ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 32 Выпуск 1

ISSN Online 2220-6426
Volume 32 Issue 1

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(1)-1

Интеграция мандатного и ролевого управления доступом и мандатного контроля целостности в верифицированной иерархической модели безопасности операционной системы

[1] *П.Н. Девянин, ORCID: 0000-0003-2561-794X <devyanin.peter@yandex.ru>*
[2,3,4] *В.В. Кулямин, ORCID: 0000-0003-3439-9534 <kuliamin@ispras.ru>*
[2,3,4] *А.К. Петренко, ORCID: 0000-0001-7411-3831 <petrenko@ispras.ru>*
[2,3,4,5] *А.В. Хорошилов, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*
[2] *И.В. Щепетков, ORCID: 0000-0002-5794-004X <shchepetkov@ispras.ru>*

[1] *РусБИТех,*
117105, Россия, Москва, Варшавское шоссе, д. 26, стр.11
[2] *Институт системного программирования имени В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[3] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1
[4] *НИУ Высшая школа экономики,*
101978, Россия, г. Москва, ул. Мясницкая, д. 20
[5] *Московский физико-технический институт,*
141701, Россия, Московская область, г. Долгопрудный, Институтский пер., 9

Аннотация. Проектирование механизма управления доступом в операционной системе (ОС), требующей высокого уровня доверия, является сложной задачей. Еще более сложной она стан овится, если требуется интеграция нескольких разнородных механизмов, таких как ролевое управление доступом (role-based access control, RBAC) и мандатное управление доступом (mandatory access control, MAC). Данная статья представляет результаты разработки иерархической интегрированной модели управления доступом и информационных потоков (hierarchical integrated model of access control and information flows, HIMACF), интегрирующей механизмы RBAC, MAC и мандатного контроля целостности (mandatory integrity control, MIC) с сохранением их ключевых свойств безопасности. Эта модель является дальнейшим развитием МРОСЛ-ДП-модели, она формализована на языке Event-B и ее корректность доказана формально. В иерархическом представлении модели каждый ее уровень (или модуль) представляет отдельный механизм управления доступом, благодаря чему модель может быть верифицирована помодульно, что снижает общую трудоемкость ее верификации за счет переиспользования при доказательстве корректности очередного уровня результатов верификации нижележащих уровней. Данная модель реализована в ОС Astra Linux Special Edition на основе инфраструктуры Linux Security Modules (LSM).

Ключевые слова: формальная модель управления доступом; управление доступом на основе ролей; мандатный контроль целостности; мандатное управление доступом; формальная верификация модели; Event-B; Astra Linux

Для цитирования: Девянин П.Н., Кулямин В.В., Петренко А.К., Хорошилов А.В., Щепетков И.В. Интеграция мандатного и ролевого управления доступом и мандатного контроля целостности в верифицированной иерархической модели безопасности операционной системы. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 7-26. DOI: 10.15514/ISPRAS–2020–32(1)–1

Благодарности. Данная работа поддержана грантом РФФИ по проекту 18-01-00378.

Integrating RBAC, MIC, and MLS in Verified Hierarchical Security Model for Operating System

[1] *P.N. Devyanin, ORCID: 0000-0003-2561-794X <devyanin.peter@yandex.ru>*
[2,3,4] *V.V. Kuliamin, ORCID: 0000-0003-3439-9534 <kuliamin@ispras.ru>*
[2,3,4] *A.K. Petrenko, ORCID: 0000-0001-7411-3831 <petrenko@ispras.ru>*
[2,3,4,5] *A.V. Khoroshilov, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*
[2] *I.V. Shchepetkov, ORCID: 0000-0002-5794-004X <shchepetkov@ispras.ru>*

[1] *RusBITech*
26, Warsaw highway, Moscow, 117105, Russia
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
[3] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[4] *National Research University, Higher School of Economics*
20, Myasnitskaya Ulitsa, Moscow, 101978, Russia
[5] *Moscow Institute of Physics and Technology (State University),*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russian Federation

Abstract. Designing a trusted access control mechanism of an operating system (OS) is a complex task if the goal is to achieve high level of security assurance and guarantees of unwanted information flows absence. Even more complex it becomes when the integration of several heterogeneous mechanisms, like role-based access control (RBAC), mandatory integrity control (MIC), and multi-level security (MLS) is considered. This paper presents results of developement of a hierarchical integrated model of access control and information flows (HIMACF), which provides a holistic integration of RBAC, MIC, and MLS preserving key security properties of all those mechanisms. Previous version of this model is called MROSL~DP-model. Now the model is formalized using Event-B formal method and its correctness is formally verified. In the hierarchical representation of the model, each hierarchy level (module) corresponds to a separate security control mechanism, so the model can be verified with less effort reusing the results of verification of lower level modules. The model is implemented in a Linux-based operating system using the Linux Security Modules infrastructure.

Keywords: Formal access control model; role-based access control; mandatory integrity control; multi-level security; formal verification; Event-B; Astra Linux

For citation: Devyanin P.N., Kuliamin V.V., Petrenko A.K., Khoroshilov A.V., Shchepetkov I.V. Integrating RBAC, MIC, and MLS in Verified Hierarchical Security Model for Operating System. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 7-26 (in Russian). DOI: 10.15514/ISPRAS–2020–32(1)–1

Acknowledgments. This study is supported by the Russian Foundation for Basic Research grant #18-01-00378.

1. Введение

Современные операционные системы (ОС) управляют достаточно сложно организованными массивами информации, выполняют множество гетерогенных функций и часто должны иметь дело с разветвленными иерархиями прав на выполнение разнообразных действий. В таком окружении использование обычного *дискреционного управления доступом* (discretionary access control, DAC) становится неудобным и требует чрезвычайно сложных действий для осуществления строгого контроля за потоками информации. Более гибким и масштабируемым с точки зрения администрирования механизмом является *управление доступом на основе ролей* (role-based access control, RBAC) [1]. Более строгие гарантии изоляции информации с разными правами доступа предоставляет *мандатное управление доступом* (mandatory access control, MAC). Еще

одной важной техникой изоляции информации и предотвращения ущерба от успешных атак является *мандатный контроль целостности* (mandatory integrity control, MIC).

Все три указанных механизма в различных комбинациях используются в современных ОС. RBAC и MAC (в виде так называемой многоуровневой защиты, multi-level security, MLS) могут быть прямо реализованы в рамках широко известной инфраструктуры защиты ОС Linux, SELinux [2]. Эта инфраструктура используется, например, в мобильной ОС Android, начиная с версии 5.0 [3]. Реализация многоуровневого MIC имеется в ОС Windows, начиная с Windows Vista [4]. Специфическая реализация контроля целостности поддерживается в macOS [5].

Каждый из этих механизмов имеет формальную модель, задающую его строгую семантику и обеспечиваемые свойства безопасности. Для MAC такой моделью является знаменитая модель Белла-ЛаПадулы (D.E. Bell, L.J. LaPadula, BLP) [6], для MIC – модель Биба (K.J. Biba) [7]. Разнообразные модели RBAC описаны в работах Сандху (R.S. Sandhu) с соавторами [1].

Однако, в опубликованных работах, кроме публикаций о предыдущих версиях HIMACF, не было представлено формальной модели безопасности, явным образом объединяющей механизмы RBAC, MAC и MIC, с сохранением их ключевых свойств безопасности, подходящей для реализации в рамках ОС. Работа [8], например, описывает, как смоделировать MAC и MIC с помощью механизма RBAC, пополненного специальными ограничениями, однако представленная в ней техника достаточно неудобна для администрирования в терминах уровней целостности или конфиденциальности. Классические простые модели недостаточны как формальное представление такого интегрированного механизма, поскольку современные ОС управляют большим разнообразием объектов с различными требованиями к ограничениям доступа к ним. Простое механическое объединение нескольких разнородных механизмов может приводить к нарушению их ключевых свойств безопасности и возникновению нежелательных потоков информации, например, использование специальных ролей для доступа к высококонфиденциальной информации может привести к ее утечке, если сами эти роли доступны для анализа пользователям с низкой конфиденциальностью.

В данной статье представлены текущие результаты разработки и верификации формальной модели, называемой иерархической интегрированной моделью управления доступом и информационных потоков (hierarchical integrated model of access control and information flows, HIMACF, ранние версии ее имели название МРОСЛ ДП-модель). [9-10]. Эта модель корректно интегрирует механизмы RBAC, MIC и MAC, сохраняя их ключевые свойства безопасности. Под ключевыми свойствами безопасности имеются в виду базовые свойства, обеспечиваемые проектными решениями в рамках рассматриваемого механизма, например, возможность доступа субъекта к объекту только через подходящую роль в RBAC, или возможность чтения субъектом объекта только при наличии у первого более высокого или равного уровня доступа в MAC. Наша модель также поддерживает тонко настраиваемое управление доступом к данным сложной структуры, которые встречаются в современных ОС. Она имеет описание на языке Event-B [11] и формально верифицирована с помощью поддерживающей этот язык среды Rodin [12]. Все ее свойства безопасности представлены как инварианты модели и строго доказаны. Представление модели иерархично, она состоит из нескольких модулей, уточняющих друг друга, причем каждый механизм описан в отдельном модуле. Такая структура позволяет представить свойства отдельного механизма в наиболее ясном виде и облегчает верификацию модели за счет того, что свойства, верифицированные в одном модуле, переиспользуются в уточняющих его модулях (не требуется их повторное доказательство). Представленная модель предназначена для использования как модель политик безопасности ОС, наличие

которой требуется стандартом Common Criteria [13-14] и нормативными документами ФСТЭК России для систем с высокими уровнями доверия. Модель частично реализована в построенной на основе ядра Linux ОС Astra Linux Special Edition [15]. Хотя в настоящее время формальное соответствие между моделью и ее реализацией в коде не может быть строго доказано, мы продолжаем исследования, нацеленные на достижение этой цели.

Статья организована следующим образом. В следующем разделе рассматриваются механизмы RBAC, MAC и MIC, их формальные модели, а также некоторые модификации, представленные в литературе. Общее описание HIMACF дано в разд. 3. Детальное формальное представление некоторых элементов модели представлено в разд. 4. Разд. 5 описывает особенности верификации модели, а заключение завершает статью и представляет некоторые дальнейшие расширения и доработки модели.

2. Базовые механизмы

В данном разделе мы напоминаем основные свойства RBAC, MIC, и MAC, а также формализующие их модели. Некоторые понятия и обозначения из этих моделей используются нами в модели HIMACF далее.

Далее процессы в ОС, способные получать доступ к данным, называются *субъектами* или *сессиями* (предпочтение отдается термину, используемому в оригинальном изложении соответствующей модели). Множество субъектов обозначается S. Данные, к которым может быть получен доступ, считаются размещенными в *объектах*, или *сущностях* системы, множество всех объектов обозначается O. Все рассматриваемые далее модели накладывают определенные ограничения на возможные виды доступа субъектов к объектам.

2.1 Модели управления доступом на основе ролей

Управление доступом на основе ролей [1] ограничивает доступ сессий к объектам с помощью *ролей* R, приписываемых сессиям, и *прав доступа* P, определяющих возможные доступы к объектам (на чтение, запись, исполнение или др. определенного объекта). Каждая роль имеет множество прав доступа, задаваемое отношением $PA \subseteq R \times P$. Пользователи системы образуют множество U. Каждый пользователь имеет множество ролей, задаваемое отношением $UA \subseteq U \times R$. Каждая сессия приписана к определенному пользователю с помощью функции $user: S \to U$ и имеет множество ролей, задаваемое отображением $roles: S \to 2^R$. На множестве ролей задан частичный порядок $RH \subseteq R \times R$, определяющий наследование ролей, роль r *наследует* r' тогда и только тогда, когда $(r, r') \in RH$. Иерархия ролей позволяет более удобно организовать права доступа сессий. Множество ролей сессии ограничено множеством ролей ее пользователя и теми, которые наследуются ими, $s \in S \Rightarrow roles(s) \subseteq \{r \mid \exists r'(r, r') \in RH \wedge (user(s), r') \in UA\}$. Состояние системы описывается кортежем $(U, R, P, S, UA, PA, RH, user, roles)$.

Модель ARBAC [18] расширяет RBAC при помощи *административных ролей* AR, которые могут иметь права на модификацию отношений UA, PA и RH. Множество административных прав обозначается AP. Предполагается, что административные и обычные права, так же, как административные и обычные роли, не пересекаются $AR \cap R = AP \cap P = \emptyset$. Привязка прав к административным ролям обозначается $APA \subseteq AR \times AP$. Привязка административных ролей к пользователям задается отношением $AUA \subseteq U \times AR$. На административных ролях также определена иерархия $ARH \subseteq AR \times AR$, являющаяся частичным порядком. Привязка ролей к сессиям переопределяется как $roles: S \to 2^{R \cup AR}$, и ограничение выглядит так: $s \in S \Rightarrow roles(s) \subseteq \{r \mid \exists r'(r, r') \in RH \cup ARH \wedge (user(s), r') \in UA \cup AUA\}$. Состояние системы задается кортежем $(U, R, AR, P, AP, S, UA, \$\$AUA, PA, APA, RH, ARH, user, roles)$.

Более детальная модель RBAC, включающая ограничения на возможные переходы между состояниями системы, описана стандартом ANSI RBAC [16].

2.2 Модель Белла-ЛаПадулы

Первая и наиболее известная модель MAC предложена Беллом и ЛаПадулой [6, 17]. В ней имеются множества субъектов S, объектов O и атрибутов доступа A. Объекты могут участвовать в отношении иерархии H. Для управления доступом используются *уровни защиты* L, имеющие частичный порядок на них. Уровень защиты объекта задается функцией $f_o: O \to L$, максимальный уровень доступа субъекта задается функцией $f_s: S \to L$, а текущий уровень доступа субъекта задан с помощью функции $f_c: S \to L$. Предполагается, что $f_c(s) \leq f_s(s)$ для всех $s \in S$. Права доступа определены с помощью матрицы прав доступа $M: S \times O \to 2^A$. Текущие доступы субъектов к объектам заданы отношением $B \subseteq S \times O \times A$. Состояние системы моделируется как (B, M, f_s, f_c, f_o, H) и называется безопасным, если выполнены следующие правила.

- Правило простой безопасности (ss-правило) требует, чтобы субъект s имел доступ на чтение к объекту o, только если $f_s(s) \geq f_o(o)$. Для доступа на запись (рассматриваемого здесь как возможность и читать, и модифицировать информацию), это тоже должно быть выполнено.
- *-правило требует, чтобы субъект s мог иметь доступ на чтение к объекту o, только если $f_c(s) \geq f_o(o)$, и s мог иметь доступ на модификацию (без чтения) к o, только если $f_c(s) \leq f_o(o)$. Для получения доступа на запись (чтение с модификацией), требуется $f_c(s) = f_o(o)$.
- Правило дискреционной безопасности (ds-правило) требует, чтобы субъект s мог иметь доступ a к объекту o, только если $a \in M(s, o\}$

Система называется безопасной, если все ее достижимые состояния безопасны. В работе [6] даны ограничения на переходы между состояниями, достаточные для того, чтобы система с безопасным начальным состоянием была безопасной.

2.3 Модель Биба

Наиболее известной моделью MIC является модель Биба [7]. Она использует упорядоченные уровни целостности I и функцию $i: O \cup S \to I$, приписывающую уровень целостности каждому объекту и субъекту системы. Модель рассматривает различные политики, ограничивающие возможные переходы между состояниями системы. Широко известна политика строгой целостности, требующая сохранения уровней целостности объектов и субъектов, а также выполнения следующих правил.

1. Субъект s может иметь доступ на чтение к объекту o, только если $i(s) \leq i(o)$.
2. Субъект s может иметь доступ на модификацию к объекту o, только если $i(o) \leq i(s)$.
3. Субъект s может иметь доступ на обращение (управление) к другому субъекту s', только если $i(s') \leq i(s)$.

В современных ОС обычно реализуется кольцевая политика, требующая выполнения правил 2 и 3.

2.4 Детализация ограничений MAC и анализ информационных потоков

В этом подразделе дается краткий обзор развития и детализации моделей мандатного управления доступом, которые делают этот механизм более практичным при использовании в современных ОС, управляющих сложными данными. Некоторые

элементы этих моделей использованы в HIMACF. Сущностями далее называются субъекты и объекты системы.

Распространение информации при работе системы недостаточно аккуратно отражается только операциями по контролю доступа. Для более точного его моделирования можно использовать понятие *информационных потоков*, в том виде, как это сделано в известной модели Take-Grant (TGM) [18-19]. Эта модель рассматривает непрямой доступ субъектов к информации через объекты и другие субъекты, который возникает в результате последовательных чтений и модификаций. За счет этого TGM полезна для анализа распространения информации и ее возможного перехода между уровнями защиты. В TGM обычные ограничения контроля доступа (как ss-правило или *-правило) названы *правилами de jure*, и они дополнены правилами *de facto*, описывающими распространение информационных потоков. Информационный поток на запись от сущности x к сущности y обозначает любой возможный способ распространения информации от x к y, например, если субъект x получает доступ на запись к y, или субъект y получает доступ на чтение к x, или другой субъект z имеет доступ на чтение к x и доступ на запись y, возможно, перенося часть данных из x к y, и т.п. Наличие информационного потока на запись от x к y эквивалентно наличию информационного потока на чтение от y к x.

- *Post-правило*: если субъект x имеет доступ или информационный поток на чтение к сущности (объекту или субъекту) z, и субъект y имеет доступ или информационный поток на запись к z, то x имеет поток на чтение к y.
- *Pass-правило*: если субъект y имеет доступ или поток на чтение к сущности x и доступ или поток на запись к сущности z, то x имеет поток на чтение к z.
- *Spy-правило*: если субъект x имеет доступ или поток на чтение к субъекту y, имеющему доступ или поток на чтение к z, то x имеет поток на чтение к z.
- *Find-правило*: если субъект z имеет доступ или поток на запись к y, имеющему доступ или поток на запись к x, то x имеет поток на чтение к z.

Видно, что эти правила эквивалентны созданию потока на чтение из каждого доступа на чтение, созданию обратного потока на чтение из каждого доступа на запись, и транзитивному замыканию получаемых потоков на чтение.

Иерархичность данных под управлением ОС также влияет на правила управления доступом. Такое влияние аккуратно описано в модели военных сообщений (Military Message System, MMS) [20], расширяющей модель Белла-ЛаПадулы. В этой модели *контейнеры* являются объектами, которые используются для доступа к другим объектам, поэтому правила контроля доступа к объекту должны учитывать уровень защиты содержащего его контейнера. В MMS требуется, чтобы контейнер имел уровень защиты больший или равный максимальному уровню защиты содержащихся в нем объектов. При этом доступ к содержащимся в контейнере объектам может быть организован по-разному. При выставленном флаге CCR (*container clearance required*) у контейнера, доступ субъекта к содержимому контейнера возможен только при возможности доступа к самому контейнеру. При опущенном флаге CCR у контейнера, доступ субъекта к содержащемуся в контейнере объекту возможен при соблюдении лишь ограничений доступа к самому этому объекту. Помимо этого, модель MMS позволяет доступ к объекту через ссылки на него при выполнении тех же ограничений, которые должны выполняться при прямом доступе к этому объекту. В этой модели пользователь может обладать множеством ролей с различными правами доступа. Модификация ролей пользователя и прав ролей возможна только от имени специальной роли *офицера защиты*. Кроме того, введена особая роль, позволяющая снижать уровень защиты объектов при необходимости. Однако, помимо перечисленных правил, MMS не специфицирует других ограничений на администрирование пользователей, ролей и объектов.

2.5 Обзор SELinux

Security-Enhanced Linux (SELinux) [2, 21] является инфраструктурой управления доступом, реализованной в рамках ядра ОС Linux на основе Linux Security Modules (LSM). LSM определяет набор функций-перехватчиков, которые могут выполнять проверки прав доступа, дополнительные к стандартному для Unix механизму дискреционного управления доступом. Вызовы этих функций-перехватчиков распределены по коду ядра ОС Linux так, что каждый раз при выполнении операции доступа выполняется обращение к нужному перехватчику (за корректность размещения вызовов отвечают разработчики ядра, так что доверие к этой инфраструктуре основано на доверии к компетентности и аккуратности группы поддержки ядра ОС Linux). SELinux предоставляет реализацию нескольких механизмов контроля доступа, а именно, RBAC, MAC, а также контроля доступа на основе типов (type enforcement, TE), в виде общего механизма контроля выполнения правил политики, при этом политика задается извне, сам по себе механизм SELinux никаких конкретных правил не реализует.

Политика безопасности описывается на специализированном языке, каждое правило политики может задавать ограничение на выполнение определенных операций над определенному классами сущностей ОС на основе атрибутов, входящих в метки безопасности. В последних версиях SELinux определено примерно 50 классов сущностей, включающих процессы, директории, файлы и сокеты различных типов, сетевые интерфейсы, семафоры, очереди и пр. Для каждого класса определено от 5 до 20 операций. Метка безопасности сущности включает идентификатор пользователя, роль, тип, уровень и множество категорий. Помимо ограничений на выполнение операций могут быть заданы правила присваивания и трансформации меток.

В дополнение, SELinux поддерживает контроль доступа к данным внутри приложений, если приложение позволяет осуществлять такой контроль в дополнение к обычному контролю доступа в ядре ОС. Примерами подобных приложений являются системы управления базами данных (СУБД), менеджеры графического интерфейса, разнообразное программное обеспечение промежуточного уровня. На текущий момент SELinux поддерживает набор специальных классов для сущностей, связанных с внутренними объектами PostgreSQL [22], D-Bus [23] и X Window System [24].

Язык описания политик SELinux достаточно выразителен и позволяет использовать совместно атрибуты, относящиеся к механизмам RBAC, MAC или TE, в произвольных выражениях, которые можно построить из логических связок, операторов сопоставления атрибутов с константами, друг с другом, операторов сравнения для уровней, значения которых считаются упорядоченными, и операторов включения для множеств категорий. Поэтому анализ и валидация политик безопасности SELinux являются очень сложными задачами. Существуют некоторые инструменты для этого (например, [25]), но они поддерживают только подмножества языка политик и недостаточны для верификации специфических свойств безопасности, которые политика должна обеспечивать.

В нескольких работах рассматривается формальное моделирование и дальнейший анализ свойств политик SELinux. Работа [26] представляет формальный язык (имеющий строго определенную семантику) для описания политик с использованием элементов TE и RBAC, но не MAC. В статье [27] этот язык использован для построения алгоритмов и инструментов анализа политик. В работе [28] механизм SELinux вкладывается в известную модель безопасности Харрисона-Руззо-Ульмана (M.A. Harrison, W.L. Ruzzo, J.D. Ullman) [29], однако, поскольку анализ достижимости небезопасных состояний в этой модели неразрешим алгоритмически, практической пользы от этого не много.

Работа [30] предлагает формальную модель для анализа политик SELinux, включающих ограничения MAC. Эта модель использует два вида правил: ограничивающие доступ

субъектов к объектам и ограничивающие модификацию меток безопасности объектов субъектами. В указанной работе эта модель используется для анализа информационных потоков между различными уровнями защиты с помощью среды языка Prolog.

Обзорная статья [25] рассматривает свойства языка описания политик SELinux на основе общих свойств таких языков, проанализированных в работе [31]. Язык называется *безопасным*, если на нем запрос с меньшими входными данными приводит к более слабым решениям (относительно некоторого естественного порядка на решениях), чем запрос с большими входными данными. Язык обладает свойством *независимой композиции*, если результат, полученный при использовании всех правил политики такой же, как и результат последовательного применения каждого из правил в некотором порядке. Язык *монотонный*, если добавление правила в политику не изменяет решения с разрешения на запрет доступа. Язык политик SELinux в этих терминах обладает свойством независимой композиции, но является немонотонным и небезопасным.

3. Общее описание модели HIMACF

Модель HIMACF (рис. 1) интегрирует механизмы RBAC, MAC и MIC. Она также включает аналогичные TGM правила построения информационных потоков, которые позволяют анализировать последствия наличия скрытых каналов передачи информации или информационных атак и обеспечивают проверку выполнения определенных гарантий безопасности даже в случае успешных атак на систему на уровне низкоцелостных процессов. Модель формализована на языке Event-B и состоит из четырех модулей, соответствующих описываемым механизмам. Первый модуль или базовый уровень модели описывает RBAC, второй – MIC, третий – MAC, четвертый – информационные потоки. Каждый следующий модуль уточняет предыдущий.

