HDBSCAN. Для более полного сравнения эксперименты и сравнения проводились на двух размеченных наборах данных: Zeisel (3005 клеток) и Romanov (2881 клетка). Для сравнения качества кластеризации использовались две внешние метрики: скорректированный индекс Рэнда и V-мера. Эксперименты продемонстрировали более высокое качество кластеризации методом HDBSCAN на наборе данных Zeisel и более низкое качество на наборе данных Romanov.

Ключевые слова: hdbscan; кластеризация данных РНК-секвенирования единичных клеток; шумоподавляющий автокодировщик

Для цитирования: Акименкова М.А., Мазнина А.А., Наумов А.Ю., Карпулевич Е.А. Применение метода HDBSCAN для кластеризации данных РНК-секвенирования единичных клеток. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 111-120 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(5)–8

1. Introduction

The cell can be considered the fundamental unit in biology. For centuries, biologists have known that multicellular organisms are characterized by many different types of cells. Cells can be distinguished by their size and shape with a microscope, and attributes based on their appearance have traditionally been the main factor in determining cell type. Advances in microfluidics have made it possible to isolate large numbers of cells, and, along with improvements in methods for isolating and amplifying RNA, it is now possible to profile the transcript of single cells using next generation sequencing technologies.

For researchers to make full use of these rich datasets, efficient computational techniques are needed. Numerous steps are neaded before clustering, such as imputation, feature selection, dimensionality reduction. Moreover, there are also software packages that implement the entire clustering workflow, such as Seurat [1]. In this article, we want to compare the use of the popular HDBSCAN [2] algorithm with the steps leading up to clustering, with a Seurat tool.

2. Methods

Many clustering algorithms can be applied to any type of data that is supplied with a measure of the distance between data points. Due to a large number of genes analysed in scRNA-seq, namely the high dimensionality, the distances between data points (i.e., cells) become similar, which is known as the «curse of dimensionality». Hence, distance differences tend to be small and therefore unreliable for identifying clustering. Applying feature selection and / or dimensionality reduction can reduce noise and speed up computations. Feature selection involves identifying the most informative genes, such as genes with the greatest variance, while decreasing dimensionality projects data into a lower-dimensional space. Many tools use variations of standard methods such as PCA [3], uMap [4], DCA [5].

Usually pipelines for scRNA-seq analysis contain tools for imputation, feature selection, dimensionality reduction, etc. This article compares the 14 pipelines shown in the Table. 1.

Table 1. Pipelines

pipeline	**imputation**	**feature selection**	**dimensionality reduction**	**clustering method**
Seurat	no	yes	PCA	Louvain
Seurat*	yes	yes	PCA	Louvain
3	yes	no	DCA	HDBSCAN
4	yes	yes	DCA	HDBSCAN
5	yes	no	uMAP	HDBSCAN

6	yes	yes	uMAP	HDBSCAN
7	yes	no	PCA	HDBSCAN
8	yes	yes	PCA	HDBSCAN
9	no	no	DCA	HDBSCAN
10	no	yes	DCA	HDBSCAN
11	no	no	uMAP	HDBSCAN
12	no	yes	uMAP	HDBSCAN
13	no	no	PCA	HDBSCAN
14	no	yes	PCA	HDBSCAN

2.1 Dimensionality reduction methods

ScRNA-seq data are always large, which increases the complexity of the analysis to some extent. Therefore, to process the initial data we used dimensionality reduction methods.

1) Principal Component Analysis

The most common dimensionality reduction technique is principal component analysis (PCA) [3], which requires no control and aims to find a lower-dimensional representation of the data.

PCA is a widely used method of uncontrolled dimensionality reduction. PCA assumes the data is normally distributed, diagonalizes the covariance matrix of the original matrix, and the resulting covariance matrix is a set of new variables for the diagonal matrix. Orthogonal transformation is used to transform a set of potential linear correlation variables into linear explanatory variables, which means that linear dimensionality reduction is realized. One of the main problems with linear dimensionality reduction algorithms is that, when they concentrate disparate data points in a lower-dimensional area, the data points are far apart.

2) Deep Count Autoencoder

The deep counting autoencoder network (DCA) [5] denoises scRNA-seq datasets. DCA accounts for the computed distribution, excess variance, and data sparsity using a negative binomial noise model with or without zero inflation, while capturing nonlinear gene-gene relationships.

One of the main advantages of DCA is that the user only needs to specify the noise model. To provide maximum flexibility, DCA implements a set of scRNA-seq-specific noise models, including negative binomial distribution with (ZINB) and no zero inflation (NB).

For example, using the ZINB noise model, DCA examines the meaning of gene-specific parameters, variance, and dropout probability based on gene expression inputs. The derived average distribution parameter is the denoised reconstruction and DCA output.

In our work, we used a 32-dimensional inner layer.

3) uMAP

Uniform Manifold Approximation and Projection (uMAP) [4] is a graph-based dimension reduction method similar to t-SNE, introduced by McInnes et al. in 2018. The algorithm builds a high-dimensional graph representation and then optimizes the low-dimensional graph so that it looks structurally as similar as possible to the original.

The advantages of the algorithm include computational efficiency (compared to t-SNE), preservation of the global structure (also compared to t-SNE). In addition, uMAP has no

restrictions on the size of the embedded layer, which allows the use of the algorithm for preprocessing to improve the performance of clustering algorithms.

The disadvantages of the uMAP algorithm are lack of interpretability and false detection of noise, uMAP tends to find a diverse structure in the noise of a dataset. uMAP is more reliable with larger datasets as the amount of structure evident to noise tends to decrease in larger datasets.

2.2 Clustering methods

Density-based methods work well even when the data is noisy and the clusters are oddly shaped.

These methods are not generally used for single cell clustering, but they have their advantages.

Both algorithms have the minimum number of samples parameter, which is the neighbor threshold for a record to become a core point. Both algorithms start by finding the core distance of each point, which is the distance between that point and its farthest neighbor, defined by the minimum samples parameter.

DBSCAN [6] is a density-based clustering algorithm – given a set of points in space, the algorithm groups together points that are closely spaced (points with many close neighbors), with lone points in low-density areas marked as outliers (farthest neighbour).

DBSCAN has the epsilon parameter, which is the radius that those neighbors have to be in for the core to form. This algorithm is well suited for clustering single cell data as it copes well with noisy data.

It also finds clusters of exotic shapes: nested and anomalous clusters, as well as low dimension folds. Additionally, there is no need to specify the number of clusters.

HDBSCAN [2] uses a density-based approach, which makes few implicit assumptions about the clusters. It is a non-parametric method that looks for a cluster hierarchy shaped by the multivariate modes of the underlying distribution. Rather than looking for clusters with a particular shape, it looks for regions of the data that are denser than the surrounding space. In addition to being better for data with varying density, it is also faster than regular DBSCAN. HDBSCAN has a minimum cluster size parameter, which defines how big a cluster needs to be in order to form.

2.3 Available workflows

The steps leading up to clustering can have a significant impact on the outcome, and numerous tools are available for each step. There are software packages that implement the entire clustering workflow, such as Seurat [1].

Satija et al. created Seurat, a single cell data analysis toolkit. The expression matrix includes the number of genes, the number of cells and the number of genes in each cell, as well as the number of cells in which each gene is expressed.

Seurat uses Louvain's graph-based algorithm [7]. The advantage is that most graph-based methods do not require the user to specify the number of clusters for identification, instead, indirect resolution parameters are used. The combination of common nearest neighbor graphs and Louvain community detection was first applied to scRNA-seq data in the PhenoGraph method, and this approach has since been incorporated into Seurat. For dimensionality reduction, PCA is used.

Because of their speed and scalability, the clustering techniques included in Seurat packages are a popular choice for large datasets. However, Louvain clustering has proved to be ineffective for small datasets.

3. Feature selection

Feature selection a collection of statistical approaches that identify and retain only variables that are most relevant to the underlying structure of the data set.

Due to the large number of genes analysed in scRNA-seq, that is, the high dimensionality, the distances between data points (i.e., cells) become similar, which is known as the «curse of dimensionality». Hence, distance differences tend to be small and therefore unreliable for identifying cell groups. Using feature selection can reduce noise and speed up calculations. Feature selection includes identifying the most informative genes, for example, with the highest variance [8].

The expression data of one cell contains a set of missing values and noise data that affects the next step in the analysis. Feature selection with variance has been used to alleviate these issues. Inspired by Prabhakaran et al. [9], we selected groups of genes with the greatest variance in expression. For the Zeisel [10] and Romanov [11], the initial sizes were 19,972 and 24,341 respectively. We took the feature selection data to select genes with high variance. Variance represents the degree of differentiation of gene expression across all cells, and high variance indicates that the gene was important for distinguishing cells. Therefore, we could easily get more biologically significant clusters. Using feature selection, a subset with top 200 genes was generated for Zeisel [9] and Romanov [11] data. We performed the following experiments with three clustering models. We compared all three clustering algorithms (ie HDBSCAN+PCA, HDBSCAN+uMAP) on the subset with the original data (19,972 and 24,341 genes without traits). Selection of the top 200 genes for each of the three algorithms enhanced clustering quality as opposed to the use of the full set of genes (19,972 and 24,341 genes).

In addition, HDBSCAN+DCA algorithm performed best among these clustering algorithms, reaching an accuracy of 0.95 on 200 gene sets. The accuracy was 9.3% higher than the result without gene selection. Meanwhile, using the gene selection method, HDBSCAN+uMAP, HDBSCAN+PCA, it was possible to increase the accuracy by 11.8% and 21.9%, respectively. These results showed that clustering with gene selection gives better performance than methods without it.

4. Imputation

The scRNA-seq data is characterized by excess zero counts, the so-called dropouts due to the low number of mRNAs sequenced within individual cells. In order to reduce the number of dropouts in some experiments, the scRNA-seq scImpute [12] method is used. ScImpute is a statistical method for imputing dropouts. ScImpute automatically detects likely dropouts and imputes only those values without changing the rest of the data. The scImpute algorithm also detects outliers in scRNA-seq data and excludes them from imputation. The effectiveness of scImpute has been demonstrated on both simulated and real human and mouse scRNA-seq data. ScImpute detects and imputes dropouts, thereby enhancing the analysis of differential expression and clustering of cell subpopulations.

In the article scImpute is used to improve the quality of clustering in combination with feature selection and dimensionality reduction methods (PCA [3], DCA [5], uMAP [4]).

5. Evaluation

To evaluate the performance of an array of popular clustering methods, we tested Seurat [1] and HDBSCAN [2] with different methods of dimensionality reduction such as DCA [5], PCA [3], uMAP [4] on 2 published datasets.

Since we have published cell type labels, for the assessment of clustering quality, we used two external quality metrics – Adjusted Rand Index [13] and V-measure [14].

The Adjusted Rand index was chosen as the first metric of clustering quality. This metric is external, i.e. the measures are based on comparing the clustering result with the a priori known division into classes. This metric is robust to the size and number of clusters.

V-measure was used as the second quality metric. The main advantage of this metric is that it is independent of the number of class labels, the number of clusters, the size of the data, and the clustering algorithm used, and is very reliable.

6. Experiments

The operation of the selected algorithms is demonstrated on two scRNA-seq gene expression datasets for house mouse cells. The Zeisel [10] contains scRNA-seq-derived cortical cell expression data from the house mouse and describes 3005 different cells of 9 different types. The data are also presented as a 19,772 x 3005 gene expression matrix. The Romanov [11] contains expression data of hypothalamic cells from the cortex of the house mouse, obtained by scRNA-seq, and describes 2881 different cells of 7 different types. The data are also presented as a 24,341 x 2881 gene expression matrix.

For a sufficient set of statistics, the Zeisel set was divided into 8 non-overlapping sets with a balanced number of cells in each cluster: 7 sets of 19,772 x 353 and one set of 19,772 x 358. The Romanov set was also divided into 8 non-overlapping sets: 7 sets of 24,341 x 346 and one sets of 24,341 x 342.

To investigate the effectiveness of clustering models with and without feature selection, as well as various dimensionality reduction techniques, we directly clustered the original data and feature selection data for 200 genes.

The results are illustrated in the Table 4 for ARI [13] metric and in the Table 5 for V-measure [14] metric.

To test the results for statistical significance, we first used the Friedman test, which the null hypothesis that repeated measurements of the same individuals have the same distribution. If the null hypothesis was rejected, we calculated pairwise comparisons using Conover post hoc test. This test is usually conducted post hoc after significant results of the Friedman test.

We discovered that HDBSCAN+DCA (algorithms 9 and 10) clustering achieved the best results on the original and feature-selected data. On the original data, HDBSCAN+DCA reached an accuracy of 0.87, which was 16.3% and 21.3% higher than those of HDBSCAN+uMAP and HDBSCAN+PCA, respectively. For 200 genes, HDBSCAN+DCA+fs achieved an accuracy of 0.95, which was 4.7%, 17.6% and 25.2% higher than Seurat, HDBSCAN+uMAP+fs and HDBSCAN+PCA+fs, respectively. P-value on Friedman test for HDBSCAN+DCA+fs, HDBSCAN+uMAP+fs, HDBSCAN+PCA+fs and Seurat pipelines is 9.8×10^{-5} and we calculated pairwise comparisons using Conover post hoc test. From the Table 2, HDBSCAN+DCA+fs demonstrated statistically significant result for all three other pipelines.

Table 2. Non-imputed algorithms posthoc p-values

	seurat	**fs+dca+hdbscan**	**fs+umap+hdbscan**	**fs+pca+hdbscan**
seurat	1	0.0417	0.0146	0.0198
fs+dca+hdbscan	0.0417	1	0.038	0.0039
fs+umap+hdbsca	0.0146	0.038	1	0.5329
fs+pca+hdbscan	0.0198	0.0039	0.5329	1

For the imputed data, P-value on Friedman test is 0.0002 we calculated pairwise comparisons using Conover post hoc test. From the Table 3, scImpute+HDBSCAN+DCA+fs demonstrated statistically significant result for all three other pipelines.

Table. 3. Imputed algorithms posthoc p-values

	seurat	**fs+dca+hdbscan**	**fs+umap+hdbscan**	**fs+pca+hdbscan**
seurat	1	0.0475	0.0024	0.0078
fs+dca+hdbscan	0.0475	1	0.038	0.0029

fs+umap+hdbscan	0.0024	0.038	1	0.2079
fs+pca+hdbscan	0.0078	0.0029	0.2079	1

Table 4. Adjusted Rand Index for different experiments

Dataset	**Seurat**	**Impute Seurat**	**Impute DCA hdbscan**	**Impute fs DCA hdbscan**	**Impute uMAP hdbscan**	**Impute fs uMAP hdbscan**	**Impute PCA hdbscan**
zeisel1	0.754	0.81	0.55	0.81	0.518	0.55	0.65
zeisel2	0.83	0.829	0.861	**0.851**	0.651	0.651	0.47
zeisel3	0.748	0.794	0.738	**0.803**	0.608	0.607	0.17
zeisel4	0.793	0.798	0.604	**0.858**	0.549	0.633	0.45
zeisel5	0.781	0.798	0.747	0.785	0.676	0.734	0.5
zeisel6	0.827	0.777	0.628	**0.865**	0.555	0.621	0.15
zeisel7	0.76	0.705	0.828	0.763	**0.893**	0.819	0.17
zeisel8	0.827	0.869	0.861	0.907	0.57	0.657	0.508
romanov1	**0.809**	0.68	0.605	0.604	0.644	0.626	0.364
romanov2	**0.772**	0.625	0.608	0.667	0.599	0.696	0.325
romanov3	0.65	0.643	0.546	0.587	0.476	0.643	0.327
romanov4	**0.696**	0.537	0.553	0.621	0.533	0.675	0.256
romanov5	**0.801**	0.535	0.516	0.614	0.568	0.654	0.077
romanov6	0.565	0.472	0.518	0.55	0.541	0.482	0.071
romanov7	0.561	0.672	0.647	0.66	0.642	0.645	0.671
romanov8	**0.704**	0.641	0.66	0.691	0.637	0.656	0.291

Table 4 (cont.)

Dataset	**Impute fs PCA hdbscan**	**DCA hdbscan**	**fs DCA hdbscan**	**uMAP hdbscan**	**fs uMAP hdbscan**	**PCA hdbscan**	**fs PCA hdbscan**
zeisel1	0.253	0.819	**0.827**	0.476	0.644	0.23	0.278
zeisel2	0.387	0.607	0.838	0.572	0.671	0.55	0.565
zeisel3	0.214	0.727	0.731	0.563	0.644	0.17	0.123
zeisel4	0.306	0.815	0.844	0.548	0.634	0.55	0.281
zeisel5	0.453	0.772	**0.907**	0.52	0.630	0.41	0.436
zeisel6	0.146	0.698	0.802	0.739	0.626	0.19	0.32
zeisel7	0.197	0.795	0.824	0.545	0.619	0.34	0.332
zeisel8	0.549	0.869	**0.95**	0.57	0.609	0.579	0.61
romanov1	0.495	0.754	0.768	0.747	0.668	0.184	0.356
romanov2	0.048	0.791	0.756	0.722	0.731	0.117	0.041
romanov3	0.195	0.641	**0.702**	0.629	0.689	0.694	0.372
romanov4	0.288	0.675	0.69	0.694	0.459	0.152	0.078
romanov5	0.064	0.795	0.797	0.771	0.671	0.152	0.218
romanov6	0.083	0.598	**0.624**	0.472	0.626	0.121	0.153
romanov7	0.447	**0.704**	**0.704**	0.64	0.613	0.799	0.655
romanov8	0.382	0.677	0.686	0.512	0.656	0.293	0.266

Table 5. V-measure for different experiments

Dataset	Seurat	Impute Seurat	Impute DCA hdbscan	Impute fs DCA hdbscan	Impute uMAP hdbscan	Impute fs uMAP hdbscan	Impute PCA hdbscan
zeisel1	0.741	0.802	0.696	**0.835**	0.695	0.71	0.412
zeisel2	0.793	0.798	0.824	**0.894**	0.709	0.785	0.424
zeisel3	0.762	0.786	0.834	**0.885**	0.702	0.705	0.411
zeisel4	0.778	0.813	0.736	**0.846**	0.677	0.702	0.402
zeisel5	0.768	0.8	0.733	0.777	0.695	0.699	0.41
zeisel6	**0.792**	0.754	0.678	0.759	0.643	0.651	0.395
zeisel7	0.779	0.745	0.829	0.851	**0.875**	0.75	0.394
zeisel8	0.817	0.869	0.822	0.873	0.71	0.794	0.42
romanov1	**0.773**	0.629	0.597	0.596	0.591	0.522	0.315
romanov2	0.773	0.58	0.597	0.665	0.512	0.584	0.325
romanov3	**0.661**	0.624	0.531	0.587	0.459	0.521	0.328
romanov4	**0.712**	0.591	0.631	0.684	0.587	0.606	0.348
romanov5	**0.76**	0.59	0.565	0.57	0.515	0.543	0.309
romanov6	0.587	0.533	0.527	0.522	0.547	0.489	0.318
romanov7	0.636	0.636	0.623	0.638	0.545	0.553	0.322
romanov8	**0.702**	0.611	0.638	0.649	0.575	0.558	0.331

Table 5 (cont.)

Dataset	Impute fs PCA hdbscan	DCA hdbscan	fs DCA hdbscan	uMAP hdbscan	fs uMAP hdbscan	PCA hdbscan	fs PCA hdbscan
zeisel1	0.413	0.774	0.826	0.664	0.69	0.415	0.4
zeisel2	0.418	0.638	0.79	0.672	0.685	0.428	0.422
zeisel3	0.407	0.765	0.775	0.718	0.709	0.409	0.402
zeisel4	0.401	0.789	0.809	0.682	0.698	0.414	0.404
zeisel5	0.394	0.765	**0.863**	0.696	0.676	0.405	0.399
zeisel6	0.396	0.713	0.778	0.744	0.635	0.402	0.405
zeisel7	0.386	0.778	0.812	0.709	0.65	0.413	0.404
zeisel8	0.406	0.819	**0.929**	0.715	0.703	0.416	0.401
romanov1	0.316	0.704	0.72	0.687	0.585	0.347	0.345
romanov2	0.317	0.733	**0.791**	0.669	0.651	0.34	0.333
romanov3	0.325	0.616	0.631	0.598	0.592	0.332	0.329
romanov4	0.339	0.669	0.677	0.669	0.513	0.347	0.327
romanov5	0.305	0.739	0.745	0.722	0.617	0.349	0.345
romanov6	0.308	0.611	**0.685**	0.48	0.569	0.343	0.332
romanov7	0.314	0.679	**0.712**	0.64	0.595	0.333	0.343
romanov8	0.321	0.648	0.65	0.54	0.588	0.345	0.329

7. Conclusion

Dimensional reduction and clustering are important when analysing scRNA-seq data. A comparative framework is pro-posed that combines three dimensionality reduction methods and feature selection with HDBSCAN [2] clustering with an en-tire Seurat [1] pipeline. Fourteen experiments were performed on two large scRNA-seq datasets using these combinations.

Two conclusions can be drawn from the results. Thus, feature selection and dimensionality reduction with DCA [5] are critical to achieving better clustering results. If the result is unsatisfactory,

imputation methods maybe introduced. HDBSCAN clustering can give satisfactory results in most cases.

Contributions

M.A., A.M. and E.K. wrote the article. A.N. designed experiments and directed research. M.A. performed research and analysed data. E.K. supervised the project. A.N. provided feedback on the text.

References

[1] A. Butler, P. Hoffman, P. Smibert, E. Papalexi, and R. Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nature biotechnology, vol. 36, no. 5, 2018, pp. 411–420.

[2] L. McInnes, J. Healy, and S. Astels. hdbscan: Hierarchical density based clustering. Journal of Open Source Software, vol. 2, no. 11, 2017, article no. 205.

[3] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. Chemometrics and intelligent laboratory systems, vol. 2, no. 1-3, 1987, pp. 37–52.

[4] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2018.

[5] G. Eraslan, L. M. Simon, M. Mircea, N. S. Mueller, and F. J. Theis. Single-cell rna-seq denoising using a deep count autoencoder. Nature communications, vol. 10, no. 1, 2019, pp. 1–14.

[6] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. ACM Transactions on Database Systems, vol. 42, no. 3, 2017, pp. 1–21.

[7] H. Lu, M. Halappanavar, and A. Kalyanaraman. Parallel heuristics for scalable community detection. Parallel Computing, vol. 47, 2015, pp. 19–37.

[8] P. Brennecke, S. Anders, J. Kim et al. Accounting for technical noise in single-cell rna-seq experiments. Nature Methods, vol. 10, 2013, pp. 1093–1095.

[9] S. Prabhakaran, E. Azizi, A. Carr, D. Pe'er. Dirichlet process mixture model for correcting technical variation in single-cell gene expression data. In Proc. of the 33rd International Conference on Machine Learning, 2016, pp 1070-1079.

[10] A. Zeisel, A.B. Mũnoz-Manchado et al. Cell types in the mousecortex and hippocampus revealed by single-cell rna-seq. Science, vol. 347, no. 6226, 2015, pp.1138–1142.

[11] R.A. Romanov, A. Zeisel et al. Molecular interrogation of hypothalamic organization reveals distinct dopamine neuronal subtypes. Nature neuroscience, vol. 20, no. 2, 2017, pp. 176–188.

[12] W. Li and J.J. Li. An accurate and robust imputation method scimpute for single-cell rna-seq data. Nature communications, vol. 9, no. 1, 2018, article no. 997.

[13] D. Steinley. Properties of the hubert-arable adjusted rand index. Psychological methods, vol. 9, no. 3, 2004, pp. 386–396.

[14] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007, pp. 410–420.

Информация об авторах / Information about authors

Maria Andreevna AKIMENKOVA – laboratory assistant, Information Systems Department. Research interests: analysis of biomedical data, analysis of single cell sequencing data.

Мария Андреевна АКИМЕНКОВА – лаборант отдела «Информационные системы». Сфера научных интересов: анализ биомедицинских данных, анализ данных секвенирования единичных клеток.

Анна Анатольевна МАЗНИНА – лаборант лаборатории геномной инженерии. Сфера научных интересов: РНК-секвенирование, онкогенетика.

Anna Anatolyevna MAZNINA – laboratory assistant at the Genomic Engineering Laboratory. Research interests: RNA sequencing, oncogenetics.

Anton Yurievich NAUMOV – research assistant, Information Systems Department. Research interests: machine learning, convolutional neural networks.

Антон Юрьевич НАУМОВ – стажер-исследователь отдела «Информационные системы». Сфера научных интересов: машинное обучение, сверточные нейронные сети.

Evgeny Andreevich KARPULEVICH – researcher, Information Systems Department. Research interests: analysis of biomedical data, information systems.

Евгений Андреевич КАРПУЛЕВИЧ – научный сотрудник отдела «Информационные системы». Сфера научных интересов: анализ биомедицинских данных, информационные системы.

DOI: 10.15514/ISPRAS–2020–32(5)–9

Application of a New Method of Noninvasive Aassessment of Carbohydrate Metabolism Disorders in the Population Screening

[1] *A.A. Berezin <artparis@mail.ru>*
[1,2] *R.S. Novikov, ORCID: 0000-0002-0656-3475 <rsnovikov@hse.ru>*
[1] *M.A. Novopashin, ORCID: 0000-0002-8919-4002 <mnovopashin@ec-leasing.ru>*
[1,2] *B.A. Pozin, ORCID: 0000-0002-0012-2230 <bpozin@ec-leasing.ru>*
[1,2] *A.V. Shmid, ORCID: 0000-0002-4672-1458 <avshmid@yandex.ru>*

[1] *«EC-leasing» Co.,*
Warshavskoye Shosse, Moscow, 117587, Russia
[2] *National Research University Higher School of Economics,*
20, Myasnitskaya st., Moscow, 101000, Russia

Abstract. Method for conducting a non-invasive screening of the population for carbohydrate metabolism disorders (CMD) has been developed and described. The novelty of the method is that there are no medical standards in the field of endocrinology for the non-invasive type of screening, so the method was based on the results of a clinical study of electrocardiographic abnormalities in patients with CMD, where the method of non-invasive determination of CMD by first-lead ECG was used. During the development of the method, an additional analysis of the ECG sample obtained during the study was performed. As a result of the analysis, it was concluded that the effectiveness of the method (sensitivity and specificity) vary slightly depending on the time of taking an ECG during the day. This means that the patient can come to the screening using the new method of non-invasive detection of CMD not only in the morning and not necessarily on an empty stomach, in contrast to the invasive methods (fasting plasma glucose test and oral glucose tolerance test). To make a decision «there is a suspicion of CMD /there is no suspicion of CMD», the patient only needs to take up to 2 ECGs.

Keywords: first-lead ECG; type 2 diabetes (T2DM); carbohydrate metabolism disorders (CMD)

For citation: Berezin A.A., Novikov R.S., Novopashin M.A., Pozin B.A., Shmid A.V. Application of a new method of noninvasive assessment of carbohydrate metabolism disorders in the population screening. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 121-130. DOI: 10.15514/ISPRAS-2020-32(5)-9.

Применение метода неинвазивного оценивания нарушений углеводного обмена при скрининге населения

[1] *А.А. Березин, <artparis@mail.ru>*
[1,2] *Р.С. Новиков, ORCID: 0000-0002-0656-3475 <rsnovikov@hse.ru>*
[1] *М.А. Новопашин, ORCID: 0000-0002-8919-4002 <mnovopashin@ec-leasing.ru>*
[1,2] *Б.А. Позин, ORCID: 0000-0002-0012-2230 <bpozin@ec-leasing.ru>*
[1,2] *А.В. Шмид, ORCID: 0000-0002-4672-1458 <avshmid@yandex.ru>*

[1] *ЗАО «ЕС-лизинг»,*
Россия, 117587, город Москва, Варшавское шоссе, д. 125, стр.1
[2] *Национальный исследовательский университет «Высшая школа экономики»,*
Россия, 101000, г. Москва, ул. Мясницкая, д. 20

Аннотация. Разработана и описана методика проведения неинвазивного скрининга населения на нарушение углеводного обмена (НУО). Новизна методики заключается в том, что по неинвазивному типу скрининга еще не существует медицинских стандартов в области эндокринологии, поэтому методика составлялась на основе результатов клинического исследования по изучению электрокардиографических отклонений у больных с НУО, где использовался метод неинвазивного определения НУО по ЭКГ первого отведения. Во время разработки методики был проведен дополнительный анализ выборки ЭКГ, полученной в ходе исследования. В результате анализа сделаны выводы, что показатели эффективности метода (чувствительность и специфичность) слабо изменяются в зависимости от времени съема ЭКГ в течение суток. Это означает, что пациент может явиться на скрининг по методике неинвазивного определения НУО не только утром и не обязательно натощак, в отличие от инвазивных методик (тест глюкозы плазмы натощак и пероральный тест толерантности к глюкозе). Для принятия решения «есть подозрение на НУО»/«нет подозрения на НУО» пациенту достаточно снять до 2 ЭКГ.

Ключевые слова: ЭКГ первого отведения; сахарный диабет 2 типа

Для цитирования: Березин А.А., Новиков Р.С., Новопашин М.А., Позин Б.А., Шмид А.В. Применение метода неинвазивного оценивания нарушений углеводного обмена при скрининге населения. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 121-130 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(5)–9

1. Introduction

Diabetes mellitus (DM) is a disease that is constantly spreading throughout Russia and the whole world [3,4]. According to the 2016 NATION study [5]: «Approximately one in five adult Russians had pre-diabetes, 5.4% had T2DM and about half of the diabetic subjects were previously undiagnosed». Type 2 diabetes mellitus (T2DM) is the main disease associated with disordered carbohydrate metabolism (CMD).

The process of developing T2DM is quite long and unnoticeable for the patient. The early stage of carbohydrate metabolism disorders on the average lasts from 5 to 10 years [6]. As a rule, the patient evaluates his condition for the presence of CMD too late, for example, when he has been suffering from T2DM for a long time. For an early detection of CMD, a periodic screening of the population at risk is carried out. Methods that can be used for screening should include those that meet the following requirements:

- easy to perform;
- security;
- standardization;
- low cost;
- reliability

The medical recommendations [2, 7] indicate that an oral glucose tolerance test (OGTT) with 75 g of glucose, a fasting plasma glucose (FPG) test, or a blood test for glycated hemoglobin should be used as a screening test. The disadvantages of these tests include the following:

- invasiveness – in order to obtain blood that measures the level of glucose or glycated hemoglobin, the patient needs to have his skin pierced;
- high cost-these tests require consumables materials (disposable needles, for OGTT - containers with 75 g of glucose, etc.), and it is also necessary to pay for the work of laboratories to analyze the collected blood samples;
- complexity of implementation – the test requires qualified medical personnel and specialized equipment. This means that in practice, these tests can only be taken in city clinics, hospitals and medical laboratories, since there are usually no properly qualified personnel and/or the necessary reagents and equipment for blood analysis in paramedic-midwifery centers (PMC). Therefore, the rural population will have to go to the nearest city medical facility to undergo a full-fledged CMD screening. In addition, to obtain reliable results of OGTT with 75 g of glucose and the FPG test, the patient must come to the screening in the morning and on an empty stomach.

Now, Russia does not yet have medical standards and recommendations for a non-invasive CMD screening, which could compensate for the shortcomings described above.

The paper [1] describes a research devoted to the study of electrocardiographic abnormalities in patients with CMD, conducted under the supervision of A.M. Mkrtumyan, MD.

The study obtained and analyzed a sample consisting of:

- 1997 first-lead ECGs taken from 127 patients with T2DM;
- 741 first-lead ECGs taken from 59 patients without T2DM.

All ECGs were taken in hospitals setting in two medical institutions: Moscow City Clinical Hospital 52 and The Loginov Moscow Clinical Scientific Center.

ECGs were taken using a portable CardioQVARK electrocardiograph [8] (registration certificate no. 2019/8124 dated February 15, 2019), then analyzed for the presence of a cardio sign of CMD with the help of an expert decision support system for taking decision by an endocrinologist developed by the company «EC-leasing» (ES DM), using the method of determining CMD described in the patent [9,10]. According to the results of the study, the sensitivity of the method for one ECG was 70%, and the specificity was 86%.

2. Setting up the problem

The aim of this research work was to develop a method for conducting a non-invasive screening of the population for the CMD based on data and results obtained in the course of a clinical study [1].

To achieve this goal, one must complete the following tasks:

- analyze the data to determine what time of the day the patient should be screened using a non-invasive method, and whether it should be performed on an empty stomach;
- develop a process for conducting a non-invasive CMD screening;
- compare the method of non-invasive detecting of CMD by OGTT with 75 g of glucose, a FPG test and a blood test for glycated hemoglobin according to the criteria for meeting the screening requirements.

3. Preliminary analysis of the sample

Before developing a screening method, it was necessary to determine at what time of the day the patient should take an ECG, and whether it should be done on an empty stomach, so that the

reliability of the method of non-invasive determination of the CMD does not suffer. To do this, we will perform an additional statistical analysis on the ECG sample obtained during the study [1].

Let us clarify the requirements for the time of screening. For that purpose, we will calculate the sensitivity and specificity of the method of non-invasive determination of the CMD for different times of taking the ECG during a day. In accordance with the study protocol [1], the patients with CMD had an ECG taken:

- before breakfast on an empty stomach;
- 2 hours after breakfast;
- before dinner;
- 2 hours after lunch;
- before dinner;
- 2 hours after dinner.

Table 1. Comparison of sensitivity and specificity of the method of noninvasive assessment of the cmd depending on the time of taking the ECG

Time of ECG taking relative to food intake	**Sensitivity (1 ECG each)**	**Specificity (1 ECG each)**
Before breakfast on an empty stomach	70% (246 out of 353 ECGs taken by patients with T2DM had a cardiosign of CMD)	88% (190 out of 216 ECGs taken by patients without T2DM did not show a cardiosign of CMD)
2 hours after breakfast	73% (248 out of 341 ECGs taken by patients with T2DM had a cardiosign of CMD)	91% (189 out of 207 ECGS taken by patients without T2DM did not show a cardiosign of CMD)
Before lunch	74% (253 out of 340 ECGs taken by patients with T2DM had a cardiosign of CMD)	89% (159 of 178 ECGs taken by patients without T2DM did not show a cardiosign of CMD)
2 hours after lunch	75% (155 out of 206 ECGs taken by patients with T2DM had a cardiosign of CMD)	83% (64 out of 77 ECG taken by patients without T2DM, did not show a cardiosign of CMD)
Before dinner	80% (336 out of 421 ECGs taken by patients with T2DM had a cardiosign of CMD)	84% (21 out of 25 ECGs taken by patients without T2DM did not show a cardiosign of CMD)
2 hours after dinner	83% (280 of 336 of the ECG taken by the patients with DM2, had a cardiosign of CMD)	87% (33 out of 38 ECGs taken by patients without DM2 did not show a cardiosign of CMD)

Table 1 shows the results of calculations of the sensitivity and specificity of the method of noninvasive determination of CMD by ECG taken at different times of the day. One can see that the sensitivity for different times of taking the ECG per day varied the sensitivity in the range of 76±6%, and the specificity – in the range of 87±4%. This in turn means the following:

- for the patient, it does not matter what time to come for a non-invasive screening. He can appear in the morning or in the afternoon;
- the patient does not have to come for a non-invasive screening on an empty stomach.