Event-B [11] позволяет описывать моделируемую систему как абстрактный *автомат* с помощью *переменных*, комбинации значений которых соответствуют состояниям автомата, а типы могут быть описаны в обычной теории множеств, операций или *событий*, соответствующих переходам между состояниями и изменяющим значения переменных, и *инвариантов*, описывающих свойства переменных, сохраняемые при переходах. Событие определяется при помощи его имени, параметров, *предусловия* и *постусловия*. Предусловие и постусловие вместе определяют *контракт события*. Предусловие описывает ограничения на значения параметров и переменных состояния, которые должны выполняться при срабатывании данного события. Одно такое ограничение называется *охранным условием*. Постусловие описывает изменения переменных состояния при срабатывании события. Одно такое правило, задающее изменение одной переменной, называется действием.

Рис. 1. Структура HIMACF
Fig. 1. HIMACF Structure

Уточнение одной моделью другой означает, что переменные, инварианты, события и контракты событий уточняемой модели наследуются уточняющей. При этом в уточняющей модели могут быть добавлены новые переменные, инварианты и события. Кроме того, в постусловиях унаследованных событий могут быть добавлены новые действия, описывающие изменения новых переменных, а также могут быть добавлены новые параметры и в предусловиях могут быть добавлены новые охранные условия. Корректно реализованное уточнение обеспечивает выполнение в уточняющей модели всех инвариантов уточняемой, поскольку их доказательства остаются валидными в уточняющей модели.

3.1 Основные идеи

Описание RBAC на базовом уровне HIMACF похоже на модель ARBAC [32]. Оно использует множества *пользователей*, *субъектов* (представляющих процессы ОС), *сущности* (представляющие любые объекты доступа, включая файлы, сокеты, устройства, очереди, семафоры, и пр.), *роли*. Сущности могут быть *контейнерами*, т.е., сущностями, способными содержать другие сущности, либо простыми *объектами*. объекты и контейнеры не пересекаются. Роли могут быть *обычными*, имеющими права обычных типов ($read, write, execute, own$), или *административными*, имеющие права на приписывание или удаление ролей, а также на модификацию прав ролей. Множества обычных и административных ролей не пересекаются.

Права доступа определенного типа или права владения представлены как *обычные права*. *Владельцем* сущности является роль, имеющая права менять права доступа к этой сущности. Владение моделируется при помощи права own. У сущности может быть не более одного владельца. *Административные права* представляют возможность приписывать обычную или административную роль субъекту или возможность модифицировать права роли. Субъект, имеющий (через административную роль) доступ $read$ к роли, может использовать ее права, т.е., наличие этого доступа к роли моделирует привязку роли к субъекту. Субъект, имеющий доступ $write$ к роли, может модифицировать ее права.

Каждый контейнер имеет строковое имя. Объект может иметь различные имена в разных контейнерах, но имя объекта внутри контейнера уникально. Иерархия сущностей ациклична, т.е., контейнер не может прямо или косвенно содержаться в себе самом. Есть единственный корень иерархии сущностей, который не содержится ни в каком контейнере.

Субъекты также имеют иерархию, субъект может иметь не более одного *родителя*, что моделирует соответствующее отношение между процессами в Unix. Эта иерархия ациклична, но может иметь несколько корневых субъектов. Для каждого субъекта имеется *роль-владелец*, имеющая права на любые операции над этим субъектом.

Роли имеют свою иерархию, иерархии обычных и административных ролей не пересекаются. У роли может быть несколько *родителей*. Иерархия ролей ациклична и может иметь несколько корней. Субъект, к которому приписана некоторая роль, имеет права этой роли, а также ее родителей и всех более далеких предков. Роли имеют строковые имена, имя роли уникально среди всего множества ролей. Есть выделенное множество *специальных административных ролей*, включающее роли, имеющие права администрирования пользователей, сущностей, субъектов и ролей. Специальные административные роли не могут быть созданы или удалены, переименованы, их права не могут быть модифицированы. Также постулируется существование специальных пользовательских ролей, имеющих права на операции, специфичные для данного

пользователя, и общих ролей, имеющих права на действия, которые может выполнить любой пользователь.

Состояние модуля RBAC задается множествами пользователей, сущностей, субъектов, ролей, привязкой (обычных и административных) ролей к субъектам, иерархиями сущностей, субъектов и ролей, отображением владения субъектов в пользователей, а также текущими (обычными и административными) доступами субъектов к сущностям и ролям.

В модуле RBAC определены события создания и удаления пользователей, сущностей, субъектов и ролей, добавления имеющейся сущности в контейнер (создания жесткой ссылки), переименования сущности или роли, модификации множества ролей, приписанных к субъекту, модификации прав роли, получения обычного доступа субъекта к сущности или административного доступа субъекта к роли.

Ключевое свойство безопасности механизма RBAC состоит в том, что для получения доступа к чему-либо субъект должен иметь привязанную роль, имеющую права на этот доступ. Это ограничение зафиксировано в предусловиях событий получения доступа, оно не может быть оформлено в виде инварианта, поскольку субъект может получить доступ к сущности через определенную роль, а затем потерять эту роль, не теряя доступ. Аналогично, после получения доступа через роль эта роль может потерять право на этот доступ.

Модуль, моделирующий механизм MIC, определяет *уровни целостности* (включающие, по крайней мере, low и $high$) и добавляет в состояние отображение пользователей, сущностей, субъектов и ролей в уровни целостности. Высокоцелостные пользователи (имеющие уровень целостности $high$) могут запускать от своего имени как высокоцелостные, так и низкоцелостные (с уровнем low) субъекты. Низкоцелостные пользователи могут запускать только низкоцелостные субъекты. Кроме того, низкоцелостный субъект может быть родителем только низкоцелостных субъектов.

Для каждого пользователя есть специальные низкоцелостные обычная и административная роли, имеющие права на все операции, которые этот пользователь может выполнить на низком уровне целостности. Для высокоцелостных пользователей, кроме того, имеются высокоцелостные обычная и административная роли для тех действий, которые он может выполнить на высоком уровне целостности. Поэтому в модели ни одно действие не может быть выполнено без использования роли с правом на это действие, что позволяет сохранить ключевое свойство безопасности RBAC.

Моделирующий MIC модуль добавляет события, позволяющие устанавливать уровни целостности пользователей, сущностей, субъектов и ролей. Все имеющиеся на базовом уровне события уточнены, в них добавлены охранные условия, говорящие, что только субъект уровня $high$ может получить доступ на модификацию чего-либо с уровнем $high$.

Основным свойством безопасности этого механизма является требование к субъекту иметь уровень $high$ для получения $write$-доступа к чему-либо с уровнем $high$, помимо предусловий оно зафиксировано в виде инвариантов. Поскольку все в системе имеет уровни целостности, все прямые или косвенные попытки нарушить это требование могут быть отслежены и предотвращены.

Модуль, моделирующий механизм MAC, добавляет частично упорядоченное множество *уровней конфиденциальности*, а в состояние – отображения пользователей, сущностей, субъектов и ролей в уровни конфиденциальности. Уровень конфиденциальности рассматривается как составная метка, содержащая всю информацию, необходимую для контроля конфиденциальности. В обычных терминах, она содержит и множество категорий, и число, соответствующее обычному уровню. Пользователь может запускать субъекты любого уровня конфиденциальности, меньше или равного уровню

пользователя. Специальные обычные и административные роли создаются для любой комбинации из уровня целостности и уровня конфиденциальности, доступной данному пользователю, эти роли обладают правами на все действия этого пользователя на данных уровнях конфиденциальности и целостности.

Основным свойством безопасности механизма MAC является требование к субъекту иметь больший или равный уровень конфиденциальности для доступа на чтение к чему-либо, и равный уровень конфиденциальности для доступа на запись. Это свойство оформлено в виде инвариантов.

Последний, четвертый модуль, добавляет переменные и события для построения информационных потоков. По аналогии с моделью Take-Grant, информационные потоки строятся между сущностями и субъектами путем построения транзитивного замыкания доступов. Вводятся события, являющиеся аналогами правил *take, pass, find*, и *spy*. Кроме того, определяется отношения контроля между субъектами. Субъект контролирует себя и любого другого субъекта, к исполнимым файлам которого он имеет доступ на запись или поток на запись, далее это отношение замыкается по транзитивности. Информационные потоки также пополняются потоками, которые могут создать контролирующие процессы через контролируемые. Таким образом, в рамках модели. включающей этот модуль можно анализировать информационные потоки в системе, а также возможные утечки информации при успешных атаках. Можно доказать, что информационные потоки не могут нарушить основные свойства безопасности – перенести информацию с более высокого уровня конфиденциальности на более низкий, даже при успешной атаке, компрометирующей субъекты на только низком уровне целостности.

3.2 Формальное представление модели

В данном подразделе дается более детальное описание формального представления базового уровня модели HIMACF, описывающего механизм RBAC. Кроме того, приведено несколько примеров спецификации событий и инвариантов других уровней для демонстрации важных свойств модели.

В моделях на языке Event-B статическая часть называется *контекстом* и содержит описание базовых типов (называемых множествами), констант и аксиом, аксиомы задают типы констант и другие ограничения. Динамическая часть модели, называемая *машиной*, содержит описание элементов автомата – переменных состояния, событий и инвариантов.

Контекст базового уровня модели HIMACF содержит следующие множества.

- *UserType* представляет тип пользователей. Это совокупность всех потенциально возможных пользователей, включая еще не зарегистрированных в системе.
- *SubjectType* – тип субъектов.
- *EntityType* – тип сущностей (объектов и контейнеров).
- *RoleType* – тип ролей.
- *Names* представляет все возможные имена сущностей и ролей.
- *Accesses* – возможные доступы к сущностям и ролям, включает доступ на чтение $ReadA$ и на запись $WriteA$.
- *AccessRights* – возможные права ролей, включает константы $Read$, $Write$, $Execute$, Own.

Корневой контейнер в иерархии сущностей соответствует константе $Root \in EntityType$. Множество специальных административных ролей *SpecialAdmRoles* тоже является константой, содержащей отдельные роли для администрирования сущностей, субъектов, пользователей, обычных ролей и административных ролей. Далее используется

конструкция $partition(U, A, B, C)$, в Event-B означающая $U = A \cup B \cup C$ и $A \cap B = A \cap C = B \cap C = \emptyset$, эта конструкция может иметь любое число аргументов, больше двух, с аналогичным смыслом. Выполнены свойства $partition\ (SpecialAdmRoles,\ \{EntitiesAR\},\ \{SubjectsAR\},\ \{UsersAR\},\ \{OrdRolesAR\},\ \{AdmRolesAR\})$, $partition\ (Accesses;\ \{ReadA\},\ \{WriteA\})$ и $partition\ (AccessRights,\ \{Read\},\ \{Write\},\ \{Execute\},\ \{Own\})$.

Базовый уровень модели HIMACF имеет следующие переменные состояния (определения типов переменных мы указываем здесь же, а не в отдельных инвариантах, как положено в Event-B).

- $Users \subseteq UserType$ – множество зарегистрированных пользователей.
- $Subjects \subseteq SubjectType$ – множество активных (работающих в данный момент) субъектов.
- $Entities \subseteq EntityType$ – множество существующих сущностей.
- $Roles \subseteq RoleType$ – множество всех существующих ролей, обычных и административных.
- $Objects$ – множество существующих объектов, сущностей, не содержащих другие сущности.
- $Containers$ – множество контейнеров, способных содержать другие сущности. Выполняются свойства $partition\ (Entities, Objects, Containers)$ и $Root \in Containers$.
- $RegRoles$ – множество обычных ролей.
- $AdmRoles$ – множество административных ролей. Выполнены свойства partition $(Roles, RegRoles, AdmRoles)$ и $SpecialAdmRoles \subseteq AdmRoles$.
- $EntityNames \in (Entities \setminus \{Root\}) \to (Containers \leftrightarrow Names)$ – отображение сущности в множество пар из контейнера и имени этой сущности в данном контейнере.
- $RoleName \in Roles \to Names$ – отображение ролей в их имена.
- $Parent \in (Containers \setminus \{Root\}) \to Containers$ – отображение контейнера в родительский контейнер. Контейнер может содержаться только в одном контейнере, для них жесткие ссылки запрещены. Объект может содержаться в нескольких контейнерах, которые можно вычислить из отображения $EntityNames$.
- $SParent \in Subjects \nrightarrow Subjects$ – частичное отображение субъектов в их родителей.
- $RParents \in Roles \to \mathcal{P}(Roles)$[1] – отображение роли в множество ее родительских ролей.
- $RoleRights \in Roles \to (Entities \leftrightarrow AccessRights)$ – отображение ролей в их права.
- $RoleAdmRights \in AdmRoles \to (Roles \leftrightarrow AccessRights)$ – отображение административных ролей в их права.
- $SubjectUser \in Subjects \to Users$ – отображение субъекта в активировавшего его пользователя.
- $SubjectOwner \in Subjects \nrightarrow Roles$ – частичное отображение субъектов в их роли-владельцы.
- $SubjectAccesses \in Subjects \to (Entities \leftrightarrow Accesses)$ – отображение субъектов в их текущие доступы к сущностям.
- $SubjectAdmAccesses \in Subjects \to (Roles \leftrightarrow Accesses)$ – отображение субъектов в их административные доступы к ролям.

[1] Здесь и далее $\mathcal{P}(X)$ используется как обозначение множества всех подмножеств X.

Инварианты, в дополнение к типовым ограничениям, содержат ограничения корректности значений переменных. Для отображений $EntityNames$ и $Parent$ должно быть выполнено следующее.

- $\forall e \cdot e \in \mathrm{dom}(EntityNames) \Rightarrow EntityNames(e) \neq \emptyset$. Каждая сущность, кроме $Root$, содержится в каком-то контейнере.
- $\forall e1, e2 \cdot \{e1, e2\} \subseteq \mathrm{dom}(EntityNames) \wedge e1 \neq e2 \Rightarrow EntityNames(e1) \cap EntityNames(e2) = \emptyset$. Имена сущностей в контейнерах уникальны.
- $\forall c1, c2 \cdot c1 \in Containers \wedge c1 \neq Root \wedge c2 \in \mathrm{dom}(EntityNames(c1)) \Rightarrow c2 = Parent(c1)$. Некорневой контейнер содержится только в одном, родительском, контейнере.
- $\forall C \cdot C \subseteq \mathrm{dom}(Parent) \wedge C \neq \emptyset \Rightarrow C \setminus Parent[C] \neq \emptyset$. В иерархии контейнеров нет циклов.

На иерархии субъектов и ролей наложены следующие ограничения.

- $\forall r \cdot r \in RegRoles \Rightarrow RParents(r) \subseteq RegRoles$.
- $\forall r \cdot r \in AdmRoles \Rightarrow RParents(r) \subseteq AdmRoles$. Иерархии обычных и административных ролей не пересекаются.
- $\forall R \cdot R \subseteq Roles \wedge R \neq \emptyset \Rightarrow (\exists r \cdot r \in R \wedge R \cap RParents(r) = \emptyset)$. В иерархии ролей нет циклов.
- $\forall S \cdot S \subseteq \mathrm{dom}(SParent) \wedge S \neq \emptyset \Rightarrow S \setminus SParent[S] \neq \emptyset$. В иерархии субъектов нет циклов.

У сущности или роли может быть не более одного владельца. $\forall r1, r2, e \cdot r1 \in Roles \wedge r2 \in Roles \wedge (e \mapsto Own) \in RoleRigths(r1) \cap RoleRights(r2) \Rightarrow r1 = r2$.

Административное право $read$ распространяется от ролей-родителей к детям. $\forall a, r, p \cdot a \in AdmRoles \wedge r \in Roles \wedge p \in RParents(r) \wedge (p \mapsto Read) \in RoleAdmRights(a) \Rightarrow (r \mapsto Read) \in RoleAdmRights(a)$.

Список операций модели (событий Event-B) на базовом уровне следующий.

- Управление пользователями: *create_user, delete_user, get_user_attr*.
- Управление доступами: *access_read_entity, access_write_entity, delete_access_entity, access_read_role, access_write_role, delete_access_role*.
- Управление правами: *grant_rights, remove_rights, set_entity_owner, set_subject_owner, grant_admin_rights, remove_admin_rights*.
- Управление сущностями: *create_object, create_container, delete_entity, create_hard_link, delete_hard_link, rename_entity, get_entity_attr, read_container, set_entity_labels, set_container_attr*.
- Управление ролями: *create_role, create_hard_link_role, delete_role, delete_hard_link_role, rename_role, get_role_attr*.
- Управление субъектами: *create_first_subject, create_subject, delete_subject, get_subject_attr*.

Например, спецификация события *access_write_entity* на базовом уровне такова.

event $access\_write\_entity$
  any
    $subject$
    $entity$
  where
    @grd1 $subject \in Subjects$
    @grd2 $entity \in Entities$
    @grd3 $\exists r \cdot r \in RegRoles \cup AdmRolesr$

$\wedge\, r \mapsto ReadA \in SubjectAdmAccesses(subject)$
$\wedge\, entity \mapsto Write \in RoleRights(r)$
@grd4 $\exists E, c \cdot E \subseteq \mathrm{dom}(Parent) \wedge (entity = Root$
$\wedge\, E = \emptyset) \vee (\, entity \neq Root$
$\wedge\, c \in \mathrm{dom}\big(EntityNames(entity)\big)$
$\wedge\, Parent[E] \cup \{c\} = E \cup \{Root\}))$
$\wedge\, (\forall o \cdot o \in E \cup \{entity\} \cup \{Root\} \Rightarrow$
$(\exists r \cdot r \in RegRoles \cup AdmRoles$
$\wedge\, r \mapsto ReadA \in SubjectAdmAccesses(subject)$
$\wedge\, o \mapsto Execute \in RoleRights(r)))$

then

@act1 $SubjectAccesses(subject) \coloneqq$
$SubjectAccesses(subject) \cup \{entity \mapsto WriteA\}$

end

Здесь параметр $subject$ – это субъект, пытающийся получить доступ, *entity* – сущность, доступ к которой запрошен. Два первых охранных условия задают типы параметров. Третье условие утверждает, что чтобы получить доступ на запись в сущность, субъект должен иметь связанную с ним роль r, имеющую право на запись в эту сущность. Четвертое условие означает, что для получения доступа субъект должен иметь доступ $Execute$ ко всем членам некоторой цепочки включающих друг друга контейнеров, начинающейся с $Root$ и заканчивающейся контейнером, содержащим нужную сущность (эта цепочка без корня обозначена как E).

Далее представлены уточненные спецификации этого же события в других модулях. В модуле, описывающем механизм MIC, к типам добавлено множество уровней целостности $Integrity$ и две константы *LowI* и $HighI$, принадлежащие этому множеству. Добавленные в переменные отображения $EntityInt: Entities \to Integrity$ и $SubjectInt: Subjects \to Integrity$ задают уровни целостности сущностей и субъектов. Отображение $CCRI: Containers \to BOOL$ задает CCRI-флаги контейнеров. Если такой флаг для контейнера выставлен в $TRUE$, то для доступа к сущностям внутри контейнера субъекту необходимо выполнить условия доступа к самому контейнеру. В этом модуле добавлены события *set_user_labels* и *set_role_labels*, позволяющие менять уровни целостности пользователей и ролей. Большая часть событий первого модуля уточнены при помощи добавления охранных условий, проверяющих соотношения уровней целостности параметров.

Инвариант, представляющий основное свойство безопасности механизма MIC, выглядит так.

$@SubjectAccesses1$
$\forall s, e \cdot s \in Subjects \wedge (e \mapsto WriteA) \in$
$SubjectAccesses(s) \Rightarrow EntityInt(e) \leq SubjectInt(s)$

В спецификацию события *access_write_entity* во втором модуле добавлены следующие два условия.

event $access\_write\_entity$ extends $access\_write\_entity$
where
$@grd5\ \exists E, c \cdot E \subseteq \mathrm{dom}(Parent) \wedge ((entity = Root$
$\wedge\, E = \emptyset) \vee (entity \neq Root$
$\wedge\, c \in \mathrm{dom}(EntityNames(entity))$
$\wedge\, Parent[E] \cup \{c\} = E \cup \{Root\}))$
$\wedge\, (\forall o \cdot o \in E \cup \{entity\} \cup \{Root\} \Rightarrow$
$(\exists r \cdot r \in RegRoles \cup AdmRoles$

$\wedge\, r \mapsto ReadA \in SubjectAdmAccesses(subject)$
$\wedge\, o \mapsto Execute \in RoleRights(r))$
$\wedge\, EntityInt(o) \leq SubjectInt(subject)$
$\vee\, CCRI(o) = FALSE))$
@grd6 $EntityInt(entity) \leq SubjectInt(subject)$
end

Пятое охранное условие требует существования цепочки включающих друг друга контейнеров, начинающейся с $Root$ и заканчивающейся контейнером, содержащим нужную сущность, для каждого из которых данный субъект должен иметь роль с правом доступа на $Execute$, кроме того, каждый из этих контейнеров должен либо иметь опущенный CCRI-флаг, либо уровень целостности меньше или равный уровня целостности субъекта, запрашивающего доступ. Шестое условие требует, чтобы субъект, запрашивающий доступ на запись к сущности, имел уровень целостности, больше или равный уровню целостности этой сущности.

Третий модуль описывает механизм MAC, добавляя в рассмотрение уровни конфиденциальности Cnf, определяемые как множество всех подмножеств некоторого непустого конечного множества $Confidentiality$. Поскольку каждое конечное частично упорядоченное множество может быть изоморфно вложено в решетку множества всех подмножеств некоторого конечного множества, такое описание не накладывает значимых ограничений на структуру возможных уровней конфиденциальности.

В качестве переменных добавлены отображения $EntityCnf: Entities \to Cnf$ и $SubjectCnf: Subjects \to Cnf$, задающие уровни конфиденциальности сущностей и субъектов. Также добавлено отображение $CCR: Containers \to BOOL$, описывающее CCR-флаги контейнеров. Если такой флаг для контейнера выставлен в $TRUE$, то для доступа к сущностям внутри контейнера субъекту необходимо выполнить условия доступа к самому контейнеру. Снова, большинство событий уточнены с помощью охранных условий, проверяющих соотношения между уровнями конфиденциальности их параметров.

Инварианты, описывающие основное свойство безопасности специфицируемого механизма, выглядят так.

$@SubjectAccesses2$
$\forall s, e \cdot s \in Subjects (e \mapsto ReadA) \in SubjectAccesses(s) \Rightarrow$
$EntityCnf(e) \subseteq SubjectCnf(s)$
$@SubjectAccesses3$
$\forall s, e \cdot s \in Subjects (e \mapsto WriteA) \in SubjectAccesses(s) \Rightarrow$
$EntityCnf(e) = SubjectCnf(s)$

Следующие условия добавлены в спецификацию $access\_write\_entity$ в третьем модуле.

event $access\_write\_entity$ $extends$ $access\_write\_entity$
$where$
@grd7 $\exists E, c \cdot E \subseteq \mathrm{dom}(Parent) \wedge ((entity = Root$
$\wedge\, E = \emptyset) \vee (entity \neq Root$
$\wedge\, c \in \mathrm{dom}(EntityNames(entity))$
$\wedge\, Parent[E] \cup \{c\} = E \cup \{Root\}))$
$(\forall o \cdot o \in E \cup \{entity\} \cup \{Root\} \Rightarrow$
$(\exists r \cdot r \in RegRoles \cup AdmRoles$
$\wedge\, r \mapsto ReadA \in SubjectAdmAccesses(subject)$
$\wedge\, o \mapsto Execute \in RoleRights(r))$
$\wedge\, (EntityInt(o) \leq SubjectInt(subject)$

$\vee\ CCRI(o) = FALSE)$
$\wedge\ (EntityCnf(o) \subseteq SubjectCnf(subject)$
$\vee$ CCR(o)=FALSE))
$@grd8\ EntityCnf(entity) = SubjectCnf(subject)$
end

Седьмое условие здесь снова требует существования цепочки включающих друг друга контейнеров, начинающейся с $Root$ и заканчивающейся контейнером, содержащим нужную сущность, для каждого из которых данный субъект должен иметь роль с правом доступа на $Execute$, кроме того, каждый из этих контейнеров должен либо иметь опущенный CCRI-флаг, либо уровень целостности меньше или равный уровню целостности субъекта, запрашивающего доступ, и должен либо иметь опущенный CCR-флаг, либо уровень конфиденциальности меньше или равный уровню конфиденциальности субъекта. Восьмое условие говорит, что для получения доступа $Write$ к сущности субъект должен иметь равный уровень конфиденциальности. Условия 5 и 7 вынуждено повторяют ограничения из условия 4, поскольку все эти ограничения должны быть выполнены для одной и той же цепочки контейнеров.

Последний модуль специфицирует модель информационных потоков между сущностями и субъектами. Для этого добавляются четыре переменных $EEMFlows: Entities \to \mathcal{P}(Entities)$, $ESMFlows: Entities \to \mathcal{P}(Subjects)$, $SEMFlows: Subjects \to \mathcal{P}(Entities)$ и $SSMFlows: Subjects \to \mathcal{P}(Subjects)$. Они представляют потоки на запись от сущностей к сущностям, от сущностей к субъектам, от субъектов к сущностям и от субъектов к субъектам. Отношение контроля между субъектами представлено отображением $DeFactoOwn: Subjects \to \mathcal{P}(Subjects)$. Правило $find$ из TGM трансформируется в два события, $find\_entity$ и $find\_subject$. Спецификация первого из них выглядит следующим образом.

event $find\_entity$
  any
    x
    y
    z
  where
    @grd1 $x \in Subjects$
    @grd2 $y \in Subjects$
    @grd3 $z \in Entities$
    @grd4 $y \in SSMFlows(x)$
    @grd5 $z \in SEMFlows(y)$
  $then$
    $@act1\ SEMFlows(x) := SEMFlows(x) \cup \{z\}$
end

4. Верификация модели

Сложность модели HIMACF делает ее верификацию без поддержки каких-либо инструментов чрезвычайно сложной задачей. Ее описание на сочетании математического и естественного языков занимает около 300 страниц, и оно увеличивается, поскольку модель постоянно развивается и в нее включаются новые элементы. Некоторые числовые характеристики модели в текущей версии представлены в табл. 1, каждая строка которой показывает значения характеристик для отдельных модулей, а последняя строка – общие числа. Обозначения модулей сокращены. В третьем ее столбце находится количество новых событий, специфицированных в

определенном модуле, в четвертом – количество уточненных в данном модуле событий из предыдущих модулей.

Табл. 1. Числовые характеристики модели HIMACF
Table 1. Numerical characteristics of the HIMACF model

Модуль	Переменные	Нов. события	Уточн. события	Инварианты	Строки кода
RBAC	21	37	0	50	1480
MIC+	11	2	37	47	1120
MLS+	5	0	34	25	700
Information Flows+	15	36	39	74	1770
Total	52	75	—	196	5070

Автоматизированная верификация модели проведена с помощью инструмента Rodin [12], поддерживающего работу с моделями на языке Event-B. Общее число доказываемых утверждений (каждое из них утверждает корректность типа переменной, изменяемой в рамках некоторого события, сохранение инварианта в рамках некоторого события, и т.д.), сгенерированных в среде Rodin, оказывается около 3600. Из них примерно 75% доказывается автоматически, остальные доказываются интерактивно, с участием человека, и для этого нужно около одного человеко-месяца. Эти оценки применимы к текущей структуре модели и работающим с ней разработчикам. Первая версия модели [10] не использовала уточнения, все механизмы были специфицированы в одном модуле, и ее разработка и верификация на основе исходного представления на естественном ярыке потребовали примерно 2 человеко-года. Текущая структура модели более удобна как для восприятия, так и для внесения изменений и развития модели, поскольку трудоемкость внесения небольших изменений и повторного доказательства инвариантов, на которые они влияют, обычно существенно ниже [33].

5. Заключение и возможное развитие

В данной статье представлены текущие результаты по разработке и верификации интегрированной модели, специфицирующей композицию управления доступом на основе ролей (RBAC), мандатного контроля целостности (MIC), мандатного управления доступом (MAC), а также информационные потоки в возникающей системе, и названной HIMACF (hierarchical integrated model of access control and information flows). Модель специфицирована на формальном языке Event-B и формально верифицирована, все ее инварианты доказаны с помощью инструмента Rodin, поддерживающего работу с моделями на Event-B.

Модель построена из нескольких модулей, уточняющих друг друга. Это позволяет более ясно описать каждый из механизмов, четко выделяя их ключевые свойства безопасности, а также облегчает выполнение верификации. Первый модуль описывает механизм RBAC, второй – MIC, третий – MAC, и четвертый модуль специфицирует информационные потоки. Ключевым свойством безопасности RBAC является требование того, чтобы субъект получал доступ на выполнение некоторой операции, только если он обладает ролью, имеющей права на ее выполнение. Ключевое требование MIC – субъект может получить доступ на модификацию сущности или вмешательство в работу другого субъекта, только если он имеет более высокий или равный уровень целостности. Аналогично, MAC требует, чтобы субъект, получающий доступ на чтение или выполнение, имел более высокий уровень конфиденциальности, а получающий доступ на модификацию – равный уровень конфиденциальности. В модели HIMACF все эти ограничения выполняются, благодаря разметке всех субъектов, сущностей и ролей уровнями конфиденциальности и целостности, а также наличию специализированных ролей, через которых пользователи выполняют все обычные операции.

Поскольку основной целью разработки модели была аккуратная интеграция нескольких механизмов управления доступом, применимых в современных ОС, контролирующих сложные иерархии процессов и файлов, получившаяся модель достаточно сложна. Тем не менее, за счет использования языка формальных спецификаций, поддержанного инструментами интерактивного дедуктивного анализа, формальная верификация модели проводится с приемлемой трудоемкостью.

Иерархическая модульная структура самой модели, использующая уточнение, позволяет эффективно развивать модель и вносить новые элементы в нее, проводя верификацию измененных версий быстро и с небольшими трудозатратами. В настоящее время уже разработаны версии модели с косвенными метками защиты (помещаемыми не на самих файлах, а на корне некоторого дерева контейнеров), специализированными объектами с ослабленными ограничениями доступа (моделирующими специальные устройства Unix без возможности сохранения информации, как, например, dev/null), запрещающими ролями (ролями, обладание которыми запрещает выполнение некоторых операций) и более сложной структурой уровней целостности в виде произвольного частично упорядоченного множества. В процессе разработки находится расширение модели, применимое для использования в качестве модели безопасности СУБД.

Модель HIMACF частично реализована в построенной на основе ядра Linux ОС специального назначения Astra Linux Special Edition [15], использующей инфраструктуру Linux Security Modules (LSM) для дополнительного контроля доступа. Наличие формальной модели позволяет построить специализированную формальную модель реализации интерфейсов LSM в этой ОС и, в идеале, провести формальную верификацию соответствия реализации кода реализации ОС самой модели, например, с использованием инструментов [34], что в итоге позволит получить формально верифицированную реализацию механизма управления доступом в рамках современной ОС. Эта работа требует еще довольно много усилий и совершенствования имеющихся инструментов верификации, информацию о прогрессе в ее продвижении можно найти, например, в [35].

Список литературы / References

[1] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman. Role-based access control models. Computer, vol. 29, no. 2, 1996, pp. 38-47.

[2] S. Smalley, C. Vance, W. Salamon. Implementing SELinux as a linux security module. Technical Report 01-043, NAI Labs, 2001.

[3] S. Smalley, R. Craig. Security Enhanced (SE) Android: Bringing Flexible MAC to Android. In Proc. of Network & Distributed System Security Symposium (NDSS), 2013, 18 p.

[4] M. Conover. Analysis of the Windows Vista security model. Technical Report, Symantec Corp., 2008.

[5] A. Cunningham, L. Hutchinson. OSX10.11 El Capitan: The Ars Technica Review. Available at: https://arstechnica.com/apple/2015/09/os-x-10-11-el-capitan-the-ars-technica-review/. Accessed 21 January 2019.

[6] D.E. Bell, L.J. LaPadula. Secure Computer Systems: Mathematical Foundations. Technical Report ESD-TR-73-278 v.~1, (also MTR-2547, v.~1), Electronic Systems Division, AFSC, Hanscom AFB, 1973.

[7] K.J. Biba. Integrity considerations for secure computer systems. Technical Report MTR-3153, The MITRE Corporation, 1977.

[8] R. Sandhu. Role hierarchies and constraints for lattice-based access controls. Lecture Notes in Computer Science, vol. 1146, 1996, pp.65-79.

[9] П.Н. Девянин. Модели безопасности компьютерных систем. Управлне доступом и информационными потоками. Горячая линия-Телеком, 2013, 338 стр. / P.N. Devyanin. Security models of computer systems. Control for access and information flows. Hotline-Telecom, 2013, 338 p. (in Russian).

[10] P.N. Devyanin, A.V. Khoroshilov, V.V. Kuliamin, A.K. Petrenko, I.V. Shchepetkov. Formal

Verification of OS Security Model with Alloy and Event-B. Lecture Notes in Computer Science, vol. 8477, 2014, pp. 309-313.
[11] J.R. Abrial. Modeling in Event-B: System and Software Engineering. Cambridge University Press, 2010, 612 p.
[12] J.R. Abrial, M. Butler, S. Hallerstede, T.S. Hoang, F. Mehta, L. Voisin. Rodin: an open toolset for modelling and reasoning in Event-B. International Journal on Software Tools for Technology Transfer, vol. 12, no. 6, 2010, pp. 447-466.
[13] ISO/IEC 15408-1:2009. Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model. ISO, 2009.
[14] ISO/IEC 15408-2:2008. Information technology – Security techniques – Evaluation criteria for IT security – Part 2: Security functional components. ISO, 2008.
[15] Astra Linux. Available at: https://en.wikipedia.org/wiki/Astra_Linux. Accessed 21 January 2019.
[16] American National Standard for Information Technology – Role Based Access Control. ANSI INCITS 359-2004, 2004.
[17] D.E. Bell, L.J. LaPadula. Secure Computer System: Unified Exposition and MULTICS Interpretation. Technical Report ESD-TR-75-306 (also MTR-2997), Electronic Systems Division, AFSC, Hanscom AFB, 1976.
[18] A.K. Jones, R.J. Lipton, L. Snyder. A linear time algorithm for deciding security. In Proc. of 17-th Annual Symposium on Foundations of Computer Science, 1976, pp. 33-41.
[19] 19. M. Bishop, L. Snyder. The Transfer of Information and Authority in a Protection System. In Proc. of 7th ACM Symposium on Operating System Principles, 1979, pp. 45-54.
[20] C.E. Landwehr, C.L. Heitmeyer, J. McLean. A security model for military message systems. ACM Transactions on Computer Systems, vol. 2, issue 3, 1984, pp. 198-222.
[21] Security-Enhanced Linux. Available at: https://www.nsa.gov/what-we-do/research/selinux/. Accessed 21 January 2019.
[22] PostgreSQL. Available at: https://en.wikipedia.org/wiki/PostgreSQL. Accessed 21 January 2019.
[23] D-Bus. Available at: https://en.wikipedia.org/wiki/D-Bus. Accessed 21 January 2019.
[24] X Window System. Available at: https://en.wikipedia.org/wiki/X_Window_System. Accessed 21 January 2019.
[25] A. Eaman, B. Sistany, A. Felty. Review of Existing Analysis Tools for SELinux Security Policies: Challenges and a Proposed Solution. Lecture Notes in Business Information Processing, vol. 289, 2017, pp. 116-135.
[26] G. Zanin, L.V. Mancini. Towards a Formal Model for Security Policies Specification and Validation in the Selinux System. In Proc. of 9th ACM Symposium on Access Control Models and Technologies, 2004, pp. 136-145.
[27] G. Zhai, T. Guo, J. Huang. SCIATool: A Tool for Analyzing SELinux Policies Based on Access Control Spaces, Information Flows and CPNs. Lecture Notes in Computer Science, vol. 9473, 2015, pp. 294-309.
[28] P. Amthor, W.E. Kühnhauser, A. Pölck. Model-based safety analysis of SELinux security policies. In Proc. of 5th International Conference on Network and System Security (NSS), 2011, pp. 208-215.
[29] M.A. Harrison, W.L. Ruzzo, J.D. Ullman. Protection in operating systems. Communications of the ACM, vol. 19, no. 8, 1976, pp. 461-471.
[30] B. Hicks, S. Rueda, L. St.Clair, T. Jaeger, P. McDaniel. A logical specification and analysis for SELinux MLS policy. ACM Transactions on Information and System Security, vol. 13, issue 3, 2010, article no. 26.
[31] M.C. Tschantz. The clarity of languages for access-control policies. PhD Thesis, Brown University, Providence, Rhode Island, USA, 2005.
[32] R. Sandhu, V. Bhamidipati, Q. Munawer. The ARBAC97 model for role-based administration of roles. ACM Transactions on Information and System Security, vol. 2, issue 1, 1999, pp. 105-135.
[33] П.Н. Девянин, В.В. Кулямин, А.К. Петренко, А.В. Хорошилов, И.В. Щепетков. Сравнение способов декомпозиции спецификаций на Event-B. Программирование, vol. 42, no. 4, 2016, pp. 17-26 / P.N. Devyanin, V.V. Kuliamin, A.K. Petrenko, A.V. Khoroshilov, I.V. Shchepetkov. Comparison of specification decomposition methods in Event-B. Programming and Computer Software, vol. 42, no. 4, 2016, pp. 198-205.
[34] J.C. Filliâtre, A.Paskevich. Why3 – Where Programs Meet Provers. Lecture Notes in Computer Science, vol. 7792, 2013, pp. 125-128.

[35] D. Efremov, M. Mandrykin, A. Khoroshilov. Deductive verification of unmodified Linux kernel library functions. Lecture Notes in Computer Science, vol. 11245, 2018, pp. 216-234.

Информация об авторах / Information about authors

Петр Николаевич ДЕВЯНИН – член-корреспондент Академии криптографии России, доктор технических наук, профессор, главный научный сотрудник ООО "РусБИТех-Астра" (ГК Astra Linux). Область интересов: теория информационной безопасности, формальные модели безопасности компьютерных систем.

Petr Nikolaevich DEVYANIN – Doctor of Technical Sciences, corresponding member of Russian Academy of Cryptography, professor, main researcher in RusBITech-Astra (Astra Linux). Field of Interest: information security theory, formal security models of computer systems.

Виктор Вячеславович КУЛЯМИН –кандидат физико-математических наук, ведущий научный сотрудник ИСП РАН, доцент кафедр системного программирования МГУ и ВШЭ. Область интересов: формальная дедуктивная верификация моделей, тестирование на основе моделей.

Viktor Vyacheslavovich KULYAMIN – Ph.D. in Physics and Mathematics, leading researcher at ISP RAS, associate professor of system programming departments at Moscow State University and the Higher School of Economics. Fields of Interest: formal deductive model verification, model-based testing

Александр Константинович ПЕТРЕНКО –доктор физико-математических наук, профессор, начальник отдела ИСП РАН, профессор МГУ и ВШЭ. Область интересов: формальные методы программной инженерии, тестирование программного и аппаратного обеспечения, формальная спецификация требований.

Alexander Konstantinovich PETRENKO – Doctor of Physical and Mathematical Sciences, Professor, Head of the Department of ISP RAS, Professor of Moscow State University and Higher School of Economics. Areas of interest: formal methods of software engineering, testing of software and hardware, formal specification of requirements.

Алексей Владимирович ХОРОШИЛОВ, ведущий научный сотрудник, кандидат физико-математических наук, директор Центра верификации ОС Linux в ИСП РАН, доцент кафедр системного программирования МГУ, ВШЭ и МФТИ. Основные научные интересы: методы проектирования и разработки ответственных систем, формальные методы программной инженерии, методы верификации и валидации, тестирование на основе моделей, методы анализа требований, операционная система Linux.

Alexey Vladimirovich KHOROSHILOV, Leading Researcher, Ph.D. in Physics and Mathematics, Director of the Linux OS Verification Center at ISP RAS, Associate Professor of System Programming Departments at Moscow State University, the Higher School of Economics, and Moscow Institute of Physics and Technology. Main research interests: design and development methods for critical systems, formal methods of software engineering, verification and validation methods, model-based testing, requirements analysis methods, Linux operating system.

Илья Викторович ЩЕПЕТКОВ – стажер-исследователь. Область интересов: разработка и верификация формальных моделей компьютерных систем

Ilya Viktorovich SHCHEPETKOV – intern researcher. Fields of Interest: development and verification of formal models of computer systems

DOI: 10.15514/ISPRAS-2020-32(1)-2

Модель мандатного контроля целостности в операционной системе KasperskyOS

В. С. Буренков, ORCID: 0000-0002-0232-774X <Vladimir.Burenkov@kaspersky.com>
Д. А. Кулагин, ORCID: 0000-0001-5055-0826 <Dmitry.Kulagin@kaspersky.com>

АО «Лаборатория Касперского»,
125212, Россия, Москва, Ленинградское шоссе, д. 39А, стр. 3

Аннотация. Существующие модели мандатного контроля целостности в операционных системах накладывают ограничения на доступы активных компонент системы к пассивным и представляют эти доступы непосредственно. Такое представление можно использовать в случае операционных систем с монолитной архитектурой, где части системы, обеспечивающие доступ к ресурсам, входят в доверенную вычислительную базу. Однако эти части являются нетривиальными компонентами со сложной реализацией, в связи с чем доказательство отсутствия в них ошибок (уязвимостей) и, следовательно, соответствия модели реальной системе – чрезвычайно трудная задача, которая на практике, как правило, полностью не решается. В данной статье представлена модель мандатного контроля целостности для микроядерной операционной системы KasperskyOS. Базирование системы на микроядре предполагает минимизацию доверенной вычислительной базы. При таком подходе доступ к ресурсам предоставляется с помощью компонент, не входящих в доверенную вычислительную базу. Это создает предпосылки для гарантии соответствия системы определенным свойствам безопасности даже в случае некорректного функционирования этих компонент. В модели для представления частей системы, обеспечивающих доступ к ресурсам, введено понятие драйвера объектов, и операции получения доступа активных компонент (сущностей) к пассивным компонентам (объектам) выполняются посредством драйверов объектов. В статье определены требования, которым должны удовлетворять драйверы объектов и некоторые другие сущности. В модель также добавлены элементы, позволяющие провести ее анализ в случае нарушения данных требований. Сформулирована и доказана теорема о постоянном нахождении модели в целостном состоянии (об отсутствии в модели информационных потоков от менее целостных компонентов к более целостным либо несравнимым) в случае удовлетворения всеми сущностями сформулированных требований, а также об ограниченном характере распространения эффекта от нарушения целостности системы в случае наличия в системе компонентов, нарушающих требования. Корректная реализация модели гарантирует, что если компрометирована часть системы, уровень целостности компонентов которой ограничен некоторыми уровнями целостности, то это не приведет к компрометации части системы с более высокими или несравнимыми уровнями целостности. Описан язык спецификации политик мандатного контроля целостности, разработанный в соответствии с правилами модели, и приведен пример использования языка для задания политики безопасности системы, работающей под управлением KasperskyOS. Целью политики является обеспечение безопасности процесса обновления системы.

Ключевые слова: мандатный контроль целостности; операционная система; KasperskyOS.

Для цитирования: Буренков В.С., Кулагин Д.А. Модель мандатного контроля целостности в операционной системе KasperskyOS. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 27-56. DOI: 10.15514/ISPRAS–2020–32(1)–2

A Mandatory Integrity Control Model for the KasperskyOS Operating System

V. S. Burenkov, ORCID: 0000-0002-0232-774X <Vladimir.Burenkov@kaspersky.com>
D. A. Kulagin, ORCID: 0000-0001-5055-0826 <Dmitry.Kulagin@kaspersky.com>
AO Kaspersky Lab.,
39A, building 3, Leningradskoe shosse, Moscow, 125212, Russia

Abstract. Existing models of mandatory integrity control in operating systems restrict accesses of active components of a system to passive ones and represent the accesses directly: subjects get read or write access to objects. Such a representation can be used in modeling of monolithic operating systems whose components that provide access to resources are part of the trusted computing base. However, the implementation of these components is extremely complex. Therefore, it is arduous to prove the absence of bugs (vulnerabilities) in them. In other words, proving such a model to be adequate to the real system is nontrivial and often left unsolved. This article presents a mandatory integrity control model for a microkernel operating system called KasperskyOS. Microkernel organization of the system allows us to minimize the trusted computing base to include only the microkernel and a limited number of other components. Parts of the system that provide resource access are generally considered untrusted. Even if some of them are erroneous, the operating system can still provide particular security guarantees. To prove that by means of a model, we introduce the notion of object drivers as intermediaries in operations on objects. We define the requirements that object drivers must satisfy. We also add the means for analysis of the consequences of violations of the requirements. We state and prove that the model either preserves integrity if all active components satisfy the requirements, or restricts the negative impact if some of the components are compromised. Correct implementation of the model guarantees that compromised components will not affect components with higher or incomparable integrity levels. We describe a policy specification language developed in accordance with the model. We provide an example of using it to describe a security policy that ensures a correct update of a system operated by KasperskyOS.

Keywords: mandatory integrity control; operating system; KasperskyOS.

For citation: Burenkov V.S., Kulagin D.A. A Mandatory Integrity Control Model for the KasperskyOS Operating System. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 27-56 (in Russian). DOI: 10.15514/ISPRAS–2020–32(1)–2

1. Введение

Контроль целостности компьютерных систем является одним из фундаментальных вопросов, рассматриваемых при разработке безопасных систем. Для обеспечения целостности в операционных системах реализуются механизмы *мандатного контроля целостности* [1]. Их реализация должна быть частью доверенной вычислительной базы.

Классический путь разработки механизмов безопасности начинается с создания модели системы [2]. Модель должна отражать существенные особенности разрабатываемой операционной системы. Достоинством наличия модели является возможность строгого доказательства того, что модель обладает определенными свойствами, которыми должна также обладать и сама система. Реализация модели приведет к получению требуемых механизмов безопасности. Если модель реализована корректно, то и система будет обладать доказанными ранее свойствами.

Существуют и официальные требования наличия формальной модели политики безопасности. Их предъявляет, например, ФСТЭК России [3].

В данной работе рассматривается разработка модели мандатного контроля целостности для ее реализации в новой микроядерной операционной системе KasperskyOS.

Концепция микроядерных операционных систем зародилась достаточно давно [4]. Также хорошо известны ее преимущества с точки зрения безопасности [5, 6, 7], главное из которых – уменьшение доверенной базы за счет вынесения большинства драйверов из

адресного пространства ядра в непривилегированный код. Несмотря на известность микроядерной архитектуры, существующие подходы к моделированию свойств безопасности операционных систем не учитывают ее специфические свойства. Так, в субъект-объектных [8] моделях управления доступом и информационными потоками получение доступов активных компонент системы к ресурсам представлено непосредственно (рис. 1). Это связано с тем, что данные модели разрабатываются для операционных систем с монолитной архитектурой, где части системы, обеспечивающие доступ к ресурсам (например, драйверы устройств и файловые системы), выполняются в адресном пространстве ядра. Это вынуждает считать их доверенными, и, как следствие, пренебрегать ими при моделировании. Однако эти части являются достаточно сложными программными компонентами, в которых высока вероятность наличия уязвимостей. По этой причине их включение в доверенную вычислительную базу означает использование предположений, которые в действительности могут быть не выполнены.

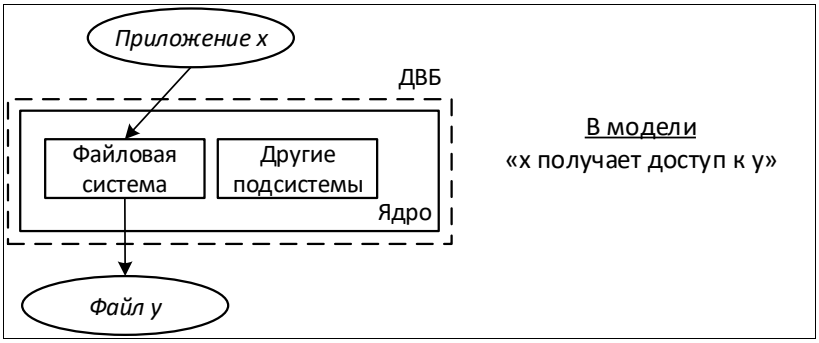


Рис. 1. Схема получения доступа к ресурсу (файлу) в монолитной операционной системе и ее трактовка в существующих формальных моделях

Fig. 1. Resource access scheme in a monolithic operating system and its interpretation in existing formal models

В отличие от монолитных операционных систем, доступ к ресурсам в рамках микроядерной операционной системы KasperskyOS происходит посредством отдельных процессов – драйверов ресурсов. Поскольку это обычные пользовательские процессы, отпадает необходимость считать их доверенными.

Рассмотрение драйверов ресурсов как обычных процессов позволяет также минимизировать начальное состояние модели системы, не включая в него большинство драйверов. Это дает возможность моделирования безопасности системы, начиная с самых ранних этапов ее загрузки: с того момента, когда загружены только ядро, модуль безопасности и очень ограниченный набор доверенных драйверов. В моделях безопасности для монолитных операционных систем достаточно сложный этап начальной загрузки системы не рассматривается, а созданные в процессе загрузки компоненты считаются частью начального состояния модели.