Next, we will calculate the maximum number of required repeated ECG readings per patient, so that each subsequent measurement can improve the accuracy of the screening result («the patient has a suspicion of CMD»/ «No suspicion of CMD»).

To do this, the first 3 ECGs taken by each patient with and without T2DM were taken from the sample (a total of 558 ECGs), then a decision tree was compiled using IBM SPSS Statistics software. The dependent variable of the tree is the sign «there is a suspicion of CMD in the patient» / «there is no suspicion of CMD», independent:

- a manifestation of a cardiosign of CMD on the first ECG of the patient («Yes/no»);
- a manifestation of a cardiosign of CMD on the second ECG of the patient («Yes/no»);
- a manifestation of a cardiosign of CMD on the third ECG of the patient («Yes/no»).

The result of building the tree is shown in fig. 1.

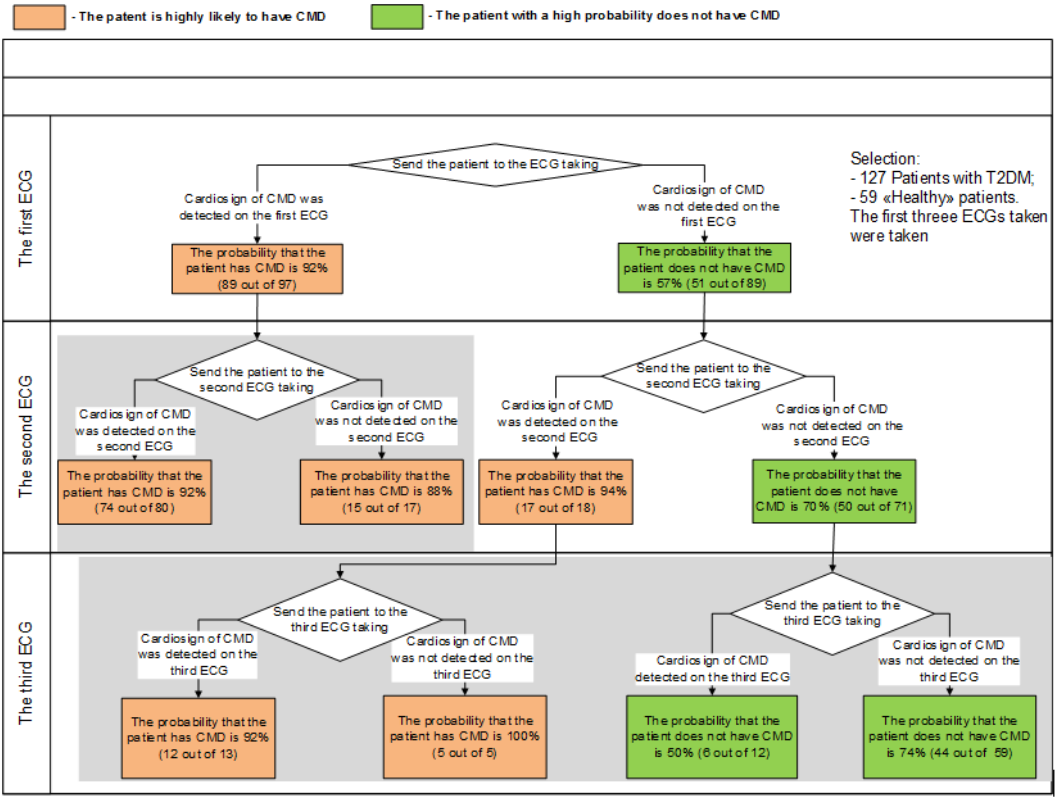


Fig. 1. The tree of taking decision whether the patient has CMD

The decision on the presence or absence of a CMD at each stage was made based on the probability of which option is higher. If at the next stage, the decision was made regardless of whether the cardiosign of the CMD is shown on the ECG or not, the stage was considered meaningless (in Figure 1, highlighted in gray). In total, to achieve maximum accuracy of the screening result, it is sufficient to take no more than 2 ECGs.

4. Description of the screening process

According to the method of non-invasive determination of CMD, screening is divided into 3 stages: ECG taking, ECG processing, and decision-making on the patient.

The ECG is taken where the patient is located. This means that the patient can be screened both in city clinics, in the PMC, or even at home. The main thing is that there should be a portable electrocardiograph on the spot that can transmit the taken ECG over the Internet for further processing.

The ECG processing was performed automatically in a separate computer center using the method of non-invasive detection of CMD. The results of processing the ECG and the value of cardio sign CMD are sent to the endocrinologist.

The decision on the patient «there is a suspicion of CMD/There is no suspicion of CMD» is made by an endocrinologist based on the results of ECG processing using a computer.

There are no requirements for the availability of medical laboratories at any stage of screening using this method, in contrast to invasive methods, where specialized equipment and laboratory conditions are required for the analysis of collected blood samples. The requirements for the number of qualified medical personnel involved have also been reduced:

- at the stage of taking the ECG, the presence of a medical professional is optional;
- it is only necessary to explain to the patient how to use a portable electrocardiograph, and track the correctness of the ECG taking process. If the patient has such an electrocardiograph with him, and he knows how to use it, then a medical personal is not required at the stage of taking the ECG;
- during the processing phase of the ECG no medical personal or laboratory is required;
- a qualified doctor is only needed at the stage of taking a decision on the patient. Moreover, thanks to the automated receipt of the patient's data and the ECG processing results, it is possible to serve the number of patients by an order more than it could be served during the screening process using the invasive methods.

The process of screening based on the method of assessing the CMD [1] and the results of statistical analysis of the sample are shown in Fig. 2.

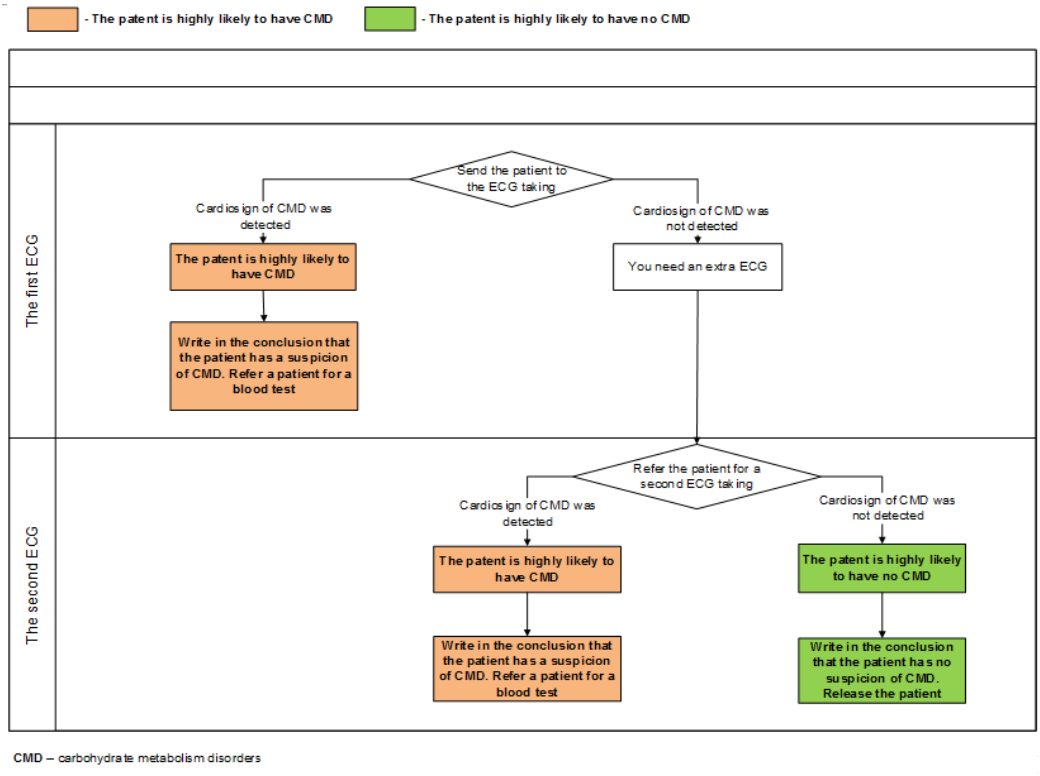


Fig. 2. The algorithm for screening process using a non-invasive method of evaluating the CMD

4.1 First take of the first lead ECG

The first-lead ECG is taken using a portable electrocardiograph. After taking the ECG is sent for processing to the computing center, where a response is received in the form of SMS to the phone of the patient and messages are sent to the endocrinologist about the detected/undetected cardiosign of the CMD. If a cardiosign is detected, the endocrinologist concludes that the patient has a suspicion

of CMD, and gives him recommendations for the diagnosis of T2DM in combination with current medical standards [2] and referral to a blood test (OGTT with 75 g of glucose, analysis for glycated hemoglobin and/or fasting plasma glucose test of the doctor's choice).

In case of non-detection of a cardiosign of the CMD, the doctor will direct the patient to re-take the ECG.

4.2 Second take of the first lead ECG

The ECG is also taken using a portable electrocardiograph. After reading, the obtained ECG is sent for processing to the computer center, where it receives a response in the form of an SMS to the patient's phone number and in the form of an appeal to the endocrinologist about the detection/non-detection of a cardiosign of CMD in patient's ECG.

If the cardiosign is detected, the endocrinologist concludes that the patient has a suspicion of CMD, and gives him recommendations for the diagnosis of T2DM in combination with current medical standards [2] and referral to a blood test (OGTT with 75 g of glucose, analysis for glycated hemoglobin and/or fasting plasma glucose analysis at the doctor's choice). In case of non-detection of cardio sign of CMD, the endocrinologist writes in the conclusion that the patient has no suspicion of CMD, and gives him a recommendation to undergo screening again in a year.

The results of a comparison of the use of invasive methods and the method of non-invasive screening for CMD presented in this paper are presented in Table 2.

Table 2. Comparison of sensitivity and specificity of the method of noninvasive assessment of the CMD depending on the time of taking the ECG

Comparison criteria	**OGTT with 75 g of glucose**	**FPG test**	**Blood test for glycated hemoglobin**	**Method of noninvasive assessment of the CMD**
Ease of execution	Qualified personnel and laboratories are required at the stage of analysis of collected blood. The patient should come to the screening in the morning on an empty stomach	Qualified personnel and laboratories are required at the stage of analysis of collected blood. The patient should come to the screening in the morning on an empty stomach	Qualified personnel and laboratories are required at the stage of analysis of collected blood	At the stage of ECG processing, medical staff and laboratories are not required. The patient does not have to come to the screening in the morning on an empty stomach
Safety	Risk of complications due to invasiveness	Risk of complications due to invasiveness	Risk of complications due to invasiveness	Risks of complications associated with invasiveness are excluded
Standardization	Standardized [2]	Standardized [2]	Standardized [7]	Not yet standardized
Cost	High (expenses for medical laboratories, disposable blood collection materials)	High (expenses for medical, laboratories, disposable blood collection materials)	High (expenses for medical laboratories, disposable blood collection materials)	Low (one only need to buy sets of portable electrocardiographs, consumables)
Reliability	High	High	High	Testing is required

The results of the comparison show that the method of non-invasive assessment of the CMD is not yet prescribed in medical recommendations and standards. Since the study [1] was conducted in a

hospital setting on a fairly small sample of patients (less than 3000), it is necessary to conduct a separate clinical trial of the method for a more accurate assessment of its reliability in real screening.

Due to its ease of implementation, low cost of the test, and noninvasiveness, this technique can be used as an adjunct to the recommended screening tests [2, 7]: OGTT with 75 g of glucose, FPG test, and blood analysis for glycated hemoglobin. Screening using a non-invasive method will allow both to reach a larger number of the covered population at the same time, and to specify which part of the population should undergo a regular invasive screening test.

5. Conclusion

The subject of research in this paper is a non-invasive screening of the population for type 2 diabetes. There are no medical standards in the field of endocrinology for the non-invasive type of screening, so the method is based on the results of a clinical study of electrocardiographic abnormalities in patients with disordred carbohydrate metabolism, where the method of non-invasive determination of CMD by the first-lead ECG was used [1]. The performance indicators of the method of noninvasive determination of CMD for one ECG differ depending on the time of taking the ECG as follows:

- the sensitivity[11] of the method is 70-83% (with the specified sensitivity, regardless of the ECG time taking is 70% [1]);
- the specificity[11] of the method is 83-91% (with the specified sensitivity, regardless of the ECG time taking is 86% [1]);

This means that the requirements for the patient to participate in a non-invasive screening have been reduced – the patient can attend the screening not only in the morning and not necessarily on an empty stomach. In addition, to make a decision by an endocrinologist about the presence/absence of suspicions of CMD, it is enough for the patient to take up to 2 of ECGs.

Based on the data from the study [1] and the results of additional analysis of a sample from the same study, a method for conducting a non-invasive screening of CMD was developed and described. It can become the basis for the development of a new standard for conducting CMD screening. Further testing of this technique is planned.

In the future, the use of non-invasive detection of CMD will significantly reduce the cost of the screening process and make it safer and easier to perform for patients in both urban and rural areas, reduce the requirements for the involvement, safety and qualification of medical personnel who conduct screening and process its results.

Список литературы / References

[1]. A. Shmid and A. Mkrtumyan. Remote Noninvasive Detection of Carbohydrate Metabolism Disorders by First-Lead ECG Screening in CardioQVARK Project. In Proc. of the International Conference on Actual Problems of Systems and Software Engineering (APSSE), 2019, pp. 139-145. doi: 10.1109/APSSE47353.2019.00025.

[2]. Дедов И.И., Шестакова М.В., Майоров А.Ю. и др. Алгоритмы специализированной медицинской помощи больным сахарным диабетом. Сахарный диабет, том 20, no. 15, 2017 г., стр. 1-121 / Dedov I.I., Shestakova M.V., Mayorov A.Yu. et al. Algorithms for specialized medical care for patients with diabetes mellitus. Diabetes Mellitus, vol. 20, no. 15, 2017, pp. 1-121 (in Russian).

[3]. World Health Organization. Global report on diabetes: executive summary. Document no. WHO/NMH/NVI/16.3, 2016.

[4]. International Diabetes Federation. IDF Diabetes Atlas, 9th ed. 2019.

[5]. Дедов И.И., Шестакова М.В., Галстян Г.Р. Распространенность сахарного диабета 2 типа у взрослого населения России (исследование NATION). Сахарный диабет, том 19, no. 2, 2016 г., стр. 104-112 / Dedov I.I., Shestakova M.V., Galstyan G.R. The prevalence of type 2 diabetes in the adult population of Russia (NATION study). Diabetes Mellitus, vol, 19m no. 2, 2016, pp. 104-112 (in Russian).

[6]. Корнеева М.Н., Поддубская Е.А., Марданов Б.У., Дудинская Е.Н. Ранние нарушения углеводного обмена в кардиологической практике: диагностика и лечение. Государственный научно-

исследовательский центр профилактической медицины, 2017, 108 стр. / Korneeva M.N., Poddubskaya E.A., Mardanov B.U., Dudinsky E.N. Early disorders of carbohydrate metabolism in cardiological practice: diagnosis and treatment. State Research Center for Preventive Medicine, 2017, 108 p. (in Russian).

[7]. Мисникова И.В., Древаль А.В., Губкина В.А., Ковалева Ю.А. Алгоритм диагностики сахарного диабета 2-го типа и контроль углеводного обмена: пособие для врачей. МОНИКИ, 2015, 21 стр. / Misnikova I.V., Dreval A.V., Gubkina V.A., Kovaleva Yu.A. Algorithm for diagnosis of type 2 diabetes and control of carbohydrate metabolism: manual for doctors. MONIKI, 2015, 21 p. (in Russian).

[8]. CardioQVARK heart monitor. Available at: http://cardioqvark.ru/, accessed: 19.10.2020.

[9]. Шмид А.В., Березин А.А., Новопашин М.А., Новиков Р.С., Позин Б.А., Мкртумян А.М., Маркова Т.Н. Компьютеризированный способ неинвазивного выявления нарушений углеводного обмена по электрокардиограмме. Патент 2728869 Российская Федерация. 2020 / Shmid A.V., Berezin A.A., Novopashin M.A., Novikov R.S., Pozin B.A., Mkrtumyan A.M., Markova T.N. Computerized method for non-invasive detection of carbohydrate metabolism disorders by electrocardiogram. Patent 2728869 Russian Federation. 2020 (in Russian)

[10]. Novopashin M.A., Shmid A.V., Berezin A.A. Fermi-Pasta-Ulam auto recurrence in the description of the electrical activity of the heart. Medical hypotheses, vol. 101, 2017, pp. 12-16.

[11]. ГОСТ Р 53022.3-2008. Технологии лабораторные клинические. Требования к качеству клинических лабораторных исследований. Часть 3. Правила оценки клинической информативности лабораторных тестов. М., Стандартинформ, 2009, 20 стр. / State Standard 53022.3-2008. Laboratory clinical technologies. Requirements for the quality of clinical laboratory research. Part 3. Rules for assessing the clinical information content of laboratory tests. Moscow, Standartinform, 2009, 20 p. (In Russian)

Информация об авторах / Information about authors

Андрей Александрович БЕРЕЗИН – консультант. Сфера научных интересов: спектральный анализ, кардиология.

Andrey Aleksandrovich BEREZIN – Consultant. Research interests: spectral analysis, cardiology.

Роман Сергеевич НОВИКОВ – старший научный сотрудник отдела математической кардиологии ЗАО «ЕС-лизинг», аспирант ВШЭ. Сфера научных интересов: кардиология, анализ данных.

Roman Sergeevich NOVIKOV – Senior Researcher, Department of Mathematical Cardiology, «EC-leasing» Co., postgraduate student, HSE. Research interests: cardiology, data analysis.

Максим Александрович НОВОПАШИН – начальник отдела математической кардиологии. Сфера научных интересов: кардиология, обработка цифровых сигналов, анализ данных.

Maxim Aleksandrovich NOVOPASHIN – Head of the Department of Mathematical Cardiology. Research interests: cardiology, digital signal processing, data analysis.

Борис Аронович ПОЗИН – д.т.н., технический директор ЗАО «ЕС-лизинг», профессор базовой кафедры ЗАО «ЕС-лизинг» «Информационо-аналитические системы» МИЭМ НИУ ВШЭ. Сфера научных интересов: информационные системы.

Boris Aronovich POZIN – Doctor of Technical Sciences, Chief Technical Officer, «EC-leasing» Co, professor, Department of Information and Analytical Systems: Joint Department with EC-leasing, HSE. Research interests: information systems.

Александр Викторович ШМИД – д.т.н., проф., генеральный директор ЗАО «ЕС-лизинг», заведующий базовой кафедрой ЗАО «ЕС-лизинг» «Информационо-аналитические системы» МИЭМ НИУ ВШЭ. Сфера научных интересов: big data.

Alexander Viktorovich SHMID – Doctor of Technical Sciences, Prof., Chief Executive Officer «EC-leasing» Co, Head of the Department of Information and Analytical Systems: Joint Department with EC-leasing, HSE. Research interests: big data.

DOI: 10.15514/ISPRAS–2020–32(5)–10

Heterogeneous Data Aggregation and Normalization in Information Security Monitoring and Intrusion Detection Systems of Large-scale Industrial CPS

M.A. Poltavtseva, ORCID: 0000-0001-9659-1244 <poltavtseva@ibks.spbstu.ru>
Peter the Great St. Petersburg Polytechnic University,
29, Politechnicheskaya st., Saint Petersburg, 195251, Russia

Abstract. Monitoring of industrial cyber-physical systems (CPS) is an ongoing process necessary to ensure their security. The effectiveness of information security monitoring depends on the quality and speed of collection, processing, and analyzing of heterogeneous CPS data. Today, there are many methods of analysis for solving security problems of distributed industrial CPS. These methods have different requirements for the input data characteristics, but there are common features in them due to the subject area. The work is devoted to preliminary data processing for the security monitoring of industrial CPS in modern conditions. The general architecture defines the use of aggregation and normalization methods for data preprocessing. The work includes the issue from the requirements for the preprocessing system, the specifics of the subject area, to the general architecture and specific methods of multidimensional data aggregation.

Keywords: security monitoring; CPS; data processing; data aggregation; normalization; distributed systems; streaming data processing; security methods; traffic analysis; data analysis; hierarchical aggregation

For citation: Poltavtseva M.A. Heterogeneous Data Aggregation and Normalization in Information Security Monitoring and Intrusion Detection Systems of Large-scale Industrial CPS. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 131-142. DOI: 10.15514/ISPRAS-2020-32(5)-10

Acknowledgements. The reported study was funded by Russian Ministry of Science (information security), project number 2/2020

Агрегация и нормализация гетерогенных данных в системах мониторинга информационной безопасности и обнаружения вторжений крупномасштабных промышленных КФС

М.А. Полтавцева, ORCID: 0000-0001-9659-1244 <poltavtseva@ibks.spbstu.ru>
Санкт-Петербургский политехнический университет Петра Великого,
195251, Россия, г.Санкт-Петербург, ул.Политехническая, дом 29

Аннотация. Мониторинг информационной безопасности промышленных киберфизических систем (КФС) является постоянным процессом, необходимым для обеспечения их безопасности. Эффективность мониторинга зависит от качества и скорости сбора, обработки и анализа гетерогенных данных КФС. Сегодня существует много методов анализа для решения задач безопасности распределенных промышленных киберфизических систем. У этих методов разные требования к характеристикам входных данных, но есть общие особенности, обусловленные предметной областью. Работа посвящена предварительной обработке данных при мониторинге безопасности промышленных КФС в современных условиях. Задача рассматривается от требований к системе предварительной обработки данных и особенностей предметной области, до специфических методов агрегации,

основанных на PCr – связях между структурами данных. Разработана архитектура обработки данных сочетающая методы агрегации и нормализации информации в системе мониторинга.

Ключевые слова: мониторинг безопасности; КФС; обработка данных; агрегация данных; нормализация, распределенные системы; потоковая обработка данных; методы обеспечения безопасности; анализ трафика; анализ данных; иерархическая агрегация

Для цитирования: Полтавцева М.А. Агрегация и нормализация гетерогенных данных в системах мониторинга информационной безопасности и обнаружения вторжений крупномасштабных промышленных КФС. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 131-142 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(5)–10

Благодарности. Исследование выполнено при финансовой поддержке Минобрнауки России (грант ИБ) в рамках научного проекта № 2/2020

1. Introduction

The modern economy requires not only stability and efficiency of enterprises, but also a high degree of modern information technologies application. In developed complexes, they cover all areas of activity, from basic actions and control of physical processes to the corporate and business segments. Intelligent computerized devices that control physical processes are commonly referred to as cyber physical objects. A set of such devices that operate in a distributed network with shared information management is called a cyber-physical system.

Despite the separation of corporate and industrial networks [1], attacks on the physical process through the corporate segment and in opposite directions remain highly relevant [2]. At the same time, the number of attacks on industrial systems and networks is constantly increasing. These are both mass and targeted (APT – advanced persistent threat) attacks. The variety of attacks, the attacker's tools, and their goals is increasing. Threats to the confidentiality of business and industrial data are not only relevant for industrial networks. It is also important to maintain the correctness of the physical process.

Protection of industrial systems is impossible without an effective solution to the monitoring problem, in order to detect intrusions into the CPS, register anomalies of processes taking into account possible malicious impact and distortion of recorded data, analyze threats and investigate security incidents [3]. Monitoring of information security of industrial CPS should not only allow solving security problems, but also do it in a timely manner [4, 5]. To do this, the data on which decisions are made must be processed and provided to the analysis tools with sufficient completeness and speed.

The work is devoted to preliminary data processing in industrial cyber-physical systems security monitoring to ensure data completeness and timeliness, taking into account the main mathematical analysis methods used in solving problems of intrusion detection and anomalies of controlled processes.

2. Related Works

Security monitoring for industrial cyber-physical systems are based on various types of input data. These are network traffic [6-8] and physical node data [9, 10]. Today, there are a large number of works aimed at analyzing network traffic in monitoring systems to solve various security problems [11-13]. Recently, this field has been dominated by works using streaming information processing technologies [14] or (earlier) combining streaming and batch technologies [6, 15]. In the field of data preprocessing methods, these works use traffic aggregation by parameters with the extraction of indicators without additional approaches to improve efficiency [15]. The authors of many works, for example [16, 17], focus on working with high-level features, adapting the processing system to the specific analysis method they use. However, the General list of methods used in industrial CPS information security monitoring is quite large [8, 9, 18-22]. Each method has its own advantages

and disadvantages. Therefore, the greatest efficiency can be achieved by using a combination of analysis methods.

Let's look at modern methods of aggregation and normalization of streaming data. Data aggregation in the cloud based on various semantic features is considered, for example, in [23]. However, semantic grouping does not imply a reduction in dimension, and the algorithm described by the authors is focused on batch, rather than streaming data processing technologies. Mathematical methods such as the principal component method [24] and the eigenvector method [25] reduce the data dimension, but their results are not used as input parameters for most analysis methods and are computationally complex. Specific aggregation methods applicable to security monitoring task data include time series aggregation [26] and aggregation of network traffic statistics [27]. However, the method from [26] creates additional calculations and is not very applicable for processing stream data. The method in [27] is the most widely used in the problems under consideration today.

Next, the paper considers the construction of stages of aggregation and normalization of streaming data for monitoring the safety of industrial CPS. The main approach is to systematize the requirements for pre-processing data from analysis methods. Data normalization is given in accordance with the approach [28] and taking into account the specifics of the cyber-physical systems. Data aggregation is based on the approach [29] for network traffic aggregation. The work also includes an approach to multidimensional data aggregation based on hierarchical relationships and the requirements of multidimensional analysis.

The novelty of the work consists in creating a generalized data processing scheme, adapting existing methods of normalization and hierarchical data aggregation, as well as developing a method of multidimensional data aggregation for monitoring information security of cyber-physical systems.

3. Data Processing Procedure in Information Security Monitoring Systems

3.1. Data Preprocessing in the CPS

Information security monitoring includes several stages. This includes collecting or registering data, pre-processing data (or preliminary data processing) including normalization and aggregation of values, generating output structures for appropriate mathematical methods, applying analysis methods, and making decisions. The general monitoring scheme is shown in fig. 1.

Preliminary data processing is an intermediate step from data collection to applying mathematical analysis methods to it. It includes a number of subtasks and links an instance of the source system (the security object) with generalized methods for solving security problems.

Data preprocessing depends on two factors. First, it depends on the types of data sources and the input data itself. Second, the data consumer is responsible for the analysis tasks. Since data preprocessing depends on the input of the monitoring system, it depends on the security object, or at least its type. Industrial systems are differing from the point of view of information security monitoring from corporate networks. Features of industrial FSCS are:

- heterogeneous data sources;
- resource restrictions;
- data loss during transmission.

Heterogeneous data sources in industrial CPS are represented by two main types of sources. These types are due to the nature of cyber-physical systems. They are:

- network traffic;
- data of physical process sensors.

The combination of these data is used to detect anomalies by various methods in industrial cyber-physical systems [8-10, 13]. The heterogeneity of the physical process sensors is an additional challenge. The same physical indicators that are comparable from an analytical point of view can be presented not only in different formats, but also in different scales (for example, the Celsius and Fahrenheit scales for temperature).

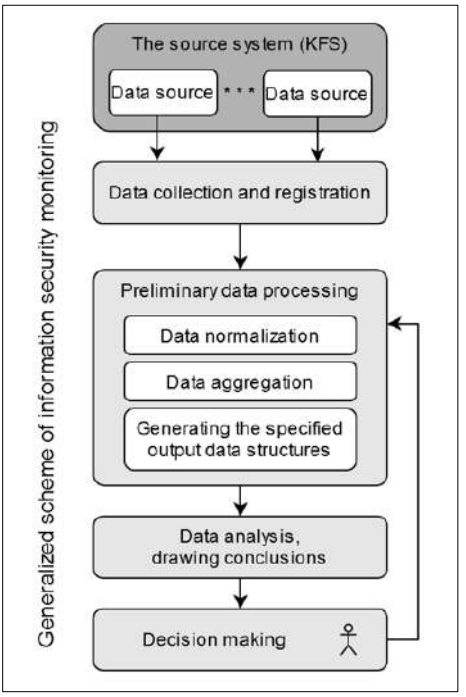


Fig. 1. Information security monitoring scheme

The security goals of a cyber-physical industrial system are achieved through the use of mathematical methods for analysis problems solving. The monitoring data consumer or analysis task has requirements for the preprocessing process:

- correspondence of output data structures to the input parameters of the analysis module;
- the reduction in data processing time.

The first requirement is mandatory for the operation of the data monitoring and analysis system. The second requirement is achieved by optimizing the data processing process. Anomaly detection methods are widely used to monitor the security of industrial cyber-physical systems [3, 4, 8-11, 18, 19, 21-23]. Their peculiarity is the approach to the analysis of network traffic and industrial processes based on time series. A feature of processes in industrial cyber-physical systems is their self-similarity [30]. Because of this, different methods perform data analysis for the same data using different time intervals. The difference between these intervals is significant (from seconds to minutes and months). Thus, the data preprocessing subsystem must convert the same set of input data into multiple time series with different aggregation periods.

Reducing data processing time is achieved due to several factors:

- excluding long operations on data;
- reducing the number of operations on data in General in the system;
- reducing the amount of data (in the system memory).

The first factor is achieved by excluding the write operation to disk during data preprocessing. The use of streaming data processing technologies eliminates long-term operations. This raises the problem of stream aggregation, which is not applicable to traditional methods of batch processing of information [31]. Reducing the number of operations is achieved by optimizing data processing structures and algorithms. It includes optimization of the order of operations (data processing scheme) and selection of optimal algorithms. Reducing the amount of data is achieved by storing in RAM only data structures that are directly used by analysis tasks. Intermediate and non-aggregated data should not take up space. This means that less data is read from memory and fewer resources

are required for the system to work. These are important factors for monitoring system efficiency and performance in large-scale distributed industrial systems.

3.2. Data Flow Processing Diagram

A large-scale distributed industrial system that monitors both network traffic and sensor data generates a large number of heterogeneous data. Achieving maximum performance of the data preprocessing subsystem requires matching the stages of normalization and aggregation of information. For monitoring the safety of large-scale industrial CPS, this agreement defines the list of initial analysis tasks. It consists in developing a data flow diagram. We introduce the following notation:

$S = \{s1, \dots, sn\}$ – data sources set, $D = \{d1, \dots, dm\}$ – set of the original pieces of data received from data sources . Mapping function $Fin: D \rightarrow S$ and its inverse define mappings between these sets, linking data fragments to sources.

$Za = \{z1, \dots zk\}$ – set of the data analysis tasks solved by a given monitoring system to achieve security goals. Each i-th problem has an input data set $Dzi = \{dzi, 1, \dots, dzi, I\}$, characterized by a structure (including the degree of aggregation).

$FA = \{a1, \dots, al\}$ – set of the data aggregation methods;

$FN = \{n1, \dots, nh\}$– set of the data normalization methods;

Then $ftr: \{D, Dzi\} \rightarrow Dzj$ – is a function for processing data and generating output sets for analysis methods, where $Ftr = \{ftr, 1, \dots, ftr, F\} = FA \cup FN$ – is a set of preprocessing functions.

The principle of a data preprocessing pipeline development is to minimize the number of data processing operations by combining the processing and data normalization stages in the most efficient way. There are two main stages for industrial CPS:

1. General stage of data preprocessing. It consists in primary aggregation of data from sources if analytical functions are applied to them separately for each stream. This stage consists of aggregation over time and does not include data normalization.
2. Local stage of data preprocessing. It consists of preparing data for joint analysis. This stage includes normalization of incoming data and repeated co-aggregation.

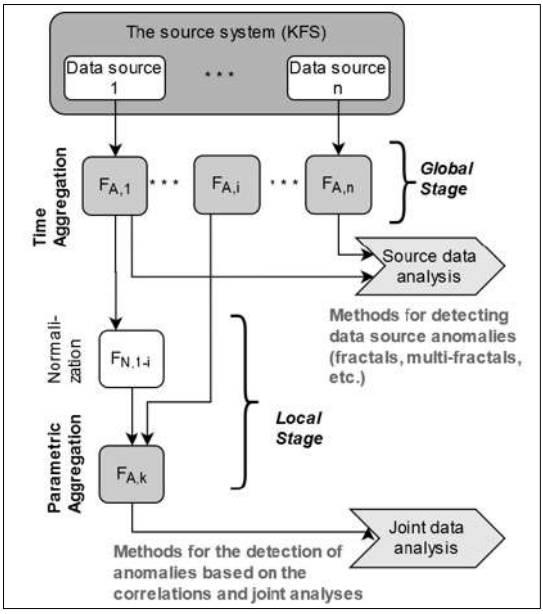


Fig. 2. Diagram of data flows and operations during preliminary data analysis in the monitoring system

Thus, the generalized data preprocessing pipeline for information security monitoring ($DataFlowTransform$) has the form: $DataFlowTransform = (FG: S \to FA) \Rightarrow (\{FL: \{\{FN, 0\} \Rightarrow FA\} \to Za\})$.

At the global FG stage, each data source is aggregated separately on the source itself, the intermediate node, or the monitoring server. The local stage consists of a sequence of normalization (if necessary) and aggregation functions. The result of each aggregation is the input data set of an analysis task. An example is shown in fig. 2.

In some systems, the data processing time optimization problem can be formulated over the given scheme, taking into account the specific characteristics of the processing tools.

3. Data Normalization

The main methods of cyber-physical industrial systems data pre-processing, in solving the security monitoring problem, are methods of aggregation (time and parametric) and data normalization. Time aggregation is also a subspecies of parametric data aggregation (the aggregation parameter is time). Despite this, we will highlight it separately, since this type of aggregation is used at the global stage and has its own characteristics. Data normalization consists of bringing heterogeneous data to a single view. It includes:

- syntactic (structural) normalization;
- semantic normalization.

Syntactic or structural normalization is the data transformation to a single view in terms of the presentation format. This is the process of matching data structures and types between incoming data and data at the input of an aggregation (analysis) method. An example of syntactic normalization is type casting. A special feature of syntactic normalization is that it does not require knowledge of the subject area and additional data about the data source.

Semantic normalization consists in bringing the values of the original data fragments to a single form, taking into account their semantics. For example, if data sources record information in different metric systems, one need to bring measurement scales and convert values. These transformations are also formalizable, but they require prior preparation based on knowledge of the subject area. For example, to aggregate temperature values together, one may need to convert indicators from the Celsius scale to the Kelvin scale, or in the opposite direction.

Stream data normalization corresponds to traditional normalization methods [28] used for industrial systems. Aggregation of the input stream additionally requires the following requirements:

- to use streaming tools or real-time modules for normalization;
- location in the memory reference metadata.

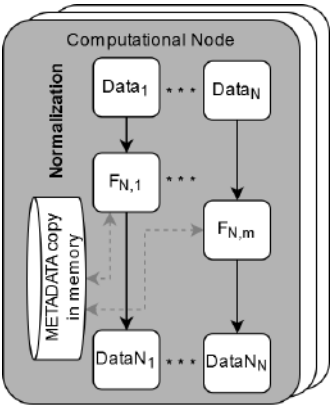


Fig.3 Diagram of data normalization process (one computational node)

In this case, an in-memory DBMS should be used to store metadata structures, or the metadata structure should be denormalized and simply placed in memory. A generalized normalization scheme for a single computing node is shown in fig. 3.