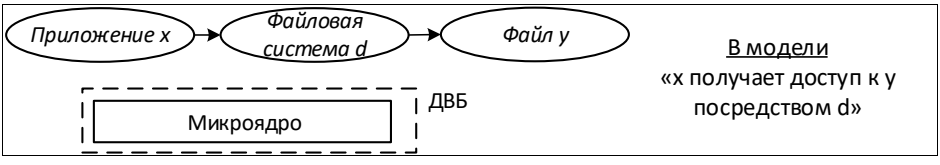


Рис. 2. Схема получения доступа к ресурсу (файлу) в микроядерной системе и необходимый вид ее трактовки в соответствующих формальных моделях

Fig. 2. Resource access scheme in a microkernel operating system and the way it should be interpreted in formal models

В то же время наличие недоверенных драйверов ресурсов означает, что при моделировании доступов к ресурсам необходимо всегда принимать во внимание, кроме

субъекта-инициатора запроса и объекта-ресурса, еще и субъект-драйвер – посредник при любом доступе к ресурсам определенного типа (рис. 2). Следовательно, чтобы в полной мере воспользоваться преимуществами микроядерной архитектуры, нужно расширить и уточнить существующие подходы к моделированию информационных потоков.

Несмотря на то, что драйверы рассматриваются как недоверенные приложения, они должны выполнять функции, присущие драйверам, и, следовательно, удовлетворять определенным требованиям. Однако если нет уверенности в том, что используемый драйвер будет отвечать этим требованиям, и требования заложены в основе модели мандатного контроля целостности, то невозможно сделать заключение о соответствии реальной системы и модели. В качестве решения этой проблемы в работе предлагается учитывать в модели возможность нарушения требований с тем, чтобы провести детальный анализ последствий такого нарушения.

Основными результатами данной работы являются следующие.

1. Разработана модель политики безопасности управления доступом и информационными потоками в KasperskyOS, описывающая мандатный контроль целостности. В отличие от существующих моделей, детально проработаны правила доступа к ресурсам посредством драйверов ресурсов, которые не обязательно являются доверенными.
2. Несмотря на то, что драйверы ресурсов и большая часть других сущностей не вносятся в доверенную вычислительную базу, свойства безопасности, обеспечиваемые моделью, зависят от конкретных аспектов поведения этих сущностей. Для учета этого обстоятельства в работе определены требования, которым должны удовлетворять драйверы ресурсов и другие сущности, участвующие в межпроцессном взаимодействии. Для анализа последствий нарушения этих требований определены дополнительные информационные потоки, возникающие в случае невыполнения требований.
3. Сформулирована и доказана теорема, утверждающая, что
   - если все сущности удовлетворяют сформулированным требованиям, то отсутствуют запрещенные информационные потоки от менее целостных компонент к более целостным или несравнимым,
   - либо, в случае присутствия сущностей, не удовлетворяющих сформулированным требованиям, характер нарушения целостности системы ограничен максимальными уровнями целостности компонентов, нарушающих требования.

   Таким образом, корректная реализация модели гарантирует, что если компрометирована часть системы вплоть до некоторого уровня целостности, то это не приведет к компрометации части системы с большими или несравнимыми уровнями целостности.
4. Помимо этого, описан язык спецификации политик мандатного контроля целостности, разработанный в соответствии с правилами модели, и приведен пример использования языка для задания политики безопасности системы с безопасным процессом обновления.

2. Существующие модели мандатного контроля целостности

Мандатный контроль целостности впервые был теоретически описан в работе Биба (K. Biba) [9]. Липнер (S. Lipner) [10]. Кларк и Уилсон (D. Clark, D. Wilson) [11], Ли (T. Lee) [12] представили ранние рассуждения о контроле целостности в коммерческих системах обработки данных. Брювер и Нэш (D. Brewer, M. Nash) [13] разработали другую классическую модель безопасности коммерческих систем, затрагивающую вопросы целостности и конфиденциальности.

Современные модели зачастую основаны на классических моделях целостности и информационных потоков. Так, Лиу (Z. Liu) и др. [14] описывают простую модель контроля целостности в стиле Белла-ЛаПадулы (D. Bell, L. LaPadula) [15].

Модели целостности Usable Mandatory Integrity Protection (UMIP) [16] и SecGuard [17] разработаны с целью простоты использования и реализованы в прототипах, основанных на ядре Linux. Описание моделей не формализовано в достаточной степени. Помимо этого, в них применяются потенциально опасные решения. Модели позволяют понижать уровень целостности субъекта, однако не рассматривают случаи, когда у такого субъекта есть открытые на запись высокоцелостные объекты. Авторы этих моделей также предлагают использовать дискреционные права доступа для задания уровней целостности. Однако при таком подходе ошибка в конфигурации дискреционных прав доступа приведет к бесполезности мандатного контроля целостности.

Модель, разработанная П.Н. Девяниным и реализованная в операционной системе Astra Linux [18], достаточно проработана теоретически. Имеются математические доказательства свойств модели. Более того, модель формализована и верифицирована в системе Event-B [19, 20]. Модель была разработана применительно к ядру Linux и подразумевает, что ядро, которое включает в себя большое количество компонент, включая драйверы и файловые системы, является доверенным. Это привело к достаточно большому количеству предположений о начальном состоянии модели, что не отвечает целям настоящей работы.

Операционная система Microsoft Windows реализует мандатный контроль целостности [21], основанный на модели Биба, начиная с версии Windows Vista (2007). Имеются положительные оценки соответствующих механизмов.

Таким образом, в литературе отсутствуют модели мандатного контроля целостности, позволяющие минимизировать доверенную вычислительную базу, в частности, вынести драйверы из пространства ядра. Данная работа восполняет этот пробел.

3. Архитектура операционной системы KasperskyOS

Модель, представленная в данной работе, была разработана для реализации в микроядерной операционной системе KasperskyOS. Основными компонентами операционной системы являются микроядро, монитор безопасности и множество драйверов, реализованных в виде непривилегированных приложений. Главными функциями микроядра являются разделение ресурсов между процессами (которые в рамках операционной системы именуются *сущностями*), предоставление механизма межпроцессного взаимодействия IPC (interprocess communication) и запуск монитора безопасности для проверки этих взаимодействий. Операционная система разрабатывалась с учетом принципа разделения ресурсов и приложений посредством минимальной доверенной вычислительной базы, направленного на поддержку политик безопасности. Этот подход известен с 1980-х годов [22]. Его современным воплощением является архитектура MILS [23].

Важным компонентом архитектуры MILS является *ядро разделения*. Если операционная система корректно реализует ядро разделения, то она гарантирует изоляцию приложений (возможно, с различными уровнями доверия) друг от друга. Изоляция приложений очень важна с точки зрения безопасности, так как она позволяет ограничить нежелательное воздействие на систему в целом, которое может оказать отдельное приложение.

Однако, во многих случаях одного только ядра разделения недостаточно, особенно, когда взаимодействуют сложные компоненты с разными уровнями доверия. В такой ситуации требуется убедиться, что взаимодействия в системе удовлетворяют требованиям безопасности (часто сложным и разнообразным). В KasperskyOS для решения этой проблемы существует Kaspersky Security System – набор инструментов для

спецификации самых разных свойств безопасности и синтезирования монитора обращений, проверяющего любые взаимодействия внутри системы на предмет соответствия политике. Для спецификации свойств (политики) используется декларативный язык Policy Specification Language (PSL). Описанная на нем политика транслируется в запускаемый образ монитора безопасности, которому ядро и другие сервисы могут направлять свои запросы.

Подход имеет много общего с архитектурой FLASK [24]. Прежде всего это:

- абстрагирование ресурсов и процессов как доменов безопасности; это позволяет монитору применять общие правила по контролю, не обращая внимание на их различную природу (когда это допустимо);
- разделение вычисления политики и механизма, инициирующего проверку события, передающего связанный с событием контекст и интерпретирующего результат проверки. Это одна из ключевых идей FLASK, позволяющая контролировать не только системные события, но и события прикладного уровня.

В KasperskyOS, кроме этого, каждый сервис должен статически декларировать свои интерфейсы. Это делает структуру передаваемых сообщений прозрачной для монитора безопасности, существенно расширяя выразительные возможности языка спецификации политик: монитор безопасности может проверить, что структура сообщения соответствует объявленному интерфейсу, проанализировать части сообщения и связать определенные правила с данным взаимодействием. Для спецификации свойств безопасности можно одновременно использовать разные модели/формализмы. Например, ролевой доступ или политики на основе конечных автоматов. Разные правила комбинируются всегда с помощью конъюнкции – чтобы какое-либо событие было разрешено, необходимо, чтобы оно было разрешено всеми правилами, ассоциированными с этим событием. В данной работе рассматривается мандатный контроль целостности, который реализован как формализм, который можно использовать совместно с другими для спецификации политики безопасности решения.

Как было отмечено, каждая сущность или ресурс является *доменом безопасности*, с которым ассоциирован *контекст безопасности*, определяемый *идентификатором*. Таким образом, с точки зрения политики безопасности, программно-аппаратная система, управляемая KasperskyOS, представляет собой множество доменов безопасности и информационных потоков между ними.

3.1 Взаимодействие между процессами

В KasperskyOS сущности обмениваются друг с другом сообщениями посредством механизма синхронного межпроцессного взаимодействия. Выделяются две роли: клиент (сущность, инициирующая взаимодействие) и сервер (сущность, обрабатывающая обращение). Клиент направляет серверу сообщение-запрос, при этом поток исполнения, из которого производится обращение, блокируется до получения ответа от сервера или ядра (в случае, например, ошибки). Серверный поток исполнения находится в ожидании сообщений. При получении запроса, этот поток получает управление, обрабатывает запрос и отсылает сообщение-ответ. При получении сообщения-ответа клиентский поток разблокируется и может продолжать исполнение.

Предположим, что поток T_1 сущности P_1 вызывает метод, объявленный в интерфейсе другой сущности P_2 (рис. 3). В этом случае P_1 отправляет *запрос* сущности P_2. В этот момент T_1 приостанавливает свое исполнение, ожидая ответа от P_2. Запрос проходит через механизм IPC, реализованный в микроядре. Механизм IPC отправляет сообщение монитору безопасности, который обрабатывает его согласно политике безопасности. Если политика безопасности разрешает выполнение запроса, то запрос направляется сущности P_2. P_2 затем обрабатывает запрос (в соответствии с реализацией метода) и

отправляет *ответ* сущности P_1. Ответ может содержать данные, вычисленные сущностью P_2 или просто сигнал о завершении обработки запроса. Ответ затем проходит путь через механизм IPC и монитор безопасности. Монитор проверяет, может ли ответ быть доставлен согласно политике безопасности. Если да, то P_1 получает ответ и T_1 продолжает свое исполнение. Если отправка запроса или ответа запрещена монитором безопасности, то попытка взаимодействия между P_1 и P_2 прерывается и T_1 продолжает свое исполнение.

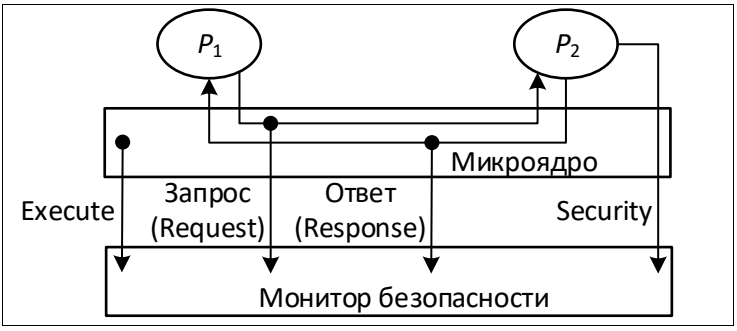


Рис. 3. Схема межпроцессного взаимодействия
Fig. 3. Interprocess communication scheme

Монитор безопасности имеет возможность контролировать любые IPC сообщения. С этой целью ядро передает монитору на проверку как сообщение-запрос, так и ответ. Передача сообщения происходит только в случае, если монитор дал разрешение. Кроме операций по контролю запросов и ответов, монитор также поддерживает две специальных операции – execute и security.

Операция *execute* используется ядром для сообщения монитору о создании новой сущности. Операция *security* позволяет сущностям отправлять сообщения монитору безопасности напрямую. Данная операция является основой для реализации сервисов (драйверов), которые предоставляют контролируемый доступ к ресурсам.

3.2 Управление доступом к ресурсам

В KasperskyOS большинство сервисов реализуется как непривилегированные приложения. Такие сервисы часто предоставляют доступ к ресурсам определенного вида. Смысл понятия «ресурс» определяется самим сервисом. Например, для файловой системы ресурсами могут быть файлы и каталоги. Для сетевой подсистемы – интерфейсы или сокеты. Поскольку данные сервисы не входят в ядро операционной системы, то контролировать доступ к ним только возможностями ядра невозможно – ядро операционной системы ничего не знает ни о семантике этих ресурсов, ни о том, какие операции над ними возможны. Тем не менее, задача разграничивать доступ к таким ресурсам возникает, и для ее решения в KasperskyOS выработан следующий поход.

Ключевым понятием является *драйвер* – это процесс-сервис, который вводит в систему новый тип ресурсов. Например, драйвер файловой системы вводит тип ресурсов «файл». Доступ к ресурсам определенного типа возможен только путем обращения к соответствующему драйверу. С каждым ресурсом драйвер ассоциирует системный дескриптор, с которым подсистема безопасности связывает информацию, необходимую для контроля доступа (например, метку целостности или роль).

Во время каждого доступа к ресурсам драйвер ресурсов должен предоставлять монитору безопасности необходимый контекст. Монитор безопасности затем разрешает или запрещает доступ. При доступе к ресурсу, драйвер использует операцию security для обращения к подсистеме безопасности. В этом обращении драйвер передает дескриптор

ресурса и любую дополнительную информацию, которую посчитает необходимой для осуществления контроля. Например, при чтении файла передается информация о том, кто (дескриптор клиента) хочет выполнить операцию (чтение файла), над чем (дескриптор файла). Монитор безопасности применяет ассоциированные с этим обращением правила и возвращает результат проверки. Ответственность драйвера – правильно проинтерпретировать ответ монитора: заблокировать операцию, если она не была разрешена. Таким образом, для реализации контроля над ресурсами, драйвер должен предоставить механизм, а политика определяется монитором безопасности. Такая архитектура позволяет подсистеме безопасности единообразно трактовать как системные ресурсы (в первую очередь, сами процессы-сущности и IPC-каналы), так и прочие ресурсы, введенные в систему драйверами. Само ядро операционной системы при этом абстрагировано от таких ресурсов и занимается только управлением (выдачей и квотированием) системных дескрипторов.

Ядро операционной системы гарантирует разделение адресных пространств сущностей, поэтому драйвер может управлять только теми ресурсами, системные дескрипторы на которые были выданы именно этому драйверу.

Безусловно, уровень доверия (в самом общем смысле) некоторого ресурса не может быть выше, чем уровень доверия драйвера этого ресурса. Любая политика безопасности должна учитывать это обстоятельство. Это становится важным в случае, когда драйвер является отдельным приложением, не входящим в доверенную вычислительную базу.

Рис. 4 иллюстрирует типовой случай выполнения доступа к ресурсу (цифрами обозначен порядок выполнения операций). Здесь сущность *P*, Драйвер и его Ресурс являются отдельными доменами безопасности. Сущность *P*, которая хочет получить доступ к ресурсу отправляет соответствующее сообщение драйверу этого ресурса. В этом сообщении ресурс может быть идентифицирован различными способами, например, по имени. Драйвер ресурса вычисляет идентификатор ресурса. Затем драйвер ресурса отправляет сообщение монитору безопасности, предоставляя контекст: дескриптор инициатора запроса *P*, дескриптор ресурса и другую информацию. Монитор безопасности определяет, может ли *P* получить доступ к ресурсу согласно политике безопасности. Если доступ разрешен, то драйвер получает доступ к ресурсу и отправляет ответ сущности *P*. Ответ содержит данные, полученные по результатам доступа к ресурсу, и идентификатор ресурса, который может быть использован сущностью в дальнейшем.

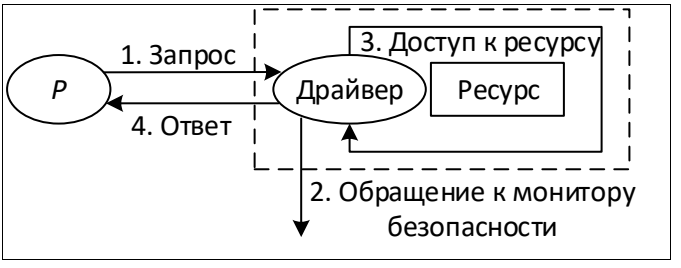


Рис. 4. Схема обращения к ресурсу посредством его драйвера
Fig. 4. Resource access via its driver

4. Модель мандатного контроля целостности

4.1 Информационные потоки

Нахождение системы в целостном состоянии означает, что сущности системы работают корректно. Для обеспечения их корректной работы механизмы мандатного контроля

целостности должны обеспечить независимость сущностей от данных с меньшим уровнем доверия (за исключением особенных, специально определенных случаев).

Для моделирования влияния данных в системе на сущности и объекты, а также для формализации свойств мандатного контроля целостности, в работе используется понятие информационного потока.

Информационным потоком от источника к приемнику называется преобразование данных в приемнике, зависящее от данных, находящихся в источнике [8]. В качестве как источника, так и приемника может выступать сущность или объект. Информационный поток от источника x к приемнику y может возникнуть в результате выполнения различных операций, например, когда сущность x получает доступ на запись в объект y или когда сущность y получает доступ на чтение к объекту x. Во всех случаях поток будем обозначать $(x, y, write_m)$ – информация передается от x к y (x «записывает» в y, отсюда название потока $write$). Индекс «m» подчеркивает, что рассматриваются информационные потоки по памяти.

4.2 Основные компоненты модели

4.2.1 Сущности и объекты

S – множество *сущностей* (субъектов) системы, представляющее активные компоненты операционной системы. O – множество *объектов* системы, представляющее пассивные компоненты операционной системы.

Для целей моделирования иерархии объектов введена *функция иерархии объектов* $HO: O \to 2^O$, которая ставит в соответствие каждому объекту $c \in O$ множество объектов $HO(c)$. Будем говорить, что объекты из $HO(c)$ непосредственно находятся в *контейнере* c.

Выделяется множество *корневых объектов* $CR \subseteq O$, не находящихся ни в каком контейнере.

Для целей моделирования управления работой с ресурсами системы посредством драйверов ресурсов, в модели введено понятия *драйвера объектов*. Драйвер объектов – сущность, посредством которой выполняются операции с объектами. Будем говорить, что драйвер объектов управляет подмножеством объектов.

Функция $OD: O \to S$ ставит в соответствие любому объекту системы его драйвер. Обратное отношение $OD^{-1} \subseteq S \times O$ позволяет определить объекты, управляемые данным драйвером.

4.2.2 Доступы и информационные потоки

$R_a = \{read_a, write_a\}$ – множество *видов доступа*, где $read_a$ обозначает доступ на чтение, $write_a$ обозначает доступ на запись.

$R_f = \{write_m\}$ – множество *видов информационных потоков*, где $write_m$ обозначает информационный поток по памяти на запись в сущность или объект.

$A \subseteq S \times O \times R_a$ – множество *доступов* сущностей к объектам.

$F \subseteq (S \cup O) \times (S \cup O) \times R_f$ – множество *информационных потоков*.

4.2.3 Уровни целостности

$(LI, \leq)$ – решетка *уровней целостности*. *Уровень целостности* сущностей и объектов определен с помощью функции $il: S \cup O \to LI$.

При работе операционной системы возникают ситуации, в которых возможно участие сущности в «опасных» взаимодействиях, то есть обращениях на чтение к объектам с

более низким либо несравнимым уровнем целостности. Для разрешения таких взаимодействий в модели введена функция $ilr: S \to LI$, определяющая для каждой сущности минимальный уровень целостности объектов, к которым эта сущность может обращаться на чтение. Модель построена таким образом, что для любой сущности $s \in S$ выполняется $ilr(s) \leq il(s)$.

Для элементов $x, y \in LI$ под $x < y$ будем иметь в виду $x \leq y \wedge x \neq y$.

$UP: S \to \{false, true\}$ – *функция привилегий*, служащая для обозначения сущностей, которым разрешено повышать уровень целостности объектов. Предполагается, что в качестве таких сущностей могут выступать доверенные сущности, для которых значение данной функции устанавливается равным $true$.

4.2.4 Система переходов

В соответствии с распространенным подходом [25], в качестве модели компьютерных систем, управляемых операционной системой, используется система переходов.

Системой переходов называется кортеж

$$TS = (States, OP, \to, G_0),$$

где

- $States$ – множество состояний,
- OP – множество правил преобразования состояний,
- $\to \subseteq States \times OP \times States$ – отношение переходов,
- $G_0 \in States$ – начальное состояние.

Для краткости под обозначением $G \overset{op}{\to} G'$ будем понимать $(G, op, G') \in \to$. Если правило преобразования состояний в данном контексте неважно, будем писать $G \to G'$.

Состояние системы переходов $TS = (States, OP, \to, G_0)$ определено как кортеж

$$G = (S, O, OD, A, F, HO, (il, ilr), UP).$$

Примечание. В рамках модели мы не рассматриваем процесс определения функции UP. Предполагается, что ее значение в каждом состоянии системы переходов определено корректно. Если $UP(s) = true$ для некоторой сущности s, то эта сущность действительно обладает привилегией повышения уровня целостности объектов. Если же сущность s не обладает такой привилегией, то $UP(s) = false$.

4.2.5 Захват контроля над сущностями и объектами

Минимизация доверенной вычислительной базы – в частности, не включение в нее драйверов ресурсов, – приводит к необходимости учета требований к сущностям, не обладающим максимальным уровнем целостности. Однако по каким-либо причинам (например, ошибка в исходном коде сущности) эти требования могут не выполняться. Описание таких причин может быть составлено на основе деталей реализации системы, что находится за рамками модели. При этом, как обсуждалось выше, целесообразно выполнить анализ последствий нарушения требований и возникновения непредвиденных событий в системе в рамках модели. Для этого в модели введено понятие *захвата контроля* над сущностью или объектом. Если над сущностью захвачен контроль, то предположения о ее поведении более не выполняются. Захват контроля над объектом означает появление в нем данных, которые он не должен содержать при корректном функционировании системы. Формально последствия захвата контроля над сущностями и объектами описаны далее в правилах преобразования состояний: если результат выполнения правила зависит от того, захвачен ли контроль над сущностью или объектом, являющимся параметром правила, то сформулирован результат такой зависимости. Далее предполагается, что в каждом состоянии модели определено

множество $CS \subseteq S$ сущностей, над которыми захвачен контроль, и множество $CO \subseteq O$ объектов, над которыми захвачен контроль.

Как будет доказано в разделе 4.5, модель исключает неограниченный захват контроля частей системы: если захвачен контроль над частью системы, максимальный уровень целостности компонент которой ниже максимально возможного уровня целостности, то это не приводит к захвату контроля над более целостными компонентами системы и, как следствие, всей системы в целом.

4.3 Требования к начальному состоянию

Представленная модель разрабатывалась таким образом, чтобы к начальному состоянию

$$G_0 = (S_0, O_0, OD_0, A_0, F_0, HO_0, (il_0, ilr_0), UP_0)$$

системы переходов TS предъявлялось как можно меньше требований. Предполагается, что в начальном состоянии существует сущность $core \in S_0$, которая представляет ядро операционной системы, и определены значения уровней целостности $il_0(core)$ и $ilr_0(core)$. Значение $il_0(core)$ является максимально возможным значением уровня целостности: ядро операционной системы является наиболее доверенной сущностью. Остальные множества, определенные в модели, являются пустыми.

В случае необходимости в качестве начального можно использовать и состояние с большим количеством непустых компонент. В таком случае к начальному состоянию модели предъявляются следующие требования.

- Для каждого объекта определен его драйвер:
$$\forall o \in O_0. \exists d \in S_0. OD_0(o) = d.$$
- Для всех объектов $o \in O_0$ определено значение $il_0(o)$.
- Для всех сущностей $s \in S_0$ определены значения $il_0(s)$ и $ilr_0(s)$.
- Уровень целостности каждого объекта не выше уровня целостности его драйвера:
$$\forall o \in O_0. il_0(o) \leq il_0\big(OD_0(o)\big).$$
- Для каждого объекта, не являющегося корневым, существует не менее целостный контейнер, в котором находится этот объект:
$$\forall o \in O_0 \setminus CR_0. \exists c. o \in HO_0(c) \wedge il_0(o) \leq il_0(c).$$
- Для всех сущностей $s \in S_0$ определено значение $UP_0(s)$.
- Множество доступов является пустым: $A_0 = \emptyset$.
- Множество информационных потоков является пустым: $F_0 = \emptyset$.

4.4 Правила преобразования состояний

В результате анализа операций межпроцессного взаимодействия, возникающих в ряде приложений KasperskyOS, был определен набор правил преобразования состояний, которые могут быть использованы для контроля таких взаимодействий с целью обеспечения целостности системы.