Dynamic load planning for data processing does not allow one to place only fragments of the metadata system on nodes to reduce resource requirements. However, this problem is not critical, since the share of data requiring semantic normalization in modern industrial systems is small. In turn, format normalization is implemented through automatic conversion of the data format during transmission to the local stage. This approach increases the load on individual nodes that do not require format validation. However, it significantly reduces the overall verification time in each individual case, since there is no access to metadata and calling an external conversion function. The choice between the two approaches is determined by the total number of types of transformations in a particular system.

The article goes on to reveal new methods of temporary data aggregation at both stages. Semantic and algorithmic related aggregation methods are used when it comes to a monoparametric data flow from a single source, and when a joint analysis of several data parameters is considered. When developing these methods, the features of the subject area were taken into account: the self-similarity of processes and data, the requirement for high reaction speed, and the requirement to reduce the load on the computer system.

5. Aggregation of streaming data

5.1. Global Aggregation Based on Time Series Hierarchies

The global stage of data aggregation is common for all parameters that enter the monitoring system from the CPS object. The main task of stream data aggregation at this stage is to generate time series of individual parameters with minimizing the number of operations and data stored in memory.

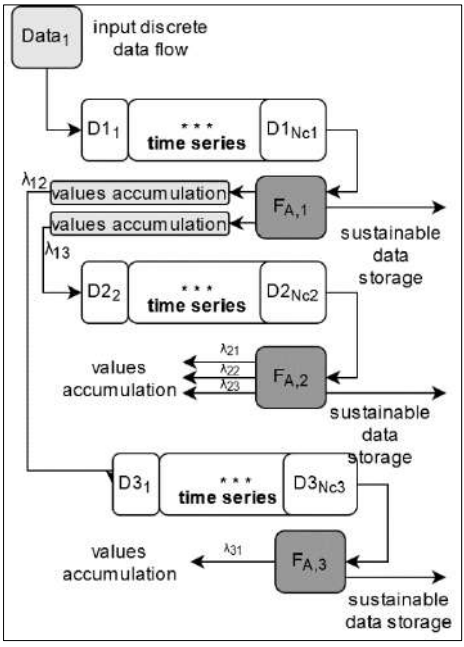


Fig. 4. Diagram of PCR Time series (Hierarchical aggregation)

To solve this problem, one can use the hierarchical aggregation based on time series, tested earlier for network traffic [30]. Aggregation of physical process sensor data over time is similar (in terms

of algorithms) to aggregation of traffic data. The approach is based on time aggregation of streaming data into related time series.

Methods for analyzing self-similar traffic and data use time series sets with different periods and a similar number of series members [8, 10, 13]. The algorithm sorts the time series in ascending order of the aggregation period. The order relation and parent – child relationship are set over time series. Time series with a common parent are located on the same level. Fig. 4 shows the flow diagram of streaming data during hierarchical aggregation in the security monitoring system.

Let's take two time series:

$W = (w_1, \dots, w_l)$ where T is the time of the aggregation window for the entire row, and dt – is the time of the aggregation window for the row element.

$W' = (w'_1, \dots, w'_l)$ where T' is the time of the aggregation window for the entire row, and dt' – is the time of the aggregation window for the row element.

A PCR (Parent-Child-Relation) relationship of the form $W \to W'$ is defined between the rows W and W' if and only if:

- $w'_1 = F_{A,i}(W) * \lambda$, where λ – the multiplier of the aggregation period, $\lambda \in N$.
- $\nexists(\lambda_i)|\lambda_i > \lambda$, λ_i – multiplier of the aggregation period for any other series.

That is, the ancestor of a given series is a series, all elements of which are aggregated to the value of the «descendant» element according to a given rule (aggregation functions $F_{A,i}(W)$ and the multiplier λ). Also, for linked $W \to W'$ series, the rule is true: $dt' = \lambda * T$.

The boundaries of the specified series are set as a set of series parameters when selecting analysis methods. Parameters are set for each time series of data:

- $dt_i \in [dt_i^{start} - \sigma, dt_i^{start} + \sigma]$, where σ – permissible error of the aggregation period of a series element, according to the analysis method;
- $Nc_i \in [Nc_i^{start} - \delta, Nc_i^{start} + \delta]$, where δ – acceptable change in the number of series elements, according to the analysis method, and Nc – is the number of series elements and Nc∈N;
- $N_i \to min$.

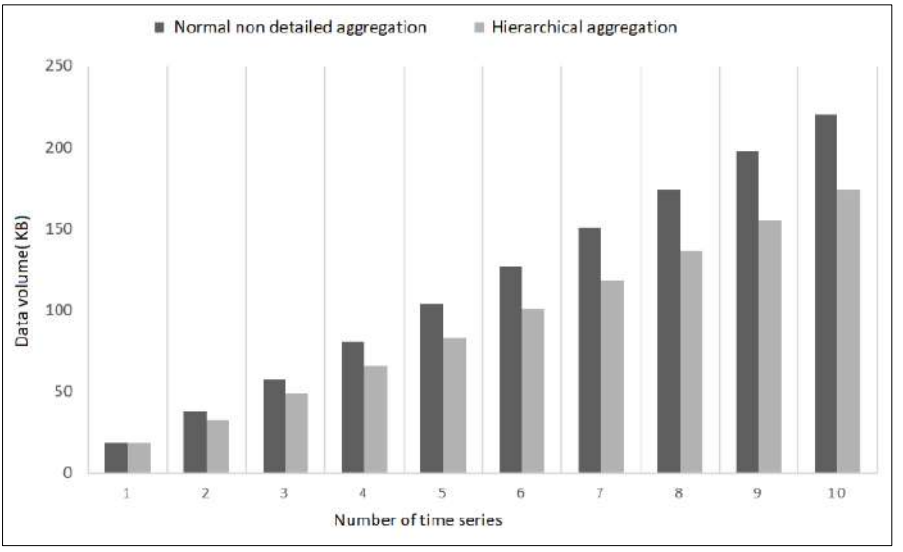


Fig. 5. Hierarchical vs Normal aggregation

Minimizing the number of elements in a series while observing other boundary conditions is a requirement to minimize computational operations during aggregation and processing of series

elements. A comparison of normal effective aggregation and adopted hierarchical aggregation is shown in fig. 5.

As a result, the number of operations with data becomes smaller. All calculations and aggregation over a data element are performed once. Data transfer from a parent with a shorter aggregation period to a child with a longer aggregation period is performed when the corresponding number of elements in the parent series is displaced. The space occupied by data also becomes smaller, since only the necessary data is stored. Storage of intermediate values is minimal.

5.2. Local Aggregation on the Basis of Related Parameters (Multidimensional Aggregation)

When solving the problem of local aggregation, there is a problem of joint analysis of parameters. There are two possible cases. The first is if the parameters of the monitoring object are generalized to a single value after normalization, which is analyzed independently (or the time series of which is analyzed independently). Then the usual hierarchical aggregation described above is used for self-similar data of the cyber-physical system. The second is if several time series of parameters are analyzed together. Then one need to solve two problems:

- consistency on time periods for time series aggregation;
- ensure fast data sharing.

One may use several approaches to solve these problems. First one is building a queue of trees. An additional hierarchy is used above the original series that sets aggregation rules from individual parameters to aggregates. Here, the hierarchy is formed in each individual element of the series. Second one is building a tree of queues. In this case, the hierarchy is set above the original time series trees. Third one is using the key graph. An additional graph (associated key graph - AKG) defines relationships between aggregated parameters, regardless of their original relationships. The graph flexibly links various parameters and queues that can be combined. The weight of the ties and set the coefficients of the transformation function data. The key graph diagram is shown in fig. 6.

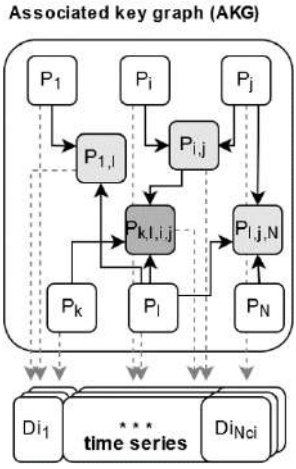


Fig.6. Diagram of associated key graph aggregation

The advantages of the key graph are:

- the possibility of movement on the graph and the changing composition of the aggregate parameters in the framework of certain relations;
- a small amount of additional data for each time window.

The drawback of the graph, as well as the previous methods, is the requirement to set relationships in advance. In other words, the parameters to be aggregated must be known in advance and explicitly

defined. Also, with increasing connectivity, the graph efficiency decreases, and the time for data processing increases. To evaluate the effectiveness, the graph parameters were determined with the growth of the number of related aggregation parameters (fig. 7) and the depth of aggregation of time series (fig. 8).

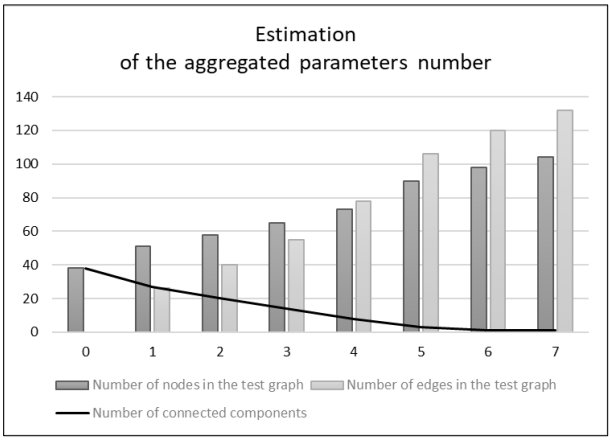


Fig.7. Evaluation of the graph effectiveness (aggregated parameters number)

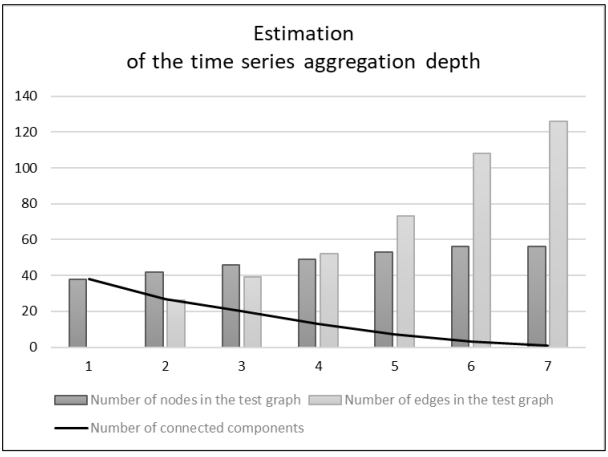


Fig. 8. Evaluation of the graph effectiveness (time series aggregation depth)

We can conclude, that this approach is most successful when the number of jointly aggregated parameters is no more than five and when the nesting depth is no more than six. This restricts the use of the method based on a connected graph. There is a need to further search for effective methods for aggregating a set of related parameters at the local aggregation stage. However, for cyber-physical systems with a small number of jointly analyzed parameters (for example, relatively homogeneous energy networks), this method can be applied.

6. Conclusion

Data pre-processing in the security monitoring task of industrial cyber-physical systems is similar to data pre-processing tasks in other areas. However, it has such features as semantic heterogeneity of input data and heterogeneity of data sources, and limitations on computing resources. The detailed flow diagram of data flows depends on the monitoring task: the number and type of parameters collected, and the requirements of analysis methods. Using a General data flow scheme that includes two stages of aggregation (local and global) and normalization allows one to organize data

processing and decompose it across distributed computing nodes, including data sources. The general scheme is unified for this type of system.

Semantic normalization in CPS security monitoring requires the presence of processing functions directories or metadata on the handler nodes. Semantic normalization transformations in industrial systems are relatively simple and not numerous (by types of transformations). In this way, metadata can be replicated in the memory of handler nodes. However, the minimum requirements for such nodes will be higher.

Hierarchical aggregation of streaming data (at the global stage) and multidimensional aggregation at the local stage allow one to reduce the number of operations on data and reduce the amount of data for online storage. The described approaches are based on the self-similarity inherent in network traffic and industrial processes. Data analysis methods for detecting anomalies and intrusions also use this property.

These methods can be used as a basis for developing an effective system for monitoring the security of primitive cyber-physical systems. They contribute to the improvement of security systems and data streaming technology. Important further areas of research are: improving the efficiency and overcoming the disadvantages of these methods, developing methods for processing secondary data exported from other systems, and ensuring semantic correspondence between methods for solving security problems and the data processing subsystem.

References

[1]. ISA/IEC 62443 Security for Industrial Automation and Control Systems. IEC technical committee 65: Industrial-process measurement, control and automation.

[2]. APT attacks on industrial companies in Russia: a review of tactics and techniques. URL: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/apt-attacks-industry-2019-rus.pdf, accessed 10.10.2020.

[3]. Kalinin M.O. Permanent Protection of Information Systems with Method of Automated Security and Integrity Control. In Proc. of the 3rd International Conference on Security of Information and Networks, 2010. pp. 118-123.

[4]. Knapp E.D., Langill J.T. Security Monitoring of Industrial Control Systems. Lecture Notes in Computer Science, vol. 9588, 2015, pp. 351-386.

[5]. Lavrova D.S., Zaitseva E.A., Zegzhda D.P. Approach to Presenting Network Infrastructure of Cyberphysical Systems to Minimize the Cyberattack Neutralization Time. Automatic Control and Computer Sciences, vol. 53, no. 5, 2019, pp. 387–392.

[6]. Dymora P., Mazurek M. An Innovative Approach to Anomaly Detection in Communication Networks Using Multifractal Analysis. Applied Sciences, vol. 10, 2020, article no. 3277.

[7]. De La Torre Parra G., Rad P., Choo K.-K. R. Implementation of deep packet inspection in smart grids and industrial Internet of Things: Challenges and opportunities. Journal of Network and Computer Applications, vol. 135. 2019, pp. 32-46.

[8]. Kalinin M.O., Lavrova D.S., Yarmak A.V. Detection of Threats in Cyberphysical Systems Based on Deep Learning Methods Using Multidimensional Time Series. Automatic Control and Computer Sciences, vol. 52, no. 8, 2018, pp. 912–917.

[9]. Coletta A., Armando A. Security Monitoring for Industrial Control Systems. Security of Industrial Control Systems and Cyber Physical Systems. Lecture Notes in Computer Science, vol. 9588, 2015, pp. 48–62.

[10]. Zegzhda D., Lavrova D., Khushkeev A. Detection of information security breaches in distributed control systems based on values prediction of multidimensional time series. In Proc. of the International Conference on Industrial Cyber Physical Systems (ICPS), 2019, pp. 780-784.

[11]. Burska K. Oslejsek R. Visual Analytics for Network Security and Critical Infrastructures. Lecture Notes in Computer Science, vol 10356, 2017. pp. 149-152.

[12]. Kalinin M.O., Minin A.A. Security Evaluation of a Wireless Ad-Hoc Network with Dynamic Topology. Automatic Control and Computer Sciences, vol. 51, no. 8, 2017, pp. 899-901.

[13]. Lavrova D.S., Alekseev I.V., Shtyrkina A.A. Security Analysis Based on Controlling Dependences of Network Traffic Parameters by Wavelet Transformation. Automatic Control and Computer Sciences, vol. 52, no. 8, 2018, pp. 931–935.

[14]. Cejka T., Zadnik M. Preserving Relations in Parallel Flow Data Processing. Security of Networks and Services in an All-Connected World. Lecture Notes in Computer Science, vol. 10356, 2017. pp. 153-156.
[15]. Bar A., Finamore A., Casas P., Golab L., Mellia M. Large-scale network traffic monitoring with DBStream, a system for rolling big data analysis. In Proc. of the International Conference on Big Data, 2014, pp. 165-170.
[16]. Mohapatra S.K., Sahoo P.K., Wu S.-L. Big data analytic architecture for intruder detection in heterogeneous wireless sensor networks. Journal of Network and Computer Applications, vol. 66, 2016, pp. 236-249.
[17]. Joshi M., Hassn Hadi T.A Review of Network Traffic Analysis and Prediction Techniques. arXiv preprint 1507.05722, 2015.
[18]. Fahad A., Tari Z., Khalil I., Habibb I., Alnuweiric H. Toward an efficient and scalable feature selection approach for internet traffic classification. Computer Networks, vol. 57, no. 9, 2013, pp. 2040-2057.
[19]. Trihinas D., Pallis G., Dikaiakos M. Low-Cost Adaptive Monitoring Techniques for the Internet of Things. IEEE Transactions on Services Computing, 2018, 14 p. DOI: 10.1109/TSC.2018.2808956.
[20]. Lv F., Wen Ch., Liu M. Representation learning based adaptive multimode process monitoring. Chemometrics and Intelligent Laboratory Systems, vol. 181, 2018, pp. 95-104.
[21]. Lavrova D.S., Popova, E.A., Shtyrkina, A.A. Prevention of DoS Attacks by Predicting the Values of Correlation Network Traffic Parameters. Automatic Control and Computer Sciences, vol. 53, no. 8, 2019, pp. 1065–1071.
[22]. Shang C., Yang F., Huang B., Huang D. Recursive Slow Feature Analysis for Adaptive Monitoring of Industrial Processes. IEEE Transactions on Industrial Electronics, vol. 65, no. 11, 2018, pp. 8895-8905.
[23]. Jiang Y., Yin S., Kaynak O. Data-Driven Monitoring and Safety Control of Industrial Cyber-Physical Systems: Basics and Beyond. IEEE Access, vol. 6, 2018, pp. 47374–47384.
[24]. Karthick N.G., Kalrani A.X. A Survey on Data Aggregation in Big Data and Cloud Computing. International Journal of Computer Trends and Technology (IJCTT), vol. 17, no 1, 2014, pp 28-32.
[25]. Pearson K. On lines and planes of closest fit to systems of points in space. Philosophical Magazine, vol. 2, 1901, pp. 559-572
[26]. Golub G.H., Van Loan C.F. Matrix Computations. Johns Hopkins University Press, 1996, 728 p.
[27]. Leonard M. J., Crowe K.E., Christian S.M., Jennifer Leigh Sloan Beeman, David Bruce Elsheimer, Edward Tilden. Computer-implemented systems and methods for efficient structuring of time series data. United States Patent US009244887B2, 2016.
[28]. David Anthony Hudhes, Pawan Kumar Singh. Hierarchical aggregation of select network traffic statistics. United States Patent US20200021506A1, 2020.
[29]. Poltavtseva M.A., Lavrova D.S., Pechenkin, A.I. Planning of aggregation and normalization of data from the Internet of Things for processing on a multiprocessor cluster. Automatic Control and Computer Sciences, vol. 50, no. 8, 2016, 703–711.
[30]. Poltavtseva M.A., Zegzhda P.D., Pankov I.D. The Hierarchial Data Aggregation Method in Backbone Traffic Streaming Analyzing to Ensure Digital Systems Information Security. In Proc. of the 2018 Eleventh International Conference on Management of Large-Scale System Sevelopment, 2018, pp. 1-5.
[31]. Sheluhin O., Atayero A., Garmashev A. Detection of Teletraffic Anomalies Using Multifractal Analysis. International Journal of Advancements in Computing Technology, vol. 3, no. 4, 2011, pp. 174-182.
[32]. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017, 640 p.

Информация об авторах / Information about authors

Мария Анатольевна ПОЛТАВЦЕВА – кандидат технических наук, доцент, доцент Института кибербезопасности и защиты информации. Сфера научных интересов: системный анализ и обработка информации, хранение данных, СУБД, Большие данные, модели данных, информационная безопасность, мониторинг информационной безопасности, поддержка принятия решений в информационной безопасности.

Maria A. POLTAVTSEVA – Candidate of Technical Sciences, associate Professor of the Institute of cybersecurity and information security since 2014. Research interests: system analysis and information processing, data engineering, DBMS, Big data, data models, information security, information security monitoring, information security decision support systems.

DOI: 10.15514/ISPRAS–2020–32(5)–11

Распределенная модульная платформа «Цифровая Лаборатория» как среда для проведения научных исследований и разработок НИЦ «Курчатовский Институт»

А.Н. Поляков, ORCID: 0000-0002-0316-409X <andrew@kiae.ru>
И.М. Енягина, ORCID: 0000-0003-0004-0582 <irina_enyagina@mail.ru>
Д.С. Коковин, ORCID: 0000-0002-6292-2591 <dskokovin@gmail.com>
НИЦ «Курчатовский Институт», 123182 г. Москва, пл. Академика Курчатова, 1

Аннотация. Курчатовский комплекс НБИКС-природоподобных технологий ориентирован на междисциплинарные исследования и разработки в области нано-, био-, информационных, когнитивных, социогуманитарных наук и технологий. Экспериментальной основой НБИКС комплекса являются Ресурсные центры, действующие в режиме коллективного пользования различными научными лабораториями и содержащие современное оборудование для проведения широкого спектра научных экспериментов. Обработка и хранение полученных экспериментальных данных производится на суперкомпьютере Вычислительного центра, пользование которым также является коллективным. Таким образом, возникают задачи обмена данными между различными зданиями, организации их обработки, анализа и упорядоченного хранения, а также совмещения гетерогенных экспериментальных данных для получения научных результатов более высокого уровня. Для решения данных вопросов на базе распределённой модульной платформы «Цифровая Лаборатория» была организована информационно-аналитическая среда как система, объединяющая в единое виртуальное пространство научное оборудование Ресурсных центров, суперкомпьютер Вычислительного центра, виртуальные машины и персональные компьютеры научных лабораторий, организуя при этом обмен данными между различными зданиями, их обработку, анализ и хранение. Работа с системой осуществляется посредством пользовательского веб-интерфейса. По запросу исследователей каждая процедура работы с экспериментальными данными заданного типа реализуется в виде автономного модуля платформы «Цифровая Лаборатория». Так, например, введен в эксплуатацию и успешно функционирует Модуль «Нейровизуализация» для обработки и анализа фМРТ/МРТ экспериментальных данных головного мозга человека, получаемых на томографе Ресурсного центра. Использование этого модуля делает задачу анализа фМРТ/МРТ данных максимально простой для пользователя, а также позволяет многократно ускорить процесс обработки данных за счёт расспараллеливания вычислений на узлах суперкомпьютера. Помимо создания модулей для работы с экспериментальными данными, система предусматривает возможность создания модулей для работы с данными иного типа. В качестве примера можно привести Модуль «Проектная деятельность» для анализа эффективности научной деятельности исследовательских лабораторий. Использование данной системы позволяет оптимизировать работу с экспериментальными данными в ходе проведения научных исследований за счёт возможности программной реализации необходимых процедур их передачи, хранения, обработки и анализа.

Ключевые слова: анализ данных; научные данные; суперкомпьютер; распределённая платформа; информационная система; экспериментальные данные

Для цитирования: Поляков А.Н., Енягина И.М., Коковин Д.С. Распределенная модульная платформа «Цифровая Лаборатория» как среда для проведения научных исследований и разработок НИЦ «Курчатовский Институт». Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 143-152. DOI: 10.15514/ISPRAS–2020–32(5)–11

Благодарности: Работы выполнены в рамках научно-исследовательской деятельности по теме «Создание распределенной модульной платформы для исследований и разработок «Цифровая лаборатория», утвержденной приказом НИЦ «Курчатовский институт» от 02 июля 2020 года №1055 и при поддержке РФФИ, в рамках научного проекта № 18-29-23020 мк.

«Digital Lab» Platform as an Environment for Scientific Research and Development at the Kurchatov Institute

A.N. Polyakov, ORCID: 0000-0002-0316-409X <andrew@kiae.ru>
I.M. Enyagina, ORCID: 0000-0003-0004-0582 <irina_enyagina@mail.ru>
D.S. Kokovin, ORCID: 0000-0002-6292-2591<dskokovin@gmail.com>
National Research Center «Kurchatov Institute»,
1 Akademika Kurchatova pl., Moscow, 123182, Russia

Abstract. The Kurchatov Complex of NBICS Nature-Like Technologies is focused on interdisciplinary research and development in the field of nano-, bio-, information, cognitive, socio-humanitarian sciences and technologies. The experimental basis of the NBICS complex is the Resource Centers operating in the mode of collective use by various scientific laboratories and containing modern equipment for conducting a wide range of scientific experiments. The processing and storage of the obtained experimental data is carried out on the supercomputer of the Computing Center, the use of which is also collective. Thus, there are problems of data exchange between different buildings, organizing their processing, analysis and orderly storage, as well as combining heterogeneous experimental data to obtain scientific results of a higher level. To solve these issues on the basis of the distributed modular platform «Digital Laboratory», an information and analytical environment was organized as a system that combines the scientific equipment of the Resource Centers, the supercomputer of the Computing Center, virtual machines and personal computers of scientific laboratories into a single virtual space, while organizing the exchange of data between various buildings, their processing, analysis and storage. The work with the system is carried out through the user web interface. At the request of researchers, each procedure for working with experimental data of a given type is implemented as an autonomous module of the «Digital Laboratory» platform. For example, the Module «Neuroimaging» for processing and analysis of fMRI / MRI experimental data of the human brain obtained on the tomograph of the Resource Center was put into operation and is successfully functioning. The use of this module makes the task of fMRI / MRI data analysis as simple as possible for the user, and also makes it possible to speed up the data processing many times over by parallelizing the computations on the supercomputer nodes. In addition to creating modules for working with experimental data, the system provides the ability to create modules for working with data of a different type. An example is the Module «Project Activity» for analyzing the effectiveness of scientific activities of research laboratories. The use of this system allows to optimize the work with experimental data in the course of scientific research due to the possibility of software implementation of the necessary procedures for their transfer, storage, processing and analysis.

Keywords: scientific data; data analysis; supercomputer; information system; experimental data

For citation: Polyakov A.N., Enyagina I.M., Kokovin D.S. «Digital Lab» platform as an environment for scientific research and development at the Kurchatov Institute. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 143-152 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–11

Acknowledgements. This works was supported by the Kurchatov Institute research activities on the project «Creation of a distributed modular research and development platform "Digital laboratory"» approved by order of the Kurchatov Institute on July 02, 2020, No. 1055 and by the RFBR research project No 18-29-23020 mk.

1. Введение

В Ресурсных центрах Курчатовского института имеется более 400 единиц современной экспериментальной техники и установок, что позволяет получать различного рода научные данные. Результаты каждого проводимого эксперимента имеют научную ценность и должны быть сохранены, с возможностью последующего многократного использования в различных исследованиях. Соответственно, необходимо обеспечить упорядоченное централизованное хранение экспериментальных данных, с возможностью автоматического формирования

запрашиваемых выборок данных в соответствии с потребностями того или иного исследования. При этом в ходе современных исследований нередко используются гетерогенные данные, полученные в результате экспериментов разного типа, на различном оборудовании, что позволяет получать научные результаты более высокого уровня. Как пример, объединение данных экспериментов ЭЭГ и фМРТ позволяет получить более точную карту функциональной нейрональной связности головного мозга человека. Однако данные, получаемые на различных экспериментальных установках, разнородны по своей структуре и формату хранения. Таким образом, актуальна задача технического обеспечения интеграции гетерогенных экспериментальных данных в целях проведения их общего анализа. Далее, для обработки и анализа научных данных привлекается суперкомпьютер Вычислительного центра, обеспечивающий высокопроизводительные вычисления. На этом этапе возникает задача обеспечения обмена данными между зданиями Ресурсных центров и зданием Вычислительного центра, а также задача разворачивания на суперкомпьютере или же виртуальных машинах специализированного программного обеспечения, необходимого для выполнения обработки, анализа и визуализации данных заданного вида, в зависимости от потребностей того или иного типа исследований. И наконец, необходим инструмент коммуникации исследователей с организованной системой хранения и анализа данных, наиболее удобным форматом которого является пользовательский веб-интерфейс.

Для решения вышеперечисленных задач на базе распределённой модульной платформы «Цифровая Лаборатория» [1] была организована единая среда как информационно-аналитическая система, обеспечивающая обмен данными между Вычислительным центром, Ресурсными центрами и научными лабораториями, с возможностью привлечения специализированного программного обеспечения заданного типа для обработки, анализа и визуализации. Данная информационная система представляет собой совокупность автономных модулей, базирующихся на платформе «Цифровая Лаборатория». Работа с системой осуществляется посредством пользовательского веб-интерфейса. Помимо общего описания системы, в качестве действующих примеров внедрения в данной статье будут описаны Модуль «Нейровизуализация» для обработки и анализа фМРТ/МРТ данных головного мозга человека [2], а также Модуль «Проектная деятельность» для учёта и анализа эффективности научной деятельности исследовательских лабораторий.

2. Платформа «Цифровая Лаборатория»

Информационно-аналитическая платформа «Цифровая лаборатория» – это распределённая модульная система для интенсивной работы с данными и метаданными в разнородном хранилище данных, с функцией динамического изменения пользователем как самих данных, так и моделей метаданных. Динамичная модификация моделей данных и типов связей между ними позволяет развивать функциональность системы и разрабатывать новые методы обработки и анализа данных [3]. Платформа «Цифровая Лаборатория» выполняет роль менеджера потоков данных для организации преобразования и передачи данных между отдельными элементами информационной среды как системы обработки и анализа данных. Ключевым компонентом платформы «Цифровая Лаборатория» является хранилище метаданных, основными элементами которого являются модели данных и метаданных. Интерфейс к этим моделям имеет стандартизованный формат и предоставляется сервисами. На базе платформы «Цифровая Лаборатория» возможно создание как простых модулей для обработки и анализа однородных данных, так и сложных модулей для обработки и анализа гетерогенных (разнородных) данных.

2.1 Архитектура

Архитектура платформы представлена на рис. 1. Ключевой компонент платформы - регистр метазаписей по данным, основными элементами которого являются модели данных и

метаданных [4]. С помощью информационных моделей описываются объекты предметной области, взаимосвязи между ними и другие метаданные. Стандартизированный интерфейс к информационным объектам предоставляется через сервисы платформы. Сервисы делятся на системные, прикладные и пользовательские. Основной функцией системных сервисов является работа с информационными объектами, в том числе: формирование информационных объектов о загружаемых данных через блок приема, запуск задач на вычислительных ресурсах, обработка запросов к системе хранения данных и др. Пользовательские сервисы служат для создания и модификации моделей данных и метаданных, поиска и визуализации, создания новых информационных объектов.

Доступ к системе осуществляется по протоколу http через веб-портал. Основной функцией портала является обеспечение доступа пользователей к информационным объектам, объектам данных и сервисам, авторизованный доступ к веб-интерфейсу взаимодействия с другими компонентами комплекса, а также администрирование информационного наполнения самого портала.

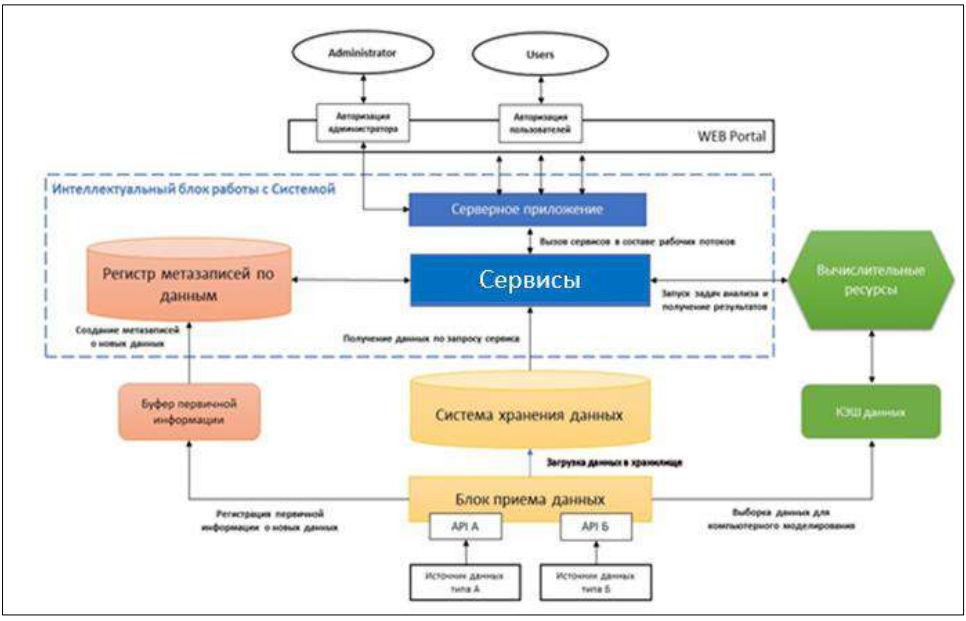


Рис. 1. Архитектура платформы «Цифровая Лаборатория» [1]
Fig. 1. Architecture of the platform "Digital Laboratory" [1]

2.2 Ключевые функции

Платформа располагает набором инструментов для обеспечения следующих функций:

- создание новых моделей данных и метаданных с функцией их динамической модификации;
- регистрация входных данных с извлечением исходных метаданных;
- многокритериальный поиск по атрибутам метаданных;
- легкая интеграция внешних приложений и сервисов пользователями;
- прием, реструктуризация и обработка заданий и рабочих процессов;
- совместное использование данных и права доступа к ресурсам инструментария;
- совместная работа для нескольких пользователей.

2.3 Программное обеспечение

При разработке платформы использовалось следующее свободно распространяемое ПО:

- языки программирования Java (версии 8 и выше), JavaScript, HTML 5;
- форматы описания метаданных XML и бизнес процессов BPMN 2.0 [5];
- СУБД MySQL версии 5.5 [6];
- интерфейсы J2EE [7];
- фреймворки Spring, Hibernate, Activiti;
- веб сервер Apache и сервер приложений Tomcat [8].

Для хранения прав доступа используется реляционная СУБД. Каждая запись определяется кортежем <ObjectId, GroupId, RoleId, AccessLevel>, где ObjectId – идентификатор объекта доступа, GroupId и RoleId – идентификатор группы и роли в этой группе, для которой требуется проверить права доступа, AccessLevel – уровень доступа.

Для повышения скорости доступа к объектам и работы сервисов системы используются права доступа в первую очередь для регулирования доступа к сервисам и функциям, предоставляемым платформой. Это уменьшает количество записей и сокращает число проверок.

3. Примеры внедрения модулей системы

3.1 Модуль «Нейровизуализация»

Создание методов МРТ и фМРТ томографии открыло новые возможности для исследования анатомической структуры и функциональной нейрональной активности головного мозга человека. Однако с увеличением количества проводимых экспериментов возникла задача быстрой обработки и анализа потоков экспериментальных МРТ/фМРТ данных, а также их упорядоченного централизованного хранения [9]. С этой целью на базе платформы «Цифровая Лаборатория» был разработан и внедрён Модуль «Нейровизуализация», представляющий собой систему хранения, анализа и визуализации экспериментальных нейробиологических данных (МРТ, фМРТ), с привлечением в качестве вычислительного ресурса кластера HPC4 суперкомпьютера НИЦ «Курчатовский Институт» [2]. Система позволяет проводить научные исследования в области изучения анатомической структуры и функциональной нейрональной архитектуры головного мозга человека, с использованием как традиционных методов и инструментов математического моделирования, так и уникальных методик, разрабатываемых исследователями. В настоящее время Модуль «Нейровизуализация» используют в Курчатовском институте для интеграции и анализа данных МРТ и фМРТ, полученных на томографе Siemens Verio Magnetom 3T Ресурсного центра «Когнимед».

Модуль «Нейровизуализазция» имеет следующую структуру.

- Хранилище данных: хранение экспериментальных МРТ/фМРТ данных, а также результатов их предобработки, обработки и анализа на суперкомпьютере.
- Вычислительный элемент: Обработка и анализ данных на суперкомпьютере с использованием свободно распространяемого специализированного программного обеспечения (FreeSurfer, FSL, SPM и др.), а также с использованием собственных разработок. Распараллеливание вычислений на узлах суперкомпьютера.
- Аналитика и статистика: формирование выборки данных по набору параметров; генерация аналитических и статистических отчётов.
- Визуализация: создание моделей нейровизуализации данных с помощью свободно

распространяемых визуальных редакторов (FreeView, Brain Browser).

- Модуль управления данными: организует обмен данными между веб-порталом, хранилищем данных, вычислительным элементом, блоками аналитики и визуализации.
- Веб-интерфейс: организует взаимодействие системы с внешними устройствами (ПК оператора томографа, ПК исследователей).

Компьютерная модель Модуля «Нейровизуализация» представлена на рис. 2.

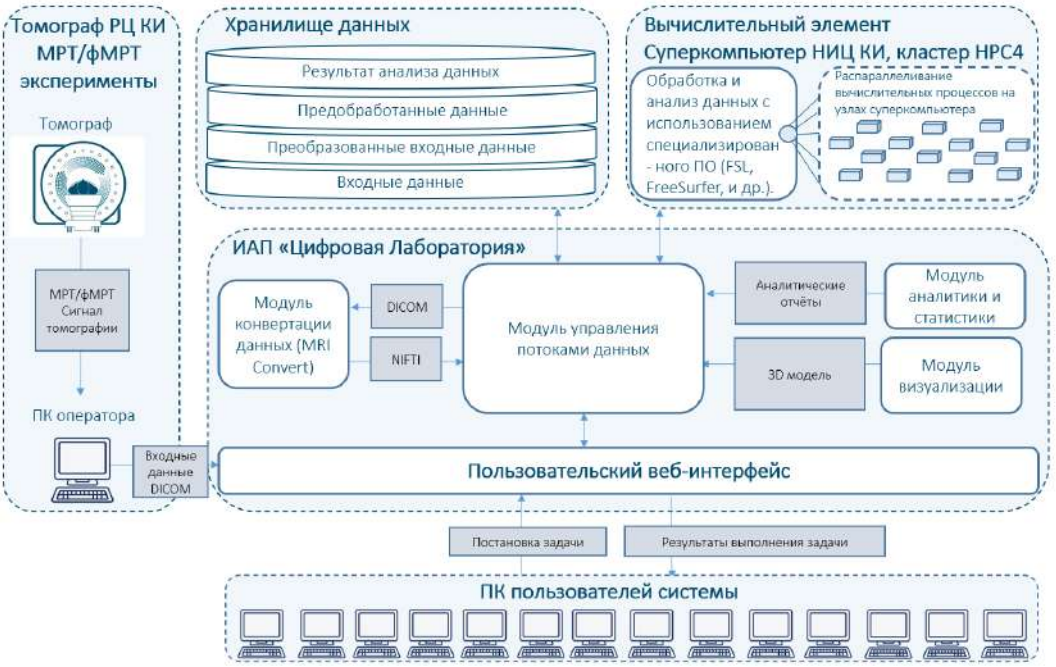


Рис. 2. Компьютерная модель Модуля «Нейровизуализация»
Fig. 2. Computer model of the Module «Neuroimaging»

Рабочий процесс можно описать следующим образом: МРТ и фМРТ эксперименты выполняются на томографе ресурсного центра, данные записываются оператором томографа в виде каталога файлов формата DICOM, которые являются «входными данными» для Модуля «Нейровизуализация». Затем система конвертирует набор DICOM файлов в один файл формата NIFTI (4D NiFti – фМРТ, 3D NiFti - МРТ), отделяя при этом непосредственно экспериментальные данные от метаданных эксперимента. Метаданные направляются в хранилище метаданных системы, тогда как данные направляются для обработки и анализа на суперкомпьютер, где развёрнуто необходимое специализированное программное обеспечение. При этом обработка одного NIFTI файла с результатами фМРТ эксперимента на ПК может занимать 24 часа и более, тогда как в рамках исследования может потребоваться оперативная обработка десятков или сотен таких файлов одновременно.

Использование Модуля «Нейровизуализиация» позволяет многократно ускорить, упростить и расширить возможности обработки и анализа данных МРТ и фМРТ за счёт распараллеливания вычислительных процессов на узлах суперкомпьютера, обеспечения широты выбора специализированных инструментов обработки и анализа, упорядоченного централизованного хранения, обеспечения доступа к данным из любого места в любое время. Полученные результаты используются в практической медицине и научно-исследовательской деятельности, для индивидуального (данные одного пациента) и группового (обычно данные от 20 до 200 пациентов) анализа. Рис. 3 представляет процесс

преобразования экспериментальных данных при использовании Модуля «Нейровизуалзация». Рис. 4 демонстрирует веб-интерфейс пользователей.

Рис. 3. Процесс преобразования экспериментальных данных МРТ/фМРТ
Fig.3. MRI/fMRI experiments data work flows

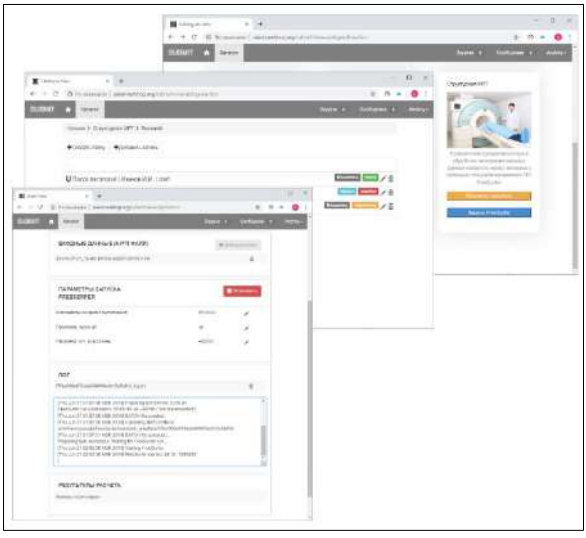


Рис. 4. Пользовательский веб-интерфейс Модуля «Нейровизуализация»
Fig.4. The user web interface of the Module "Neuroimaging"

3.2 Модуль «Проектная деятельность»

Основной целью создания Модуля «Проектная деятельность» является повышение эффективности организации и управления научно-технической деятельностью подразделений НИЦ «Курчатовский Институт». В рамках этой задачи система обеспечивает сбор и структурированное хранение данных о научной деятельности сотрудников (проекты, исследования, публикации, патенты и прочее), предоставляет данные о текущем статусе и эффективности выполнения проектов, обеспечивая тем самым возможность всестороннего

анализа и оперативного принятия необходимых решений. Модуль позволяет фиксировать результаты научной деятельности лабораторий, рабочих групп и отдельных сотрудников лабораторий, предоставляя тем самым возможность оперативного доступа к перечню работ той или иной группы исследователей или же того или иного проекта. Модуль также содержит функционал расширенного поиска сотрудников и их достижений, организуя в том числе автоматическое формирование текстовых и графических статистических отчётов. Гибкость и ёмкость Модуля обеспечивается за счёт разделения входных данных на непосредственно данные и метаданные (данные о данных, содержащие признаки данных). Рисунок 5 представляет компьютерную модель Модуля «Проектная деятельность».

Рис. 5. Компьютерная модель Модуля «Проектная деятельность»
Fig.5. Computer model of the Module «Project Activity»

Основные элементы архитектуры системы «Проектная деятельность».

- Модуль управления потоками данных. Разделяет полученные от пользователей системы данные на инфрмационные данные (далее «данные») и метаданные. Также модуль обеспечивает взаимообмен потоками данных между пользовательским веб-интерфейсом, хранилищем метаданных, хранилищем данных, модулем анализа эффективности научной деятельности.
- Хранилище данных. Предназначено для упорядоченного хранения данных, загружаемых пользователями системы. Система подразумевает хранение не только объектов (метазаписей), но и сторонних объектов данных (созданных за пределами платформы), связанных с объектом, например, изображений, текстовых документов и т.д. Все сторонние объекты хранятся в виде файлов и должны быть связаны как минимум с одним базовым объектом. Это позволяет контролировать все процессы, связанные с жизненным циклом файла, и отслеживать такие события как: загрузка файла в систему, имя пользователя, загрузившего файл, время загрузки, изменения файла (замена другой версией).
- Хранилище метаданных. Предназначено для хранения метаданных моделей, загружаемых пользователями системы. Интерфейс к этим моделям имеет стандартизированный формат и будет предоставляться сервисами. Сервисы представляют три группы: система, приложение и пользователь. Одна из основных функций для системных сервисов – извлечение начального набора метаданных из

входных данных, а также создание дополнительных метаданных во время использования системы пользователем. Системные сервисы обеспечивают, среди прочего, загрузку работы на вычислительные ресурсы, обработку данных, формирование запросов в систему хранения.

- Модуль анализа эффективности научной деятельности. Предназначен для обработки и систематизации загруженных в систему данных о выполняемых проектах и исследованиях, с целью анализа эффективности научной деятельности по заданным признакам. Функционал модуля предполагает возможность автоматического формирования аналитических отчётов.
- Веб-интерфейс. Доступ к системе осуществляется по протоколу http через веб-портал. Основной функцией портала является обеспечение доступа пользователей к базовым объектам, объектам данных и сервисам, авторизованный доступ к веб-интерфейсу взаимодействия с другими компонентами комплекса, а также администрирование информационного наполнения самого портала.

Модуль подразумевает хранение не только информационных объектов (метазаписей), но и сторонних объектов данных (созданных за пределами платформы), связанных с информационным объектом, например, изображений, текстовых документов и т.д. Все сторонние объекты хранятся в виде файлов и должны быть связаны как минимум с одним информационным объектом. Это позволяет контролировать все процессы, связанные с жизненным циклом файла, и отслеживать такие события как, загрузка файла в систему, имя пользователя, загрузившего файл, время загрузки, изменения файла (замена другой версией). Взаимодейсвие Модуля с пользователями осуществляется посредством веб-интерфейса. Модуль «Проектная деятельность» объединяет все данные, участвующие в процессе выполнения научных проектов, в единое многомерное пространство на уровне метаданных, а также предоставляет механизмы их поиска, обработки и актуализации. На основе собранных данных происходит динамическое формирование отчетов о текущем состоянии дел подразделения, что даёт возможность оперативно выявлять возникшие затруднения и перераспределять ресурсы подразделения по критически важным задачам.

4. Заключение

В результате выполненных работ была организована единая информационно-аналитическая среда для проведения научных исследований и разработок НИЦ «Курчатовский Институт» как система на базе распределённой модульной платформы «Цифровая Лаборатория». Данная система позволяет формировать по мере необходимости автономные модули, реализующие различные сценарии сбора, обработки, анализа и хранения экспериментальных научных данных, получаемых на оборудовании Ресурсных центров. Исследователи имеют возможность работать с системой посредством пользовательского веб-интерфейса.

Создание такой системы позволяет упростить и ускорить процедуры работы с экспериментальными данными, а также организовать упорядоченное централизованное хранение таких данных, с возможностью автоматического формирования необходимых выборок и их многократного использования в различных научных исследованиях. Помимо работы с экспериментальными данными, система предоставляет возможность хранения и анализа данных другого типа, таких, например, как индикаторы научной деятельности по проектам.

В настоящее время система введена в эксплуатацию для подразделений Курчатовского комплекса НБИКС-природоподобных технологий, используются модули «Проектная деятельность» и «Нейровизуализация». Помимо функционирования уже созданных модулей, система предусматривает создание и запуск новых модулей по запросу научно-исследовательских групп.

Список литературы / References

[1] Polyakov A., Kokovin D., Poyda A., Zhizhin M., Andreev A., Govorov A., Ilyin V. Toolkit for intensive work with metadata in specialized information systems. Procedia Computer Science, vol. 119, 2017, pp. 59-64.

[2] Enyagina I.M., Polyakov A.N., Poyda A.A., Ushakov V.L. System for Automatic Processing and Analysis of MRI/fMRI Data on the Kurchatov Institute Supercomputer. EPJ Web of Conferences, vol. 226, 2020, article no. 03006.

[3] Bubenko Janis A., jr. From Information Algebra to Enterprise Modelling and Ontologies - a Historical Perspective on Modelling for Information Systems. In Conceptual Modelling in Information Systems Engineering, Springer, 2007, pp. 1-18.

[4] Burgess N. Scaling an RNS number using the core function. In Proc. of the 16th IEEE Symposium on Computer Arithmetic, 2003. pp. 262-269.

[5] Jenn Riley. Understanding Metadata: What is Metadata, and What is it For? National Information Standards Organization, 2017, 49 p.

[6] Extensible Markup Language (XML) 1.1 (Second Edition). W3C Recommendation, 2006. URL: https://www.w3.org/TR/xml11/, accessed 20.11.2020.

[7] MySQL 5.1 Release Notes. URL: https://docs.oracle.com/cd/E17952_01/mysql-5.1-relnotesen/index.html, accessed 20.11.2020.

[8] Java EE Overview. URL: http://www.oracle.com/technetwork/java/javaee/overview/index.html, accessed 20.11.2020.

[9] An open source framework for creating Java EE web applications. URL: https://struts.apache.org/, accessed 20.112020.

[10] Poldrack Russell A., Mumford Jeanette A., Nichols Thomas E. Handbook of Functional MRI Data Analysis, Cambridge University Press, 2011, 238 p.

Информация об авторах / Information about authors

Андрей Николаевич ПОЛЯКОВ – кандидат технических наук, заместитель начальника отдела нейрокогнитивных наук, интеллектуальных систем и робототехники Комплекса НБИКС-природоподобных технологий. Сфера научных интересов: информационные распределённые системы, анализ и хранение научных данных, динамические данные.

Andrey POLYAKOV – PhD, Deputy Head of the Department of Neurocognitive Sciences, Intelligent Systems and Robotics of the Kurchatov Complex of NBICS Nature-Like Technologies. Research interests: information distributed systems, analysis and storage of scientific data, dynamic data.

Ирина Михайловна ЕНЯГИНА – младший научный сотрудник Лаборатории технологий искусственного интеллекта. Сфера научных интересов: нейровизуализация, информационные распределённые системы, анализ и хранение научных данных.

Irina ENYAGINA – Researcher at the Laboratory of Artificial Intelligence Technologies. Research interests: information distributed systems, analysis and storage of scientific data, neuroimaging.

Дмитрий Сергеевич КОКОВИН – инженер-исследователь Лаборатории технологий искусственного интеллекта. Сфера научных интересов: СУБД, информационные системы.

Dmitry KOKOVIN – Research Engineer at the Laboratory of Artificial Intelligence Technologies. Research interests: data base, information systems, analysis and storage of scientific data.

DOI: 10.15514/ISPRAS–2020–32(5)–12

Модельный подход к обеспечению безопасности и надежности Web-сервисов

[1,2] *Е.М. Лаврищева, ORCID: 0000-0002-1160-1077 <lavr@ispras.ru>*
[1,3] *С.В. Зеленов, ORCID: 0000-0003-0446-0541 <zelenov@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, 141701, Россия, Московская обл., г. Долгопрудный, Институтский пер., д. 9*
[3] *Национальный исследовательский университет «Высшая школа экономики» 101000, Россия, г. Москва, ул. Мясницкая, д. 204*

Аннотация. Дается анализ проблематики надежности и безопасности в мировой практике и в нашей стране. Рассмотрены аспекты моделирования программно-технических систем (ПТС) из сервисных ресурсов и готовых КПИ с обеспечением надежности и безопасности. Приводятся сформировавшиеся базовые и теоретические основы проблемы моделирования, опыта использования современных сервисных средств SOA, SCA, SOAP в ПТС и Web-системах с обеспечением их надежности и безопасности в Интернет. Отмечается, что ПТС и Web-системы создаются методом сборки в современных средах: IBM WSDK + WebSphere, Apache Axis + Tomcat; Microsoft .Net + IIS и др. Должны проводиться верификация и тестирование систем для поиска ошибок, возникающих при исключительных ситуациях (кибератаках, запрещенных доступах к БД и др.). Описаны методы анализа таких ситуаций и применения методов надежности и безопасности для обеспечения устойчивой и безотказной работы сервисных компонентов ПТС в информационной среде Интернет.

Ключевые слова: безопасность; надежность; качество; сервисный ресурс; сервисные службы; Web-системы; анализ уязвимости; ошибочные ситуации; оценка качества

Для цитирования: Лаврищева Е.М., Зеленов С.В. Модельный подход к обеспечению безопасности и надежности Web-сервисов. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 153-166. DOI: 10.15514/ISPRAS–2020–32(5)–12

Благодарности: Работа поддержана грантом РФФИ № 19-01-00206.

Model-Based Approach to Ensuring Reliability and Security of Web-services

[1,2] *E.M. Lavrischeva, ORCID: 0000-0002-1160-1077 <lavr@ispras.ru>*
[1,3] *S.V. Zelenov, ORCID: 0000-0003-0446-0541 <zelenov@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 (MIPT)*
141700, Russia, Moscow region, Dolgoprudny, Campus per., 9.
[3] *National Research University Higher School of Economics (HSE)*
20 Myasnitskaya Ulitsa, Moscow, 101000, Russia

Abstract. In the paper, we analyze problems of reliability and security in the world practice and in Russia. We consider aspects of modeling software/hardware systems from service resources and ready-made reuses with ensuring reliability and security. We present the formed basic and theoretical foundations of the modeling problem, the experience of using modern service tools SOA, SCA, SOAP in software/hardware systems and Web systems to ensure their reliability and security on the Internet. We note that software/hardware systems and Web systems are created by the assembly build method in modern environments: IBM WSDK + WebSphere, Apache Axis + Tomcat; Microsoft .Net + IIS, etc. Verification and testing of systems should be conducted for searching of errors that occur in exceptional cases (cyber-attacks, forbidden access to the database, etc.). We describe methods for analyzing such situations and applying reliability and security methods to ensure stable and trouble-free operation of software/hardware systems service components in the Internet information environment.

Keywords: security; reliability; quality; service resource; service services; Web systems; vulnerability analysis; erroneous situations; quality assessment

For citation: Lavrischeva E.M., Zelenov S.V. Model-Based Approach to Ensuring Reliability and Security of Web-services. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 153-166 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–12

Acknowledgments. This work was supported by the RFBR grant no. 19-01-00206.

1. Введение

1.1 Исторические аспекты надежности техники

Теория надежности как самостоятельная научная дисциплина начала складываться после окончания 2-й мировой войны. Военное ведомство США начало интенсивно финансировать работы, связанные с качеством вооружений самого разного назначения. В СССР работы по надежности техники, связанные с военной радиофизической техникой в авиационной, космической, автомобильной и морской областях, были инициированы военно-промышленным комплексом (ВПК) страны.

С 1950г. в СССР проводились разработки по создания надежной техники и оборудования в военной отрасли по постановлению Минрадиопрома СССР и проведения конкретных разработок по программе ВПК (1950-1992). Такие работы проводятся и сейчас в оборонной промышленности в рамках защиты наших земных и атмосферных границ.

В компьютерной науке в 1986 г. появилось усовершенствованное направление классической теории надежности для технических средств – dependability (гарантоспособность), включающее в себя вопросы обеспечения отказоустойчивости, готовности, живучести, достоверности, целостности и конфиденциальности. Современные сервис-ориентированные компьютерные системы (КС) должны использовать надежные гарантоспособные сервисные Web-услуги Интернет для создания программно-технических систем (ПТС) со свойствами работоспособности, безотказности, долговечности и ремонтопригодности.

По программе ВПК (1950-1992) такие средства были разработаны в СССР для авиации, космоса, энергетики, межконтинентальных баллистических ракет, боевых самолетов, вертолетов и бортовой космической техники. В настоящее время проводятся активные работы по созданию новых видов работоспособного оборудования для оборонной промышленности и медицины страны, учитывающие катастрофические жертвы, пожары, вирусные заболевания и т.п.

Отличный экскурс в историю теории надежности в Советском Союзе и за рубежом содержится в публикации И. Ушакова [1].

1.2 Гарантоспособность вычислений на компьютерах

Представитель технического комитета IFIP «Гарантоспособные вычисления и отказоустойчивость» Дж. К. Лапри (J.C. Laprie) в 1992 году написал книгу «Гарантоспособность. Основные определения и терминология» [2]. В Брюсселе в 1992 году были опубликованы «Критерии оценки безопасности информационных технологий (ITSEC)» [3], принятые в Германии, Франции, Великобритании, Италии и США. Эти критерии постоянно совершенствуются и согласуются с разными странами-участницами IFIP. Так, департамент компьютерных наук университета Вирджинии (Department of Computer Science University of Virginia) занимается повышением живучести информационных систем (ИС) в критических инфраструктурах.

Центр надежных и высокоэффективных вычислений Иллинойского университета (Center for Reliable and High-Performance Computing University of Illinois) модифицировал программный продукт Chameleon до уровня функциональной надежности КС с сетевой структурой. В нем Fault-Tolerance Manager должен обеспечивать адаптацию гарантоспособных средств в разных вычислительных средах (однородной, гетерогенной и кластеризованной).

Департамент компьютерных наук университета в Теннесси (Department of Computer Science University of Tennessee) разрабатал средства для отказоустойчивых сетевых вычислений NetSolve, усовершенствуя клиент-серверную структуру Интернет. В ней модульность продукта дает возможность вести гарантоспособные вычисления на двух уровнях: внутреннем и межсерверном.

Ведущими учреждениями в области гарантоспособности США являются: Лаборатория реактивного движения (JPL) Калифорнийского технологического института; Корпорация электронных систем коммутации Bell System (США, 1953 г.), ныне AT&T; компания IBM с Исследовательским центром им. Дж. Уотсона; Лаборатория им. Ч. Дрейпера Массачусетского технологического института; Международный Стэнфордский научно-исследовательский институт; корпорация Boeing. Вопросами гарантоспособности ТС занимаются: во Франции – Лаборатория автоматики и системного анализа (LAAS) Тулузского национального научно-исследовательского центра и компания Schneider Electric, в Англии – компании SRC и CSR, в Германии – компания Siemens и другие.

В зарубежных работах ([4-6]) приводятся основные результаты исследований в области гарантоспособных и надежных отказоустойчивых КС.

Имеются многочисленные отечественные работы в области обеспечения надежности и безопасности ПТС, выполненные в рамках ВПК в период 1950-1992, а также в современных космических и авиационных системах и для критических инфраструктур, в том числе военного и промышленного назначения ([7-14]).

2. Подходы к обеспечению функциональности надежности ПТС

Наиболее весомые практические результаты по обеспечению надежности и качества получены в ходе проекта с МНИИПА (1975-1987) в рамках ВПК. Под руководством В.В. Липаева ([7-9]) была разработана технология сборочного программирования и обеспечения высокого качества создаваемых ПТС для авиационной, космической, морской

отраслях промышленности и систем специального военного назначения ([15-17]). При этом сформировалась методология обеспечения качественных ПТС:

- тестирование каждого ТС и систем из них на тестовых наборах и сбора сведений об ошибках;
- интеграция, сборка отдельных связанных системных и прикладных ресурсов в систему и проведения интеграционного тестирования;
- оценивание надежности и качества созданных специализированных ПТС для применения в авиационной, морской и космической областях;
- анализ рисков ПТС и отказов аппаратных средств бортовых компьютеров и проведение мероприятий по выпуску высоконадежной и качественной новой продукции.

В критических системах результаты обработки информации и управляющие воздействия определяют работоспособность и качество применения сложных систем в чрезвычайных ситуациях, военными силами, космическими объектами, атомных электростанций и т.п., где происходят аварии и катастрофы вследствие недостаточной безопасности функционирования программных средств. Гибнут спутники и самолеты: Марс 1 (1976), Боинг (2018) и др.

Как правило, ошибки возникают вследствие простых ошибок и дефектов в программах, ПО. Часто это связано с жизнью и здоровьем людей и большими материальными потерями. *Информационная безопасность* определяется защищенностью от неслучайных воздействий. Ее можно рассматривать как преднамеренное ухудшение характеристик функционирования системы, которые специально могут искажать информацию Поэтому требуется защищать программы и данные от злоумышленников с помощью систем защиты криптографии и аутентификации данных и пользователей.

Функциональная безопасность зависит от отказов, влияющих на работоспособность и выполнение основных функций, при выполнении которых могут быть дефекты в аппаратуре, программах и данных. Безопасность функционирования определяется:

- техническими отказами и искажениями информации;
- случайными сбоями и разрушениями информации;
- дефектами и ошибками в программах обработки информации и данных;
- недостатками в средствах обнаружения отказов и восстановления работоспособности систем, программ и данных.

Понятия и характеристики функциональной безопасности систем близки надежности. В надежности учитываются все отказы, а в функциональной безопасности только те, которые приводят к катастрофическим ущербам. Поэтому методы и средства функциональной безопасности базируются на методах определения надежности функционирования комплексов программ, систем и БД.

В настоящее время проблемы обеспечения безотказности, отказоустойчивости, информационной безопасности реализуются в проектах ИСП РАН. Основные направления исследований — борьба с авариями, катастрофами и кибератаками на объекты критической инфраструктуры и коммуникационных средств Интернет и прикладных систем.

Принимая во внимание статистику инцидентов, аварий и катастроф, вызванных отказами КС, рост интенсивности влияния пассивных и активных факторов внешней среды, такие задачи приобретают все большую значимость для банковских и медицинских комплексов, систем электронной коммерции, ГИС и систем мониторинга окружающей среды, приложений распределенного хранения и обработки данных.

На данный момент разработка компьютерных систем и ПТС основывается на сервисных и системных компонентах в рамках моделей SOA/SCA с использованием Web-сервисов и Web-служб Semantic Web и Client-Server Architecture глобальной сети Интернет, как среды взаимодействия разных компонентов Web-систем. Основу технологий КС и ПТС составляет гарантоспособность, надежность, основанная на следующих положениях:

- безотказность (reliability);
- готовность (availability);
- достоверность (high confidence);
- приспособленность к обслуживанию или ремонтопригодность (maintainability);
- информационная безопасность (security) и ее составляющие – конфиденциальность (confidentiality) и целостность (integrity);
- аварийная безопасность (safety).

Теория надежных систем развивается с учетом современных достижений в области элементной базы и технологий моделирования, применения нового математического аппарата теории надежности и конфигурирования сложных ПТС из готовых сетевых и сервисных ресурсов Интернет ([18-21]) и проведения анализа и оценки рисков, устойчивости, прогнозирования неисправностей, отказоустойчивости, защиты, функциональной безопасности, качества систем и др.

3. Моделирование Web-систем из сервисных ресурсов Интернет

3.1 Модели Web-систем

В настоящее время в Интернет среде представлены модели SOA (Service Oriented Architecture) [22] и SCA (Service Component Architecture) [23]], которые образуют используемые для моделирования Web-систем и сайтов из сервисных и готовых reusable компонентов, находящихся в библиотеках и репозиториях Интернет.

SOA – это архитектура из сервисных и системных элементов, которые группируются в среде сервера Интернет с помощью сервисных служб, интерфейсов, входных/выходных данных и портов обмена метаданными, задаваемыми в языке WSDL [24]. Элементы SOA задают описание некоторой функции (Function) с заданным качеством сервиса (Quality service) в IT-стандартах комитета W3C.

SCA – это архитектура из сервисов и компонентов, которые могут быть разработаны различными организациями, как готовые компоненты типа reusable. К ним относятся сетевые сервисы, объекты планирования, доступа к БД и к EJB серверу приложений J2EE. Эта архитектура включает удаленные компоненты повторного использования (КПИ, reuses), которые обмениваются между собой гетерогенными данными из множества общих сетевых Web-сервисов Интернет [25]. Элементы архитектуры могут собираться в систему методом интеграции или конфигурационной сборки требуемых сервисных ресурсов. Данные модели SOA и SCA используются нами при моделировании Web-приложений, Web-систем и сайтов.

3.2 Сервисы, используемые при описании Web-систем

Наряду с компонентным подходом широкое распространение получил сервисный подход. С его помощью реализуется Web-система из готовых системных, функциональных и прикладных компонентов ([17-21], [26]).

Главная идея сервисного подхода Интернет и Semantic Web [27]] состоит в том, чтобы накапливать независимые сервисные компоненты и собирать для разных задач. Функциональные и прикладные сервисы в Интернет – это обычные программные ресурсы, которые реализуют отдельные задачи (в математике, промышленности, экономике, авиации и др.) и накапливаются в Web-библиотеках Интернет. Они обладают способностью к взаимодействию в локальных и/или глобальных сетях Интернет и описываются в языках IDL [28], WSDL [24] и др.

Существуют следующие виды сервисов:

- общие сервисы системных сред (CORBA, DCOM, J2EE, .Net, Appach, JAVA и др.), устанавливают связи с другими сервисами и компонентами через механизмы вызова

RPC, RMI и службы именования, каталогизации и др.;
- сетевые сервисы стандартной модели OSI, API, модели SOA, SCA, которые выполняют обработку пользовательских сервисных ресурсов в сети Интернет;
- готовые программные и сервисные ресурсы (services, artifacts, reuses, assets и др.) пользовательской системы, которые используются как многоразовые КПИ для решения разного рода вычислительных задач, бизнес задач и др. Некоторые из сервисов стали обязательной частью общесистемных средств (VS.Net, IBM, Intel, Linux и др.), а также используются в специальных областях знаний (медицина, биология, генетика, геофизразведка и др.).

3.3 Системные и прикладные сервисы

Применение сервисов производится с помощью клиент-серверной архитектуры. Такая архитектура первоначально была реализована в системе CORBA для управления объектной моделью и ее элементами. Она включает:

- брокер объектных запросов (Object Request Broker — ORB) для взаимодействия клиент-объектов с сервер-объектами на ЯП (Smalltalk, Cobol, Ada-95, Lisp, PL/1, C++, Python, Java, IDLScript и др.);
- общие объектные сервисы (Common Object Services — COS) для управления изменениями, реализациями, транзакциями, подпроцессами и т.п.;
- общие средства обслуживания (Common Facilities — CF) для объединения в различные конфигурации сервисных объектов;
- объектные приложения (Application Objects — AO), над которыми могут производиться операции – открыть, инсталлировать, переместить, поместить, выполнить.

Объект-клиент и объект-сервер обмениваются между собой данными с помощью запросов, каждый из которых обрабатывается брокером ORB на основе описания интерфейсов объектов для клиента, сервера и ядра ORB.

Интерфейс клиента (Client Interface) обеспечивает взаимодействие с объектом-сервера и состоит из трех базовых интерфейсов:

- stub-интерфейс содержит описание внешних параметров и операций в IDL;
- интерфейс динамического вызова (Dynamic Invocation Interface — DII) объекта для выполнения программы клиента при поиске интерфейса в репозитории интерфейсов и программ в репозитории реализаций;
- интерфейс сервисов ORB (ORB Services Interface), содержащий набор сервисных функций, которые клиент запрашивает у сервера через брокер ORB.

Stub-интерфейс обеспечивает взаимосвязь клиента с ORB через операцию удаленного вызова RPC и посылает параметры серверу в запросе. Интерфейс DII обеспечивает доступ объектов и их интерфейсов во время выполнения. В каждом вызове указывается тип объекта, тип запроса и параметры в IDL или WSDL.

3.4 Процесс проектирования Web-систем

Процесс проектирования Web-системы состоит в следующем:

- формирование модели Web-системы;
- определение интерфейсов взаимодействия между сервисными компонентами;
- определение или поиск готовых функциональных сервисов и их размещение в библиотеках Интернет с уникальными именами;
- определение схемы связи сервисных функций или компонентов в ЯП через операторы вызова CALL/RMI/RPC и протоколы связи ISO, в которых задаются входные и выходные параметры;
- использование Web-сервера и Web-клиента для передачи запросов от клиента к серверу

из модели Web-системы.
- верификация и тестирование ресурсов ПТС и проведение метрического анализа рисков, отказов, защиты данных, надежности, информационной безопасности и др.

3.5. Сервисы поддержки Web-систем в Интернет

Для сборки (композиции) сервисов используется инструмент Jopera for Eclipse [29], который выполняет:

- композицию сервисов (типа Agile) и визуальный мониторинг отладки композиций сервисов;
- управление изменением интерфейсов сервиса с помощью сообщений об изменениях в Jopera;
- масштабируемость и автономное исполнение процесса запуска Web-систем с помощью сообщений Jopera.

Jopera содержит набор Eclipse-плагинов для связи различных программных элементов и допускает итеративную композицию сервисов (через маршрутизаторы SOAP и RESTful Web-сервис, Grid-сервисы, Java snipets и др.) после моделирования процессов в сети. Для поиска сервисов по их семантическим описаниям используются Feta Client и Feta Engine [30]. Feta Client – это GUI-плагин системы Интернет Taverna, используемый для описания сервиса, а Feta Engine – для задания Web-сервиса.

Для разработки Web-систем в Semantic Web используются средства:

- RDF стандарта W3C (2004) для описания сетевых, семантических ресурсов и метаданных (данные о данных). Служит каркасом для создания отдельных компонентов семантической паутины.
- RDFS (RDF Schema) — надстройка над RDF, которая позволяет создавать классы и свойства объектов.
- OWL (Web Ontology Language) построен на форматах RDF и RDFS, предназначен для описания онтологий, логики и согласуется с современными сетевыми стандартами.
- SPARQL (Protocol And RDF Query Language) — язык запросов для быстрого доступа к данным RDF для получения необходимой информации из сети.
- RIF (Rule Interchange Format) — формат обмена правилами и др.

WSDL – язык описания входных и выходных данных в запросов Интернет, сервисы которого описываются в языках: WSCI (Web Services Choreography Interface); WSCL (Web Services Conversation Language); BPMN (Business process and model and notation); BPEL (Business Process Execution Language for Web Services) и др.

4. Моделирование устойчивости и безопасности Web-систем

Устойчивость КС определяется их способностью обнаруживать и фиксировать отказы и сбои, часть из которых не было предусмотрено на этапе проектирования прикладных систем. Особенно остро такая задача стоит при построении Web-систем в среде Web- Services и Cloud Computing, которые характеризуются глобальной распределённостью компонентов, гетерогенностью и высокой сложностью. При построении Web-систем предусматривается возможность появления исключительных ситуаций (ошибок, отказов и разного рода сбоев) с помощью методов их обнаружения и анализа.

Создаваемые Web-системы из сервисных и сервисно-компонентных элементов представляют собой композитную структуру из ресурсов Web-сервисов Интернет, обеспечивающую параллельное функционирование компонентов ПТС ([15-17], [25]).