Каждое правило преобразования состояний описывает переход из состояния $G = (S, O, OD, A, F, HO, (il, ilr), UP)$ в состояние $G' = (S', O', OD', A', F', HO', (il', ilr'), UP')$ и представлено с помощью пред- и постусловий. Если все части предусловия правила истинны в состоянии G, то постусловие этого правила определяет отношение между компонентами состояния G' и соответствующими компонентами состояния G. В определении правил указаны только те компоненты G', которые изменяются при осуществлении перехода.

Предусловия правил, моделирующих действия сущностей, должны быть реализованы в операционной системе. В связи с этим предусловия разрабатывались так, чтобы они были по возможности простыми. Постусловия в основном служат для анализа модели.

Правила преобразования состояний принимают параметры, список которых указывается в скобках после имени правила.

Правила, описывающую работу с объектами, разработаны таким образом, что в предусловии каждого из них присутствует требование того, что уровень целостности объекта меньше либо равен уровню целостности драйвера этого объекта. В связи с этим информационный поток от драйвера к объекту не несет угрозы целостности. Информационный поток от объекта к драйверу может оказать влияние на драйвер только в случае захвата контроля над драйвером. Однако в этом случае уже захвачен контроль над не менее целостной сущностью, чем объект, в связи с чем информационный поток от объекта к этой сущности не приведет к расширению зоны захвата (подробнее это разобрано ниже). Эти особенности модели позволяют не рассматривать в правилах потоки от драйверов к их объектам и от объектов к их драйверам.

В постусловиях некоторых правил используются конструкции if-then-else (как правило, для описания эффектов от факта компрометации параметров правила), смысл которых такой же, как и во многих популярных языках программирования.

4.4.1 Получение доступа на чтение к объектам

Операции получения доступа к ресурсам операционной системы выполняются посредством драйверов ресурсов. В корректно работающей системе должно выполняться следующее свойство.

Свойство драйверов ресурсов 0

При доступе сущности к ресурсу драйвер этого ресурса должен инициировать сравнение уровней целостности сущности и ресурса, реализованное в механизме мандатного контроля целостности.

Таким образом, в модели предполагается, что при доступе сущности s к объекту o, если над драйвером объекта o не захвачен контроль, то будет выполнено сравнение уровней целостности сущности s и объекта o. Если над драйвером захвачен контроль, то будем считать, что сравнение уровней выполнено не будет. В связи с этим модель разработана таким образом, чтобы уровень целостности драйвера был не меньше уровня целостности объектов, которыми он управляет, а также выполнялись определенные гарантии по поводу соотношения уровней целостности драйвера объектов и сущности, получающей доступ к этим объектам.

Рассмотрим операцию получения сущностью x доступа на чтение к объекту y. Эта операция выполняется посредством драйвера d объекта y: драйвер получает доступ к объекту и передает данные из объекта в сущность x. Возникающие информационные потоки показаны на рис. 5.

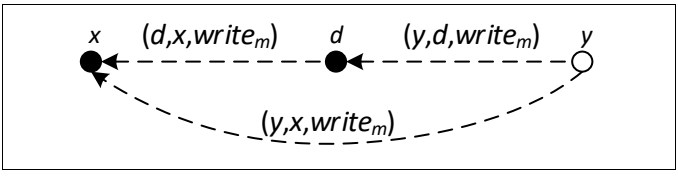


Рис. 5. Возможные информационные потоки при получении сущностью x доступа на чтение к объекту y посредством драйвера d

Fig. 5. Information flows that may arise if entity x has acquired read access to object y via driver d

В корректно работающей системе драйверы ресурсов должны обладать следующим свойством.

Свойство драйверов ресурсов 1

1. Драйвер ресурсов должен быть реализован таким образом, чтобы информационные потоки от ресурсов, которыми он управляет, не влияли на его корректную работу.
2. При доступе на чтение сущностей к ресурсу, управляемому драйвером, драйвер должен передавать данные из ресурса в неизменном виде.

Свойство 1.1 позволяет не учитывать поток $(y, d, write_m)$. Свойство 1.2 позволяет не учитывать поток $(d, x, write_m)$. Из этого свойства следует, что достаточно учитывать только информационный поток $(y, x, write_m)$ от объекта y к сущности x, получающей к нему доступ на чтение.

В случае, если над драйвером d захвачен контроль, d более не обладает свойством 1. Важным становится информационный поток $(d, x, write_m)$, поскольку d может передавать в x произвольные данные. Информационный поток $(y, d, write_m)$ можно не учитывать и в этом случае, поскольку над d и так уже захвачен контроль.

Получение доступа на чтение к данным с большим или равным уровнем целостности является стандартной операцией.

Случай, когда $\neg\big(il(x) \le il(y)\big) \wedge \big(ilr(x) \le il(y)\big)$, и над сущностью x не захвачен контроль, является особенным: предполагается, что сущность x может безопасно считывать менее целостные данные (либо данные, чей уровень целостности не сравним с $il(x)$, но сравним с $ilr(x)$; в дальнейшем будем называть такие данные просто менее целостными), то есть продолжать функционировать согласно своей спецификации. В этом случае информационных потоков к сущности x не возникает (в предположении, что над сущностью x не захвачен контроль).

В случае, если над драйвером d захвачен контроль, сравнение уровней целостности x и y может не выполняться. Однако ядром операционной системы обеспечивается аналогичное сравнение уровней целостности x и d. Ядро операционной системы также гарантирует выполнение условий $OD(y) = d$ и, следовательно, $il(y) \le il(d)$.

Таким образом, правило имеет вид:

$read(x, d, y)$: сущность x получает доступ на чтение к объекту y посредством драйвера объектов d.

Предусловие: $x, d \in S$; $y \in O$; $OD(y) = d$;

$\big(il(x) \le il(d)\big) \vee \Big(\neg\big(il(x) \le il(d)\big) \wedge \big(ilr(x) \le il(d)\big)\Big)$;

$d \notin CS \Rightarrow \big(il(x) \le il(y)\big) \vee \Big(\neg\big(il(x) \le il(y)\big) \wedge \big(ilr(x) \le il(y)\big)\Big)$;

$il(y) \le il(d)$.

Постусловие: $A' = A \cup \{(x, y, read_a)\}$;

if $d \notin CS$ then {

  if $il(x) \le il(y) \wedge il(x) \le il(d) \vee x \in CS$ then $F' = F \cup \{(y, x, write_m)\}$

}

else {

  if $il(x) \le il(d) \vee x \in CS$ then $F' = F \cup \{(y, x, write_m), (d, x, write_m)\}$

}.

4.4.2 Получение доступа на запись к объектам

Рассмотрим операцию получения сущностью x доступа на запись к объекту y. Эта операция выполняется посредством драйвера d объекта y: драйвер получает данные для записи от сущности x, доступ к объекту y и записывает полученные данные в объект. Возникающие информационные потоки показаны на рис. 6.

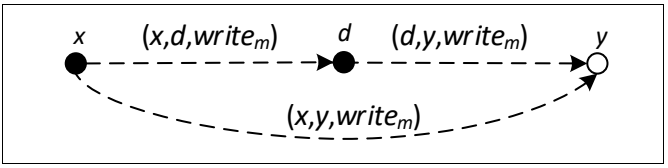


Рис. 6. Возможные информационные потоки при получении сущностью x доступа на запись к объекту y посредством драйвера d

Fig. 6. Information flows that may arise if entity x has acquired write access to object y via driver d

В корректно работающей системе драйвера ресурсов должны обладать следующим свойством.

Свойство драйверов ресурсов 2

1. Драйвер ресурсов должен быть реализован таким образом, чтобы информационные потоки от сущностей, использующих его сервисы по записи в ресурсы, которыми он управляет, не влияли на его корректную работу.
2. При доступе на запись сущностей к ресурсу, управляемому драйвером, драйвер должен записывать в ресурс те же данные, которые ему были переданы сущностью.

Свойство 2.1 позволяет не учитывать поток $(x, d, write_m)$. Свойство 2.2 позволяет не учитывать поток $(d, y, write_m)$.

В случае, если над драйвером d захвачен контроль, d более не обладает свойством 2. Важным становится информационный поток $(x, d, write_m)$, поскольку данные от x могут привести к неспецифицированному поведению драйвера d. Информационный поток $(d, y, write_m)$ можно не учитывать и в этом случае, поскольку уровень целостности объекта y не больше уровня целостности сущности d, и в рамках модели можно описать захват контроля над y: с помощью последовательности правил $write(d, d, y)$ и $control(y)$, которые будут описаны далее.

Таким образом, правило имеет вид:

$write(x, d, y)$: сущность x получает доступ на запись к объекту y посредством драйвера объектов d.

Предусловие: $x, d \in S$; $y \in O$; $OD(y) = d$;

$d \notin CS \Rightarrow il(y) \leq il(x)$;

$il(y) \leq il(d)$.

Постусловие: $A' = A \cup \{(x, y, write_a)\}$;

if $d \notin CS$ then $F' = F \cup \{(x, y, write_m)\}$

else $F' = F \cup \{(x, y, write_m), (x, d, write_m)\}$.

4.4.3 Создание сущностей и объектов, удаление, перемещение, повышение уровня целостности объектов

$execute(x, y, s, ils, ilsr)$: сущность x создает (запускает) сущность s из объекта y с установлением уровней целостности сущности s.

Предусловие: $x \in S$; $y \in O$; $s \notin S \cup O$; $ils \leq il(y)$; $ilsr \leq ils$.

Постусловие: $S' = S \cup \{s\}$; $il'(s) = ils$; $ilr'(s) = ilsr$;

if $y \in CO$ then $F' = F \cup \{(y, s, write_m)\}$.

Если над объектом y захвачен контроль, то возникает информационный поток от y к сущности s с тем, чтобы в дальнейшем можно было моделировать распространение захвата контроля с помощью правила $control(s)$.

В правилах $create$, $create\_root$, $move$ и $delete$ возникновение информационных потоков объясняется аналогично правилу $write$.

$create(x, y, z, d, ily)$: сущность x создает объект y посредством драйвера d, включает y в состав контейнера z и устанавливает уровень целостности y в ily.

Предусловие: $x, d \in S$; $y \notin S \cup O$; $z \in O$; $(x, z, write_a) \in A$; $(d, z, write_a) \in A$;
$ily \le il(x)$; $ily \le il(z)$; $ily \le il(d)$.
Постусловие: $O' = O \cup \{y\}$; $HO'(z) = HO(z) \cup \{y\}$; $HO'(y) = \emptyset$; $OD'(y) = d$; $il'(y) = ily$;
if $d \in CS$ then { $F' = F \cup \{(x, y, write_m), (x, d, write_m)\}$; $CO' = CO \cup \{y\}$ }.

$create\_root(x, y, d, ily)$: сущность x создает корневой объект y посредством драйвера d и устанавливает уровень целостности y в ily.

Предусловие: $x, d \in S$; $y \notin S \cup O$; $ily \le il(x)$; $ily \le il(d)$.
Постусловие: $O' = O \cup \{y\}$; $CR' = CR \cup \{y\}$; $HO'(y) = \emptyset$; $OD'(y) = d$; $il'(y) = ily$;
if $d \in CS$ then { $F' = F \cup \{(x, y, write_m), (x, d, write_m)\}$; $CO' = CO \cup \{y\}$ }.

$move(x, y, d, from, to)$: сущность x перемещает объект y из контейнера $from$ в контейнер to посредством драйвера d.

Предусловие: $x, d \in S$; $y \in O$; $OD(y) = d$; $from, to \in O$; $from \ne to$; $y \ne from$; $y \ne to$;
$y \in HO(from)$;
$(x, from, write_a) \in A$; $(x, to, write_a) \in A$; $(d, from, write_a) \in A$; $(d, to, write_a) \in A$;
$d \notin CS \Rightarrow il(y) \le il(x)$;
$il(y) \le il(d)$; $il(y) \le il(to)$.
Постусловие: $HO'(from) = HO(from) \setminus \{y\}$;
$HO'(to) = HO(to) \cup \{y\}$.
if $d \in CS$ then $F' = F \cup \{(x, y, write_m), (x, d, write_m)\}$.

$delete(x, y, d, z)$: сущность x удаляет объект y из контейнера z посредством драйвера d.

Предусловие: $x, d \in S$; $y \in O$; $z \in O$; $y \in HO(z)$; $OD(y) = d$;
$(x, z, write_a) \in A$; $(d, z, write_a) \in A$; $HO(y) = \emptyset$; $il(y) \le il(x)$; $il(y) \le il(d)$.
Постусловие: $O' = O \setminus \{y\}$;
if $y \in CO$ then $CO' = CO \setminus \{y\}$;
$A' = A \setminus \{(s, y, \alpha_a) : s \in S, \alpha_a \in R_a\}$;
$F' = F \setminus (\{(x, y, write_m) : x \in S \cup O\} \cup \{(y, x, write_m) : x \in S \cup O\})$;
$HO'(z) = HO(z) \setminus \{y\}$;
if $d \in CS$ then $F' = F' \cup \{(x, d, write_m)\}$.

$upgrade(x, y, z, d, ily)$: сущность x изменяет уровень целостности объекта y посредством драйвера d, объект y находится в контейнере z, новое значение уровня целостности объекта y равно ily.

Предусловие: $x \in S$; $y \in O$; $OD(y) = d$; $z \in O$; $UP(x) = true$;
$il(y) \le il(x)$; $ily \le il(x)$; $y \in HO(z)$; $ily \le il(z)$; $ily \le il(d)$; $il(y) < ily$.
Постусловие: $il'(y) = ily$.

4.4.4 Взаимодействие сущностей

Наиболее распространенным случаем межпроцессного взаимодействия является вызов одной сущностью метода другой сущности с тем, чтобы вторая сущность выполнила определенное действие и вернула результат. Если сущность x вызывает метод сущности y, то в рамках такого взаимодействия возможны два информационных потока:

- $(x, y, write_m)$ – данные, передаваемые сущностью x при обращении к сущности y за сервисом сущности y (параметры вызываемого метода сущности y).
- $(y, x, write_m)$ – данные, возвращаемые сущности x после того, как вызванный метод сущности y завершил свою работу.

В корректной системе должно выполняться следующее предположение.

Предположение 1. Поведение сущности не должно приводить к некорректному состоянию системы независимо от того, какие данные ей переданы в результате вызова ее метода.

Таким образом, в случае, если над сущностью y не захвачен контроль, то поток $(x, y, write_m)$ учитывать не нужно.

Соотношения уровней целостности сущностей x и y аналогичны случаю получения сущностью доступа на чтение к объекту.

Описанный случай взаимодействия сущностей представлен в модели с помощью правила $call$:

$call(x, y)$: сущность x вызывает метод сущности y с целью получения данных от сущности y.

Предусловие: $x, y \in S$; $\big(il(x) \le il(y)\big) \vee \Big(\neg\big(il(x) \le il(y)\big) \wedge \big(ilr(x) \le il(y)\big)\Big)$.

Постусловие: if $il(x) \le il(y) \vee x \in CS$ then {

if $y \notin CS$ then $F' = F \cup \{(y, x, write_m)\}$

else $F' = F \cup \{(y, x, write_m), (x, y, write_m)\}$ }.

В рамках особого случая межпроцессного взаимодействия сущность передает данные менее целостной сущности. Например, сущность может быть подписана на получение данных от более доверенной сущности. В модели такое взаимодействие задано с помощью правила $invoke$.

При моделировании такого взаимодействия необходимо всегда учитывать информационный поток от сущности, передающей данные к сущности, получающей их. При этом в корректной системе должно выполняться следующее предположение.

Предположение 2. Если сущность вызывает метод другой сущности посредством $invoke$, то данные-ответ, полученные от последней в ходе IPC-взаимодействия, не влияют на корректность работы первой.

Таким образом, если над сущностью x не захвачен контроль, то поток $(y, x, write_m)$ учитывать не нужно.

$invoke(x, y)$: сущность x вызывает метод сущности y с целью передачи данных сущности y.

Предусловие: $x, y \in S$; $il(y) \le il(x)$.

Постусловие: if $x \notin CS$ then $F' = F \cup \{(x, y, write_m)\}$

else $F' = F \cup \{(x, y, write_m), (y, x, write_m)\}$.

4.4.5 Возникновение неявных информационных потоков

Ранее представленные (де-юре) правила моделируют действия сущностей в системе и определяют информационные потоки, возникающие в результате таких действий. Однако информационные потоки также могут возникать неявно, в результате ранее совершенных действий. Для моделирования возникновения таких потоков в модель добавлены правила, основанные на идее *де-факто передач* [26]. Следуя [8, 27], будем называть такие правила де-факто правилами.

В отличие от де-юре правил, де-факто правила не предназначены для реализации в операционной системе и используются для целей, связанных с синтезом и анализом модели.

$pass(x, z, y)$: «z передает (passes) данные из x в y».

Предусловие: $z \in S; x \in O; y \in S \cup O; x \neq y; (z, x, read_a) \in A; (z, y, write_m) \in F;$

$\neg\Big(\neg\big(il(z) \leq il(x)\big) \wedge ilr(z) \leq il(x) \vee \neg\Big(il(z) \leq il\big(OD(x)\big)\Big) \wedge ilr(z) \leq il\big(OD(x)\big)\Big) \vee z \in CS.$

Постусловие: $F' = F \cup \{(x, y, write_m)\}$.

Если сущность z может безопасно считывать менее целостные данные с меньшим либо равным уровнем целостности (из объекта x) и над ней не захвачен контроль, то сущность z безусловно не передает эти данные в сущность или объект y, следовательно, информационный поток от x к y не возникает. В остальных случаях передача данных происходит.

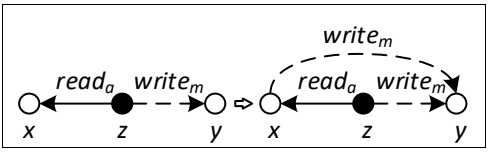


Рис. 7. Реализация информационного потока с помощью правила pass
Fig. 7. Realization of information flow via the pass de facto rule

$post(x, z, y)$: «x отправляет (posts) данные y через z».

Предусловие: $x, y \in S; z \in O; x \neq y; (y, z, read_a) \in A; (x, z, write_m) \in F;$

$\neg\Big(\neg\big(il(y) \leq il(z)\big) \wedge ilr(y) \leq il(z) \vee \neg\Big(il(y) \leq il\big(OD(z)\big)\Big) \wedge ilr(y) \leq il\big(OD(z)\big)\Big).$

Постусловие: $F' = F \cup \{(x, y, write_m)\}$.

Элемент предусловия $\neg\Big(\neg\big(il(y) \leq il(z)\big) \wedge ilr(y) \leq il(z) \vee \neg\Big(il(y) \leq il\big(OD(z)\big)\Big) \wedge ilr(y) \leq il\big(OD(z)\big)\Big)$ говорит о том, что сущность y получила доступ на чтение к объекту z не как сущность, которая может безопасно считывать менее целостные данные. В ином случае, аналогично правилам $read$ и $call$, поток от x к y не возникает.

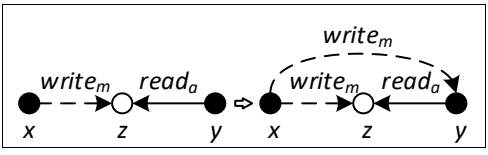


Рис. 8. Реализация информационного потока с помощью правила post
Fig. 8. Realization of information flow via the post de facto rule

$find(x, z, y)$: «y находит (finds) данные из x через z».

Предусловие: $x, z \in S; y \in S \cup O; x \neq y; \{(x, z, write_m), (z, y, write_m)\} \subseteq F.$

Постусловие: $F' = F \cup \{(x, y, write_m)\}$.

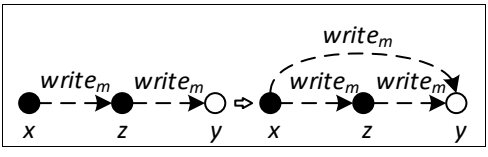


Рис. 9. Реализация информационного потока с помощью правила find
Fig. 9. Realization of information flow via the find de facto rule

4.4.6 Распространение зоны захвата контроля

Для моделирования распространения зоны захвата контроля в модели введено де-факто правило, описывающее изменения множества $CS \cup CO$: если к сущности или объекту имеется поток от сущности или объекта из $CS \cup CO$, то эта сущность или объект попадает в множество сущностей и объектов, над которыми захвачен контроль. Предусловие этого правила проверяет наличие потока от элемента множества $CS \cup CO$, а постусловие добавляет того, к кому этот поток, в множество $CS \cup CO$. Таким образом, наличие потока в текущем состоянии приведет к попаданию во множество захваченных в следующем состоянии. В случае, если захват контроля осуществляется над драйвером объектов, контроль также захватывается и над всеми объектами, которыми управляет этот драйвер.

$control(x)$: захват контроля над сущностью или объектом x.

Предусловие: $x \in (S \cup O) \setminus (CS \cup CO)$; $\exists y \in CS \cup CO.(y, x, write_m) \in F$.

Постусловие: $CS' \cup CO' = CS \cup CO \cup \{x\}$;

if $x \in S$ then $CO' = CO' \cup OD^{-1}(x)$.

4.5 Свойства безопасности, обеспечиваемые моделью

Операция $upgrade$, которая должна выполняться в реальной системе только доверенными сущностями, подразумевает отсутствие информационных потоков к объекту, уровень целостности которого предполагается повысить. В противном случае источник такого информационного потока может стать менее целостным, чем приемник. В реальной системе для гарантии отсутствия таких потоков используются специальные средства (например, криптографические). Такие средства не рассматриваются в рамках модели. Поэтому при анализе модели на предмет свойств безопасности, которые она обеспечивает, мы рассматриваем только такие вычисления системы переходов, на которых доверенные сущности не выполняют операции, которые могут нарушить целостность системы.

Назовем *вычислениями без кооперации доверенных и недоверенных сущностей для реализации информационных потоков* такие вычисления системы переходов TS, на которых доверенные сущности не инициируют применение правила $upgrade$.

Теорема 1 показывает, что либо возникающие информационные потоки направлены от не менее целостных сущностей или объектов, либо расширение зоны захвата контроля носит строго ограниченный характер. Ограничение заключается в том, что в этом случае в множестве захваченных компонентов всегда уже имеется сущность с уровнем целостности большим либо равным уровню целостности компонента, к которому реализован информационный поток (рис. 10).

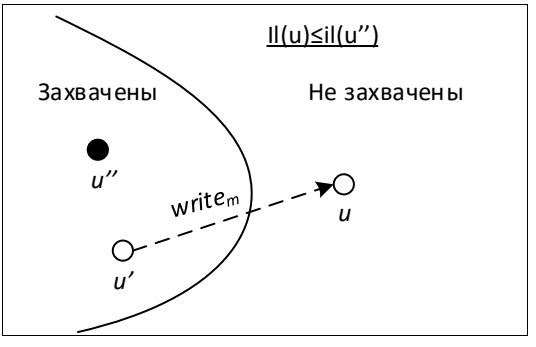


Рис. 10. Ограничение расширения зоны захвата контроля
Fig. 10. Restriction of the area of compromised components

Теорема 1. Для всех состояний $G = (S, O, OD, A, F, HO, (il, ilr), UP) \in States$ системы переходов $TS = (States, OP, \rightarrow, G_0)$, достижимых из начального состояния G_0 и принадлежащих вычислениям системы переходов TS без кооперации доверенных и недоверенных сущностей для реализации информационных потоков, выполняется:

Для всех сущностей и объектов $u, u' \in S \cup O$, если существует информационный поток от u' к u, то либо уровень целостности u меньше либо равен уровню целостности u', либо в множестве сущностей и объектов, находящихся под контролем, существует сущность u'', такая, что уровень целостности u меньше либо равен ее уровню целостности:

$$\forall G \in Reach(TS). \forall u, u' \in S \cup O. (u', u, write_m) \in F \Rightarrow il(u) \leq il(u') \vee \exists u'' \in CS. il(u) \leq il(u'').$$

Доказательство. Доказательство утверждения теоремы проведем методом математической индукции по длине пути системы переходов TS. Выберем произвольный путь $G_0 G_1 G_2$ ... системы TS.

Базис индукции. Длина пути $n = 0$.

В соответствии с требованиями к начальному состоянию, $F_0 = \emptyset$. Следовательно, доказываемое утверждение верно.

Индуктивная гипотеза. Пусть для некоторого $n \geq 0$ верно:

$$\forall k \leq n. \forall u, u' \in S_k \cup O_k. (u', u, write_m) \in F_k \Rightarrow il_k(u) \leq il_k(u') \vee \exists u'' \in CS_k. il_k(u) \leq il_k(u'').$$

Шаг индукции.