Созданная Web-система должна проверяться на:

- нестабильность функционального состава и структуры Web-системы;
- многовариантность результатов обслуживания и обработки вызовов КПИ средствами

Web-сервиса на корректность и функциональную безопасность при выполнении;
- значение характеристики надежности отдельных компонентов и защиту информации (безотказность, достоверность и др.);
- характеристики оперативности Web-сервисов при обработки запросов и непредсказуемости изменения сетевой задержки для клиента, что обусловлено глобальностью, невысоким качеством сети Интернет;
- сложность применения традиционных средств повышения надежности и оперативности из-за невозможности точного диагностирования возникающих отказов, а также отсутствия достоверной информации о значении надежностных характеристик Web-сервисов и оперативности обслуживания.

Таким образом, сегодняшние устоявшиеся методы и технологии измерения, оценки и прогнозирования характеристик Web-сервисов и Web-систем являются недостаточно мощными в среде Интернет. Это в некоторой степени создает проблемы при прогнозировании характеристик компонентно-интегрируемых сервисно-компонентных Web-систем и синтеза (сборки) таких ПС с требуемыми характеристиками: требуется развитие традиционных методов обеспечения отказоустойчивости и повышения надежности ПТС. Современные Web-системы должны обрабатывать возникающие отказы, сбои и потенциально-опасные события, способствовать повышению качества сервисных ресурсов и обеспечивать функциональную безопасность ПТС в глобальной среде Интернет.

Возможность создавать качественную инфраструктуру Web-сервисной ПТС является критическим условием развития науки, обороны, медицины физических экспериментов и эффективного функционирования предприятий и органов государственного аппарата. Такое видение развития технологий Web-сервисов находится в соответствии с правительственными и международными источниками (программами NEC, NESSI, NECTISE, EPSRC, DTI, EC, FP7 и др.).

Исследование Web-систем проводятся по следующим направлениям:
- адаптация традиционных механизмов обеспечения отказоустойчивости (включая репликацию, восстановление после сбоя, основанное на контрольных точках или коллективной обработки исключений) для сервис-ориентированной архитектуры;
- проведение оценочных испытаний, экспериментальной оценки характеристик надежности;
- анализ Web-сервисов и сервис-ориентированных систем в условиях экстремальных внешних и внутренних воздействий на основе стрессового тестирования и техники «засева» ошибок;
- расширение описания WSDL Web-сервисов за счет включения параметров качества обслуживания QoS и их использования в качестве дополнительных критериев поиска и отбора сервисов.

Концепция сервис-ориентированной архитектуры SOA и сервисно-компонентной архитектуры SCA была предложена для решения проблем обеспечения надежного и безопасного взаимодействия сложных распределенных систем. SOA и SCA обеспечивают построение современных Web-систем на основе слабо связанных программных сервисных модулей (служб) с общедоступными интерфейсами в языке WSDL и специальными механизмами взаимодействия с помощью протокола SOAP (Simple Object Access Protocol) [31]. Описание таких сервисных модулей могут быть обнаружены другими системами в специальных реестрах UDDI (Universal Description, Discovery and Integration), после чего модули могут быть вызваны с помощью SOAP-сообщений в языке XML, передаваемых стандартным интернет-протоколом, таким как HTTP, SMTP, FTP и др.

4.1 Сервисы поддержки надежности Web-систем

При создании и эксплуатации Web-систем в среде облачных вычислений (Cloud Computing) может возникнуть потенциальная уязвимость сервисных ресурсов Интернет к кибер-атакам и информационным вторжениям, что нарушает доступность предоставляемых услуг, целостность и конфиденциальности данных. В связи с этим актуальным является провести исследование механизмов обнаружения исключительных ситуаций, их диагностирование и выявить уязвимости ПО Web-сайтов, систем и Cloud ([12-21], [26-27]).

Главной задачей исследования Web-систем является обеспечение информационной безопасности Web-приложений, Web-систем Cloud Computing, измерение показателей надежности, защита данных Web-систем и облачных Web-систем.

Проверка обработки исключительных ситуаций (exception handling) Web-сервисов в Интернет проводится путем засева ошибок (fault injection) в распределенные Web-системы и облачные системы. Web-услуга воспринимает надежность Web-сервиса через среду коммуникации и функционирования. Распределенная модель взаимодействия, инкапсуляция деталей реализации и отсутствие контроля сервиса Web-услуг значительно сужают номенклатуру воспринимаемых характеристик надежности и сокращают набор доступных показателей для их оценки.

Современная концепция надежности (гарантоспособность) не учитывает вероятностно-временную взаимосвязь между различными вариантами обслуживания, возникновение которой обусловлено асинхронным распределенным характером взаимодействия потребителя и провайдера Web-услуг, а также временными задержками в коммуникационной среде Интернет.

Для достижения высокой надежности и устойчивости Web-систем и сайтов из сервисов требуется знание точных причин и источников возникновения исключительных ситуаций, возникающих во время работы Web-системы и применять наиболее подходящие методы обеспечения отказоустойчивости и восстановления системы после исправления ошибок. К сервис-ориентированным Web-системам можно применять два механизма отказоустойчивости:

- backward error recovery (возврат системы к предыдущему безошибочному состоянию);
- forward error recovery (переход системы в одно из безошибочных состояний).

Это зависит от логики приложения и использования механизмов обработки исключений.

Поскольку возврат к предыдущему безошибочному состоянию не всегда применим к Web-службам из-за того, что большинство из них не хранят своего состояния (т.е. являются stateless), обработка исключений становится наиболее популярным методом обеспечения отказоустойчивости и восстановления после ошибок Web-служб.

В связи с этим необходимо проводить анализ механизмов распространения исключений в Web-системах в среде IBM WebSphere SDK, Apache Axis, Microsoft .Net и др.

Для выполнения такого анализа используется методика засева ошибок/дефектов (fault injection). Засев дефектов является хорошо зарекомендовавшим себя методом оценки надежности и отказоустойчивости вычислительных систем. Несмотря на то, что применение данного метода для исследования распределённых систем в целом является достаточно хорошо изученным, существует определённый дефицит работ в области его применения к Web-сервисам и сервис-ориентированным системам. Важную роль в анализе и оценке надежности Web-сервисов играет тестирование устойчивости Web-систем к некорректным входным параметрам вызова, которые используются при оценке надежности Web-сервисов. В Web-системах могут возникать ошибки (errors), дефекты (faults, bugs), отказы и сбои (failures) в Web-службах.

Отказ/сбой системы диагностируется, когда ошибочный результат распространяется за пределы интерфейса системы. Дефект в КС, например, дефект в ПО, допущенный вследствие

ошибки программиста, может привести к отказу или сбою Web-системы. Обычно различают три группы дефектов КС: дефекты проектирования и разработки, физические неисправности и ошибки взаимодействия (рис.1).

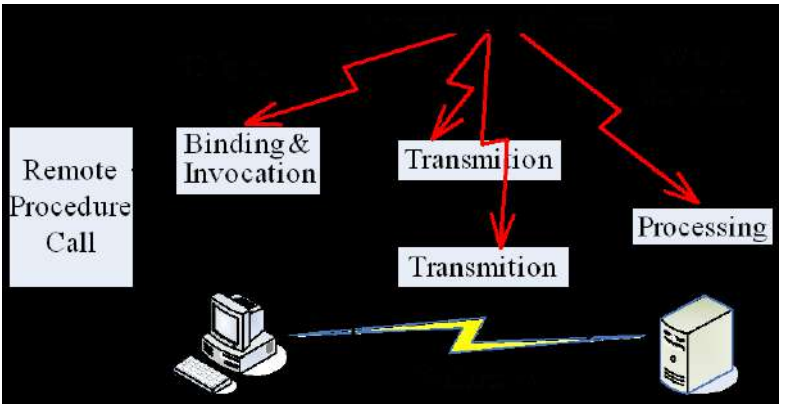


Рис. 1. Потенциальные места возникновения ошибок, отказов и сбоев в сервис-ориентированных Web-системах

Fig. 2. Potential sources of errors, faults, and failures in service-oriented Web-sistems

Опыт показывает, что в Web-сстемах сервисного типа возникают:

- сбои сети и отказы удаленной Web-службы;
- внутренние ошибки и сбои Web-службы;
- ошибки привязки на стороне клиента.

Указанные ошибки/дефекты являются общими и могут проявляться в любой Web-службе во время её работы. К традиционным сетевым отказам и сбоям относятся недоступность службы DNS или же потери и искажения сетевых пакетов. Кроме того, безотказная работа Web-службы зависит от безотказного функционирования ПО, такого как Web-сервер, сервер приложений и система управления базами данных.

Такие сбои могут возникать при принудительном и неожиданном прекращении работы серверов приложений WebSphere, Apache Tomcat и IIS.

Ошибки могут возникать и на стороне клиента при раннем связывании или вызове динамического интерфейса (Dynamic Invocation Interface). Например, «Ошибка в пространстве имен целей», «Ошибка в имени Web-службы» и т.д. происходят из-за изменений параметров вызова и/или несоответствий между WSDL описанием Web-службы и фактическим интерфейсом вызова. Сбои и отказы самих Web-служб могут быть связаны с программными и системными ошибками времени выполнения (run-time errors), которые генерируют пользовательские или системные исключения. Ошибки времени выполнения, такие как «переполнение стека» или «нехватка памяти», приводят к исключениям на уровне системы в целом. Исключение, возникающее вследствие выполнения операции типа «Деление на ноль» также перехватывается и генерируется на системном уровне.

Типичными примерами ошибок времени выполнения приложений являются «Несоответствие типа операнда» или «Выход индекса за пределы массива».

Ошибки связывания сервисов являются, по сути, набором тестов робастности и реализуются на стороне клиента с помощью передачи Web-сервису ошибочных значений параметров вызова. Трассировку и документирование появления исключительных ситуаций можно сделать с помощью стандартного инструментария Java или C#.

В качестве технологий реализации Web-сервисов (конкретной реализации прикладного программного интерфейса для создания Web-служб, а также сервера приложений для их развертывания) предлагается использовать инструменты:

- IBM WSDK + WebSphere;

- Apache Axis + Tomcat;
- Apache Axis + Glassfish;
- Microsoft .Net + IIS.

В качестве интегрированной среды разработки могут быть использованы Eclipse IDE, Netbeans IDE (для Java Web-сервисов), а также Microsoft Visual Studio (для .Net Web-сервисов).

4.2 Исследование уязвимостей NVD и CVE Баз данных

Созданная Web-система из готовых сервисных ресурсов и компонентов может работать с БД Интернет, где размещаются большие данные, к которым идет обращение из других систем и сайтов [17-18]. В БД могут содержаться сведения об уязвимостях к кибератакам, информационным вторжениям, нарушения доступности к сервисным услугам, целостности и конфиденциальности данных. Требуется провести обнаружение аварийных ситуаций, диагностирование выявленных исключительных ситуаций и уязвимостей в Web-системе. Причиной возникновения неадекватных ситуаций могут быть:

- сбои в сети и ошибки удаленной службы при выполнении операторов link;
- сбои и ошибки внутреннего типа при вычислении некоторого сервиса компонента;
- ошибки взаимодействия между клиентом и сервером.

Ошибки на стороне клиента происходят из-за изменений параметров или передаваемых данных и несоответствия типов описания данных в WSDL и/или IDL.

Ошибки сборки (связывания) реализуются на стороне клиента путем передачи ошибочных значений передаваемых данных через параметры. В случае возникновения ошибок передачи данных проводится *трассировка стека исключительных ситуаций* на стороне клиента и Web-системы. Эту трассировку проводит IBM WSDK в процессе взаимодействии с Web-сервисом.

Пример трассировки стека исключений, соответствующего перехвату ошибки («Несовпадение типов операндов») на стороне клиента, использующего реализацию JAX-RPC от Sun Microsystems привеведен на рис.2.

```
java.rmi.ServerException: JAXRPC.TIE.04:
Internal Server Error
   (JAXRPCTIE01: java.lang.NumberFormatException:
   For input string: "578ER")
at com.sun.xml.rpc.client.dii.BasicCall.invoke(BasicCall.java:497)
at ai.c1.xai12.wstest.InvoceWS.invoce(InvoceWS.java:125)
at ai.c1.xai12.wstest.InvoceWS.invoceByVector(InvoceWS.java:75)
at wstest.Main.main(Main.java:42)
```

Рис. 2. Пример трассировки стека исключений
Fig. 2. Exception stack trace example

Информацию об уязвимости получается из общедоступных источников или специализированных БД уязвимости (БДУ), которые предоставляют информацию об уязвимости, возникшей в связи с атаками, нарушениями несанкционированного доступа к защищенным данным БД и др. Поставщиком уязвимостей в БД является общий словарь CVE в виде XML-формата или SGL-дампов (www.osvdl.org). На основе обнаруженных ошибок, исключительных ситуаций и отказов проводится оценка надежности ПО Web-систем, изготовленных из сервисных и компонентных ресурсов Интернет средствами Web-служб.

5. Заключение

В работе представлен подход к построению Web-систем из сервисных и сервисно-компонентных ресурсов, содержащихся в Web-services, в библиотеках и хранилищах

Интернет с обеспечением надежности ПТС в глобальной среде Интернет. Описываются стандартные сервисные ресурсы SOA, SCA, SOAP для создания ПТС для некоторой предметной области знаний. Эти сервисные ресурсы отрабатываются в средах: IBM WSDK + WebSphere, Apache Axis + Tomcat, + Glassfish; Microsoft .Net + IIS.

Приводится классификация возникающих аварийных, исключительных ситуаций и проведения трассировок уязвимостей (атак, запрещенных доступов в БД и др.). Описаны методы моделирования ПТС из готовых ресурсов Интернет и подходы к обеспечению отказоустойчивости, безопасности, защиты и оценки надежности и безопасности сервисных ресурсов и Web-систем с учетом собранных сведений об ошибках и исключительных ситуациях.

Список литературы / References

[1]. Ushakov I. Is Reliability Theory still alive? Reliability: Theory & Applications, vol. 2, no 1, 2007, pp. 6-19.

[2]. Laprie J.C. Dependability: Basic Concepts and Terminology. Springer, 1992, 245 p.

[3]. Information Technology Security Evaluation Criteria (ITSEC): Preliminary Harmonised Criteria. Document COM(90) 314, Version 1.2. Commission of the European Communities, 1991.

[4]. Avizienis A., Laprie J.-C., Randell B., Landwehr C., Dobson I.E Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Trans. on Dependable and Secure Computing, vol. 1, no. 1, 2004, pp. 11-33.

[5]. Chan P.P.W., Lyu M.R., Malek M. Making Services Fault Tolerant. Lecture Notes in Computer Science, vol. 4328, 2006, pp. 43–61.

[6]. Tartanoglu F., Issarny V., Romanovsky A., Levy N. Coordinated Forward Error Recovery for Composite Web Services. In Proc. of the 22nd Symposium on Reliable Distributed Systems (SRDS), 2003, pp. 167-176.

[7]. Липаев В.В. Надежность программного обеспечения. М., СИНТЕГ, 1998, 231 стр. / Lipaev V.V. Reliability of the software. M., SINTEG, 1998 г., 231 p. (in Russian).

[8]. Липаев В.В. Методы обеспечения качества крупномасштабных программных систем. М., СИНТЕГ, 2003 г., 510 стр. / Lipaev V.V. Quality assurance methods for large-scale software systems. M., SINTEG, 2003, 510 p. (in Russian).

[9]. Липаев В.В. Надежность и функциональная безопасность комплексов программ реального времени. Москва, Светлица, 2013 г., 193 стр. / Lipaev V.V. Reliability and functional safety of real-time software complexes. Moscow, Svetlitsa, 2013, 193 p. (in Russian).

[10]. Андон Ф.И., Коваль Г.И. и др. Основы инженерии качества программных систем. Киев, Наукова думка, 2007 г., 670 стр. / Andon F.I., Koval G.I. et al. Fundamentals of quality engineering of software systems. Kiev, Naukova Dumka, 2007, 670 p. (in Russian).

[11]. Горбенко А.В., Засуха С.А. и др. Безопасность ракетно-космической техники и надежность компьютерных систем. Авиационно-космическая техника и технология, no. 1(78), 2011 г., стр. 9–20 / Gorbenko A.V., Zasukha S.A. and other Safety of rocket and space technology and reliability of computer systems. Aerospace engineering and technology, no. 1 (78), 2011, pp. 9–20.

[12]. Лаврищева Е.М., Пакулин Н.В., Рыжов А.Г., Зеленов С.В. Анализ методов оценки надежности оборудования и систем. Практика применения методов. Труды ИСП РАН, том 30, вып.3, 2018 г., стр. 99-120 / Lavrischeva E.M., Pakulin N.V., Ryzhov A.G., Zelenov S.V. Analysis of methods for assessing the reliability of equipment and systems. Practice of methods. Trudy ISP RAN/Proc. ISP RAS, том 30, вып. 3, 2018 г., стр. 99-120 (in Russian). DOI: 10.15514/ISPRAS-2018-30(3)-8.

[13]. Lavrischeva E.M., Mutilin V.S., Ryzhov A.G. Designing variability models for software, operating systems and their families. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 5, 2017, pp. 93-110. DOI: 10.15514/ISPRAS-2017-29(5)-6.

[14]. Тарасюк О.М., Горбенко А.В. Безопасность и устойчивость Веб- и облачных систем. Практикум. Министерство образования и науки Украины, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», 2017 г., 40 стр. / Tarasyuk O.M., Gorbenko A.V. Security and Resilience of Web and Cloud Systems. Workshop. Ministry of Education and Science of Ukraine, National Aerospace University «Kharkiv Aviation Institute», 2017, 40 p. (in Russian).

[15]. Лаврищева Е.М., Грищенко В.Н. Связь разноязыковых модулей в ОС ЕС ЭВМ. Москва, Финансы и статистика, 1982 г., 137 стр. / Lavrischeva E.M., Grishchenko V.N. Linking multilingual modules in the OS of ES computers. Moscow, Finance and Statistics, 1982, 137 p. (in Russian).
[16]. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Киев, Наукова думка, 1991 г., 282 стр. / Lavrischeva E.M., Grishchenko V.N. Assembly programming. Kiev, Naukova Dumka, 1991, 282 p. (in Russian).
[17]. Липаев В.В., Позин Б.А., Штрих А.А. Технология сборочного программирования. М., Радио и связь, 1992 г., 287 стр. / Lipaev V.V., Pozin B.A., Shtrikh A.A. Assembly programming technology. M., Radio and communication, 1992, 287 p. (in Russian).
[18]. Лаврищева Е.М., Петренко А.К. Моделирование семейств программных систем. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 49-64 / 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.
[19]. Лаврищева Е.М. Рыжов А.Г. Применение теории общих типов данных стандарта ISO/IEC 11404 GDT к Big Data. Евразийский союз ученых, no. 31, 2016 г., стр. 99-110 / Lavrischeva E.M., Ryzhov A.G. application of the theory of general data types standard ISO/IEC 11404 GDT to Big Data. Eurasian Union of Scientists, no. 31, 2016, pp. 99-110 (in Russian).
[20]. Лаврищева Е.М., А.Г. Рыжов. Подход к моделированию систем и сайтов из готовых ресурсов. Труды XX Всероссийской конференции «Научный сервис в сети Интернет», 2018 г., стр. 321-345 / Lavrischeva E.M., Ryzhov A.G. Approach to the modeling of systems and sites from ready resources. In Proc. of the XX All-Russian Conference on Scientific Services on the Internet, 2018, pp. 321-345 (in Russian),
[21]. Лаврищева Е.М., Петренко А.К. Технология сборки интеллектуальных и информационных ресурсов Интернет. Труды XXI Всероссийской конференции «Научный сервис в сети Интернет», 2019 г., стр. 469-488 / Lavrischeva K.M., Petrenko A.K. Technology of Assembly of intellectual and information resources of the Internet, In Proc. of the XXI All-Russian Conference on Scientific Services on the Internet, 2019, pp. 469-488 (in Russian).
[22]. Service-Oriented Architecture Ontology. The Open Group, 2010, 114 p.
[23]. Paik H., Lemos A.L., Barukh M.C., Benatallah B., Natarajan A. Service Component Architecture (SCA). In Web Service Implementation and Composition Techniques. Springer, 2017, pp. 2-3-250.
[24]. Web Services Description Language (WSDL), Version 2.0, Part 1: Core Language. W3C Recommendation, 26 June 2007.
[25]. Боркус В. Web-сервисы: современные стандарты. Аналитический обзор. PCWeek, 2005 г. / Borkus V. Web services: modern standards. Analytical review. PCWeek, 2005 (in Russian),
[26]. Лаврищева Е.М. Software Engineering компьютерных систем. Парадигмы. Технологии. CASE-системы программирования. Киев, Наукова думка, 2013, 280 стр. / Lavrischeva E.M. Software Engineering of Computer Systems. Paradigms. Technology. CASE- programming systems. Kiev, Naukova Dumka, 2013, 280 p. (in Russian).
[27]. Matthews B. Semantic Web Technologies. E-Learning, vol. 6, no. 6, 2005.
[28]. Interface Definition Language. The Object Management Group. URL: https://www.omg.org/spec/IDL, accessed 20.10.2020.
[29]. Jopera for Eclipse. URL: http://www.jopera.ethz.ch, accessed 20.10.2020
[30]. Taverna plugins. URL: http://www.mygrid.org.uk/usermanual1.7/taverna_plugins.html, accessed 20.10.2020
[31]. SOAP Version 1.2, Part 1: Messaging Framework (Second Edition). W3C Recommendation, 27 April 2007.

Информация об авторах / Information about authors

Екатерина Михайловна ЛАВРИЩЕВА – доктор физико-математических наук, профессор, главный научный сотрудник ИСП РАН, профессор МФТИ. Область интересов: надежность и качество, моделирование сложных систем, Web-системы, сборочное программирование.

Ekaterina Mikhailovna LAVRISCHEVA – Doctor of Physical and Mathematical Sciences, Professor, Principal Researcher at ISP RAS, Professor at Moscow Institute of Physics and Technology. Areas of interest: reliability and quality, modeling og complex systems, Web-systems, assembly programming

Сергей Вадимович ЗЕЛЕНОВ – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, доцент кафедры системного программирования ВШЭ. Область интересов: методы проектирования и разработки ответственных систем, формальные методы программной инженерии, методы верификации и валидации, тестирование на основе моделей.

Sergey Vadimovich ZELENOV – Ph.D. in Physics and Mathematics, Senior Researcher at ISP RAS, Associate Professor of System Programming Department at HSE. Areas of interest: design and development methods for critical systems, formal methods of software engineering, verification and validation methods, model-based testing.

DOI: 10.15514/ISPRAS–2020–32(5)–13

Модификация алгоритма Marching Cubes для получения трехмерного представления плоского изображения

[1] *Д.И. Хернандес Фарсас, ORCID: 0000-0002-7435-7753 <di.hernandez@ugto.mx>*
[1] *Р. Гусман Кабрера, ORCID: 0000-0002-9320-7021 <guzmanc@ugto.mx>*
[1] *Т. Кордова Фрага, ORCID: 0000-0002-6486-7530 <theo@_sica.ugto.mx>*
[1] *Х.З. Уамани Луна, ORCID: 0000-0003-1776-9820 <zajorize@gmail.com>*
[2] *Х.Ф. Гомез Агилар, ORCID: 0000-0001-9403-3767 <jfranciscogoma@hotmail.com>*

[1] *Университет Гуанахуато,*
Мексика, 36000, Гуанахуато, ул. Ласкурен де Ретана, 5
[2] *Национальный совет по науке и технологиям,*
Мексика, 03940, Мехико, район Бенито Хуарес, Аллея Повстанцев, 1582

Аннотация. Совмещение трехмерной модели с изображением можно рассматривать как установку визуальных соответствий, извлекаемых из данных, описывающих эти изображения. Эта непростая задача еще более усложняется, если при построении изображений используются разные методы визуализации. В статье представлен подход, позволяющий сопоставлять характеристики, обнаруживаемые в двух различных видах изображений – фотографиях и 3D-моделях – с использованием общего 2D-представления. Наш подход основан на модификации алгоритма Marching Cubes, позволяющей избежать неоднозначных решений, не добавляя вычислений при обработке каждого куба. Мы разделяем идею о решающей важности разделения случаев эквивалентности на два класса. С учетом всех возможных состояний внутри и снаружи в четырех углах одной грани куба имеются только четыре нетривиальных варианта после исключения эквивалентных вариантов путем вращения. Полученные результаты демонстрируют применимость предлагаемой методики.

Ключевые слова: Marching Cubes; трехмерное представление; изоповерхности

Для цитирования: Эрнандес Фариас Д.И., Гусман Кабрера Р., Кордова Фрага Т., Уамани Луна Х.З., Гомез Агилар М. Модификация алгоритма Marching Cubes для получения трехмерного представления плоского изображения. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 167-180. DOI: 10.15514/ISPRAS–2020–32(5)–13

Modification of the Marching Cubes Algorithm to Obtain a 3D Representation of a Planar Image

[1] *D.I. Hernández Farías, ORCID: 0000-0002-7435-7753 <di.hernandez@ugto.mx>*
[1] *R. Guzmán Cabrera, ORCID: 0000-0002-9320-7021 <guzmanc@ugto.mx>*
[1] *T. Cordova Fraga, ORCID: 0000-0002-6486-7530 <theo@_sica.ugto.mx>*
[1] *J.Z. Huamaní Luna, ORCID: 0000-0003-1776-9820 <zajorize@gmail.com>*
[2] *J.F. Gomez Aguilar, ORCID: 0000-0001-9403-3767 <jfranciscogoma@hotmail.com>*

[1] *Universidad de Guanajuato,*
Lascuráin de Retana No. 5, Col. Centro C.P. 36000, Guanajuato, Gto., México
[2] *Consejo Nacional de Ciencia y Tecnología,*
Av. Insurgentes Sur 1582, Alcaldía Benito Juárez, C.P. 03940, Ciudad de México

Abstract. The registration of a 3D model over an image can be seen as the alignment of visual correspondences extracted from these two data. This is a challenging task and it is even more complex when the two images have a different modality. This paper introduces an approach that allows matching features detected in two different modalities: photographs and 3D models, by using a common 2D representation. Our approach is based on a modifcation of the Marching Cubes algorithm aiming to remove ambiguous cases without adding further calculations in each cube. We share the idea about the crucial importance of splitting the equivalence cases into two classes. Considering all the possible states inside/outside in the four corners of a cube side, indeed, there are only four non-trivial cases after eliminating those equivalences through the rotation. The obtained results allow us to validate the feasibility of the proposed methodology.

Keywords: marching cubes; 3D representation; Iso-surfaces

For citation: Hernández Farías D.I., Guzmán Cabrera R., Cordova Fraga T., Huamaní Luna J.Z., Gomez Aguilar J.F. Modification of the marching cubes algorithm to obtain a 3D representation of a planar image. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020. pp. 167-180 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–13

1. Введение

Важность объемной визуализации как инструмента анализа данных привела к разработке нескольких методов, позволяющих создавать проекции на основе двухмерных изображений. Методы визуализации подразделяются на три основные категории: мультипланарный рендеринг (multi-planar rendering), объемный рендеринг (volume rendering) и поверхностный рендеринг (surface rendering) [1]. В каждая из этих категорий имеется своя техника реконструкции изображения.

Мультипланарный рендеринг позволяет визуализировать оттенки серого в произвольных поперечных сечениях на основе пространственных данных. *Объемный рендеринг* используется при генерации изображения с использованием рейкастинга (ray casting, бросание лучей). *Поверхностный рендеринг* обеспечивает визуализацию заданного объекта на основе данных, представленных в виде набора базовых элементов (вокселей, voxel), которые определяют границы заданных структур.

Существуют различные алгоритмы поверхностного рендеринга, в число которых входит алгоритм *Marching Cubes* (MC, алгоритм *шагающих кубиков*), широко распространенный в таких областях, как биология, биохимия, биомедицина, экология и т.д. [2, 3]. MC может успешно применяться для генерации трехмерных медицинских изображений на основе нескольких источников, таких как компьютерная томография (КТ) или магнитно-резонансная томография (МРТ).

Реконструкция медицинских изображений может способствовать развитию межплатформенных систем, которые можно использовать для улучшения клинической

диагностики, а также для выполнения точных хирургических операций с низким уровнем риска [1, 4].

В данной работе мы представляем модифицированную версию алгоритма MC, поддерживающей генерации предельно детализированных изображений. Работа затрагивает следующие аспекты визуализации.

- Редактирование небольших участков поверхности без потребности в реконструкции всей поверхности, содержащейся на изображении.
- Генерация сетки с как можно меньшим числом искажений между треугольниками или областями.
- Оптимизация использования имеющихся ресурсов хранения данных путем совершенствования процесса генерации поверхности. Это достигается за счет обеспечения наличия у кубов общей.

Статья имеет следующую структуру. В разд. 2 представлен краткий обзор литературы по данной тематике. В разд. 3 описан наш подход, основанный на модифицированном алгоритме MC. В разд. 4 обсуждаются проведенные эксперименты, а также полученные результаты. В заключение, в разд. 5 представлены выводы и указаны направления будущих исследования.

2. Родственные работы

Алгоритм MC был впервые предложен Уильямом Э. Лоренсеном (William E. Lorensen) и Харви Э. Клайном (Harvey E. Cline) [5]. Это один из наиболее широко используемых методов поверхностного рендеринга, который считается надежным и простым. Более подробно с данным алгоритмом можно ознакомиться в [3]. В *алгоритме MC* используется подход «разделяй и властвуй» для последовательной обработки пространственных данных с использованием *вокселей*, эквивалентных *кубам*. С каждым кубом ассоциируется изоповерхность с изозначением h (задаваемым в качестве входного аргумента алгоритма), на которой генерируется триангуляционная сетка путем определения того, как куб пересекается с этой поверхностью, после чего алгоритм переходит к следующему кубу. Пересечение поверхностей с кубами находится путем сравнения значений их вершин с h. Если значение вершины больше или равно h, то вершина считается внутренней. В противном случае она считается внешней. Имеется 256 возможных способов пересечения поверхности с кубом. Это число можно сократить из-за того, что некоторые пары вариантов пересечения являются инверсными или симметричными, оставив лишь 15 вариантов, показанных на рис. 1. После определения типа пересечения выполняется построение триангуляционной сетки.

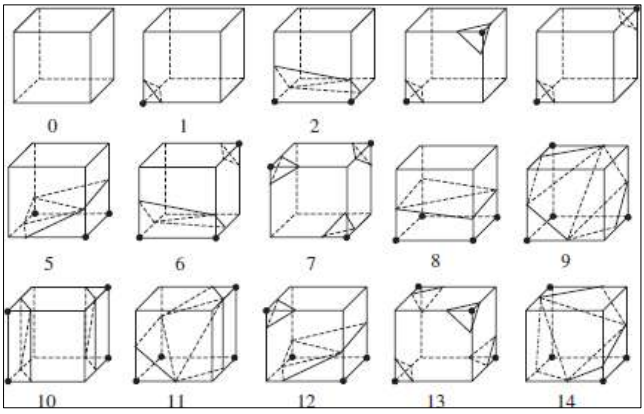


Рис. 1. Пятнадцать вариантов пересечения, учитываемых в традиционном алгоритме Marching Cubes

Fig. 1 Fifteen intersection cases considered in the traditional Marching Cubes algorithm

Алгоритм MC применялся для решения широкого спектра задач, и при этом были выявлены различные возможности для его усовершенствования. Кроме того, важно отметить, что у алгоритма шагающих кубиков есть некоторые недостатки, такие как вероятность появления видимых артефактов огранки, сложность выбора изозначения для обеспечения корректной аппроксимации поверхности, потеря информации при генерации небольших структур и т.д.

В некоторых описанных в литературе исследованиях [3-5] в основном изучались следующие аспекты:

(i) топология;

(ii) избыточность и неоднозначность при генерации сетки;

(iii) тип данных и скорость.

В [6] Дарст (Durst) описывает проблему наличия нескольких способов триангуляции для вариантов пересечения в алгоритме MC. Неоднозначность выбора интерполяции рассматривается в работе [7]. В [8] впервые было предложено расширить набор вариантов алгоритма MC. Позднее для устранения внутренних неоднозначностей авторы [9] предложили расширить набор вариантов триангуляции до 33 (этот вариант MC принято называть *Marching Cubes 33 – MC33*). Расширенная версия *MC33* представлена в [10]. Предлагаемый метод предусматривает наличие дополнительных меток узлов, позволяющих избежать вырождения треугольников. Одним из важнейших усовершенствований алгоритма является возможность сохранения топологии трилинейной интерполяции. Авторы экспериментируют как с реальными, так и со случайно сгенерированными наборами данных. Сравнение производительности предложенной версии с другими алгоритмами, основанными на MC, позволяет заострить внимание на их различиях, относящихся к топологическим проблемам в генерируемой сетке.

Одной из областей, нуждающихся в развитии технологий 3D-реконструкции, является медицина. В [11] представлен подход для 3D-реконструкции в медицинской области, сочетающий в себе алгоритм MC с алгоритмом начального заполнения (seed-occupying algorithm). Для генерации изоповерхности вместо треугольников используются многоугольники. Затем уменьшается число граней, а также смягчается проблема неоднозначности. Результаты экспериментов на реальных медицинских КТ-снимках человеческого черепа свидетельствуют о преимуществах предложенного подхода по сравнению с традиционным алгоритмом с точки зрения требуемых объема памяти и времени обработки. Лонг (Long) и Нагамуне (Nagamune) в [4] описывают алгоритм MC для реконструкции эндоскопических изображений. В их подходе используется метод выбора средней точки для получения позиции пересечения поверхности, и авторы демонстрирует перспективные результаты. В работе [12] авторы выполняют трехмерную реконструкцию поверхности на основе данных КТ брюшной полости (тканей печени), применяя алгоритм наращивания областей (region growing algorithm) для сегментации изображения, а также методы математической морфологии. В предлагаемом подходе построение контура поверхности производится аналогично стандартному алгоритму MC, однако пересечение выполняется только по пикселям со значением 1 бинаризованных изображений. Кроме того, предлагается усовершенствование алгоритма, позволяющее уменьшить число избыточных операций, выполняемых в ходе построения. Наконец, в работе [1] представлен обзор различных методов трехмерной реконструкции на основе алгоритма MC, применяемых в области черепно-лицевой хирургии. В этих методах применяется параллельный обход, а также сокращается число фрагментов изоповерхности (iso-surface patches).

Как и в других областях, связанных с вычислениями, улучшению методов объемной визуализации способствовали продвижения в разработке более мощного графического вычислительного оборудования (такого как графические процессоры, GPU). В [2] описан подход к ускорению алгоритма MC на основе использования GPU. Кроме того, авторы также

обсуждают различные способы повышения производительности MC с помощью применения вспомогательных пространственных структурах данных. Некоторые эксперименты, проведенные над различными наборами пространственных данных, подтвердили эффективность предложенной модели. Предложенный метод демонстрирует существенную разницу в скорости рендеринга – в 18 раз – по сравнению подходом, основанном на использовании только обычных процессоров.

3. Наш подход

3.1 Алгоритм Marching Cubes

Предлагаемые модификации алгоритма MC направлены на устранение неоднозначных вариантов без потребности в дополнительных вычислениях в каждом кубе. Мы разделяем идею о решающем значении разделения эквивалентных вариантов на два класса [13].

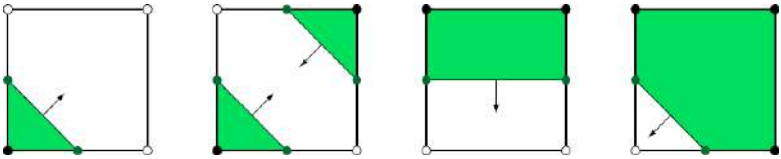


Рис. 2. Нетривиальные конфигурации, допустимые в предложенных модификациях алгоритма MC. Зеленая область обозначает сплошное пространство. Стрелки представляют собой направление нормали поверхности по отношению к внешней части ребер

Fig. 2. Non-trivial configurations allowed by the proposed modifications to the MC algorithm. The green region represents the solid space. The arrows represent the surface normal direction towards outside in the edges

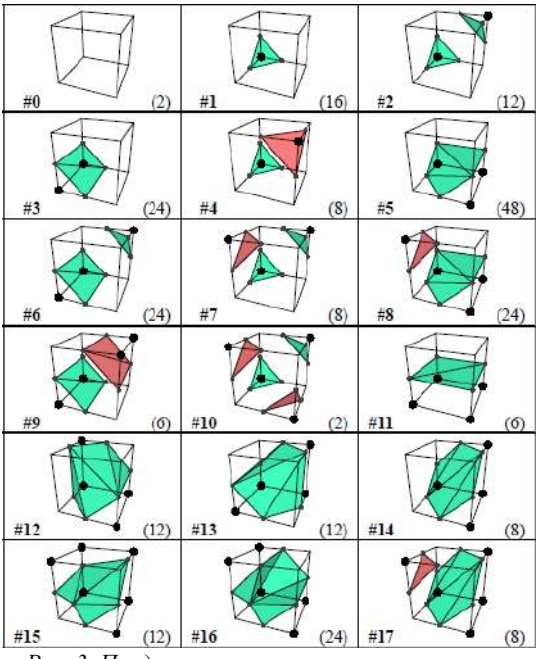


Рис. 3. Предлагаемые классы эквивалентности

Fig. 3. Equivalence classes proposed

С учетом всех возможных расположений углах одной стороны куба внутри и снаружи изоповерхности можно установить, что в действительности существуют только четыре нетривиальных варианта после исключения эквивалентных вариантов путем вращения.

Требование *пространственного соответствия ребер* выполняется, когда все вершины соединены на смежных гранях, разделяющих нижний угол.

Далее, в соответствии со стратегией *предпочтительной полярности* (preferred polarity), четыре варианта, показанные на рис. 2, являются нетривиальными конфигурациями ребер, допустимыми в нашем подходе [14]. Черные точки обозначают внутренние вершины, белые – внешние [13].

Мы предлагаем разрешать эту проблемную ситуацию, устраняя инверсии отношения эквивалентности для классов, порождающих неоднозначность, и оставляя только вращение. Для вариантов с четырьмя внутренними углами на поверхности изображения мы образуем три новых класса. Образуется набор из 18 классов эквивалентности, показанных на рис. 3. Классы 15, 16 и 17 являются новыми классами, состоящими из инверсного представления классов 2, 6 и 7 и их соответствующих вращений [15].

В каждой ячейке таблицы число в левом нижнем углу означает номер класса, а число в правом нижнем углу – количество вариантов, относящихся к данному классу эквивалентности, из 256 всех вариантов. Черная точка в углу обозначает, что он находится в пределах объема изображения, а угол без точки обозначает, что он находится снаружи. Зеленые треугольники находятся перед гранью куба, а красные треугольники – позади материала куба [16].

В идеале, число вариантов триангуляции при переходе от одного куба к другому должно быть одного и того же порядка. Для достижения этого можно разделить каждый переходной блок на два блока меньшего размера, как показано на рис. 4. Таким образом, переходный куб делится на части максимального и среднего разрешения плоскостью, параллельной грани. Левый блок (максимального разрешения) триангулируется с помощью выборочных значений с его граней. Значения углов *A, B, C* и *D* дублируются на противоположной грани куба. Затем производится переход внутри куба, что позволяет использовать более контролируемые данные. С другой стороны, в правом блоке триангуляция производится обычным способом с использованием данных среднего разрешения [17].

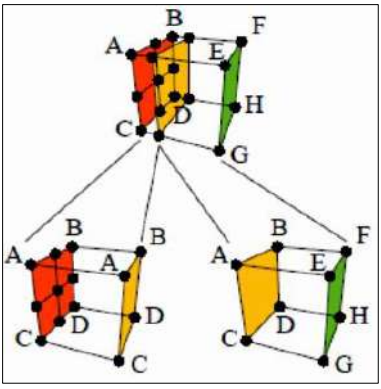


Рис. 4. Переходный куб разделяется на два блока меньшего размера. Левый куб триангулируется на основе выборочных значений с грани максимального разрешения, а правый – по традиционному алгоритму MC

Fig. 4. A transition cube is divided into two smaller blocks. The left side cube is triangulated by exploiting sample values coming from the maximum resolution face, while the one in the right is done with the traditional MC algorithm

3.2 Предлагаемая методика

На рис. 5 схематично показана методика, предлагаемая в нашей работе. Как можно видеть, задается двухмерное изображение, из которого нужно получить трехмерное представление. Это изображение может быть, например, медицинским снимком. Представленные в нашей

статье результаты получены при использовании в качестве исходного изображения снимка МРТ. Ниже приводится описание каждого из этапов методики.

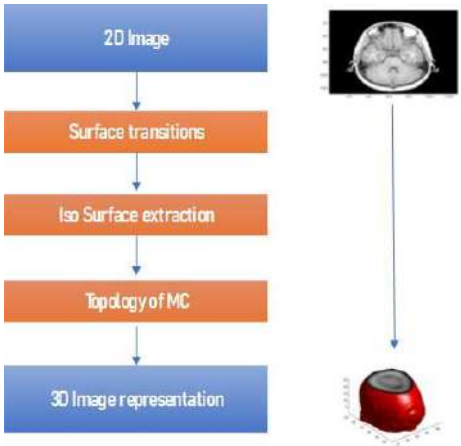


Рис. 5 Предлагаемая методика
Fig. 5. Proposed methodology

3.2.1 Переходы кубов на поверхности

Сетки разделяются блоками кубов, соединенных со следующим уровнем восьмью кубами. Позиции вторичных вершин вычисляются линейным преобразованием внутренних позиций граничного куба таким образом, что полные кубы масштабируются до меньшего размера в зависимости от их расположения [18]. Для этого для координат (x, y, z) вычисляются $(\Delta x, \Delta y, \Delta z)$ с использованием формулы (1):

$$\Delta x = \begin{cases} (1 - 2^{-k}x)w(k) & if \quad x < 2^k \\ 0 & if \quad 2^k \le x \le 2^k(s-1) \\ (s - 1 - 2^{-k}x)w(k) & if \quad x \le 2^k(s-1) \end{cases}. \tag{1}$$

Здесь k – значение уровня детализации (Level of Detail – LOD), а s – размер блока. Функция $w(k)$ задает ширину переходов куба с LOD равным k. В нашем подходе она определяется как $w(k) = 2^{k-2}$. Δy и Δz вычисляются аналогично. Вершины на гранях полного разрешения переходных кубов остаются неизменными. Добавление смещений приводит к тому, что в области перехода, как показано на рис. 6, поверхность становится настолько плоской, что могут возникать деформации.

Эти проблемы могут быть устранены путем проецирования вектора смещения вершины на касательную плоскость, проходящую через исходное положение вершины, с использованием формулы:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 1 - N_x^2 - N_xN_y - N_xN_z \\ -N_xN_y \; 1 - N_y^2 - N_yN_z \\ -N_xN_z - N_yN_z \; 1 - N_z^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}, \tag{2}$$

где N – вершина блока нормальной длины.

На рис. 6-(a) к вершинам низкого разрешения применяется линейное преобразование в соответствии с (1), чтобы освободить место для переходного куба. Это может привести к нежелательным последствиям при создании плоской области и вогнутости. Для решения этих проблем применяется формула (2) для проецирования разности вершин на касательную плоскости относительно нормальной вершины N.

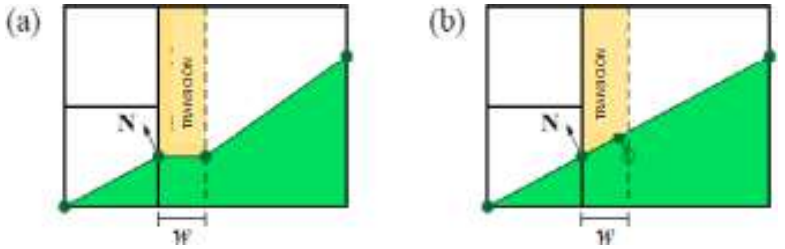


Рис. 6. Различные эффекты проекций
Fig. 6. Different effects of projections

При выборе первичной и вторичной вершин принято использовать основную вершину в качестве позиции углов, общих для блоков разного уровня детализации.

3.2.2 Извлечение изоповерхностей

Основной целью является построение точного контура заданного трилинейного скалярного поля по следующему параметрическому описанию:

$$s(x,y,z) = (1-x)(1-y)(1-z)c_{000} + (1-x)(1-y)zc_{001} + (1-x)y(1-z)c_{010} + (1-x)yzc_{011} + x(1-y)(1-z)c_{100} + x(1-y)zc_{101} + xy(1-z)c_{110} + xyzc_{111}, \quad (3)$$

где $c_{ijk}\ (i,j,k)\ \in\ [0,1]$ – скалярные значения в вершине заданного куба.

M\ Контур задается фиксированным пространством τ, состоящим из всех точек в $(x,y,z)^T$, преобразованных в соответствии с

$$s(x,y,z) = \tau. \quad (4)$$

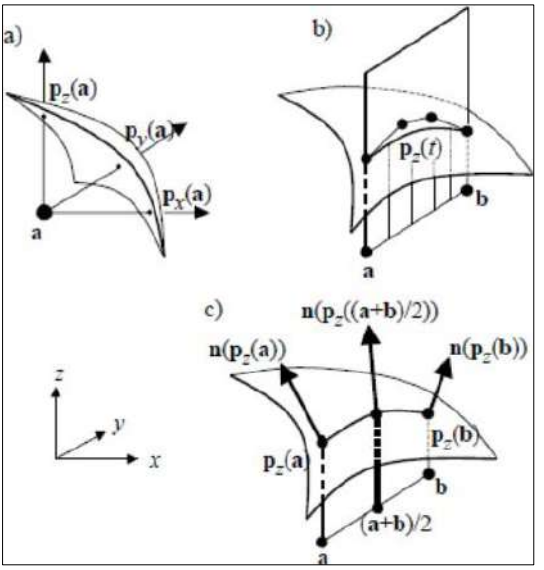


Рис. 7. Проекции a на контур в направлениях x −, y − и z −
Fig. 7 Projections of a over the contour in x −, y −, and z − directions

При рассмотрении точки $a = (x_a, y_a, z_a)^T$ нам было бы интересно найти пересечение заданной линии с контуром, определенном формулами (3) и (4), поскольку требуется решить кубическое уравнение [19].

В частном случае, когда имеется линия, параллельная одной из осей координат, задачу можно упростить до решения линейного уравнения. Пусть $p_x(a)$ является пересечением линии $a\ +$

$\lambda(1,0,0)^T$ с контуром, определяемым соотношениями (3) и (4), $p_y(a)$ – пересечением линии $a + \lambda(0,1,0)^T$ с этим контуром, и $p_z(a)$ – пересечением линии $a + \lambda(0,0,1)^T$. Тогда $p_x(a)$, $p_y(a)$ и $p_z(a)$ – это проекции a на контур по направлениям $x-$, $y-$ и $z-$. Это означает, что можно построить три точки на контуре для заданной точки простым образом. На рис. 7 приводится пример. На рис. 7a представлены проекции $p_x(a)$, $p_y(a)$ и $p_z(a)$ для заданной точки в направлении контура $x-$, $y-$, $z-$. Как видно из рис. 7b, проекция $p_z(t)$ отрезка линии $(1-t)a + tb$ на контуре является рациональной кубической кривой. Наконец, конфигурация для расчета точки привязки $p_z(t)$ показана на рис. 7c.

Имея точку a, которая служит для построения контура, определяемого формулой (3), можно вычислить его нормаль – вектор $n(a)$ по формуле:

$$n(a) = \left(s_x(x_a, y_a, z_a), s_y(x_a, y_a, z_a), s_z(x_a, y_a, z_a)\right)^T, \quad (5)$$

где s_x, s_y и s_z являются частичными производными s, определяемого формулой (3). Кривые контура строятся по проекциям отрезков линий.

Расширяя понятие контурных кривых до параметрических поверхностей в контуре, определенном с формулами (3) и (4), можно получить треугольник:

$$x(\mu, \nu, \omega) = \mu a + \nu b + \omega c. \quad (6)$$

Координатами барицентра являются вершины a, b и c. Положим $\mu + \nu + \omega = 1$. Тогда $p_x(x(\mu,\nu,\omega))$, $p_y(x(\mu,\nu,\omega))$ и $p_z(x(\mu,\nu,\omega))$ являются проекциями x на контур, определенный с помощью формул (3) и (4). См. рис. 8.

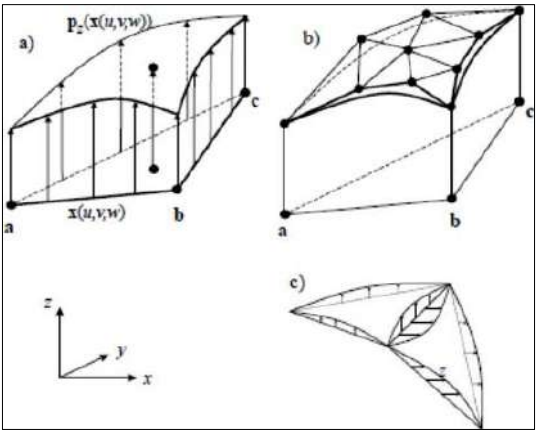
1. 

Рис. 8. Проекция $p_z(x(\mu,\nu,\omega))$. (a) значение $p_z(x(\mu,\nu,\omega))$ получается путем проецирования каждой из формулы (6) в направлении $z-$ в контуре, определенном формулами (3) и (4). (b) $p_z(x(\mu,\nu,\omega))$ является рациональной кубической поверхностью. (c) два смежных треугольника, образованных алгоритмом MC в разных направлениях

Fig. 8. Projections of $p_z(x(\mu,\nu,\omega))$. (a) the value of $p_z\big(x(\mu,\nu,\omega)\big)$ is obtained by means of the projection of each point of Eq. (6) in the $z-$ direction in the contour defined by Eq. (3) and 4. (b) $p_z\big(x(\mu,\nu,\omega)\big)$ is a rational cubic surface. (c) two adjacent triangles generated by MC in different directions

Поверхности $p_x(x(\mu,\nu,\omega))$, $p_y(x(\mu,\nu,\omega))$ и $p_z\big(x(\mu,\nu,\omega)\big)$ являются рациональными кубическими поверхностями. Например, $p_z\big(x(\mu,\nu,\omega)\big)$ можно описать как треугольную рациональную поверхность Безье:

$$p_z\big(x(\mu,\nu,\omega)\big) = \frac{\sum_{i+j+k=3} \omega_{ijk} b_{ijk} B^3_{ijk}(\mu,\nu,\omega)}{\sum_{i+j+k=3} \omega_{ijk} B^3_{ijk}(\mu,\nu,\omega)}, \quad (7)$$

где B^3_{ijk} — многочлены [20] и

$$
\begin{gathered}
w_0 = z_{n(p_z(a))}, \\
w_1 = \frac{4}{3} z_{n\left(p_z\left(\frac{a+b}{2}\right)\right))} - \frac{1}{3} z_{n(p_z(b))}, \\
w_2 = \frac{4}{3} z_{n\left(p_z\left(\frac{a+b}{2}\right)\right)} - \frac{1}{3} z_{n(p_z(a))}, \\
w_3 = z_{n(p_z(b))}, \\
b_0 = p_z(a), \\
b_3 = p_z(b).
\end{gathered} \tag{8}
$$

Значения $w_0, \ldots, w_3$ определяются координатами $z-$ некоторых нормальных векторов контура. Например, w_0 определяется компонентом $z-$ нормального вектора $p_z(a)$. Кривые относительно $p_x(t)$ и $p_y(t)$ можно получить аналогичным образом.

В базовом случае для получения контура (определенного с помощью формул (3) и (4)), нежно воспользоваться алгоритмом MC вместе с проекциями каждого исходящего треугольника в контуре. К сожалению, это может привести к проблемам в смежных треугольниках из-за направлений проекции. Чтобы избежать этого, все поверхности срезов быть треугольными рациональными кубическими, а не треугольными [19].

Треугольник (a, b, c), полученный с помощью алгоритма MC представляет собой контур с учетом того, что требуется:

1. определить проекции в направлениях $q_{ab}, q_{bc}, q_{ca} \in x, y, z$ на краевых кривых;
2. определить проекцию в направлении q_{abc} всего треугольника.
3. спроецировать границы треугольника (a, b, c) по контуру в соответствии с (1); границами треугольника (a, b, c) являются кривые x_{ab}, x_{bc} и x_{ca}:

$$
\begin{aligned}
x_{ab}(t) &= p_{q_{ab}}((1-t)a + tb) \\
x_{bc}(t) &= p_{q_{bc}}((1-t)b + tc) \\
x_{ca}(t) &= p_{q_{ca}}((1-t)c + ta)
\end{aligned}
$$

4. спроецировать x_{ab}, x_{bc} и x_{ca} в направлении q_{abc} на плоскость, определенную a, b и c; для получения кривых y_{ab}, y_{bc} и y_{ca} на плоскости (a, b, c) используются краевые кривые области в окончательном варианте.
5. рассчитать обрезанную поверхность в проекции $p_{q_{ab}}(\mu a, \nu b, \omega c)$ с $\mu + \nu + \omega = 1$; область обрезанной поверхности задается границами y_{ab}, y_{bc} и y_{ca} [21].

3.2.3 Топология Marching Cubes

Мы стремимся определить набор внутренних точек, которые служат для получения топологически точной триангуляции для каждого варианта. Вершины многоугольника помечаются метками v_1, v_2, v_3, v_4, v_5 и v_6. Триангуляция этого многоугольника производится в ребрах (v_2, v_6) и (v_3, v_5); их не следует использовать, так как у контура нет ребер между этими вершинами в верхней грани ячейки [19].

Предположим, что все точки контура имеют нормальное направление в (x, y, z). Для контура, полученного с помощью соотношений (3) и (4), существует не менее двух точек x_0 и x_1 с нормалью в направлении x.

Определим d:

$$
d = ar^2 + br + c \tag{9}
$$

Теперь можно описать алгоритм генерации *топологически точных* шагающих кубиков:

1. создать многоугольники, вмещающие полное сглаженное изображение; эти многоугольники называются *наружными кольцами*;
2. рассчитать *внутренние кольца* по формуле (9);

3. если внутреннее кольцо не является реальным или находится за пределами изображения, то внешние кольца триангулируются независимо; в противном случае продолжить;
4. проверить связность между *внутренним кольцом* и каждым из *наружных колец*; если внутреннее кольцо и одно из внешних колец принадлежат к одному и тому же сегменту контура, то зона между ними триангулируется.

4. Эксперименты

Нашей основной целью является создание трехмерной модели заданного объекта на основе 2D-изображения, полученного с помощью сканера. Для оценки эффективности предложенного подхода был использован реальный снимок МРТ. Далее будет описано, как генерировалась 3D-изображение с использованием различных этапов, описанных ранее.

Сначала для определения границ на поверхности исходного изображения мы используем среднюю интенсивность пикселей и обозначаем эти границы разными цветами (рис. 9).

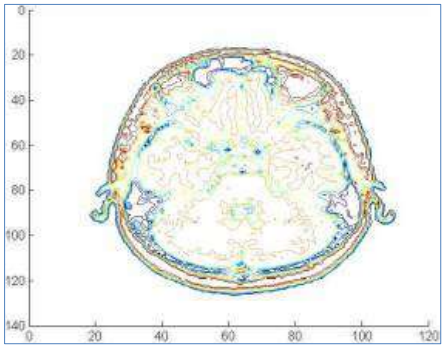


Рис. 9. Определение границ на поверхности изображения на плоскости
Fig. 9. Defining the borders on the surface of an image in the plane

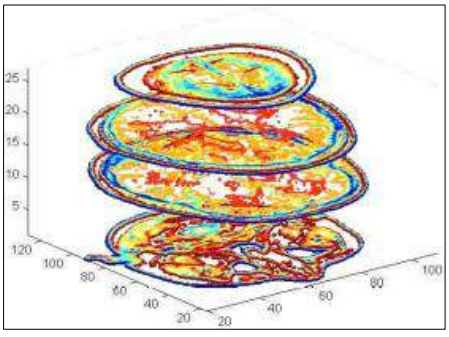


Рис. 10. Создание изображения в виде срезов на плоскости
Fig. 10. Generate a sliced image, by defining the edges in a plane

Следующий шаг заключается в генерации изображения, представленного в виде срезов. Это делалось путем определения ребер на плоскости, как описывалось в предыдущем разделе. На рис. 10 приводится изображение в виде срезов.

Как можно заметить, удалось определить плоскости различной интенсивности, в которых показаны текстурные пластины с четырьмя изображениями в матрице размером 1024x1024, спроецированные из единственной текстуры карты расположения двухмерных подслоев (матрицы 3x3) в поднаборе данных. В одной восьмой части текстуры каждого среза копируются противоположные данные для частичного воспроизведения виртуальных стыков этих плоскостей, что дает конечный результат в виде трехмерного изображения. Затем нужно заполнить изображение в виде срезов, чтобы получить трехмерную версию, см. рис. 11. Это

достигается путем предварительного извлечения из поверхностного слоя отфильтрованного образца, так как в поверхностной сетке отдельные текстуры часто повторяются.

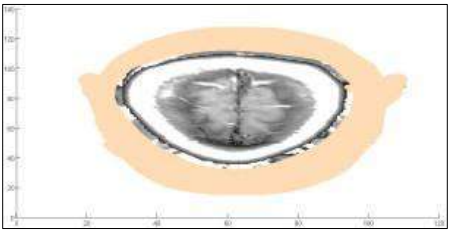


Рис. 11 Изображение в виде срезов для получения 3D-версии
Fig. 11 Sliced image in order to produce a 3D version

Для этого требуется применять методы сглаживания и заполнения пустот. Мы использовали метод, описанный в 3.2.2.

На рис. 12 изображен один срез изображения при увеличении интенсивности пикселей. Как можно заметить, что под воздействием этого процесса уменьшаются контрастность.

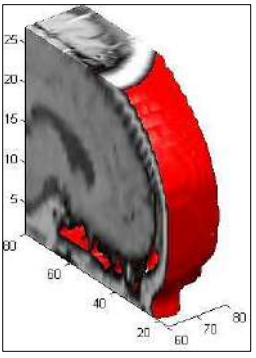


Рис. 12 Срез изображения при увеличении интесивности пикселей
Fig. 12 Slide of the image when its pixels are elevated

Как видно, увеличение яркости всех пикселей достигается за счет фильтрации краев. На рис. 12 показано увеличение интенсивности 10 элементов, что позволяет наблюдать внутреннюю часть изображения на определенном уровне яркости и, таким образом, иметь возможность оценивать эффект при различных вариантах увеличения пикселей.

5. Заключение

В этой статье описана методика получения 3D-представления двухмерного изображения. Предложенная методика заключается в модификации алгоритма Marching Cubes и включает три этапа, первый из которых помогает нам получить выборку первичных и вторичных вершин, где в качестве главной вершины используются углы, в которых соединяются блоки различных уровней детализации. На втором этапе полученный алгоритмом MC треугольник представляется контуром и, наконец, идентифицируется топология, которая определяет внутренние точки, необходимые для получения топологически точного треугольника для каждого варианта. Полученные результаты показывают практичность реализованной методики. В рамках дальнейшей работы планируется разработать метрики, которые позволили бы измерить производительность предложенного алгоритма.

Список литературы / References

[1] T. Senthil Kumar and Anupa Vijai. 3D Reconstruction of Face from 2D CT Scan Images. Procedia Engineering, vol. 30, 2012, pp. 970-977.

[2] Marcos Cirne and Helio Pedrini. Marching Cubes Technique for Volumetric Visualization Accelerated with Graphics Processing Units. Journal of the Brazilian Computer Society, vol. 19, 2013, pp. 223-233.

[3] Timothy Newman and Hong Yi. A survey of the Marching Cubes algorithm. Computers Graphics, vol. 30, issue 5, 2006, pp. 854-879.

[4] Zhongjie Long and Kouki Nagamune. A Marching Cubes Algorithm: Application for Three-dimensional Surface Reconstruction Based on Endoscope and Optical Fiber. Information, International Information Institute, vol. 18, issue 4, 2015, pp.1425-1437.

[5] William E. Lorensen and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. ACM Computer Graphics, vol. 21, issue 4, 1987, pp. 163-169.

[6] M.J. Durst. Letters: Additional Reference to Marching Cubes. ACM Computer Graphics, vol. 22, issue 2, 1988, pp, 72-73.

[7] Gregory Nielson and Bernd Hamann. The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes. In Proc. of the 26th IEEE Conference on Visualization, 1991, pp. 83-91.

[8] B. Natarajan. On Generating Topologically Consistent Isosurfaces from Uniform Samples. The Visual Computer, vol. 11, 1994, pp. 52–62.

[9] Evgeni V. Chernyaev. Marching Cubes 33: Construction of Topologically Correct Isosurfaces. Technical report, Institute for High Energy Physics, 1995.

[10] Lis Custodio, Sinesio Pesco, and Cláudio Silva. An Extended Triangulation to the Marching Cubes 33 Algorithm. Journal of the Brazilian Computer Society, vol. 25, 2019, article no. 6.

[11] F. Gong and X. Zhao. Three-Dimensional Reconstruction of Medical Image Based on Improved Marching Cubes Algorithm. In Proc. of the International Conference on Machine Vision and Human-machine Interface, 2010, pp. 608-611.

[12] S. Liu and J. Peng. Optimization of Reconstruction of 2D Medical Images based on Computer 3D Reconstruction Technology. Journal of Digital Information Management, vol. 13, issue 3, 2015, pp. 142-146.

[13] Maxim A. Olshanskii, Arnold Reusken, and Jörg Grande. A Finite Element Method for Elliptic Equations on Surfaces. SIAM Journal on Numerical Analysis, vol. 47, issue 5, 2009, pp. 3339-3358.

[14] Alexey Y. Chernyshenko and Maxim A. Olshanskii. An Adaptive Octree Finite Element Method for PDEs posed on Surfaces. Computer Methods in Applied Mechanics and Engineering, vol. 291, 2015, pp. 146-172.

[15] A. Bonito, R.H. Nochetto, and M.S. Pauletti. Dynamics of Biomembranes: Effect of the Bulk Fluid. Mathematical Modelling of Natural Phenomena, vol. 6, issue 5, 2011, pp. 25-43.

[16] Matteo Cacciari and Gavin P. Salam. Dispelling the N 3 myth for the kt jet-finder. Physics Letters B, vol. 641 issue 1, 2006, pp. 57-61.

[17] Thatcher Ulrich. Rendering Massive Terrains Using Chunked Level of Detail Control. In Proc. of the 29th Annual Conference on Computer Graphics and Interactive Techniques, 2002.

[18] Gregory M. Nielson, Liyan Zhang, Kun Lee, and Adam Huang. Parameterizing Marching Cubes Isosurfaces with Natural Neighbor Coordinates. Lecture Notes in Computer Science book series, vol. 4975, 2008, pp. 315-328.

[19] Allen Van Gelder and Jane Wilhelms. Topological Considerations in Isosurface Generation. ACM Transactions on Graphics, vol. 13, issue 4, 1994, pp. 337-375.

[20] Gerald E. Farin. Curves and Surfaces for Computer-Aided Geometric Design: A Practical Code. 4th edition. Academic Press, 1996, 429 p.

[21] Claudio Montani, Riccardo Scateni, and Roberto Scopigno. A modified look-up table for implicit disambiguation of Marching Cubes. The Visual Computer, vol. 10, 1994, pp. 353-355.

Информация об авторах / Information about authors

Делия Иразу ХЕРНАНДЕС ФАРСАС, PhD, доцент, факультет науки и техники. Научные интересы включают искусственный интеллект, машинное обучение, обработку естественного языка.

Delia Irazu HERNÁNDEZ FARÍAS, PhD, Associated Professor, Division of Sciences and Engineering. Research interests include Artificial Intelligence, Machine Learning, Natural Language Processing.

Рафаэль ГУСМАН КАБРЕРА – PhD, профессор кафедры электротехники, заведующий лабораторией. Область научных интересов: искусственный интеллект, машинное обучение, обработка изображений, интеллектуальный анализ текстов, обработка естественного языка.

Rafael GUZMÁN CABRERA – PhD, titular professor at the Electrical Engineering Department, Head of Laboratory. Research interests include Artificial Intelligence, Machine Learning, Image Processing, Text Mining, Natural Language Processing.

Теодоро КОРДОВА ФРАГА – PhD, профессор кафедры инженерной физики. Область научных интересов: физика, биомагнетизм, биомедицинские инструменты, обработка цифровых сигналов и изображений.

Teodoro CORDOVA FRAGA – PhD, Professor at the Department of Engineering Physics. Research interests: Physics, Biomagnetism, Biomedical Instrumentations, Digital Signal and Imaging Processing.

Хосе Сакариас УАМАНИ ЛУНА – магистр наук в области физики. Область научных интересов: физика, цифровая обработка изображений, медицинские приложения.

Jose Zacarías HUAMANÍ LUNA – Master of Sciences, Physics. Research interests: Physics, Digital Image Processing, Medical Applications.

Хосе Франсиско ГОМЕС АГИЛАР – PhD, профессор, заведующий лабораторией. Область прикладная математика, медицинская физика, дробное исчисление.

Jose Francisco GOMEZ AGUILAR – PhD, Professor, Head of Laboratory. Research interests include Applied Mathematics, Medical Physics, Mathematical Physics, Fractional Calculus.

DOI: 10.15514/ISPRAS–2020–32(5)–14

Моделирование инфразвукового пистонфона

Д.В. Головин, ORCID: 0000-0002-9488-6505 <golovin@vniiftri.ru>
Всероссийский научно-исследовательский институт
физико-технических и радиотехнических измерений, Росстандарт,
141570, Московская область, Солнечногорский район, г.п. Менделеево

Аннотация. Представлены результаты численного моделирования прикладной акустической задачи – моделирования газовых процессов, протекающих в измерительной камере инфразвукового пистонфона 3202 при различных частотах колебаний поршня (0,1 – 1000 Гц) и характеризующихся крайне малыми значениями числа Маха ($9{,}1\cdot10^{-7}\div9{,}1\cdot10^{-3}$). Моделирование проведено с использованием квазигазодинамических (КГД) и квазигидродинамических (КГиД) уравнений вязкого сжимаемого теплопроводного газа, определены границы устойчивости алгоритмов КГД и КГиД в данной задаче.

Ключевые слова: акустика; инфразвук; численное моделирование; квазигазодинамические уравнения; квазигидродинамические уравнения

Для цитирования: Головин Д.В. Моделирование инфразвукового пистонфона. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 181-198. DOI: 10.15514/ISPRAS–2020–32(5)–14

Благодарности. Автор благодарен Елизаровой Т.Г. за поддержку, консультации и обсуждение результатов при проведении данной работы.

Simulation of Infrasound Pistonphone

D.V. Golovin, ORCID: 0000-0002-9488-6505 <golovin@vniiftri.ru>
All-Russian Scientific Research Institute of
Physical Technical and Radiotechnical Measurements, Rosstandart,
Mendeleevo, Solnechnogorsky district, Moscow region, 141570

Abstract. There are presented the results of numerical simulation of an applied acoustic problem – modeling of gas processes occurring in the measuring chamber of the infrasound pistonphone 3202 at different frequencies of piston oscillation (0.1 – 1000 Hz) and characterized by extremely small Mach numbers ($9.1\cdot10^{-7}\div9.1\cdot10^{-3}$). The simulation was performed using quasi-gas-dynamic (QGD) and quasi-hydrodynamic (QHD) equations of a viscous compressible heat-conducting gas with the use of a time-explicit difference scheme, all spatial derivatives was approximated by central differences. It is shown that QGD and QHD models can be used for a simulation of applied acoustics and, in particular, to the simulation of infrasonic pistonphone: the stability limits of the QGD and QHD algorithms in this problem were determined, the dependence of sound pressure on the tuning parameter α is investigated and it is shown that this dependence is quite small. The spectra of sound pressure at the control point calculated by QGD and QHD are given, their dependence on the tuning parameter α is shown, both models equally predict the value of the sound pressure amplitude at the fundamental frequency oscillations. At the end of the article, the sound pressure at the control point at the fundamental frequency oscillations obtained by using QGD and QHD is compared with the values calculated by using semi-empirical formula of sound pressure at closed volume for a case of small oscillations using the polytropic index obtained by Henry Gerber instead of the adiabatic coefficient.

Keywords: acoustic; infrasound; numerical simulation; quasi gas dynamic equations; quasi hydro dynamic equations

For citation: Golovin D.V. Simulation of infrasound pistonphone. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 181-198 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–14

Acknowledgements. The author is grateful to T.G. Elizarova for support, advice and discussion of the results in carrying out this work.

1. Введение

Актуальность исследования обусловлена появлением за последние 10 лет нового поколения инфразвуковых микрофонов, применяемых на станциях мониторинга за соблюдением Договора о всеобъемлющем запрещении ядерных испытаний (см., например, [1-3]). В России аналогом являются дифференциальные конденсаторные микробарометры К-304-АМ1И и ISGM-03M (вариант К-304-АМ1) с диапазоном частот вплоть до 0,001 – 63 Гц (рабочий диапазон частот от 0,02 до 4,0 Гц) и максимальной амплитудой входного сигнала не менее 100 Па, разработанные и производимые НТЦ «Геофизические измерения». Таким образом, для поверки и калибровки данных средств измерений необходимо разработать лабораторную установку, реализующую один из первичных (абсолютных) методов калибровки с рабочим диапазоном частот от 1 мГц до 50-100 Гц.