Рассмотрим переход $G_n \rightarrow G_{n+1}$ системы переходов TS. Докажем, что

$$\forall u, u' \in S_{n+1} \cup O_{n+1}. (u', u, write_m) \in F_{n+1} \Rightarrow il_{n+1}(u) \leq il_{n+1}(u') \vee \exists u'' \in CS_{n+1}. il_{n+1}(u) \leq il_{n+1}(u'').$$

Рассмотрим произвольные $u, u' \in S_{n+1} \cup O_{n+1}$. Предположим, что $(u', u, write_m) \in F_{n+1}$ (гипотеза). Докажем, что

$$il_{n+1}(u) \leq il_{n+1}(u') \vee \exists u'' \in CS_{n+1}. il_{n+1}(u) \leq il_{n+1}(u'').$$

Рассмотрим все правила, порождающие переход $G_n \rightarrow G_{n+1}$, и для каждого правила – информационные потоки, которые могут возникнуть в результате применения этого правила.

Правило $read(x, d, y)$.

Случай 1. $d \notin CS_n$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это поток $(y, x, write_m)$. Если $x \notin CS_n$, то, согласно предусловию правила, $il_n(x) \leq il_n(y)$. Следовательно, $il_{n+1}(x) \leq il_{n+1}(y)$. Если $x \in CS_n$, то в качестве u'' можно выбрать x, поскольку правило $read$ не изменяет множество CS и $x \in CS_{n+1}$.

Случай 2. $d \in CS_n$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это один из следующих потоков.

$(y, x, write_m)$. Если $x \notin CS_n$, то $il_n(x) \leq il_n(d)$. Следовательно, $il_{n+1}(x) \leq il_{n+1}(d)$, и в качестве u'' можно выбрать d. Если $x \in CS_n$, то в качестве u'' можно выбрать x, поскольку правило $read$ не изменяет множество CS и $x \in CS_{n+1}$.

$(d, x, write_m)$. Если $x \notin CS_n$, то $il_n(x) \leq il_n(d)$. Следовательно, $il_{n+1}(x) \leq il_{n+1}(d)$, и в качестве u'' можно выбрать d. Если $x \in CS_n$, то в качестве u'' можно выбрать x.

Правило $write(x, d, y)$.

Случай 1. $d \notin CS_n$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это поток $(x, y, write_m)$. Согласно предусловию правила, $il_n(y) \leq il_n(x)$. Следовательно, $il_{n+1}(y) \leq il_{n+1}(x)$.

Случай 2. $d \in CS_n$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это один из следующих потоков.

$(x, y, write_m)$. Согласно предусловию, $il_n(y) \leq il_n(d)$. Следовательно, $il_{n+1}(y) \leq il_{n+1}(d)$, и в качестве u'' можно выбрать d.

$(x, d, write_m)$. В качестве u'' можно выбрать саму сущность d.

Правило $execute(x, y, s, ils, ilsr)$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это поток $(y, s, write_m)$. Согласно предусловию правила, $ils \leq il_n(y)$. Следовательно, $il_{n+1}(s) \leq il_{n+1}(y)$.

Правила $create(x, y, z, d, ily)$, $create\_root(x, y, d, ily)$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это один из следующих потоков.

$(x, y, write_m)$. Согласно предусловию, $ily \leq il_n(x)$. Следовательно, $il_{n+1}(y) \leq il_{n+1}(x)$.

$(x, d, write_m)$. В качестве u'' можно выбрать саму сущность d.

Правило $move(x, y, d, from, to)$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это один из следующих потоков.

$(x, y, write_m)$. Так как при этом $d \in CS_n$, то, в качестве сущности u'' можно выбрать сущность d, поскольку в соответствии с предусловием $il_n(y) \leq il_n(d)$.

$(x, d, write_m)$. В качестве u'' можно выбрать саму сущность d.

Правило $delete(x, y, d, z)$.

Информационный поток из гипотезы – это поток $(x, d, write_m)$, возникающий, если $d \in CS_n$. В этом случае $d \in CS_{n+1}$, и эту сущность можно выбрать в качестве u''.

Правило $control$ не добавляет информационные потоки рассматриваемого вида, не изменяет уровни целостности и не удаляет сущности. Поэтому в соответствии с индуктивной гипотезой в этом случае утверждение шага индукции оказывается истинным.

Правило $call(x, y)$.

Случай 1. $y \notin CS_n$. Информационный поток из гипотезы – это поток $(y, x, write_m)$. Если $x \notin CS_n$, то $il_n(x) \leq il_n(y)$. Следовательно, $il_{n+1}(x) \leq il_{n+1}(y)$. Если $x \in CS_n$, то в качестве u'' можно выбрать x.

Случай 2. $y \in CS_n$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это один из следующих потоков.

$(y, x, write_m)$. Если $x \notin CS_n$, то $il_n(x) \leq il_n(y)$. Следовательно, $il_{n+1}(x) \leq il_{n+1}(y)$. Если $x \in CS_n$, то в качестве u'' можно выбрать x.

$(x, y, write_m)$. В качестве u'' можно выбрать саму сущность y.

Правило $invoke(x, y)$.

Случай 1. $x \notin CS_n$. Информационный поток из гипотезы – это поток $(x, y, write_m)$. Согласно предусловию, $il_n(y) \leq il_n(x)$. Следовательно, $il_{n+1}(y) \leq il_{n+1}(x)$.

Случай 2. $x \in CS_n$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это один из следующих потоков.

$(x, y, write_m)$. Поскольку $il_n(y) \leq il_n(x)$, то $il_{n+1}(y) \leq il_{n+1}(x)$.

$(y, x, write_m)$. В качестве u'' можно выбрать саму сущность x.

Правило $pass(x, z, y)$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это поток $(x, y, write_m)$.

Согласно предусловию правила $pass$, $(z, x, read_a) \in A_n$. Анализ постусловий всех правил показывает, что доступ $(z, x, read_a) \in A_n$ мог быть получен только с помощью правила $read(z, OD(x), x)$, где $OD(x)$ – драйвер объекта x. Другими словами, одним из $op_i, i = 1, \dots, n$ на вычислении $G_0 op_1 G_1 \dots op_n G_n$ является $op_l = read(z, OD(x), x), 1 \leq l \leq n$. Поскольку случай, когда сущность z может безопасно получать доступ на чтение менее целостных данных в данном правиле не приводит к возникновению информационных потоков, то, согласно предусловию правила $read$ для случая $z \notin CS_l$

(поскольку драйвер объекта остается неизменным на вычислениях системы переходов TS, для его обозначения будем просто писать OD, без индекса состояния):

1. если $OD(x) \notin CS_l$, то $il_l(z) \leq il_l(x)$;
2. если $OD(x) \in CS_l$, то $il_l(z) \leq il_l\big(OD(x)\big)$.

Поскольку уровни целостности и факты захвата контроля не меняются на вычислениях системы переходов TS, итого имеем

$$\big(OD(x) \notin CS_n \wedge il_n(z) \leq il_n(x)\big) \vee \Big(OD(x) \in CS_n \wedge il_n(z) \leq il_n\big(OD(x)\big)\Big) \vee z \in CS_n.$$

Согласно предусловию правила $pass$ также имеем $(z, y, write_m) \in F_n$. Согласно индуктивной гипотезе,

$$il_n(y) \leq il_n(z) \vee \exists u'' \in CS_n . il_n(y) \leq il_n(u'').$$

Если $\exists u'' \in CS_n . il_n(u) \leq il_n(u'')$, то сразу заключаем $\exists u'' \in CS_{n+1} . il_{n+1}(y) \leq il_{n+1}(u'')$.

Если $il_n(y) \leq il_n(z)$, то имеем три случая:

1. $il_n(y) \leq il_n(z) \wedge il_n(z) \leq il_n(x)$, следовательно, $il_n(y) \leq il_n(x)$, а значит $il_{n+1}(y) \leq il_{n+1}(x)$;
2. $il_n(y) \leq il_n(z) \wedge il_n(z) \leq il_n\big(OD(x)\big)$, следовательно, $il_n(y) \leq il_n\big(OD(x)\big)$, а значит $il_{n+1}(y) \leq il_{n+1}\big(OD(x)\big)$. Поскольку в этом случае $OD(x) \in CS_{n+1}$, то в качестве искомого $u'' \in CS_{n+1}$ можно взять $OD(x)$;
3. $il_n(y) \leq il_n(z) \wedge z \in CS_n$, следовательно, $z \in CS_{n+1} \wedge il_{n+1}(y) \leq il_{n+1}(z)$, и в качестве искомого $u'' \in CS_{n+1}$ можно взять z.

Таким образом, проанализированы все возможные случаи, и можно заключить, что

$$il_{n+1}(y) \leq il_{n+1}(x) \vee \exists u'' \in CS_{n+1} . il_{n+1}(y) \leq il_{n+1}(u'').$$

Правило $post(x, z, y)$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это поток $(x, y, write_m)$.

Согласно предусловию правила $post$, $(y, z, read_a) \in A_n$. Следовательно,

$$\big(OD(z) \notin CS_n \wedge il_n(y) \leq il_n(z)\big) \vee \Big(OD(z) \in CS_n \wedge il_n(y) \leq il_n\big(OD(z)\big)\Big) \vee y \in CS_n.$$

Согласно предусловию правила $post$ также имеем $(x, z, write_m) \in F_n$. В соответствии с индуктивной гипотезой,

$$il_n(z) \leq il_n(x) \vee \exists u'' \in CS_n . il_n(z) \leq il_n(u'').$$

Если $\exists u'' \in CS_n . il_n(z) \leq il_n(u'')$, то $\exists u'' \in CS_{n+1} . il_{n+1}(z) \leq il_{n+1}(u'')$. Имеем три случая:

1. $\exists u'' \in CS_n . il_n(z) \leq il_n(u'') \wedge il_n(y) \leq il_n(z)$, следовательно, $\exists u'' \in CS_n . il_n(y) \leq il_n(u'')$, а значит $\exists u'' \in CS_{n+1} . il_{n+1}(y) \leq il_{n+1}(u'')$.
2. $\exists u'' \in CS_n . il_n(z) \leq il_n(u'') \wedge il_n(y) \leq il_n\big(OD(z)\big)$, следовательно, $il_{n+1}(y) \leq il_{n+1}\big(OD(z)\big)$. Поскольку в этом случае $OD(z) \in CS_{n+1}$, то в качестве искомого $u'' \in CS_{n+1}$ можно взять $OD(z)$;
3. $\exists u'' \in CS_n . il_n(z) \leq il_n(u'') \wedge y \in CS_n$, следовательно, $y \in CS_{n+1}$, и в качестве искомого $u'' \in CS_{n+1}$ можно взять саму сущность y.

Если $il_n(z) \leq il_n(x)$, то имеем три случая:

1. $il_n(z) \leq il_n(x) \wedge il_n(y) \leq il_n(z)$, следовательно, $il_n(y) \leq il_n(x)$, а значит $il_{n+1}(y) \leq il_{n+1}(x)$;
2. $il_n(z) \leq il_n(x) \wedge il_n(y) \leq il_n\big(OD(z)\big)$, следовательно, $il_{n+1}(y) \leq il_{n+1}\big(OD(z)\big)$. Поскольку в этом случае $OD(z) \in CS_{n+1}$, то в качестве искомого $u'' \in CS_{n+1}$ можно взять $OD(z)$;

3. $il_n(z) \le il_n(x) \wedge y \in CS_n$, следовательно, $y \in CS_{n+1}$, и в качестве искомого $u'' \in CS_{n+1}$ можно взять саму сущность y.

Таким образом, проанализированы все возможные случаи, и можно заключить, что

$$il_{n+1}(y) \le il_{n+1}(x) \vee \exists u'' \in CS_{n+1}.\, il_{n+1}(y) \le il_{n+1}(u'').$$

Правило $find(x, z, y)$. Информационный поток $(u', u, write_m) \in F_{n+1}$ из гипотезы – это поток $(x, y, write_m)$.

Согласно предусловию правила, $\{(x, z, write_m), (z, y, write_m)\} \subseteq F_n$. Следовательно, в соответствии с индуктивной гипотезой,

$$\big(il_n(z) \le il_n(x) \vee \exists u'' \in CS_n.\, il_n(z) \le il_n(u'')\big) \wedge \big(il_n(y) \le il_n(z) \vee \exists u'' \in CS_n.\, il_n(y) \le il_n(u'')\big).$$

Итого имеем четыре случая:

1. $il_n(z) \le il_n(x) \wedge il_n(y) \le il_n(z)$, следовательно, $il_{n+1}(y) \le il_{n+1}(x)$;
2. $il_n(z) \le il_n(x) \wedge \exists u'' \in CS_n.\, il_n(y) \le il_n(u'')$, следовательно, $\exists u'' \in CS_{n+1}.\, il_{n+1}(y) \le il_{n+1}(u'')$.
3. $il_n(y) \le il_n(z) \wedge \exists u'' \in CS_n.\, il_n(z) \le il_n(u'')$, следовательно, $\exists u'' \in CS_{n+1}.\, il_{n+1}(y) \le il_{n+1}(u'')$.
4. $\big(\exists u'' \in CS_n.\, il_n(z) \le il_n(u'')\big) \wedge \big(\exists u'' \in CS_n.\, il_n(y) \le il_n(u'')\big)$, следовательно, $\exists u'' \in CS_{n+1}.\, il_{n+1}(y) \le il_{n+1}(u'')$.

Таким образом, проанализированы все возможные случаи, и можно заключить, что

$$il_{n+1}(y) \le il_{n+1}(x) \vee \exists u'' \in CS_{n+1}.\, il_{n+1}(y) \le il_{n+1}(u'').$$

Теорема доказана.

5. Пример использования политики мандатного контроля целостности

Рассмотрим применение политик мандатного контроля целостности для обеспечения процесса безопасного обновления системы, работающей под управлением KasperskyOS. Под «безопасным» в данном случае понимается такое обновление, перед осуществлением которого была проверена цифровая подпись образа обновления. С точки зрения процесса обновления, компоненты системы могут получать и применять обновления. В этом процессе непосредственно принимают участие следующие приложения (сущности):

- `Downloader` загружает образ обновления с удаленного ресурса. Это приложение напрямую взаимодействует с недоверенной удаленной системой, и потенциально над ним может быть захвачен контроль. Поэтому данному приложению назначен низкий уровень целостности.
- `Verifier` проверяет цифровую подпись образа обновления. Это доверенное приложение, которое может считывать низкоцелостный образ обновления с целью проверки его цифровой подписи.
- `Updater` обновляет систему. Это доверенное приложение, которое должно получать данные только из высокоцелостных файлов.
- `FileSystem` реализует файловое хранилище (является драйвером для файлов).

Эти приложения реализуют следующий сценарий. `Downloader` загружает образ обновления и сохраняет его с использованием сервисов `FileSystem`. Затем `Verifier` проверяет цифровую подпись образа обновления. Если подпись является корректной, приложение дает инструкцию `FileSystem` на создание нового высокоцелостного файла, который является копией оригинального образа обновления. Наконец, `Updater` считывает новый файл и применяет обновление.

5.1 Определение интерфейсов

Перед написанием политики безопасности необходимо определить интерфейсы компонентов системы.

Следующий фрагмент определяет интерфейс операционной системы, который используется ядром для сообщения монитору безопасности о создании новых сущностей:

```
interface execute {
  exec (in Handle image);
}
```

Указанный в этом интерфейсе метод `exec` используется для создания новых сущностей. Метод принимает идентификатор образа, из которого необходимо создать новую сущность.

Следующий фрагмент определяет интерфейс, по которому сущность `FileSystem` передает необходимый контекст монитору безопасности:

```
interface FileSystem_Security {
  create (in Handle client,
          in Handle directory,
          in Handle object,
          in Label label);
  // …
}
```

Интерфейс определяет, как `FileSystem` взаимодействует с монитором безопасности для контроля доступа к файлам. В частности, когда `FileSystem` создает новые объекты, оно отправляет сообщение с использованием метода `create`. В ответ сущность получает решение о предоставлении доступа, в соответствии с которым она должна продолжить свою работу. Следовательно, драйвер ресурсов предоставляет контекст и действует согласно решениям о предоставлении доступа к его ресурсам, а монитор безопасности реализует политику безопасности.

Также определяется интерфейс самой сущности `FileSystem`:

```
interface FileSystem {
  // …
  write (in Handle object,
         in Data data,
         out DataLength count,
         out RetCode ret);
  read (in Handle object,
        in DataLength count,
        out Data data,
        out RetCode ret);
}
```

Другие приложения могут использовать этот интерфейс для работы с файлами.

5.2 Определение политики безопасности

Для определения политики безопасности системы необходимо разработать спецификацию на языке PSL. Мандатный контроль целостности представлен в языке *классом политик* `mandatory_integrity_control`, который определяет множество *правил*, соответствующих правилам модели. Спецификация политики безопасности определяет соответствие этих правил и взаимодействий в системе. При каждой попытке взаимодействия монитор безопасности исполняет правила для определения решения о допустимости взаимодействия.

Для использования класса политик необходимо создать на его основе *объект политики*, указав для него конфигурацию. Для объекта класса `mandatory_integrity_control` необходимо указать уровни целостности.

Политика безопасности определяет для каждого межпроцессного взаимодействия, какие правила класса политик будут исполнены монитором для разрешения или запрета взаимодействия. В число таких взаимодействий входят создание сущностей и объектов и другие взаимодействия.

В нашем примере сначала определяется объект политик `integrity` как экземпляр класса `mandatory_integrity_control`:

```
policy object integrity =
  mandatory_integrity_control {
    config : {
      levels : [“LOW”, “MEDIUM”, “HIGH”]
    }
}
```

В конфигурации этого объекта политик указаны три уровня целостности `LOW`, `MEDIUM` и `HIGH`. Если уровни целостности заданы в виде списка значений (как в данном примере), то задан полный порядок на множестве уровней целостности. Следовательно, `LOW < MEDIUM < HIGH`.

Далее задается политика создания сущностей:

```
execute method=exec, dst=Downloader {
  integrity.execute {
    target : dst,
    image  : message.image,
    level  : “LOW” }
}
execute method=exec, dst=Verifier {
  integrity.execute {
    target : dst,
    image  : message.image,
    level  : “HIGH”, levelR: “LOW” }
}
execute method=exec, dst=Updater {
  integrity.execute {
    target : dst,
    image  : message.image,
    level  : “HIGH” }
}
execute method=exec, dst=FileSystem {
  integrity.execute {
    target : dst,
    image  : message.image,
    level  : “HIGH” }
}
```

Язык спецификации политик предоставляет директиву `execute` для назначения правил, которые должны быть исполнены монитором безопасности при создании сущностей, и предоставления монитору необходимого контекста. В данном примере ядро предоставляет контекст посредством метода `exec` интерфейса `execute`. Доступ к этому контексту может быть получен посредством объекта `message` (например, `message.image`). Также неявно передаются два идентификатора: `src` – идентификатор сущности, обратившейся к ядру для создания новой сущности с идентификатором `dst`.

Правило `integrity.execute` имеет следующие параметры: `target` определяет создаваемую сущность, `image` определяет объект, из которого должна быть создана сущность, `level` и `levelR` задают уровни целостности (*ils* и *ilsr* в правиле *execute*) новой сущности. Если `levelR` опущен, то `levelR=level`.

Секция `request` соответствует всем IPC-запросам (первым частям синхронного межпроцессного взаимодействия) и подразумевает два неявным параметра: `src` для инициатора запроса и `dst` для получателя. Следующий фрагмент определяет, что для каждого IPC-запроса, `integrity.call` применяет правило *call* с аргументами `source` (`src`) и `target` (`dst`).

```
request {
  integrity.call {
    source : src,
    target : dst }
}
```

Секция `security` соответствует прямым обращениям сущностей к монитору безопасности. В ней есть неявный параметр `src` – идентификатор этой сущности.

При создании файлов `FileSystem`, как драйвер файловых ресурсов, предоставляет контекст монитору безопасности путем вызова метода `create` интерфейса `FileSystem_Security`:

```
security src=FileSystem {
  match method=create {
    integrity.create {
      initiator : message.client,
      target    : message.object,
      container : message.directory,
      driver    : src,
      level     : message.label }
  }
}
```

Сообщения, которые `FileSystem` отправляет монитору безопасности, предоставляют следующий контекст:

- идентификатор сущности, которая инициирует запрос на создание объекта (`message.client`);
- идентификатор объекта, который должен быть создан (`message.object`). Этот идентификатор создается сущностью `FileSystem`;
- идентификатор контейнера, в котором объект должен быть создан (`message.directory`). Контейнер определяется сущностью `FileSystem` и не может иметь уровень целостности, больший, чем уровень целостности сущности `FileSystem`;
- идентификатор драйвера файловых ресурсов (`src`);
- метка, определяющая уровень целостности, который должен быть назначен создаваемому файлу (`message.label`).

Если `container` не указан, то данная операция соответствует правилу *create_root*, и `initiator` должен быть равен `driver`. Только драйверы могут создавать корневые контейнеры для ресурсов, которые они предоставляют. Следовательно, уровень целостности драйвера никогда не может быть ниже уровня целостности его ресурсов.

Правила, соответствующие запросам на чтение и запись к `FileSystem`, определяются следующим образом:

```
request dst=FileSystem {
  match method=read {
    integrity.read {
```

```
      reader : src,
      object : message.object }
    }
  match method=write {
    integrity.write {
      writer : src,
      object : message.object }
  }
}
```

Методы `read` и `write` сущности `FileSystem` соответствуют правилам модели $read(reader, FileSystem, object)$ и $write(writer, FileSystem, object)$ соответственно, то есть происходит автоматическое указание драйвера объекта равным параметру методов `dst`.

5.3 Применение политики

Рассмотрим, как разработанная политика позволяет обеспечивать целостность системы (рис. 11). В соответствии с данной политикой, загруженный образ обновления имеет уровень целостности `LOW`. `Verifier` может считать данный файл и проверить его цифровую подпись. Если верификация успешна, `Verifier` создает новый файл с содержимым старого файла и уровнем целостности `HIGH`. Для применения обновления сущность `Updater` должна получить доступ на чтение к образу обновления посредством метода `read` сущности `FileSystem`. Определение того, будет ли данный доступ предоставлен или нет, происходит в соответствии с правилом $read$. Это означает, что монитор безопасности выполняет следующие проверки:

$$\big(il(Updater) \le il(f)\big) \vee \Big(\neg\big(il(Updater) \le il(f)\big) \wedge \big(ilr(Updater) \le il(f)\big)\Big),$$
$$il(f) \le il(FileSystem),$$

где f – идентификатор образа обновления. Если файл был верифицирован, то $il(f) = HIGH$, и условие $il(Updater) \le il(f)$ истинно. Следовательно, монитор безопасности разрешит предоставление рассматриваемого доступа, и `Updater` сможет применить обновление.

Если файл не был верифицирован, то $il(f) = LOW$, и условие $il(Updater) \le il(f)$ ложно. Вторая часть дизъюнкции также ложна, поскольку хоть и $\neg\big(il(Updater) \le il(f)\big)$ истинно, но $ilr(Updater) \le il(f)$ ложно. Следовательно, монитор безопасности запретит выдачу такого доступа на чтение, и `Updater` не сможет применить обновление. Таким образом, монитор безопасности предотвратит попытки обновления системы с использованием образов обновления, уровень целостности которых не равен `HIGH`.

Предположим, что существует еще одна файловая система с уровнем целостности `MEDIUM`. В этом случае с использованием ее сервисов сущность `Downloader` смогла бы записать образ обновления, сущность `Verifier` – проверить его, но сущность `Updater` не смогла бы получить доступ на чтение к этому файлу. Это связано с тем, что правило по получению доступа на чтение к объекту проверяет, что уровень целостности этого объекта не выше уровня целостности драйвера, посредством которого осуществляется доступ. Другими словами, для доступа к файлу f посредством драйвера $driver$ условие

$$il(Updater) = HIGH \wedge il(Updater) \le il(f) \wedge il(f) \le il(driver)$$

может быть удовлетворено только если $HIGH \le il(driver)$, что не выполняется, если $il(driver) = MEDIUM$. Таким образом, если система содержит драйверы с различными

уровнями целостности, то политика мандатного контроля целостности автоматически принимает это во внимание при контроле доступа к ресурсам.