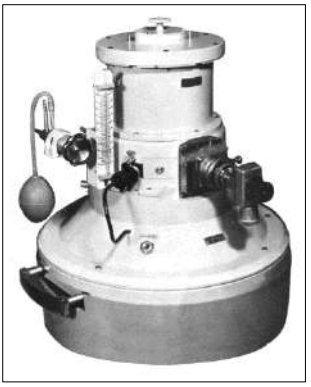


Рис. 1. Общий вид пистонфона 3202
Fig. 1. General view of pistonphone 3202

На таких частотах представляется удобным использовать классический метод пистонфона [4-7].Во ВНИИФТРИ в 1960-ые гг. был разработан пистонфон 3202, обладающий рабочим диапазоном частот 0,1 – 100 Гц (рис. 1). Поршень в основании измерительной камеры посредством электродинамической системы с подвижной катушкой приводится в движение, изменяет величину рабочего объема и тем самым создает колебания давления воздуха, действующее на устанавливаемый на верхку камеры эталонный измерительный микрофон. Перед проведением измерений проводится проверка на герметичность камеры с помощью избыточного давления, нагнетаемого резиновой грушей, и водяного манометра, соединенного с внутренним объемом камеры через штуцер. По величине двойного хода поршня, фиксируемого с помощью микроскопа по нанесенной на шток поршня светящейся метке, диаметра поршня и объема используемой измерительной камеры рассчитывается амплитуда звукового давления P, воздействующего на калибруемое средство измерения:

$$P = \frac{C|\boldsymbol{N}|P_0 S}{V}\frac{X}{2}, \qquad (1)$$

где $C = kh/\sin(kh)$ – коэффициент волнового распределения звукового давления вдоль оси камеры; k – волновое число; h – высота камеры; $\boldsymbol{N}$ – коэффициент политропы, зависящий от частоты колебаний; P_0 – атмосферное давление в момент герметизации камеры; S – площадь поперечного сечения поршня пистонфона; V – объем камеры; X – двойная амплитуда колебаний поршня.

Коэффициент политропы $\boldsymbol{N}$ рассчитывается согласно исследованию Г. Гербера (H. Gerber) [8] для случая жесткого поршня по следующим формулам:

$$\boldsymbol{N} = 1 + (\gamma - 1)\boldsymbol{E}, \qquad \boldsymbol{E} = D\sum \frac{N_n}{1 - \frac{M_n}{2\pi}\frac{\alpha}{fl^2}i},$$

где γ – коэффициент адиабаты; $\boldsymbol{E}$ – комплексная передаточная функция температуры; D, N_n, M_n, l – члены, зависящие от геометрии камеры; α – коэффициент температуропроводности при постоянном объеме; f – частота колебаний воздуха в камере: i – мнимая единица.

Для камеры цилиндрической формы D, N_n, M_n и l имеют следующий вид:

$$D = \frac{8}{\pi^2}, \qquad N_{mn} = \frac{1}{\left(m + {}^1\!/_2\right)^2 x_n^2}, \qquad M_{mn} = \frac{\left(m + {}^1\!/_2\right)^2 \pi^2 + x_n^2 R^2}{(2R+1)^2}, \qquad l = \frac{V}{S},$$

где m=0,1,2,..; n=0,1,2,..; x_n – n-ый корень функции Бесселя нулевого порядка; $R = {}^h\!/_d$ – отношение внутренней высоты h к внутреннему диаметру d камеры; V и S – внутренний объем и внутренняя площадь стенок камеры.

Подобный расчет автоматически несет в себе допущения, принятые в [8] при выведении формулы зависимости коэффициента политропы от частоты колебаний (например, рассматривается случай малых колебаний при осреднении плотности, температуры и давления газа по всему объему камеры, т.е. не рассматриваются их объемные распределения). За последние 15 лет появилось несколько работ [9-11], посвященных исследованию результатов [8] и показывающих, что они недостаточно хорошо соответствуют современному уровню точности акустических измерений и что необходимо пересмотреть международный стандарт о первичной калибровке измерительных лабораторных микрофонов типа LS1 и LS2 методом взаимности IEC 61094-2 [12] в части калибровки микрофонов на инфразвуковых частотах, в которой используются результаты работы Г. Гербера. Таким образом, целью данной работы является проверка совпадения результатов численного моделирования звукового давления для сжимаемого вязкого теплопроводного воздуха со значением, вычисляемым по формуле (1), и результаты которой могут послужить для оптимизации характеристик пистонфона 3202.

2. Постановка задачи и газодинамические параметры

Измерительная камера пистонфона представляет собой цилиндр диаметром d=69,97 мм и высотой h=66,06 мм, поршень в основании камеры имеет диаметр d_p=20 мм. Рассматриваются частоты колебаний поршня 0,1 – 1000 Гц, при которых поршень колеблется с амплитудой X=0,5 мм (необходимо отметить, что при таком колебании поршня в рассматриваемом диапазоне частот задача находится в рамках линейной акустики, поэтому моделирование не должно показать каких-либо сильных нелинейных эффектов). Начальные параметры воздуха: температура T_0=296,15 K, давление p_0=101325 Па, плотность ρ_0=1,186 кг/м^3, коэффициент вязкости μ=1,83·10^{-5} Па·с, газовая постоянная R=288,5 Дж/(кг·К), коэффициент адиабаты γ=1,4, коэффициент теплопроводности χ=0,0254 Вт/(м·К), число Прандтля Pr=0,728, показатель межмолекулярного взаимодействия ω=0,74.

Для описания движения поршня требуется учет движения границы и применение подробных пространственных сеток. Желательно также провести сопоставление между собой результатов, полученных при помощи различных численных подходов. Все это сложно выполнимо в рамках индивидуального кода, но может быть сделано в рамках имеющихся открытых пакетов, например, пакета OpenFoam. В данной работе выполнен упрощенный вариант постановки задачи в плане ее геометрии и условий на границе.

Рис. 2. Расчетная область измерительной камеры пистонфона
Fig. 2. Calculation area of the pistonphone measurement coupler

Область расчета (рис. 2) полагается цилиндрической, а колебательно течение воздуха в камере пистонфона – осесимметричным. Для численного решения используются полные уравнения Навье-Стокса для вязкого сжимаемого теплопроводного газа в отсутствие внешних сил и источников тепла с применением их регуляризации двух видов – КГД и КГиД моделей [13-14]:

$$\frac{\partial \rho}{\partial t} + \frac{1}{r}\frac{\partial (r j_{mr})}{\partial r} + \frac{\partial j_{mz}}{\partial z} = 0,$$

$$\frac{\partial (\rho u_r)}{\partial t} + \frac{1}{r}\frac{\partial (r j_{mr} u_r)}{\partial r} + \frac{\partial (j_{mz} u_r)}{\partial z} + \frac{\partial p}{\partial r} = \frac{1}{r}\frac{\partial (r \Pi_{rr})}{\partial r} + \frac{\partial \Pi_{zr}}{\partial z} - \frac{\Pi_{\varphi\varphi}}{r},$$

$$\frac{\partial (\rho u_z)}{\partial t} + \frac{1}{r}\frac{\partial (r j_{mr} u_z)}{\partial r} + \frac{\partial (j_{mz} u_z)}{\partial z} + \frac{\partial p}{\partial z} = \frac{1}{r}\frac{\partial (r \Pi_{rz})}{\partial r} + \frac{\partial \Pi_{zz}}{\partial z},$$

$$\frac{\partial E}{\partial t} + \frac{1}{r}\frac{\partial (r j_{mr} H)}{\partial r} + \frac{\partial (j_{mz} H)}{\partial z} + \frac{1}{r}\frac{\partial (r q_r)}{\partial r} + \frac{\partial q_z}{\partial z} = \frac{1}{r}\frac{\partial}{\partial r}[r(\Pi_{rr} u_r + \Pi_{rz} u_z)] + \frac{\partial}{\partial z}(\Pi_{zr} u_r + \Pi_{zz} u_z),$$

где u_r и u_z – проекции вектора скорости $\vec{u}$ на оси r и z; E – полная энергия единицы объема; H – полная удельная энтальпия:

$$E = \rho \frac{u_r^2 + u_z^2}{2} + \frac{p}{\gamma - 1}, \qquad H = \frac{E + p}{\rho},$$

$$p = \rho R T.$$

Компоненты вектора плотности вычисляются по формулам:

$$j_{mr} = \rho(u_r - \omega_r), \qquad j_{mz} = \rho(u_z - \omega_z)$$

Величины ω_r и ω_z, компоненты тензора вязкости Π и проекции на оси r и z теплового потока в КГД и КГиД считаются по-разному.

Граница расчетной области состоит из оси симметрии, поршня и стенок камеры.

Движение поршня в произвольный момент времени t описывается с помощью граничного условия для азимутальной скорости:

$$u^{piston} = u_z^{piston} = 2\pi f X \cdot \cos(2\pi f t).$$

Также на границе поршня задано условие прилипания для радиальной скорости, постоянство температуры и непроницаемости:

$$u_r = 0, \qquad T = T_0, \qquad \frac{\partial p}{\partial z} = 0.$$

На оси симметрии заданы условия симметрии:

$$\frac{\partial \rho}{\partial r} = 0, \qquad \frac{\partial u_z}{\partial r} = 0, \qquad u_r = 0, \qquad \frac{\partial p}{\partial r} = 0.$$

На стенках камеры – условия прилипания для скорости, постоянства температуры и непроницаемости:

$$u_r = 0, \qquad u_z = 0, \qquad T = T_0, \qquad \frac{\partial p}{\partial n} = 0,$$

где *n* – нормаль к поверхности.

В качестве колебания давления $\Delta p(t)$ (и, впоследствии, звукового давления) в момент времени *t* рассматривается величина

$$\Delta p(t) = p(t) - p_0,$$

где *p(t)* – полное давление в момент времени *t*; p_0 – атмосферное давление.

2.1 Уравнения КГД

Величины ω_r и ω_z, компоненты тензора вязкости *П* и проекции на оси *r* и *z* теплового потока в КГД и КГиД считаются по-разному. В случае КГД формулы расчета выглядят следующим образом:

$$\omega_r^{QGD} = \frac{\tau}{\rho}\left[\frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r^2) + \frac{\partial}{\partial z}(r\rho u_r u_z) + \frac{\partial p}{\partial r}\right] \qquad \omega_z^{QGD} = \frac{\tau}{\rho}\left[\frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r u_z) + \frac{\partial}{\partial z}(r\rho u_z^2) + \frac{\partial p}{\partial z}\right]$$

$$\Pi_{rr}^{QGD} = \Pi_{rr}^{NS} + u_r\omega_r^* + R^*, \qquad \Pi_{rr}^{NS} = 2\mu\frac{\partial u_r}{\partial r} - \frac{2}{3}\mu\, div\vec{u},$$

$$\Pi_{rz}^{QGD} = \Pi_{rz}^{NS} + u_r\omega_z^*, \qquad \Pi_{rz}^{NS} = \Pi_{zr}^{NS} = \mu\left(\frac{\partial u_r}{\partial z} + \frac{\partial u_z}{\partial r}\right),$$

$$\Pi_{zr}^{QGD} = \Pi_{zr}^{NS} + u_z\omega_r^*,$$

$$\Pi_{\varphi\varphi}^{QGD} = \Pi_{\varphi\varphi}^{NS} + R^*, \qquad \Pi_{\varphi\varphi}^{NS} = 2\mu\frac{u_r}{r} - \frac{2}{3}\mu\, div\vec{u},$$

$$\Pi_{zz}^{QGD} = \Pi_{zz}^{NS} + u_z\omega_z^* + R^*, \qquad \Pi_{zz}^{NS} = 2\mu\frac{\partial u_z}{\partial z} - \frac{2}{3}\mu\, div\vec{u}.$$

При этом величины ω_r^*, ω_z^* и R^* вычисляются так:

$$\omega_r^* = \tau\left[\rho u_r\frac{\partial u_r}{\partial r} + \rho u_z\frac{\partial u_r}{\partial z} + \frac{\partial p}{\partial r}\right],$$
$$\omega_z^* = \tau\left[\rho u_r\frac{\partial u_z}{\partial r} + \rho u_z\frac{\partial u_z}{\partial z} + \frac{\partial p}{\partial z}\right],$$
$$R^* = \tau\left[u_r\frac{\partial p}{\partial r} + u_z\frac{\partial p}{\partial z} + \gamma p\, div\vec{u}\right].$$

Формулы для компонент теплового потока:

$$q_r^{QGD} = q_r^{NS} - u_r R^q, \qquad q_z^{QGD} = q_z^{NS} - u_z R^q$$
$$q_r^{NS} = -\chi\frac{\partial T}{\partial r}, \qquad q_z^{NS} = -\chi\frac{\partial T}{\partial z},$$
$$R^q = \tau\rho\left[\frac{u_r}{\gamma-1}\frac{\partial}{\partial r}\left(\frac{p}{\rho}\right) + \frac{u_z}{\gamma-1}\frac{\partial}{\partial z}\left(\frac{p}{\rho}\right) + pu_r\frac{\partial}{\partial r}\left(\frac{1}{\rho}\right) + pu_z\frac{\partial}{\partial z}\left(\frac{1}{\rho}\right)\right].$$

2.2 Уравнения КГиД

По своему виду КГиД уравнения проще уравнений КГД, и уравнения замыкания в ней выглядят как:

$$\omega_r^{QHD} = \tau\left(u_r\frac{\partial u_r}{\partial r} + u_z\frac{\partial u_r}{\partial z} + \frac{1}{\rho}\frac{\partial p}{\partial r}\right) \qquad \omega_z^{QHD} = \tau\left(u_r\frac{\partial u_z}{\partial r} + u_z\frac{\partial u_z}{\partial z} + \frac{1}{\rho}\frac{\partial p}{\partial z}\right)$$

$$q_r = q_r^{NS} \qquad q_z = q_z^{NS}$$

$$\Pi_{rr}^{QHD} = \Pi_{rr}^{NS} + u_r\omega_r^{QHD}, \qquad \Pi_{rz}^{QHD} = \Pi_{rz}^{NS} + u_r\omega_z^{QHD},$$

$$\Pi_{zr}^{QHD} = \Pi_{rz}^{NS} + u_z\omega_r^{QHD}, \qquad \Pi_{\varphi\varphi}^{QHD} = \Pi_{\varphi\varphi}^{NS},$$

$$\Pi_{zz}^{QHD} = \Pi_{zz}^{NS} + u_z\omega_z^{QHD}.$$

Следует отметить, что при τ=0 уравнения КГиД и КГД переходят в уравнения Навье-Стокса.

2.3 Сеточная аппроксимация уравнений и обезразмеривание газодинамических переменных

Для сеточной аппроксимации уравнений используются явные аппроксимации по времени и центральные разностные производные по пространству на равномерной сетке с одинаковыми шагами по координатным направлениям [13, 5.5]. Регуляризующий параметр τ был задан как

$$\tau = \alpha\frac{h}{\mathrm{c}},$$

где α – схемный параметр, не изменяющийся в процессе расчета; h – шаг сетки; c – скорость звука в невозмущенной среде.

Как будет показано далее, шаг интегрирования по времени необходимо выбирать в соответствии с критерием Куранта.

Для удобства работы с уравнениями основные величины приведены к безразмерному виду для выделения параметров подобия, от которых зависит решение задачи (числа Маха и Рейнольдса).

Связь между размерными и безразмерными величинами (обозначены тильдой) выглядит следующим образом:

$$\rho = \tilde{\rho}\rho_0, \qquad p = \tilde{p}\rho_0(2\pi fX)^2, \qquad u = \tilde{u}\cdot 2\pi fX,$$

$$r = \tilde{r}d_p, \qquad z = \tilde{z}d_p, \qquad t = \tilde{t}\frac{d_p}{2\pi fX}, \qquad T = \tilde{T}\frac{(2\pi fX)^2}{\gamma R}.$$

Такое обезразмеривание не изменяет вид уравнений КГД и КГиД, но изменит вид уравнения состояния воздуха:

$$\tilde{p} = \frac{\tilde{\rho}\tilde{T}}{\gamma}.$$

Безразмерные коэффициенты вязкости, теплопроводности и параметр τ (знак тильда у безразмерных величин опущен) вычисляются следующим образом:

$$\mu = \frac{1}{Re}(M^2T)^{\omega}, \qquad \chi = \frac{\mu}{\mathrm{Pr}(\gamma-1)}, \qquad \tau = \alpha hM,$$

где M и Re – числа Маха и Рейнольдса:

$$M = \frac{2\pi fX}{c}, \qquad Re = \frac{\rho_0(2\pi fX)d_p}{\mu_0}.$$

Для рассматриваемых частот 0,1 – 1000 Гц значения чисел Маха и Рейнольдса меняются от $9{,}1\cdot10^{-7}$ и 0,204 до $9{,}1\cdot10^{-3}$ и 2040 соответственно.

3. Результаты моделирования

3.1 Устойчивость уравнений явной аппроксимации по времени КГД и КГиД и влияние параметра α

Для случая явной аппроксимации по времени используемых уравнений, согласно теоретическим оценкам [12-13], шаги по времени Δt и пространству h подчиняются критерию Куранта:

$$\Delta t = \beta \frac{h_{min}}{c_{max}}, \qquad (2)$$

где $0 < \beta < 1$ – число Куранта; h_{min} – минимальный шаг применяемой сетки; c_{max} – максимальная локальная скорость звука в расчетной области на текущем шаге по времени.

Для рассматриваемой задачи, как было сказано выше, характерны крайне малые значения числа Маха, поэтому, при наличии в уравнениях регуляризующих слагаемых, вопрос изучения границ применимости формулы (2) является естественным.

Исследование устойчивости проводилось следующим образом: для двух частот колебаний поршня (10 и 100 Гц) и двух равномерных сеток (53x100 и 79x150) фиксировалось число α в диапазоне от 0,1 до 1,0 и выбиралось такое число β, при котором расчет являлся устойчивым вне зависимости от наличия схемных осцилляций в распределениях плотности, давления, температуры и скоростей. Дополнительным критерием являлась устойчивость колебаний давления с течением времени. Границы устойчивости определялись относительно грубо: если при зафиксированном α расчет расходился, то первая значащая цифра числа Куранта β уменьшалась на единицу. Таким образом подбиралось максимально возможное значение числа Куранта β устойчивого счета для заданного значения α.

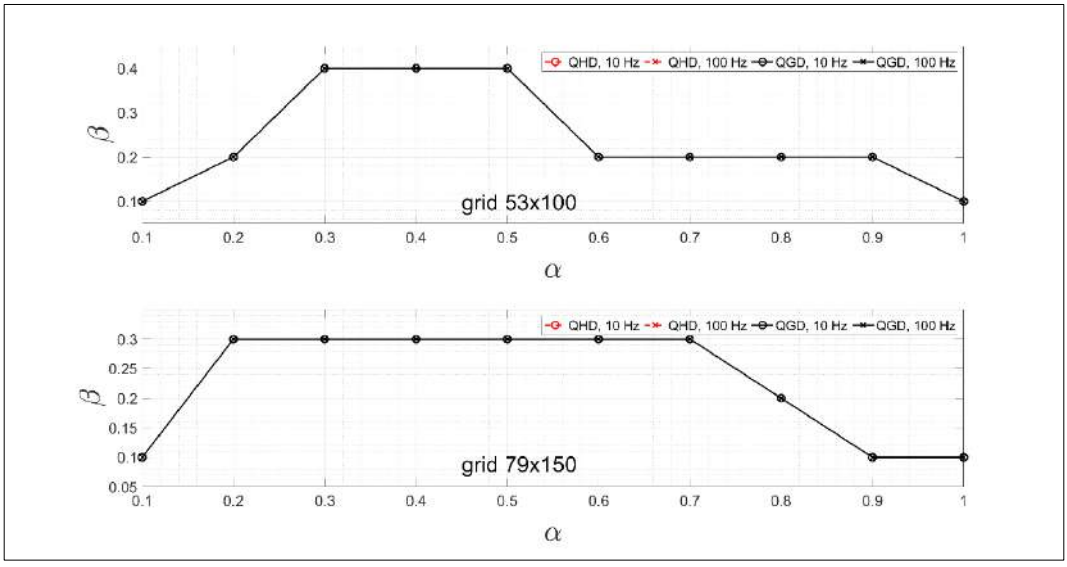


Рис. 3. Зависимость числа Куранта β от параметра α на частотах 10 Гц и 100 Гц для сеток 53x100 и 79x150

Fig. 3. The dependence of Courant number β from parameter α at the frequencies of 10 Hz and 100 Hz, the grids 53x100 and 79x150

На рис. 3 представлены результаты этого исследования. Как видно из рисунков, даже при таких малых значениях чисел Маха и Рейнольдса выполняется условие Куранта (2), а зависимость β(α) определяется размером сетки и в рамках заданной точности подбора β не зависит от частоты колебаний и одинакова для моделей КГД и КГиД.

На рис. 4-7 приведены колебания давления в контрольной точке на частотах 10 и 100 Гц для моделей КГД и КГиД.

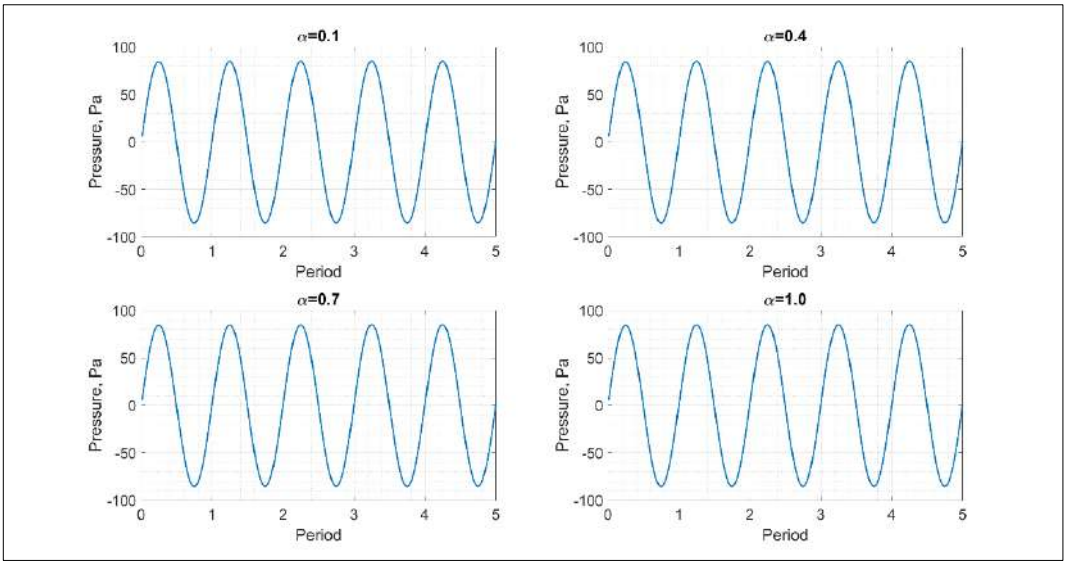


Рис. 4. Колебания давления в контрольной точке для различных значений параметра α на частоте колебаний поршня 10 Гц для модели КГД, сетка 53x100

Fig. 4. Pressure deviation in control point at different value of parameter α at the piston frequency oscillation of 10 Hz, QGD model, the grid 53x100

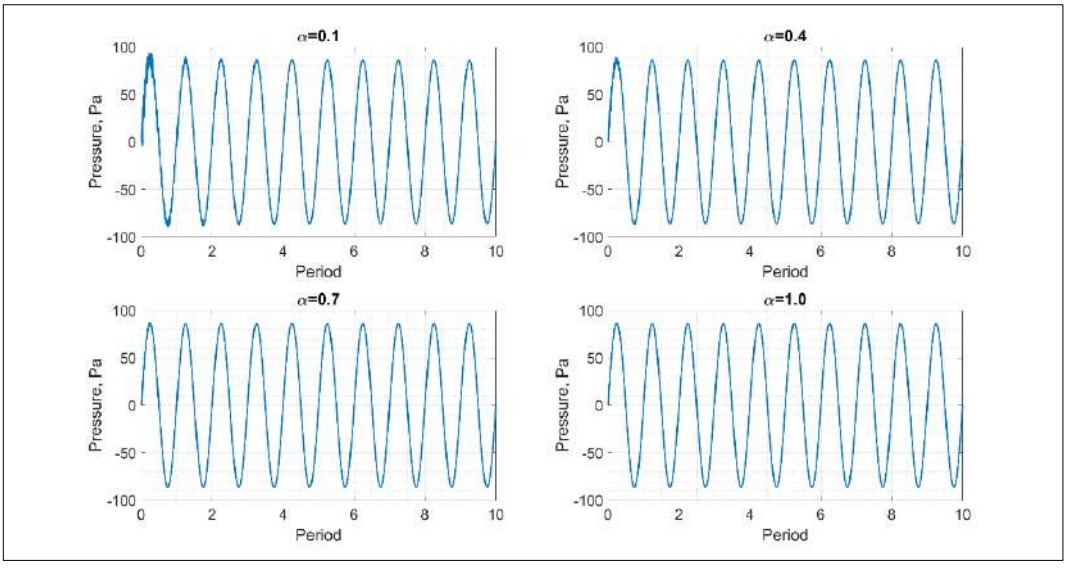


Рис. 5. Колебания давления в контрольной точке для различных значений параметра α на частоте колебаний поршня 100 Гц для модели КГД, сетка 53x100

Fig. 5. Pressure deviation in control point at different value of parameter α at the piston frequency oscillation of 100 Hz, QGD model, the grid 53x100

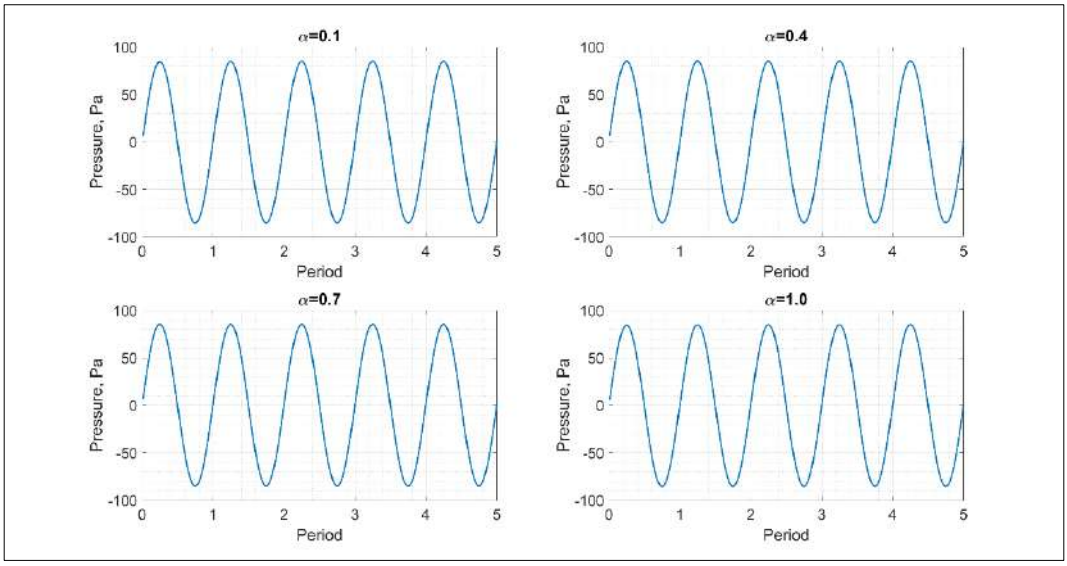


Рис. 6. Колебания давления в контрольной точке для различных значений параметра α на частоте колебаний поршня 10 Гц для модели КГиД, сетка 53x100

Fig. 6. Pressure deviation in control point at different value of parameter α at the piston frequency oscillation of 10 Hz, QHD model, the grid 53x100

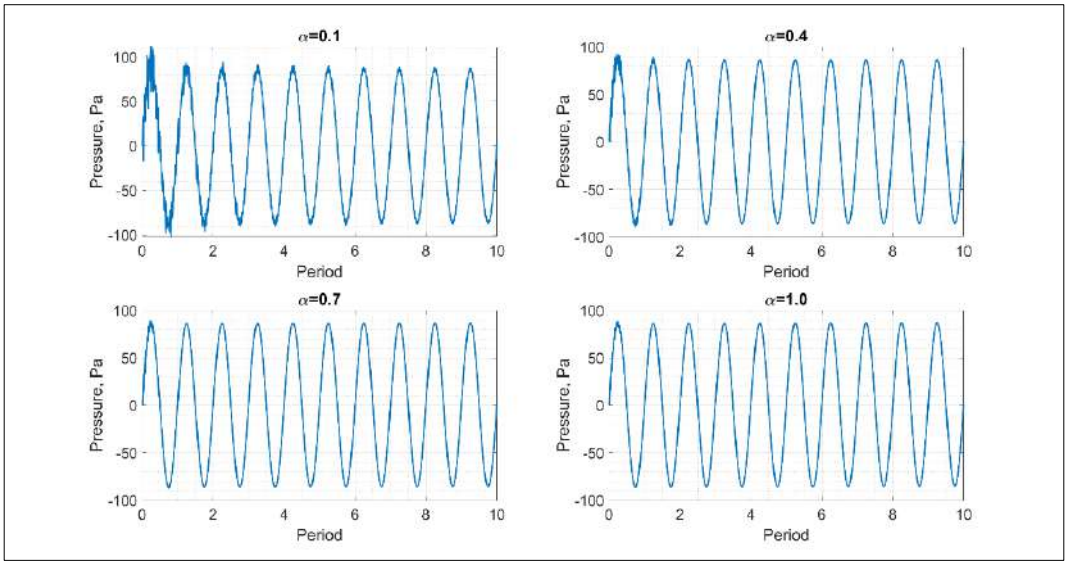


Рис. 7. Колебания давления в контрольной точке для различных значений параметра α на частоте колебаний поршня 100 Гц для модели КГиД, сетка 53x100

Fig. 7. Pressure deviation in control point at different value of parameter α at the piston frequency oscillation of 100 Hz, QHD model, the grid 53x100

По мере уменьшения числа α для модели КГиД при частоте колебаний поршня 100 Гц увеличиваются осцилляции на кривой колебания давления (рис. 7). Если еще больше уменьшить значение α (меньше 0,1), то устойчивых колебаний давления не получится, хотя для модели КГиД в области α<0,1 получаются гладкие распределения плотности, давления, температуры и скоростей (при α>0,1 на сетке 53x100 на исследованных частотах присутствуют незначительные схемные осцилляции, уменьшающиеся при увеличении размеров сетки).

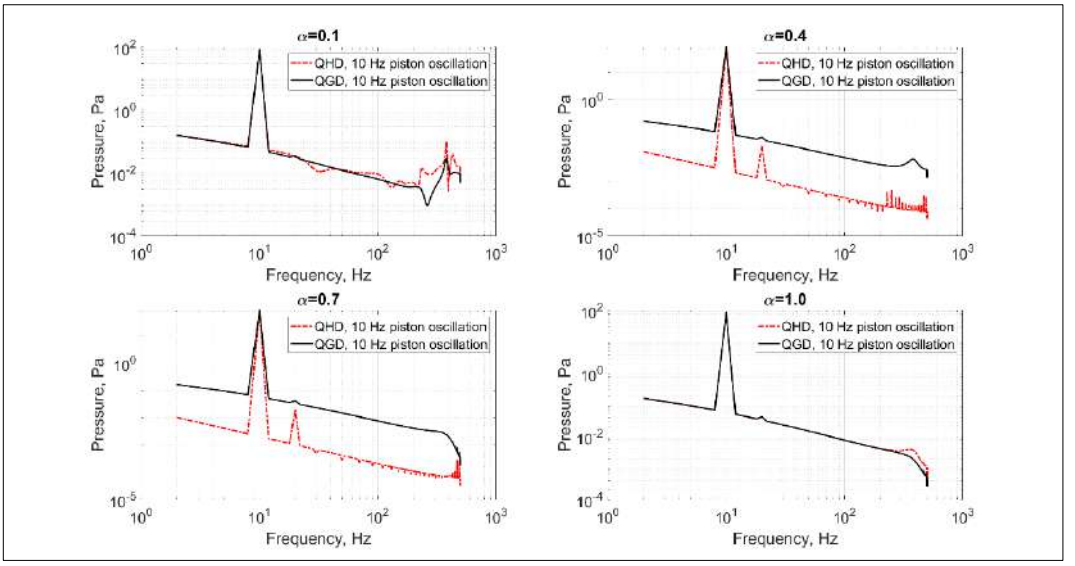


Рис. 8. Фурье-спектр давления в контрольной точке для различных значений параметра α на частоте колебаний поршня 10 Гц для моделей КГД и КГиД, сетка 53x100

Fig. 8. Fourier pressure amplitude spectrum in control point at different value of parameter α at the piston frequency oscillation of 10 Hz, QGD and QHD models, the grid 53x100

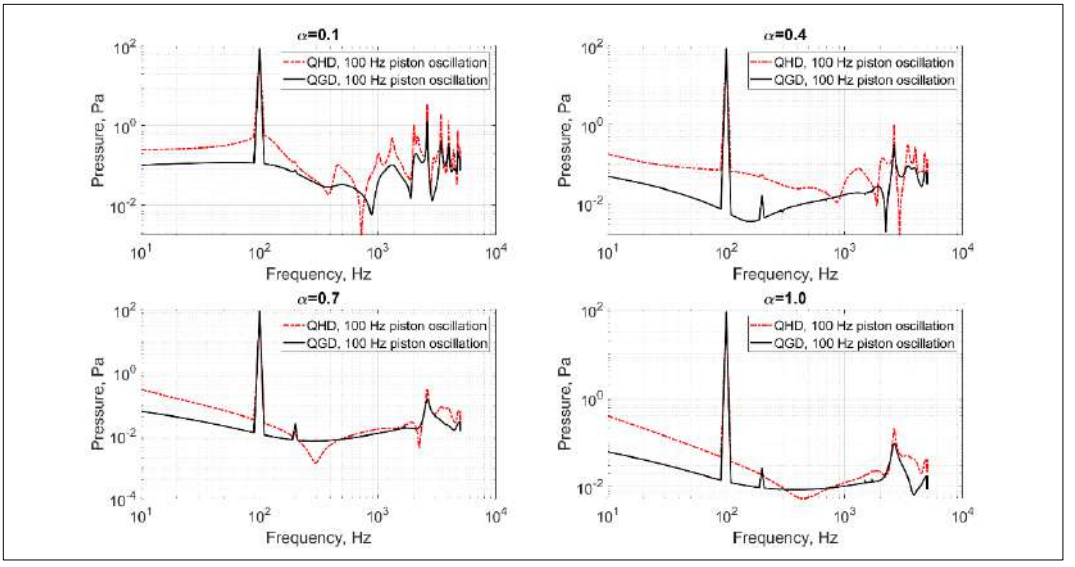


Рис. 9. Фурье-спектр давления в контрольной точке для различных значений параметра α на частоте колебаний поршня 100 Гц для моделей КГД и КГиД, сетка 53x100

Fig. 9. Fourier pressure amplitude spectrum in control point at different value of parameter α at the piston frequency oscillation of 100 Hz, QGD and QHD models, the grid 53x100

На рис. 8-9 представлены спектры колебаний давления в контрольной точке, соответствующие рис. 4-7. Модели КГД и КГиД предсказывают достаточно одинаковые значения амплитуды давления на основной частоте колебаний поршня, при определенных значениях числа α даже совпадают значения вторых гармоник – на частоте 10 Гц при α=1 спектры давления для КГД и КГиД очень хорошо совпали друг с другом (рис. 8). Если выбрать контрольную точку в другой части камеры, то на частотах 40-100 Гц значение

амплитуды колебаний давления основной частоты будут отличаться до 1 Па при расчете по КГД и КГиД.

Также была исследована зависимость амплитуды колебания давления в контрольной точке от значения параметра α – определенное влияние α присутствует, однако его сложно назвать серьезным (рис. 10).