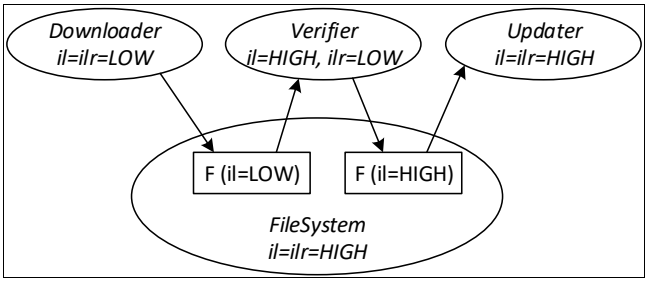


Рис. 11. Схема обработки и применения образа обновления. F – файл-образ обновления
Fig. 11. Update image processing and application scheme. F is the image file

6. Заключение

В работе представлена модель мандатного контроля целостности для микроядерной операционной системы KasperskyOS. В отличие от других моделей, данная модель учитывает наличие драйверов ресурсов при доступах к ресурсам, что позволяет избавиться от предположения доверенности драйверов (присущего ранее существовавшим моделям), которое не позволяло моделировать микроядерные системы с драйверами разного уровня доверия. Тем не менее, даже недоверенные драйверы должны выполнять определенные присущие им функции. В связи с этим сформулированы требования, которым должны удовлетворять драйверы ресурсов и другие сущности, при выполнении которых не возникает угрозы целостности системы (не возникают информационные потоки от менее целостных компонент системы к более целостным либо компонентам с несравнимым уровнем целостности). Проведен анализ модели в случае, когда эти требования не выполняются (например, когда захвачен контроль над некоторыми компонентами системы).

Сформулирована и доказана теорема о допущении моделью либо только разрешенных информационных потоков, либо об ограничении зоны захвата контроля в случае наличия захваченных компонентов. Разработан язык спецификации политик мандатного контроля целостности. Транслятор этого языка создает исполнимый монитор безопасности, выполняющий проверки мандатного контроля целостности при доступе к ресурсам приложений, работающих под управлением операционной системы KasperskyOS. Продемонстрирован пример использования языка для разработки политики безопасности системы с безопасным обновлением. Показано, как мандатный контроль целостности запрещает использование образов обновления с непроверенной цифровой подписью.

Список литературы / References

[1]. Jaeger T. Operating System Security. Morgan and Claypool Publishers, 2008, 220 p.

[2]. Landwehr C. Formal Models for Computer Security. ACM Computing Surveys, vol. 13, issue 3, 1981, pp. 247-278.

[3]. Федеральная служба по техническому и экспортному контролю. Информационное сообщение о требованиях по безопасности информации, устанавливающих уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий от 29 марта 2019 г. / FSTEC Russia. Information message on information security requirements establishing levels of confidence in information security tools and information technology security tools dated March 29, 2019. Available at: https://fstec.ru/normotvorcheskaya/informatsionnye-i-analiticheskie-materialy/1812-

informatsionnoe-soobshchenie-fstek-rossii-ot-29-marta-2019-g-n-240-24-1525 (in Russian), accessed 14.01.2020.

[4]. Young M. et al. The duality of memory and communication in the implementation of a multiprocessor operating system. In Proc. of the 11th Symposium on Operating Systems Principles, 1987, pp. 63-76.

[5]. Hohmuth M., Peter M., Härtig H., Shapiro J. Reducing TCB Size by Using Untrusted Components: Small Kernels versus Virtual-Machine Monitors. In Proc. of the 11th ACM SIGOPS European Workshop, 2004..

[6]. Biggs S., Lee D., Heiser G. The Jury Is. In Proc. of the 9th Asia-Pacific Workshop on Systems. 2018, Article No. 16.

[7]. Herder J. N., Bos H., Gras B., Homburg P., Tanenbaum A. S. Construction of a Highly Dependable Operating System. In Proc. of the Sixth European Dependable Computing Conference, 2006, pp. 3-12.

[8]. Девянин П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Горячая линия-Телеком, 2013, 338 с. / Devyanin P.N. Security models of computer systems. Control for access and information flows. Hotline-Telecom, 2013, 338 p. (in Russian).

[9]. Biba K. Integrity Considerations for Secure Computer Systems. Technical report MTR-3153, The MITRE Corporation, 1977.

[10]. Lipner S. Non-Discretionary Controls for Commercial Applications. In Proc. of the 1982 IEEE Symposium on Security and Privacy, 1982. pp. 2-10.

[11]. Clark D., Wilson D. A Comparison of Commercial and Military Computer Security Policies. In Proceedings of the 1987 IEEE Symposium on Security and Privacy, 1987, pp. 184-195.

[12]. Lee, T. Using Mandatory Integrity to Enforce “Commercial” Security. In Proc. of the 1988 IEEE Symposium on Security and Privacy, 1988, pp. 140-146.

[13]. Brewer D., Nash M. The Chinese Wall Security Policy. In Proc. of the 1989 IEEE Symposium on Security and Privacy, 1989, pp. 206-214.

[14]. Liu Z., Wang T., Li W. An Integrity Control Model for Operating System. In Proc. of the 2009 International Conference on Management and Service Science, 2009, pp. 1-4.

[15]. Bell D., LaPadula L. Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report MTR-2997 Rev. 1, The MITRE Corporation, 1976.

[16]. Li N., Mao Z., Chen H. Usable Mandatory Integrity Protection for Operating Systems. In Proc. of the 2007 IEEE Symposium on Security and Privacy, 2007, pp. 164-178.

[17]. Zhai E. et al. SecGuard: Secure and Practical Integrity Protection Model for Operating Systems. In Proc.of the 13th Asia-Pacific Web Conference, 2011, pp. 370-375.

[18]. Devyanin P. et al. Using Refinement in Formal Development of OS Security Model. Lecture Notes in Computer Science, vol. 9609. 2015, pp. 107-115.

[19]. Devyanin P. et al. Formal Verification of OS Security Model with Alloy and Event-B. Lecture Notes in Computer Science, vol. 8477, 2014, pp. 309-313.

[20]. П.Н. Девянин и др. Моделирование и верификация политик безопасности управления доступом в операционных системах. Горячая линия–Телеком, 2019б 214 стр. / P.N. Devyanin, D.V. Efremov, V.V. Kulyamin, A.K. Petrenko, A.V. Khoroshilov, I.V. Shchepetkov. Modeling and verification of access control security policies in operating systems. Hotline – Telecom, 2019, 214 p.

[21]. Yosifovich P., Russinovich M., Solomon D., Ionescu A. Windows Internals, Part 1: System architecture, processes, threads, memory management, and more (7th Edition). Microsoft Press. 2017. 800 p.

[22]. Rushby J. Design and Verification of Secure Systems. In Proc. of the 8th ACM Symposium on Operating Systems Principles, 1981, pp. 12-21.

[23]. Alves-Foss J., Oman P., Taylor C. The MILS Architecture for High-Assurance Embedded Systems. International Journal of Embedded Systems, vol. 2, no. 3/4, 2006, pp. 239-247.

[24]. Spencer R., Smalley S., Loscocco P., Hibler M., Andersen D., Lepreau J. The Flask Security Architecture: System Support for Diverse Security Policies. In Proc. of the 8th USENIX Security Symposium, 1999.

[25]. Baier C., Katoen J.-P. Principles of Model Checking. The MIT Press. 2008. 975 pp.

[26]. Bishop M., Snyder L. The Transfer of Information and Authority in a Protection System. In Proc. of the 7th ACM symposium on Operating systems principles, 1979, pp. 45-54.

[27]. Bishop M. Conspiracy and Information Flow in the Take-Grant Protection Model. Journal of Computer Security, vol. 4, no. 4, 1996, pp. 331-359.

Информация об авторах / Information about authors

Владимир Сергеевич БУРЕНКОВ – кандидат технических наук, разработчик-исследователь. Сфера научных интересов: модели программно-аппаратных систем, формальные методы верификации.

Vladimir Sergeevich BURENKOV – PhD, research developer. Research interests: models of computer systems, formal verification methods.

Дмитрий Александрович КУЛАГИН – кандидат технических наук, ведущий разработчик. Сфера научных интересов: информационная безопасность, компиляторы, операционные системы.

Dmitry Aleksandrovich KULAGIN – PhD, development team lead. Research interests: information security, compilers, operating systems.

DOI: 10.15514/ISPRAS-2020-32(1)-3

Система визуализации для авиационной ОС реального времени JetOS

[1] *Б.Х. Барладян, ORCID: 0000-0002-2391-2067 <bbarladian@gmail.com>*
[1] *Л.З. Шапиро, ORCID: 0000-0002-6350-851X <pls@gin.keldysh.ru>*
[2] *К.А. Маллачиев, ORCID: 0000-0002-4112-5403 <mallachiev@ispras.ru>*
[2,3,4,5] *А.В.Хорошилов, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*
[6] *Ю.А. Солоделов, ORCID: 0000-0001-5891-7645 <yasolodelov@2100.gosniias.ru>*
[1] *А.Г.Волобой, ORCID: 0000-0003-1252-8294 <voloboy@gin.keldysh.ru>*
[1] *В.А. Галактионов, ORCID: 0000-0001-6460-7539 <vlgal@gin.keldysh.ru>*
[6] *И.В. Ковернинский, ORCID: 0000-0002-8571-324X <ivkoverninsk@2100.gosniias.ru>*

[1] *Институт прикладной математики им. М.В. Келдыша РАН,*
125047, Россия, Москва, Миусская пл., д. 4
[2] *Институт системного программирования имени В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[3] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1
[4] *НИУ Высшая школа экономики,*
101978, Россия, г. Москва, ул. Мясницкая, д. 20
[5] *Московский физико-технический институт,*
141701, Россия, Московская область, г. Долгопрудный, Институтский пер., 9
[6] *Государственный научно-исследовательский институт авиационных систем,*
125319, Россия, Москва, ул. Викторенко, 7

Аннотация. В работе рассматриваются вопросы создания систем визуализации для бортовых комплексов гражданской авиации. Все программное обеспечение, используемое на борту судна, должно соответствовать международно-принятым стандартам безопасности. Это накладывает дополнительные требования и к используемому оборудованию, и к процессу разработки системы. Данная работа посвящена специфике использования многоядерных процессоров в авиационных встраиваемых системах для повышения производительности программной реализации библиотеки OpenGL SC. Возможность использования многоядерных процессоров в критических для безопасности системах обеспечивается в перспективной российской операционной системе реального времени (ОСРВ) JetOS. Рассматриваются также реализация многооконной визуализации с использованием библиотеки OpenGL SC.

Ключевые слова: OpenGL SC; встроенные системы; операционная система реального времени; многоядерные вычисления; ускорение рендеринга; многооконность; компоновщик

Для цитирования: Барладян Б.Х., Шапиро Л.З., Маллачиев К.А., Хорошилов А.В., Солоделов Ю.А., Волобой А.Г., Галактионов В.А., Ковернинский И.В. Система визуализации для авиационной ОС реального времени JetOS. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 57-70. DOI: 10.15514/ISPRAS–2020–32(1)–3

Rendering System for the Aircraft Real-Time OS JetOS

[1] *B.Kh. Barladian, ORCID: 0000-0002-2391-2067 <bbarladian@gmail.com>*
[1] *L.Z. Shapiro, ORCID: 0000-0002-6350-851X <pls@gin.keldysh.ru>*
[2] *K.A. Mallachiev, ORCID: 0000-0002-4112-5403 <mallachiev@ispras.ru>*
[2,3,4,5] *A.V. Khoroshilov, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*
[6] *Y.A. Solodelov, ORCID: 0000-0001-5891-7645 <yasolodelov@2100.gosniias.ru>*
[1] *A.G. Voloboy, ORCID: 0000-0003-1252-8294 <voloboy@gin.keldysh.ru>*
[1] *V.A. Galaktionov, ORCID: 0000-0001-6460-7539 <vlgal@gin.keldysh.ru>*
[6] *I.V. Koverninskiy, ORCID: 0000-0002-8571-324X <ivkoverninsk@2100.gosniias.ru>*

[1] *Keldysh Institute of Applied Mathematics Russian Academy of Science,*
4, Miusskaya sq., Moscow, 125047, Russia
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
[3] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[4] *National Research University, Higher School of Economics*
20, Myasnitskaya Ulitsa, Moscow, 101978, Russia
[5] *Moscow Institute of Physics and Technology (State University),*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russian Federation
[6] *State Research Institute of Aviation Systems*
7, Viktorenko street, Moscow, 125319, Russia

Abstract. The paper discusses the creation of rendering systems for airborne civil aviation systems. All software used on board must comply with internationally accepted safety standards. This imposes additional requirements on both the hardware used and the system development process. This work is devoted to the specifics of using multi-core processors in aviation embedded systems to improve the performance of software implementation of the OpenGL SC library. The possibility of using multi-core processors in safety-critical systems is provided by the Russian real-time operating system JetOS. Implementation of multi-window rendering using the software OpenGL SC library is also considered.

Keywords: OpenGL SC; embedded systems; real-time operating system; multi-core calculations; rendering acceleration; multi-windowing; compositor

For citation: Barladian B.Kh., Shapiro L.Z., Mallachiev K.A., Khoroshilov A.V., Solodelov Y.A., Voloboy A.G., Galaktionov V.A., Koverninskiy I.V. Rendering System for the Aircraft Real-Time OS JetOS. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 57-70 (in Russian). DOI: 10.15514/ISPRAS–2020–32(1)–3

1. Введение

Современные комплексы бортового оборудования самолетов проектируются на основе концепции интегрированной модульной авионики [1, 2], в основе которой лежит объединение приборов и бортовых процессоров в единую сеть, управляемую операционной системой реального времени (ОСРВ). Использование этой концепции и современного оборудования (например, компьютерных дисплеев вместо механических приборов) позволяет снизить количество кабелей и устройств на борту и, тем самым, уменьшить взлетный вес лайнера. Таким образом, возникает задача создания программного обеспечения дисплеев в кабине пилота, которое должно обеспечивать надежное отображение информации с интерактивной скоростью, используя ресурсы процессора с пониженным энергопотреблением, который устанавливается на борту.

Процесс создания авиационной техники должен следовать международным стандартам для авиационной промышленности, включая и соответствующие стандарты на разработку программного обеспечения. Без соблюдения этих стандартов и получения

соответствующих сертификатов допуск к полетам и экспорт построенных в России самолетов невозможен. Международный стандарт ARINC 653 [3] описывает требования к ОСРВ и программный интерфейс между прикладным авиационным ПО и операционной системой. Также отображение информации на пилотном дисплее должно удовлетворять стандарту OpenGL SC (Safety Critical) [4]. Важным требованием является также возможность сертификации в соответствии с требованиями авиационных стандартов DO-178C [5]. Так как сертификация графического ускорителя без участия производителя невозможна, то мы не рассматриваем вопрос создания системы визуализации с их использованием.

В работе [6] рассматривалась программная реализация графической библиотеки OpenGL SC, предназначенная для работы под управлением перспективной российской бортовой операционной системы реального времени JetOS [7]. Хотя нам удалось значительно ускорить стандартную программную реализацию OpenGL, однако достичь скорости реального времени на одном ядре типичного авиационного процессора не представляется возможным. При проведении исследований операционная система JetOS еще не была полностью реализована, в частности, не поддерживались многоядерные процессоры. В соответствии со стандартом ARINC 653 в авиационных ОСРВ нельзя использовать многопоточность. Поэтому часть исследований по возможному распараллеливанию алгоритмов была сделана под операционной системой Linux. В настоящее время JetOS поддерживает специальное расширение стандарта ARINC 653, позволяющее использовать многоядерные процессоры. Расширение называется Asymmetric Multi-Processing (AMP). Это позволило перейти к задаче ускорения визуализации, используя многоядерные процессоры, не отклоняясь от разрешенных стандартов.

2. Технология AMP

Следует отметить существенную разницу между технологией AMP и многопоточностью, используемой в Linux для ускорения OpenGL SC на многоядерных компьютерах. В то время как многопоточность обеспечивает эффективную конкурентную работу нескольких потоков в одном адресном пространстве, технология AMP в JetOS поддерживает возможность запуска нескольких модулей (т.е. экземпляров JetOS), каждое из которых работает на своем ядре процессора независимо от других модулей. Этот подход обеспечивает безопасность и надежность, необходимые для бортового программного обеспечения, но является менее эффективным в сравнении с производительностью, достигаемой при использовании технологии многопоточности. Конфигурация проекта, использующего AMP технологию, называется AMP проектом. Таким образом, AMP проект позволяет создавать приложения с параллельными вычислениями, отдельные части которых выполняются на различных ядрах процессора под управлением различных экземпляров JetOS.

Разработанное нами ускорение визуализации для многоядерного компьютера базируется на подходе распараллеливания библиотеки SC OpenGL, предложенном в [6]. В его основе лежит параллельная генерация нескольких последовательных кадров. Основными проблемами при реализации такого подхода являетются обмен информацией между различными частями приложения (различными модулями) и синхронизация их работы.

AMP проект поддерживает именованные разделяемые блоки памяти, доступ к которым возможен из различных модулей. Использование таких блоков памяти, обеспечивает обмен информацией между модулями. Для синхронизации работы модулей мы будем использовать специальные объекты, называемые **событиями**.

Для синхронизации процессов, выполняемых модулями A и B, мы используем объект событие (AMP_EVENT), доступный одновременно в обоих модулях. Для создания этого объекта мы использовали небольшие блоки памяти, доступ к которым возможен из этих

модулей. Состояние объекта **событие** определяется значением целочисленной переменной, хранящейся в этом блоке. Мы будем рассматривать только два состояния этого объекта: ***Событие взведено*** **(AMP_UP)** – значение переменной 1 и ***Событие сброшено*** (**AMP_DOWN)** – значение 0. Невозможность одновременного изменения состояния объекта **события** из двух модулей обеспечивается с помощью использования атомарных операций для доступа к объекту **событие**.

Для удобства работы с событиями нами был разработан набор функций, обладающих мнемоническими именами (см. листинг 1).

```
#define AMP_UP          1
#define AMP_DOWN    0
typedef int*  AMP_EVENT;
/// Get event state.
int AMP_GetEventState(AMP_EVENT ev)
{
  return ev[0];
}
/// Set the event in the  state "up".
void AMP_SetEvent(AMP_EVENT ev)
{
  atomic_int *atomic = (atomic_int *)ev;
  atomic_store(atomic, AMP_UP);
}
/// Set the event in the  state "down".
void AMP_ResetEvent(AMP_EVENT ev)
{
  atomic_int *atomic = (atomic_int *)ev;
  atomic_store(atomic, AMP_DOWN);
}
/// Infinitely wait while event is in state "down".
void AMP_WaitEvent(AMP_EVENT ev)
{
  RETURN_CODE_TYPE ret;
  while (ev[0] == AMP_DOWN)
  {
    TIMED_WAIT(MILLISECOND, &ret);
  }
}
```

Листинг 1. Функции для работы с событиями
Listing 1. Functions to deal with events

Функция *TIMED_WAIT(),* выполняющая необходимое ожидание, обеспечивается операционной системой JetOS. Отметим еще, что указатели, определяющие события в разных модулях, могут отличаться друг от друга в силу разных адресных пространств в них, но переменные по этим указателям имеют одинаковые значения.

3. Повышение производительности библиотеки OpenGL

Рассмотрим использование AMP технологии для повышения производительности программной реализации библиотеки OpenGL SC. Схема работы библиотеки OpenGL на самом верхнем уровне при визуализации одного кадра приложения включает в себя три основных этапа, представленных на рис. 1.

Первый и третий этапы должны выполняться строго последовательно. На первом этапе каждый кадр обрабатывается последовательно по мере поступления новых инструкций из приложения. На третьем этапе изображение копируется из внутреннего буфера в буфер экрана. Этот этап выполняется драйвером кадрового буфера. Копирование должно выполняться последовательно кадр за кадром в порядке их создания приложением. Таким

образом, выполнение первого и третьего этапов не может быть распараллелено, но они могут выполняться одновременно со вторым этапом для разных кадров. К счастью, время выполнения первого этапа для типичных приложений авионики обычно относительно мало.

Рис. 1. Схема работы OpenGL для генерации одного кадра
Fig. 1. OpenGL working scheme for generating one frame

Для распараллеливания вывода на экран с использованием технологии JetOS AMP на разных ядрах (разных экземплярах JetOS) выполняется параллельно генерация нескольких последовательных кадров. Количество их будет на единицу меньше чем количество экземпляров JetOS. В нашей схеме эти экземпляры JetOS будут выплнять роль серверов. Один экземпляр JetOS будет работать как клиент. В нем также реализуются первый и третий этапы, представленные на рис. 1.

В отличие от многопоточной реализации, AMP технология не позволяет запустить произвольное количество экземпляров JetOS. Количество экземпляров JetOS не может превышать количество ядер для данного процессора. Типичный авиационный процессор PowerPC (P3041) [8], используемый в наших исследованиях, имеет четыре ядра. Потенциально в авиационном оборудовании может использоваться процессор PowerPC (P4080) с восемью ядрами. В нашем случае возможно параллельно запустить четыре экземпляра JetOS. Они использовались следующим образом:

a. клиент – обрабатывает команды OpenGL и выводит подготовленное изображение на экран;
b. три сервера, которые выполняют генерацию кадров параллельно.

Теперь работа OpenGL с параллельной генерацией кадров может быть представлена схемой, показанной на рис. 2.

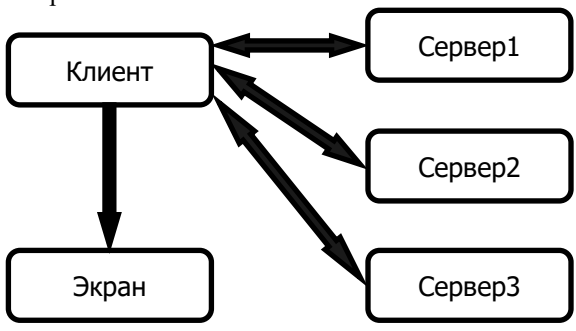


Рис. 2. Схема работы OpenGL с использованием четырех экземпляров JetOS
Fig. 2. OpenGL operational pattern using four JetOS instances

Обмен данными между клиентом и сервером осуществляется через блок общей памяти. Этот блок содержит контекст OpenGL, в котором хранятся инструкции OpenGL для генерируемого данным сервером кадра, обработанные на первом этапе (см. рис. 1). В этом контексте также выделены все соответствующие буферы, необходимые для работы OpenGL. В частности, здесь размещается буфер изображения, в котором будет создано конечное изображение данного кадра.

Для синхронизации работы клиентского модуля с несколькими серверами используются два события для каждого сервера:

a. **Start_render** – взводится клиентом, когда данные, необходимые для генерации данного кадра, готовы и сервер может ее начать;
b. **End_render –** взводится сервером, когда он завершил генерацию кадра. Тогда клиент может передать подготовленное изображение на дисплей с помощью библиотеки кадрового буфера, и продолжить обработку инструкций OpenGL для следующего кадра.

Для реализации этой пары событий введен доступный клиенту и серверу 16-байтовый блок памяти. Первая половина этого блока используется для события **Start_render**, а вторая половина для события **End_render**. Псевдокод алгоритма работы клиента показан на листинге 2.

```
1. indx=0;
2.  Обработка команд OpenGL для текущего кадра;
3.  AMP_SetEvent(Start_render [indx]);
4.  indx++
5.   While(indx < S_NUM) go to p.2
6.  indx = 0;
7. AMP_WaitEvent(End_render[indx]);
8. AMP_ResetEvent(End_render [indx]);
9. Вывод изображения, полученного от сервера, на экран;
10.    Обработка команд OpenGL для следующего кадра;
11.    AMP_SetEvent(Start_render [indx]);
12.    indx++;  indx = indx % S_NUM
13.    Go to p.7
```

Листинг 2. Псевдокод работы клиента
Listing 2. Pseudo-code of client operation

Константа S_NUM равна количеству серверов (т.е. 3 в нашем случае).

Псевдокод алгоритма работы *i*-того сервера, каждый из которых работает со своими 16-битовыми блоками, можно записать так, как показано на листинге 3.

```
1.   AMP_WaitEvent(Start_render [i]);
2.   AMP_ResetEvent(Start_render [i]);
3.   Генерация кадра;
4.   AMP_SetEvent(End_Render[i]);
5.   Go to p.1;
```

Листинг 3. Псевдокод работы i-го сервера
Listing 3. Pseudo-code of client operation

Во время инициализации все события **Start_render** устанавливаются в AMP_DOWN, а события **End_render** в AMP_UP. После запуска клиент сначала обрабатывает инструкции OpenGL для первых трех кадров и передает заполненные контексты на серверы. И только затем, после взведения события **End_render**[0], начинает передачу сгенерированных изображений на дисплей.

Также необходимо было решить проблему с указателями на контекст OpenGL, которые устанавливаются в одном модуле (на клиенте), а используются в другом (на сервере). Проблема заключается в том, что модули имеют разные адресные пространства. Данные, совместно используемые общими блоками памяти, имеют одинаковые значения, но в

адресных пространствах разных модулей указатели принимают разные значения. Таким образом, их следует корректировать в зависимости от модуля, поскольку адреса начала блока памяти в разных модулях разные. Чтобы решить эту проблему, в контексте OpenGL сохраняется значение указателя на него в адресном пространстве клиентского модуля. Используя разницу между текущим значением указателя на контекст в модуле сервера и сохраненным, корректируются указатели на сервере перед их использованием для генерации кадра, а затем они восстанавливаются перед передачей обработки очередного кадра клиенту.