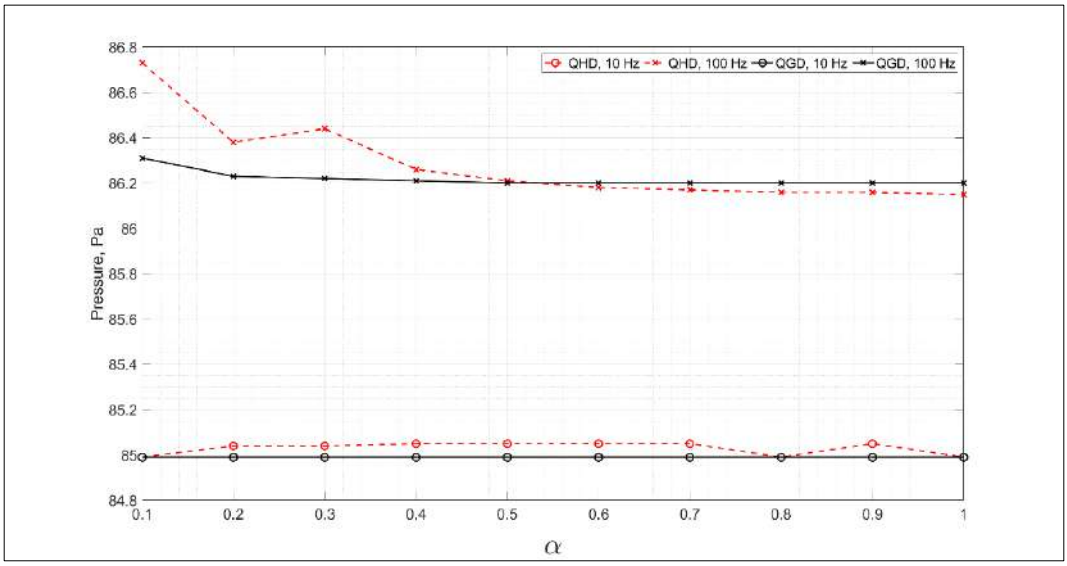


Рис. 10. Зависимость амплитуды колебаний давления на частоте колебания поршня в контрольной точке от параметра α на частотах 10 Гц и 100 Гц для моделей КГД и КГиД, сетка 53x100

Fig. 10. The dependence of pressure deviation amplitude at the piston frequency oscillation in control point from parameter α at the frequencies of 10 Hz and 100 Hz, the QGD and QHD models, the grid 53x100

В заключение остается добавить, что при $\alpha = \tau = 0$ не удалось подобрать такого значения шага по времени, чтобы вычисления были устойчивыми.

3.2 Звуковое давление в камере пистонфона

Теперь, убедившись на основе результатов подраздела 3.1, что модели КГД и КГиД действительно предсказывают колебания давления, слабо зависящие от настроечного параметра α и которые с хорошей точностью можно считать синусоидальными, далее будем использовать термин «звуковое давление».

В контрольной точке (см. рис. 2) на основе вычислений от 5 (на частотах 0,1 – 10 Гц для КГД и 1 – 10 Гц для КГиД) до 20 (в остальном диапазоне частот) периодов колебаний на сетке 53x100 определялся Фурье-спектр звукового давления и значение амплитуды звукового давления на основной частоте колебаний, которое затем сравнивалось с рассчитанным по формуле (1). Расчеты проводились при α=0,5.

Примеры линий тока и распределения звукового давления согласно КГД показаны для двух моментов времени на рис. 11-14.

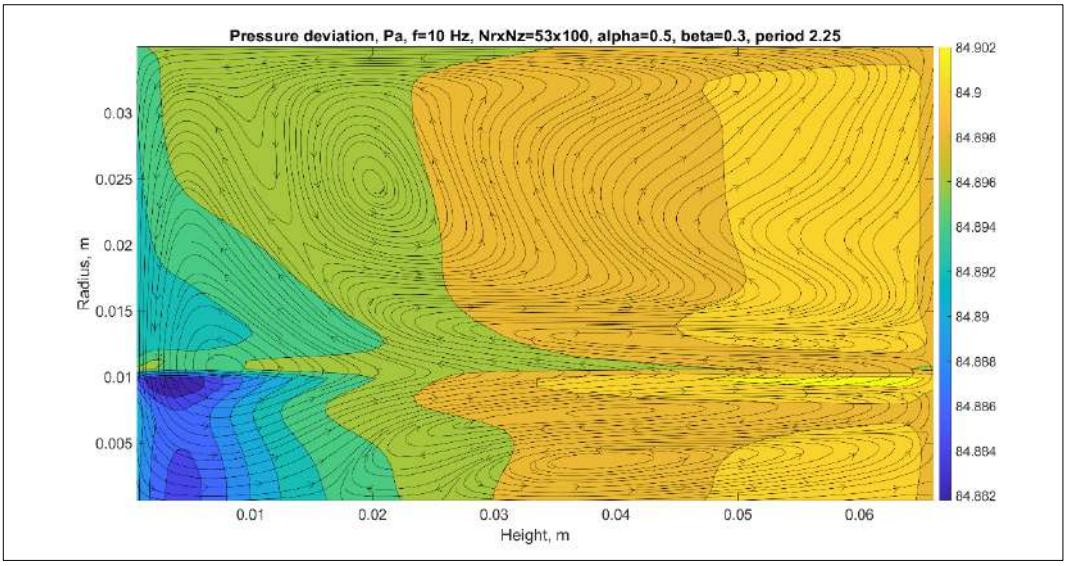


Рис. 11. Линии тока и изолинии звукового давления на частоте f=10 Гц в расчетной области на сетке 53x100, параметры α=0,5 и β=0,3, число периодов с начала колебаний – 2,25. Расчет согласно КГД
Fig. 11. Streamlines and sound pressure isolines at the frequency of 10 Hz at the calculation area on the grid 53x100, parameters α=0.5 and Courant number β=0.3, the number of periods since the beginning of oscillations – 2.25. Simulation according to QGD

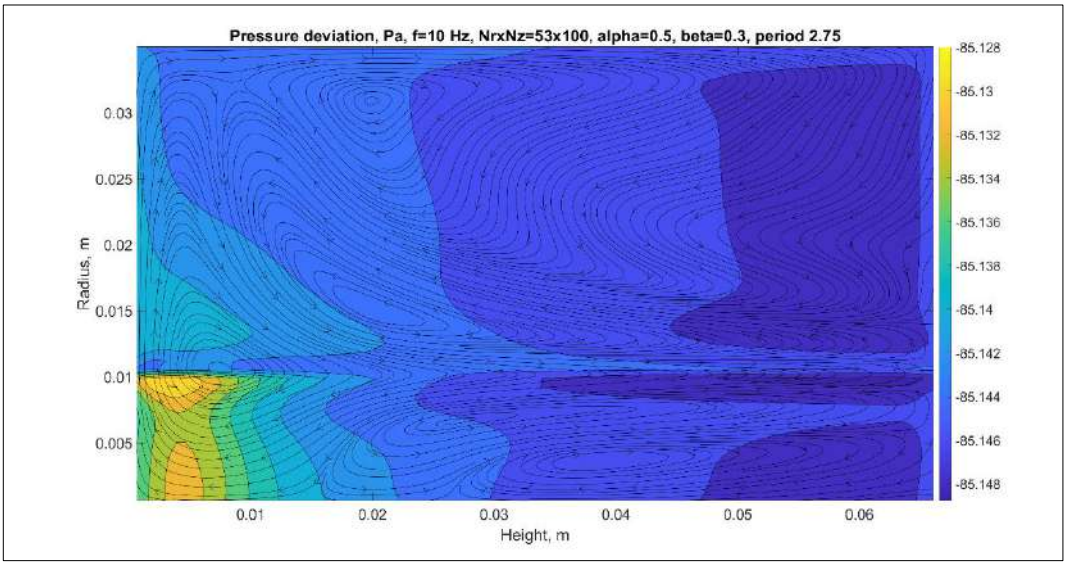


Рис. 12. Линии тока и изолинии звукового давления на частоте f=10 Гц в расчетной области на сетке 53x100, параметры α=0,5 и β=0,3, число периодов с начала колебаний – 2,75. Расчет согласно КГД
Fig. 12. Streamlines and sound pressure isolines at the frequency of 10 Hz at the calculation area on the grid 53x100, parameters α=0.5 and Courant number β=0.3, the number of periods since the beginning of oscillations – 2.75. Simulation according to QGD

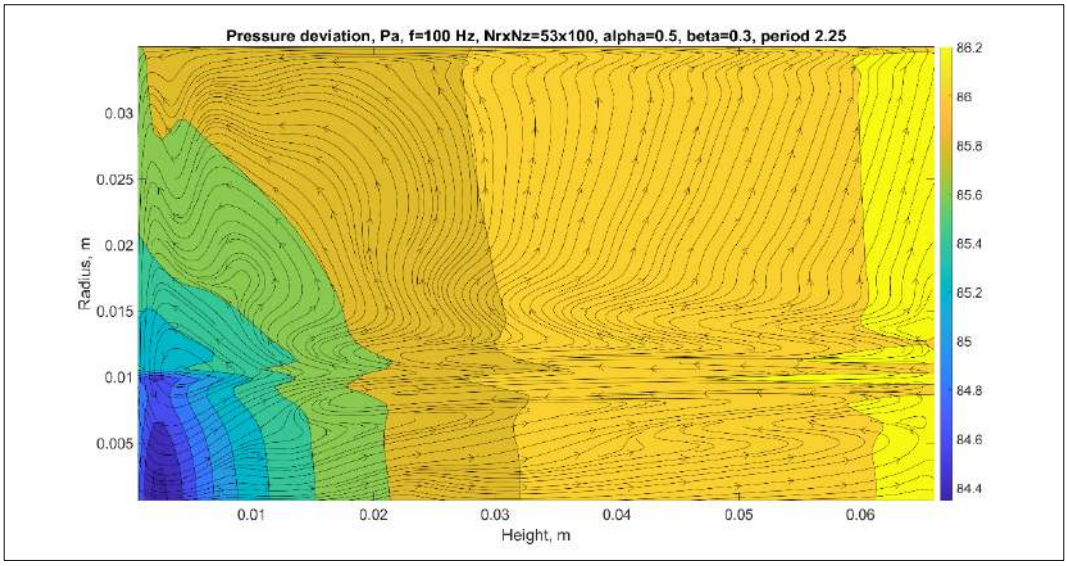


Рис. 13. Линии тока и изолинии звукового давления на частоте f=100 Гц в расчетной области на сетке 53x100, параметры α=0,5 и β=0,3, число периодов с начала колебаний – 2,25. Расчет согласно КГД

Fig. 13. Streamlines and sound pressure isolines at the frequency f=100 Hz at the calculation area on the grid 53x100, parameters α=0.5 and Courant number β=0.3, the number of periods since the beginning of oscillations – 2.25. Simulation according to QGD

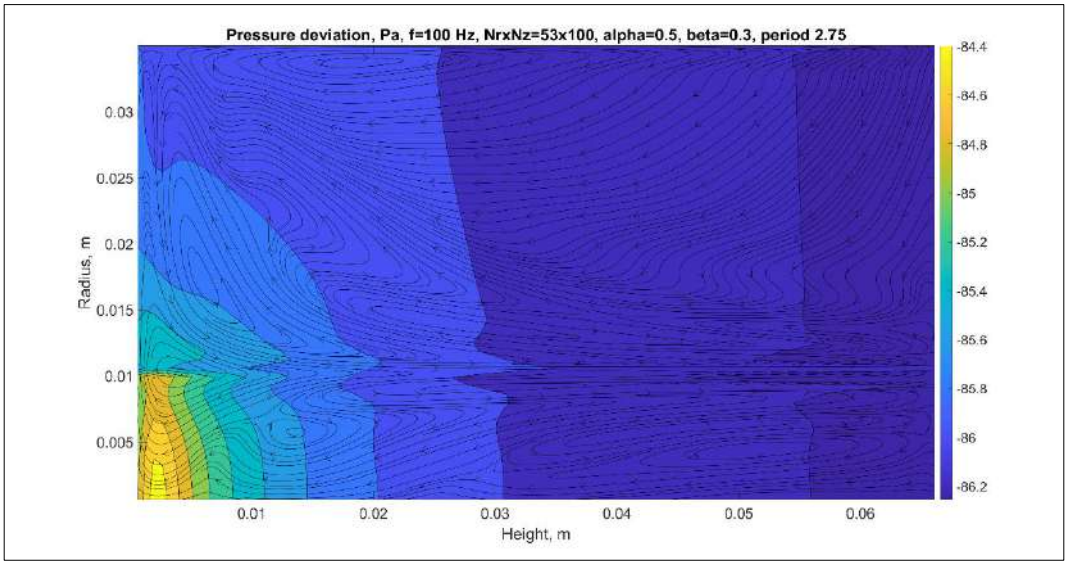


Рис. 14. Линии тока и изолинии звукового давления на частоте f=100 Гц в расчетной области на сетке 53x100, параметры α=0,5 и β=0,3, число периодов с начала колебаний – 2,75. Расчет согласно КГД

Fig. 14. Streamlines and sound pressure isolines at the frequency f=100 Hz at the calculation area on the grid 53x100, parameters α=0.5 and Courant number β=0.3, the number of periods since the beginning of oscillations – 2.75. Simulation according to QGD

На рис. 15-18 приведены распределения звукового давления и линии тока, рассчитанные по КГиД.

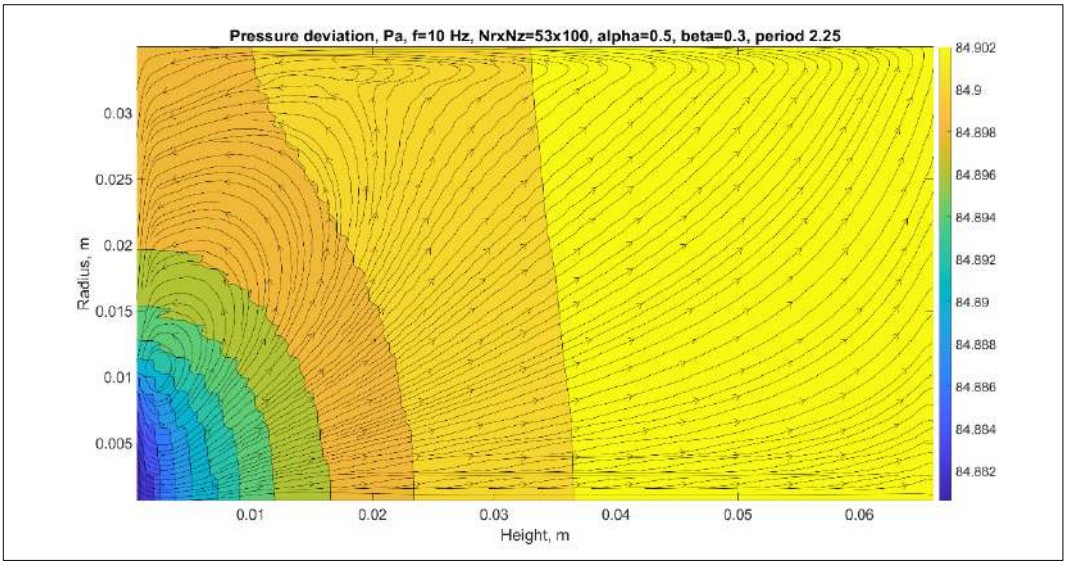


Рис. 15. Линии тока и изолинии звукового давления на частоте f=10 Гц в расчетной области на сетке 53x100, параметры α=0,5 и β=0,3, число периодов с начала колебаний – 2,25. Расчет согласно КГиД
Fig. 15. Streamlines and sound pressure isolines at the frequency f=10 Hz at the calculation area on the grid 53x100, parameters α=0.5 and Courant number β=0.3, the number of periods since the beginning of oscillations – 2.25. Simulation according to QHD

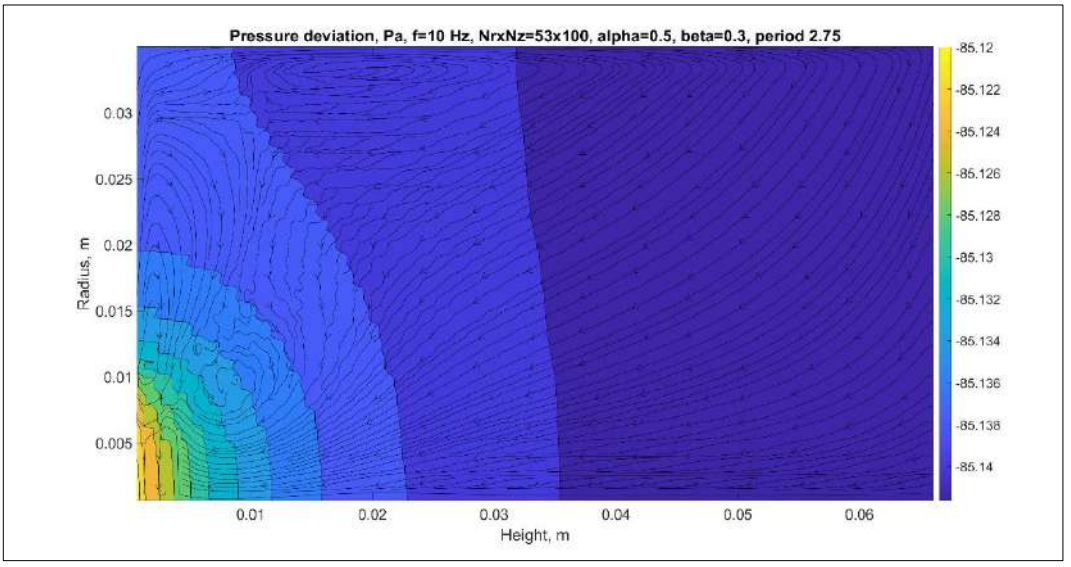


Рис. 16. Линии тока и изолинии звукового давления на частоте f=10 Гц в расчетной области на сетке 53x100, параметры α=0,5 и β=0,3, число периодов с начала колебаний – 2,75. Расчет согласно КГиД
Fig. 16. Streamlines and sound pressure isolines at the frequency f=10 Hz at the calculation area on the grid 53x100, parameters α=0.5 and Courant number β=0.3, the number of periods since the beginning of oscillations – 2.75. Simulation according to QHD

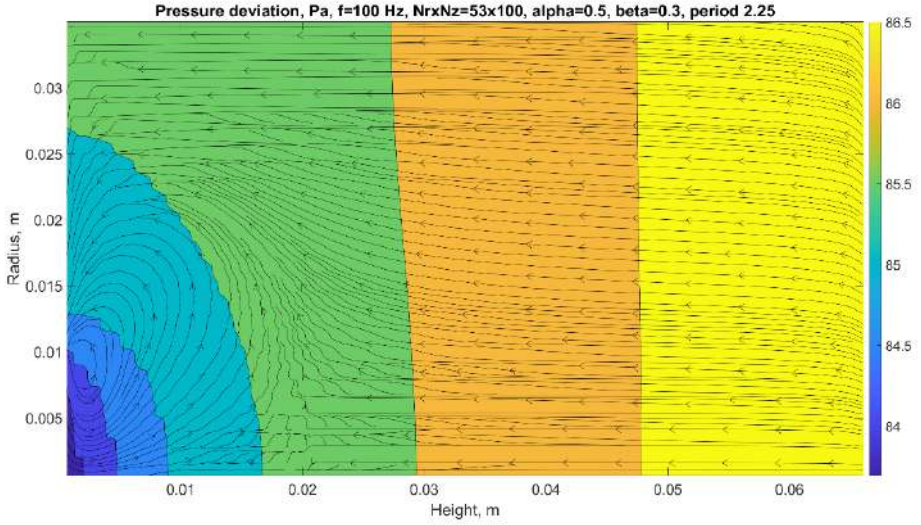


Рис. 17. Линии тока и изолинии звукового давления на частоте f=100 Гц в расчетной области на сетке 53x100, параметры α=0,5 и β=0,3, число периодов с начала колебаний – 2,25. Расчет согласно КГиД

Fig. 17. Streamlines and sound pressure isolines at the frequency f=100 Hz at the calculation area on the grid 53x100, parameters α=0.5 and Courant number β=0.3, the number of periods since the beginning of oscillations – 2.25. Simulation according to QHD

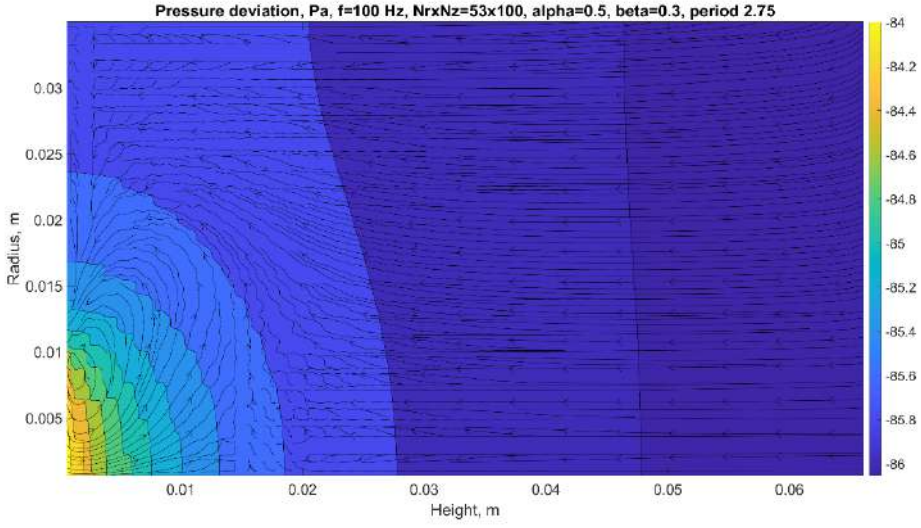


Рис. 18. Линии тока и изолинии звукового давления на частоте f=100 Гц в расчетной области на сетке 53x100, параметры α=0,5 и β=0,3, число периодов с начала колебаний – 2,75. Расчет согласно КГиД

Fig. 18. Streamlines and sound pressure isolines at the frequency f=100 Hz at the calculation area on the grid 53x100, parameters α=0.5 and Courant number β=0.3, the number of periods since the beginning of oscillations – 2.75. Simulation according to QHD

Хотя на кривых звукового давления для расчета по КГиД при α=0,5 практически нет осцилляций (только в начале первого периода колебаний, как на рис. 6-7), линии тока и изолинии звукового давления говорят о наличии схемных осцилляций на используемой сетке, как упоминалось выше. Также из приведенных выше рисунков можно видеть различие в предсказаниях изолиний звукового давления и линий тока у КГД и КГиД, но это

практически не сказывается на значении звукового давления в контрольной точке на основной частоте колебаний, в чем можно убедиться по спектрам на рис. 8-9.

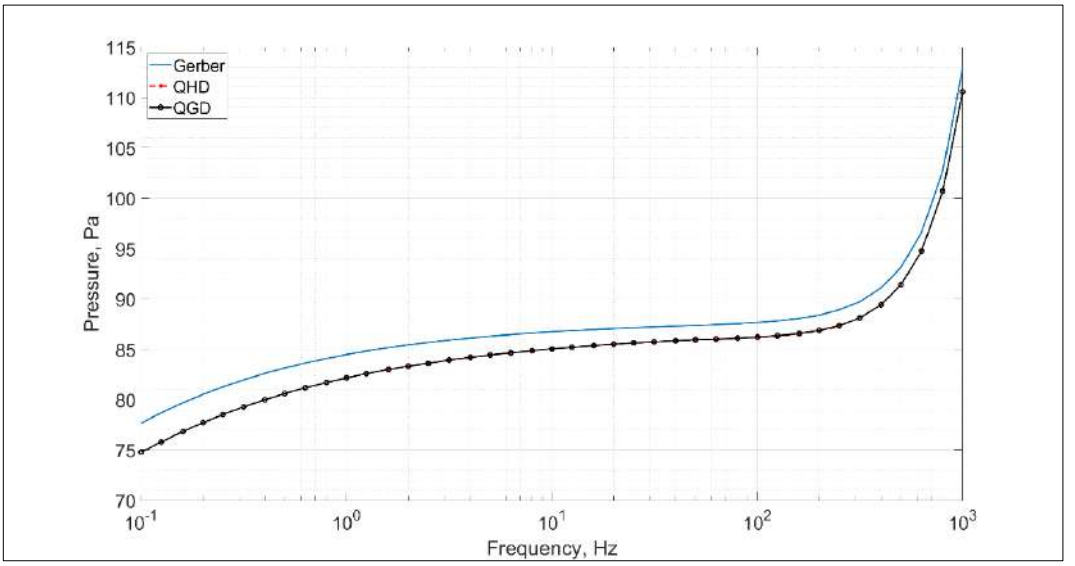


Рис. 19. Сравнение результатов численного моделирования звукового давления на сетке 53x100 с вычислением по формуле (1)
Fig. 19. Comparison of numerical simulation results of sound pressure on the grid 53x100 with calculation by the formula (1)

Звуковое давление для модели КГД вычислено в диапазоне частот 0,1 – 1000 Гц, для КГиД – от 1 до 250 Гц (на рис. 19). Это различие вызвано прежде всего тем, что для моделирования частот ниже 1 Гц требуется существенный объем времени. При этом на частотах выше 100 Гц размах схемных осцилляций распределения звукового давления для расчета по КГиД становится значительным, хотя вычислительная устойчивость сохраняется вплоть до исследованной частоты 1000 Гц. Следует отметить, что в области частот 1 – 250 Гц результаты моделирования с помощью КГиД уравнений очень хорошо совпадают со значениями, полученными применением уравнений КГД, и, основываясь на этих данных, можно ожидать, что совпадут также и на частотах 0,1 – 1 Гц.

Если предположить, что зависимость звукового давления от частоты слабо зависит от размера расчетной сетки, то, возможно, имеет смысл сравнить относительное изменение кривых звукового давления для аналитической формулы (1), КГД и КГиД:

$$\Delta = 20 \lg\left(\frac{P}{P_{ref}}\right) \text{ [дБ}/dB\text{]},$$

где Δ – изменение амплитуды звукового давления P в контрольной точке на частоте колебаний поршня относительно ее значения P_{ref} на опорной частоте 100 Гц.

Соответствующий график представлен на рис. 20. Как по нему видно, до частоты 20 Гц относительное изменение давления для численного моделирования и вычисленного по формуле (1) хорошо согласуются друг с другом, но на частоте 0,1 Гц различие составляет около 0,2 дБ (примерно 2 %). При этом результаты моделирования по КГД и КГиД в рассчитанном для КГиД диапазоне частот 1 – 250 Гц практически полностью совпали друг с другом.

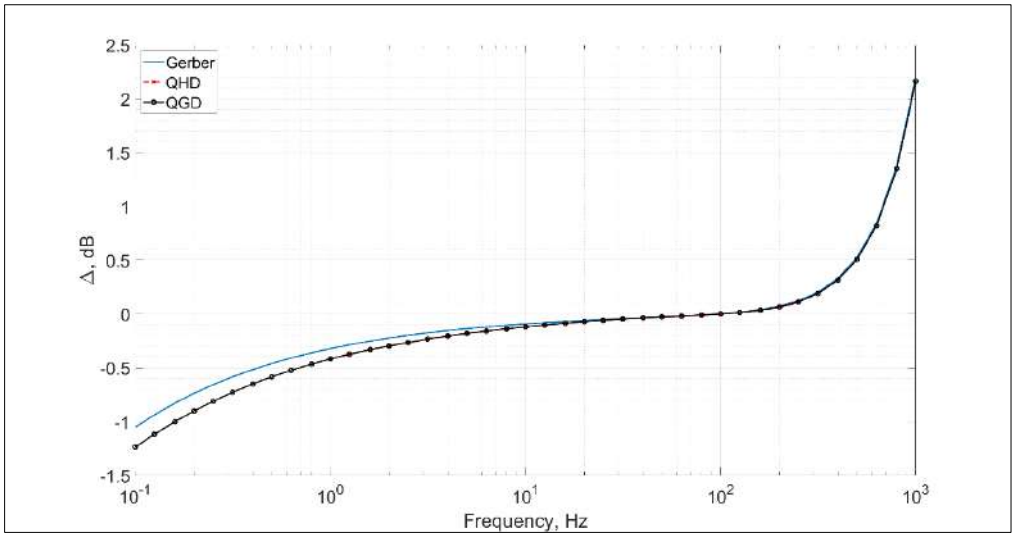


Рис. 20. Сравнение изменения величины звукового давления относительно его значения на частоте 100 Гц для КГД, КГиД и расчета по формуле (1)
Fig. 20. Comparison of changes of sound pressure value relative to its value at the frequency of 100 Hz for QGD, QHD and calculation by the formula (1)

4. Заключение

Изначально КГД и КГиД модели были выбраны из-за простоты написания явной разностной схемы, ее распараллеливания и того факта, что эти уравнения хорошо описывают нестационарные течения. И оказалось, что модели КГД и КГиД можно применять к задачам акустики без использования специальных методов и лимитеров, что появляются при моделировании акустических колебаний посредством уравнений Навье-Стокса. При этом анализ устойчивости обеих моделей для чисел Маха, характерных в этой задаче, выполнен впервые.

Полученная разность между значением, рассчитанным по формуле (1), и значениями, рассчитанными для сжимаемого вязкого теплопроводного воздуха по КГД и КГиД, оказалась меньше 0,05 дБ в диапазоне частот 20 – 1000 Гц, но по мере снижения частоты разности между аналитическим и полученными с помощью КГД и КГиД моделей значениями увеличиваются и достигают 0,2 дБ (около 2 %) на частоте 0,1 Гц. При этом, как показано выше, результаты моделирования по КГД и КГиД моделям слабо зависят от параметра α в его исследованной области значений и достаточно точно совпадают друг с другом. При подтверждении результатов моделирования с помощью эксперимента это будет свидетельствовать как в пользу корректности применения КГД и КГиД для задач акустики и сверхмедленных течений (следует напомнить, что значения чисел Маха в рассмотренной области частот – $9{,}1\cdot10^{-7}\div9{,}1\cdot10^{-3}$), так и наличия систематической погрешности для калибруемых микрофонов при использовании акустической теории Г. Гербера на инфразвуковых частотах.

Дополнительной сложностью является оценка численного моделирования – прямое сравнение с результатами эксперимента на момент написания статьи невозможно из-за подготовки пистонфона 3202 к его модернизации и расширения рабочего частотного диапазона до 0,001 – 100 Гц. Схожей задачей – по своему исполнению – является численное моделирование камер связи установки первичной калибровки микрофонов Bruel&Kjaer Type 9699, в которой реализуется метод взаимности для абсолютной калибровки измерительных лабораторных микрофонов типа LS1 и LS2. Результаты такого моделирования уже можно сравнить с экспериментом, но подобное исследование, по сути, будет являться самостоятельным, и выходит за рамки данной работы.

В заключение следует отметить, что серьезным недостатком примененной явной разностной схемы является то, что для численного моделирования частот колебаний ниже 1 Гц тратится большое количество времени: если для расчета 20 периодов колебаний с частотой 100 Гц потребовалось около 25 минут на ПК с ЦПУ Intel i9-9900K (4,8 ГГц по всем ядрам), то расчет 5 периодов колебаний давления с частотой 0,1 Гц занял примерно 3,5 суток.

Список литературы / References

[1]. Merchant B. John. Hyperion 5113/A Infrasound Sensor Evaluation. Technical Report SAND2015-8097, Sandia National Lab., 2015, 43 p.

[2]. Slad George William, Merchant Bion J. Chaparral Model 60 Infrasound Sensor Evaluation. Technical Report SAND-2016-1902, Sandia National Lab., 2016, 42 p.

[3]. Merchant Bion J., McDowell Kyle D. MB3a Infrasound Sensor Evaluation. Technical Report SAND2014-20108, Sandia National Lab., 2014, 59 p.

[4]. Коньков А.В. О методе пистонфона. Сборник научных трудов ВНИИФТРИ, вып. 23 (33), 1975 г., стр. 5–11 / Konkov A.V. On the pistonphone method. Collection of scientific papers of VNIIFTRI, issue 23 (33), 1975, pp. 5–11 (in Russian).

[5]. Головин Д.В., Коньков А.В. Влияние внешних условий на звуковое давление в камере пистонфона в инфразвуковом диапазоне частот. Измерительная техника, №9, 2020 г., стр. 67-72 / Golovin D.V., Konkov A.V. The influence of external conditions on sound pressure in the pistonphone at the infrasound frequency range. Measuring technology, №9, 2020, pp. 67-72. (in Russian).

[6]. Zhang Fan, He Wen, He Longbiao, Rong Zuochao. Acoustic properties of pistonphones at low frequencies in the presence of pressure leakage and heat conduction. Journal of Sound and Vibration, vol. 358, 2015, pp. 324-333.

[7]. He Wen, He Longbiao, Zhang Fan, Rong Zuochao, and Jia Shushi. A dedicated pistonphone for absolute calibration of infrasound sensors at very low frequencies. Measurement Science and Technology, vol. 27, no.2, article no. 025018.

[8]. Gerber H. Acoustic Properties of Fluid-Filled Chambers at Infrasonic Frequencies in the Absence of Convection. Journal of Acoustical Society of America, vol. 36, issue 8, 1964, pp. 1427–1434.

[9]. Guianvarc'h C., Durocher J.-N., Bruneau M., Bruneau A.-M. Acoustic transfer admittance of cylindrical cavities. Journal of Sound and Vibration, vol. 292, issues 3–5, 2006, pp. 595-603.

[10]. Jackett R., Avison J. Realizing the primary standard for sound pressure: The trouble with IEC 61094-2. In Proc. of the 44rd International Congress on Noise Control Engineering, 2015, 9 p.

[11]. Vincent P., Rodrigues D., Larsonnier F., Guianvarc'h C., Durand S. Acoustic transfer admittance of cylindrical cavities in infrasonic frequency range. Metrologia, vol. 56, no. 1, 2019, article no. 015003.

[12]. IEC 61094-2:2009 Electroacoustics - Measurement microphones - Primary method for pressure calibration of laboratory standard microphones by the reciprocity technique. International Electrical Commission, Geneva, 2009

[13]. Елизарова Т.Г. Квазигазодинамические уравнения и методы расчета вязких течений. М., Научный Мир, 2007 г., 350 стр. / Elizarova T.G. Quasi-Gas Dynamic Equations. Springer Science & Business Media, 2009, 286 p.

[14]. Шеретов Ю.В. Динамика сплошных сред при пространственно-временно́м осреднении. М., Ижевск, НИЦ «Регулярная и хаотическая динамика», 2009 г., 400 стр. / Sheretov Y.V. Continuum dynamics under spatiotemporal averaging. M., Izhevsk, Scientific and Publishing Center «Regular and Chaotic Dynamics», 2009, 400 p. (in Russian).

Информация об авторе / Information about author

Дмитрий Витальевич ГОЛОВИН – младший научный сотрудник отдела акустики. Его научные интересы включают калибровку средств измерения звукового давления в воздухе на инфразвуковых частотах и численное моделирование задач акустики.

Dmitrii Vital`evich GOLOVIN – researcher of acoustic laboratory. His research interests include calibration of measuring instruments of sound pressure in the air at infrasound frequencies and simulation of applied acoustics.