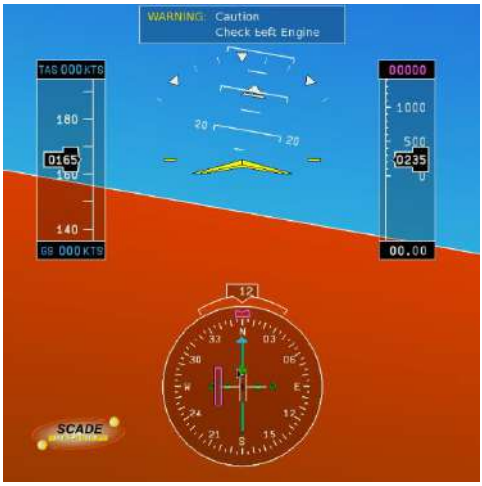


Рис. 3. Приложение GlassCockpit
Fig. 3. GlassCockpit application

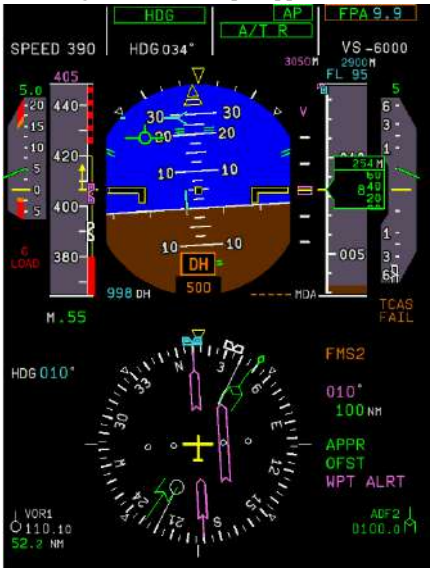


Рис. 4. Основной дисплей полета (PFD)
Fig. 4. Primary Flight Display (PFD)

Результаты тестов для приложений GlassCockpit и PFD (рис. 3 и 4), а также приложения визуализации рельефа местности SVS представлены на рис. 5. Использование трех серверов (четырех процессорных ядер) ускоряет визуализацию примерно в 2,9 раза для

приложения PFD, в 2,8 раза для приложения ClassCockpit и в 3,1 раза для приложения SVS.

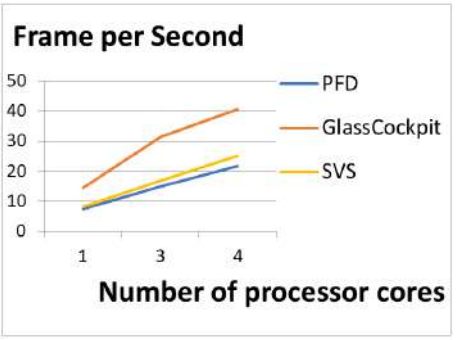


Рис. 5. Зависимость скорости визуализации от количества используемых процессорных ядер
Fig. 5. Dependence of visualization speed on the number of processor cores used

4. Многооконная визуализация

При разработке кабин современных самолетов существует тенденция использовать большие дисплеи, чтобы объединить в себе информацию о полетной навигации и состоянии оборудования самолета. Рис. 6 показывает кабину самолета MS-21, иллюстрирующую данную тенденцию.

Рис. 6. Кабина самолета MS-21
Fig. 6. Cabin of the aircraft MS-21

Количество дисплеев в кабине самолета MS-21 уменьшено (например, по сравнению с кабиной самолета Airbus A320), но дисплеи стали намного шире и позволяют отображать больше информации.

В итоге информация о полете и работе оборудования, генерируемая многочисленными системами управления полетом, должна отображаться на широкоэкранных, многофункциональных дисплеях. Она должна отображаться одновременно и в удобной для восприятия форме. В частности, этой информацией является скорость полета, указатель положения, высотомер, указатель поворота и скольжения, указатель вертикальной скорости и т. д. При этом также должны отображаться такие технические характеристики, как частота вращения двигателя, давление масла и количество топлива. Кроме того, необходимо визуализировать карту местности, различные пневматические, гидравлические и электрические цепи, данные метеорологических радаров, различные виды предупреждений и многое другое. Эта информация обычно генерируется независимыми системами, и они не должны мешать друг другу в соответствие с требованиями стандарта ARINC 653.

Для изображения информации от нескольких систем на одном экране современные операционные системы реализуют многооконный интерфейс, когда содержимое каждого приложения отображается в собственном окне. Упрощенный подход состоит в том,

чтобы позволить каждому приложению открывать неперекрывающееся с другими окно на дисплее. Такой подход позволяет ускорить визуализацию, в то же время его реализация для систем, критичных для безопасности, требует значительных усилий. Эта задача решается с помощью разработки компоновщика (compositor), который обеспечивает поддержку эффективной многооконной визуализации.

Различные подходы к реализации компоновщика для систем, критичных для безопасности, рассмотрены в [9]. Одной из реализаций компоновщика является расширение EGL_EXT_compositor для CoreAVI, который обеспечивает многооконную визуализацию для OpenGL SC 1.0.1 и OpenGL SC 2.0 [10].

Разрабатываемая нами библиотека OpenGL SC [6] предназначена для работы под операционной системой JetOS. Это определяет специфику разработки и предъявляет существенные требования к разрабатываемому коду и алгоритмам. В частности, для сертификации системы требуется полный доступ к исходным кодам библиотеки OpenGL SC и компоновщика. С другой стороны, при разработке компоновщика мы можем воспользоваться AMP технологией ОСРВ JetOS.

В работе [9] рассматриваются два основных типа графической компоновки: компоновка на аппаратном уровне и компоновка в кадровый буфер. Несмотря на то, что преимущества компоновки уровня оборудования включают хорошую производительность, энергосбережение и эффективность при работе с большим количеством обновлений, этот подход требует дополнительной полосы пропускания для отображения всех окон. Это также требует специальной поддержки драйвера кадрового буфера, который недоступен для нас на используемом в настоящее время оборудовании.

Компоновка в кадровый буфер объединяет элементы из нескольких приложений и внеэкранных буферов в один кадровый буфер. Затем кадровый буфер выводится на экран. Такая компоновка требует только одного слоя для отображения всех буферов. Фактически это единственный доступный подход в нашем случае. Схема визуализации данных при компоновке в кадровый буфер показана на рис. 7.

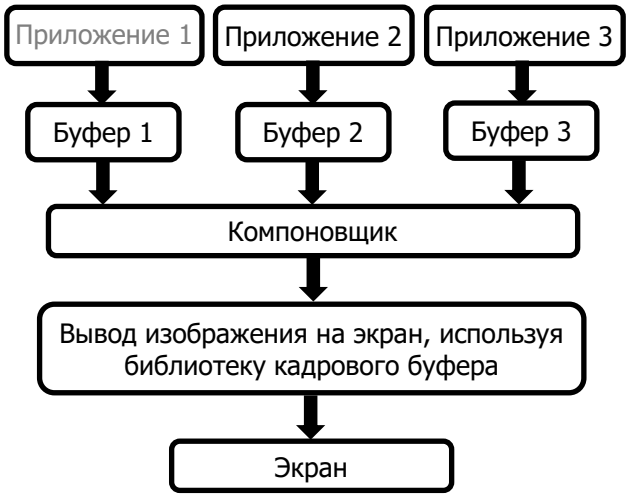


Рис. 7. Схема визуализации данных для компоновщика кадрового буфера
Fig. 7. Data visualization scheme for the framebuffer composer

Каждое приложение отображает данные с использованием библиотеки OpenGL SC в собственном буфере. Эти буферы затем передаются в компоновщик. Он формирует из них единый слой кадрового буфера и визуализирует его на дисплее с помощью

библиотеки кадрового буфера. Основной проблемой здесь является эффективная синхронизация независимо работающих приложений и компоновщика.

Как было подробно описано в докладе [11], были исследованы два варианта реализации многооконной визуализации. Первый использует стандартные средства ARINC 653 – разделы, а второй использует AMP технологию JetOS.

5. Использование разделов для многооконной визуализации

В соответствии с требованиями ARINC 653 JetOS обеспечивает разделение памяти и времени между разделами. Диспетчеризация работы разделов осуществляется на постоянной, циклической основе. Каждому разделу отводится определенная фиксированная часть от полного периода работы системы. Таким образом, обеспечивается детерминистическое поведение системы. Каждое приложение и компоновщик работают в своем разделе операционной системы. Синтезированные изображения передаются из приложения в компоновщик с помощью специальных блоков общей памяти. Каждое приложение использует собственный блок памяти для синтеза изображений. Компоновщик имеет доступ к этому блоку памяти только для чтения.

Синхронизация между приложениями и компоновщиком обеспечивается сообщениями, передаваемыми между разделами по специальным каналам связи стандарта ARINC 653. Два канала используются между каждым приложением и компоновщиком. Первый канал используется приложением для информирования компоновщика о готовности изображения к визуализации. Второй канал используется компоновщиком для информирования приложения о том, что изображение было визуализировано, и приложение может снова использовать буфер для генерации изображения следующего кадра.

Рис. 8. Компоновка двух приложений
Fig. 8. A composition of two applications

Пример результата работы такого компоновщика приведен на рис. 8. Два приложения: PFD (приведенное на рис. 4) слева и простое приложение Counter справа работают одновременно (фактически – по очереди в соответствии с требованиями ARINC 653), а изображения, создаваемые приложениями, визуализируются компоновщиком.

Предложенный подход работает правильно, но скорость визуализации в данном примере недостаточна для авиационных приложений. Оба приложения работают со скоростью ~

5 кадров в секунду. Есть несколько причин для такого поведения. Во-первых, типичный авиационный процессор PowerPC [8] обладает относительно низкой производительностью. Вторая причина заключается в том, что все разделы работают на одном ядре процессора и имеют заранее определенное время его использования. Мы можем лишь попытаться оптимизировать это разделение времени с учетом реальных потребностей приложений. В данном примере время кадра состояло из 45 мс для приложения PFD, 15 мс для приложения Counter и 16 мс для компоновщика. Этот распределение обеспечивает относительно сбалансированный доступ к процессору для данных приложений. Очевидно, что для других приложений соотношения времен будет другим. А дальнейшее ускорение возможно только за счет использования всех процессорных ядер.

6. Использование технологии AMP для многооконной визуализации

В этом случае каждое приложение и компоновщик могут выполняться на своем экземпляре операционной системы. Для передачи изображения из приложения в компоновщик каждое приложение использует собственный блок памяти, имеющий общий доступ с компоновщиком. В настоящее время AMP технология не обеспечивает пересылку сообщений между модулями, работающими на разных ядрах процессора. Поэтому для синхронизации используется механизм событий, описанный в разд. 2. Для синхронизации взаимодействия каждого приложения с компоновщиком необходимо два события:

a. **Start_copy** – взводится приложением, когда сгенерированное им изображение готово для визуализации компоновщиком;
b. **End_copy** – взводится компоновщиком, когда изображение уже было визуализировано и соответствующий блок памяти может снова использоваться приложением для генерации следующего кадра.

Тогда алгоритм работы приложения можно записать следующим образом (листинг 4):

```
While (true)
{
  AMP_WaitEvent(End_copy);
  AMP_ResetEvent(End_copy);
  Генерация изображения;
  AMP_SetEvent(Start_copy);
}
```

Листинг 4. Алгоритм работы приложения
Listing 4. Application operation algorithm

Обработка изображений компоновщиком происходит по алгоритму, показанному на листинге 5.

```
for (int i = 0; i < application number; i++)
{
  If (AMP_GetEventState(Start_copy[i]) == AMP_UP)
  {
     AMP_ResetEvent(Start_copy[i]);
     Вывод изображения на экран;
     AMP_SetEvent(End_copy[i]);
  }
}
```

Листинг 5. Обработка изображений компоновщиком
Listing 5. Image processing by the composer

На стадии инициализации все события **End_copy** взведены в состояние AMP_UP, а **Start_copy** сброшены в состояние AMP_DOWN.

Примеры изображений, полученных с помощью предложенного многооконного подхода, реализованного с использованием AMP технологии, показаны на рис. 9-11.

Рис. 9. Многооконный дисплей трех приложений: GlassCockpit, Counter, карта
Fig. 9. Multi-window display of three applications: GlassCockpit, Counter, map

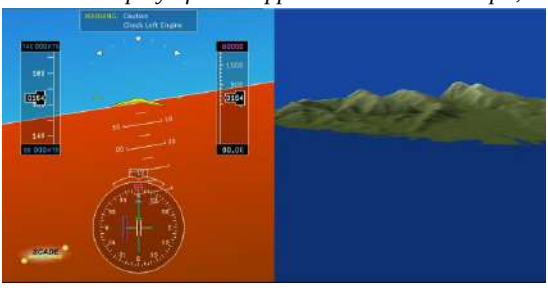


Рис. 10. Многооконный дисплей двух приложений: GlassCockpit и рельеф местности
Fig. 10. Multi-window display of three applications: GlassCockpit and topographical relief

Рис. 11. Многооконный дисплей двух приложений: PFD и состояние дверей
Fig. 11. Multi-window display of two applications: PFD and door status

Наша реализация многооконной визуализации показала следующие скорости на процессоре PowerPC e500mc (4 ядра, 1 ГГц).

Рис. 9: все три приложения показывались со скоростью 16 кадров в секунду. Рис. 10: приложение GlassCockpit показывалось со скоростью 16 кадров в секунду, а визуализация рельефа местности – со скоростью 9.2 кадра в секунду. Рис. 11: приложение PFD (основной дисплей полета) обновлялось со скоростью 10.3 кадра в секунду, а более простое приложение, показывающее состояние дверей, – 21.7 кадра в секунду.

7. Заключение

Разработка системы визуализации данных о полете и отображения информации о состоянии самолета обладает своей спецификой, связанной с критически важными вопросами безопасности. Эта специфика часто не позволяет применять готовые, известные решения для той или другой функциональности. При разработке программного обеспечения для бортового оборудования необходимо также учитывать то, что на борт ставятся энергосберегающие и относительно низкоэффективные процессоры. В то же время система должна обеспечивать интерактивную скорость визуализации.

Проведенные исследования показали возможность повышения скорости визуализации программной реализации библиотеки OpenGL SC на типичном авиационном

компьютере на базе относительно слабого процессора PowerPC (1 ГГц) за счет использовании нескольких ядер процессора. С помощью AMP технологии, разрешенной для авиационных систем, было достигнуто ускорение как для работы одиночного приложения, так и для программного компоновщика, который обеспечивает компоновку в один кадровый буфер изображений, сгенерированных несколькими отдельными приложениями. При визуализации многооконного экрана достигнута интерактивная скорость ~15-20 кадров в секунду.

Список литературы / References

[1]. Федосов Е.А. Проект создания нового поколения интегрированной модульной авионики с открытой архитектурой. Полет, №8, 2008 г., стр. 15-22 / Fedosov E.A. Project On New-Generation Open Architecture Integrated Modular Avionics Development. Flight, №8, 2008, pp. 15-22 (in Russian).

[2]. Федосов Е.А., Косьянчук В.В., Сельвесюк Н.И. Интегрированная модульная авионика. Радиоэлектронные технологии, №1, 2015 г., стр. 66-71 / Fedosov E.A., Kosyanchuk V.V., Selvesyuk N.I. Integrated Modular Avionics. Radioelectronic Technologies, №1, 2015, pp. 66-71.

[3]. ARINC Standards Store. Available at https://www.aviation-ia.com/product-categories, accessed 15.12.2019.

[4]. Safety Critical Working Group. Available at https://www.khronos.org/openglsc, accessed 15.12.2019.

[5]. DO-178C Software Considerations in Airborne Systems and Equipment Certification. Available at https://my.rtca.org/NC__Product?id=a1B36000001IcmqEAC, accessed 15.12.2019.

[6]. Б.Х. Барладян, А.Г. Волобой, В.А. Галактионов, В.В. Князь, И.В. Ковернинский, Ю.А. Солоделов, В.А. Фролов, Л.З. Шапиро. Эффективная реализация OPENGL SC для авиационных встраиваемых систем. Программирование, том 44, № 4, 2018 г., стр. 3-10 / B.Kh. Barladian, A.G. Voloboy, V.A. Galaktionov, V.V. Knyaz', I.V. Koverninskii, Yu.A. Solodelov, V.A. Frolov, L.Z. Shapiro. Efficient Implementation of OpenGL SC for Avionics Embedded Systems. Programming and Computer Software vol. 44, № 4, 2018, pp. 207–212

[7]. Солоделов Ю.А., Горелиц Н.К. Сертифицируемая бортовая операционнаяс истема реального времени JetOS для российских проектов воздушныхсудов. Труды ИСП РАН, том 29, вып. 3, 2017 г., стр. 171-178 / . Solodelov Yu.A., Gorelits N.K. Certifiable onboard real-time operation system JetOS for Russian aircrafts design. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 3, 2017. pp. 171-178 (in Russian). DOI: 10.15514/ISPRAS-2017-29(3)-10.

[8]. Central Processing Module (CPM/ P3041-VPX 3U). Available at http://www.nkbvs.ru/en/products/elektronnie-modyli/vpx-3u/moduli-universalnogo-protsessora-dannix-mypd-p3041/, accessed 15.12.2019.

[9]. A Safety Critical Compositor for OpenGL SC. Available at http://www.coreavi.com/sites/default/files/compositor_whitepaper_final.pdf, accessed 15.12.2019.

[10]. EGL_EXT_compositor. FACE-aligned Safety Critical Compositor. Available at https://coreavi.com/wp-content/uploads/2018/08/coreavi_product_brief_-_egl_ext_compositor.pdf, accessed 15.12.2019.

[11]. B.Kh. Barladian, L.Z. Shapiro, K.M. Mallachiev, A.V. Khoroshilov, Y.A. Solodelov, A.G. Voloboy, V.A. Galaktionov, I.V. Koverninskiy. Multi-windows rendering using software OpenGL in avionics embedded systems. In Proc. of the 29th International Conference on Computer Graphics and Vision. CEUR Workshop Proceedings, vol. 2485, 2019, paper 7.

Информация об авторах / Information about authors

Борис Хаимович БАРЛАДЯН, старший научный сотрудник, кандидат технических наук, доцент. Научные интересы: компьютерная графика, компьютерное моделирование.

Boris Haimovich BARLADYAN, Senior Researcher, Candidate of Technical Sciences, Associate Professor. Research interests: computer graphics, computer modeling.

Лев Залманович ШАПИРО, старший научный сотрудник, кандидат технических наук, доцент. Научные интересы: компьютерная графика, компьютерное моделирование, вычислительная оптика.

Lev Zalmanovich SHAPIRO, Senior Researcher, Candidate of Technical Sciences, Associate Professor. Research interests: computer graphics, computer modeling, computational optics.

Курбанмагомед Абдурагимович МАЛЛАЧИЕВ, младший научный сотрудник. В 2018 окончил аспирантуру ВМК МГУ им. М.В. Ломоносова. Научные интересы: верификация программного обеспечения, операционные системы реального времени.

Kurbanmagomed Abdurahimovich MALLACHIEV, Junior Researcher. In 2018 completed his postgraduate education at the CMC faculty of Lomonosov Moscow State University. Research interests: software verification, real-time operating systems.

Алексей Владимирович ХОРОШИЛОВ, ведущий научный сотрудник, кандидат физико-математических наук, директор Центра верификации ОС Linux в ИСП РАН, доцент кафедр системного программирования МГУ и ВШЭ. Основные научные интересы: методы проектирования и разработки ответственных систем, формальные методы программной инженерии, методы верификации и валидации, тестирование на основе моделей, методы анализа требований, операционная система Linux.

Alexey Vladimirovich KHOROSHILOV, Leading Researcher, Ph.D. in Physics and Mathematics, Director of the Linux OS Verification Center at ISP RAS, Associate Professor of System Programming Departments at Moscow State University and the Higher School of Economics. Main research interests: design and development methods for critical systems, formal methods of software engineering, verification and validation methods, model-based testing, requirements analysis methods, Linux operating system.

Юрий Алексеевич СОЛОДЕЛОВ, начальник сектора, окончил Московский авиационный институт в 2009 году. Научные интересы: операционные системы реального времени, разработка сертифицируемого бортового ПО.

Yuri Alekseevich SOLODELOV, Head of Sector, graduated from Moscow Aviation Institute in 2009. Research interests: real-time operating systems, development of certified on-board software.

Алексей Геннадьевич ВОЛОБОЙ, ведущий научный сотрудник, доктор физико-математических наук, доцент. Научные интересы: компьютерная графика, оптика, оптическое моделирование.

Alexey Gennadievich VOLOBOY, Leading Researcher, Doctor of Physical and Mathematical Sciences, Associate Professor. Research interests: computer graphics, optics, optical modeling.

Владимир Александрович ГАЛАКТИОНОВ, главный научный сотрудник, доктор физико-математических наук, профессор. Научные интересы: компьютерная графика, вычислительная оптика, компьютерная лингвистика, научная визуализация.

Vladimir Alexandrovich GALAKTIONOV, Chief Researcher, Doctor of Physical and Mathematical Sciences, Professor. Research interests: computer graphics, computational optics, computer linguistics, scientific visualization.

Игорь Викторович КОВЕРНИНСКИЙ, заместитель начальника отделения, окончил МФТИ в 1972 г. Научные интересы: интегрированная модульная авионика.

Igor Viktorovich KOVERNINSKY, deputy head of department, graduated from Moscow Institute of Physics and Technology in 1972. Research interests: integrated modular avionics.

DOI: 10.15514/ISPRAS-2020-32(1)-4

Технологии автоматического тестирования программных комплексов реалистичной компьютерной графики

Е.Ю. Денисов, ORCID: 0000-0002-0614-9100 <eed@spp.keldysh.ru>
А.Г. Волобой, ORCID: 0000-0003-1252-8294 <voloboy@gin.keldysh.ru>
Е.Д. Бирюков, ORCID: 0000-0003-4297-6813 <birukov@gin.keldysh.ru>
М.С. Копылов, ORCID: 0000-0002-9526-0766 <pmk@gin.keldysh.ru>
И.А. Калугина, ORCID: 0000-0001-5558-3045 <qik@gin.keldysh.ru>

Институт прикладной математики им. М.В.Келдыша РАН,
125047, Россия, Москва, Миусская пл., д.4

Аннотация. В статье описаны технологии автоматического тестирования программного обеспечения применительно к промышленным системам компьютерной графики и оптического моделирования. Автоматизация тестирования становится жизненно необходимой в условиях ограниченности ресурсов при частом выпуске версий, которые нередко возникают у производителей программного продукта. Представлены как методы регрессионного тестирования вычислительного ядра таких комплексов, так и способы тестирования пользовательского интерфейса. Для регрессионного тестирования используется механизм сценариев на языке Python. Рассмотрены методы его распараллеливания, которые позволяют значительно сократить время тестирования. Поскольку в оптическом моделировании широко применяются стохастические методы, результаты расчетов могут отличаться, что осложняет регрессионное тестирование. В этом случае предлагается применять некоторый порог при сравнении результатов. Автоматизированные тесты для тестирования пользовательского интерфейса разработаны на основе инструмента AutoIt. Отдельно описаны подходы к тестированию пользовательского интерфейса систем, реализованных в виде дополнений (plug-in) к существующим комплексам автоматизации проектирования, исходный код которых закрыт и недоступен для авторов автоматических тестов.

Ключевые слова: тестирование ПО; автоматическое тестирование; компьютерная графика; моделирование освещенности; надежность программного продукта.

Для цитирования: Денисов Е.Ю., Волобой А.Г., Бирюков Е.Д., Копылов М.С., Калугина И.А. Технология автоматического тестирования программного комплекса реалистичной компьютерной графики. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 71–88. DOI: 10.15514/ISPRAS–2020–32(1)–4

Technologies for automatic testing of a software package for realistic computer graphics

E.Y. Denisov, ORCID: 0000-0002-0614-9100 <eed@spp.keldysh.ru>
A.G. Voloboy, ORCID: 0000-0003-1252-8294 <voloboy@gin.keldysh.ru>
E.D. Birukov, ORCID: 0000-0003-4297-6813 <birukov@gin.keldysh.ru>
M.S. Kopylov, ORCID: 0000-0002-9526-0766 <pmk@gin.keldysh.ru>
I.A. Kalugina, ORCID: 0000-0001-5558-3045 <qik@gin.keldysh.ru>

Keldysh Institute of Applied Mathematics of RAS,
125047, Russia, Moscow, Miusskaya sq., 4

Abstract. The article describes the technology of automatic software testing in relation to industrial systems of computer graphics and optical simulation. Test automation becomes vital in the face of limited resources with the frequent release of product versions, which often occur among software product manufacturers. There are presented both methods of regression testing the computational kernel of such systems, and methods of testing the user interface. Scripting mechanism based on Python is used for regression testing, its multithreading capabilities which allow significant decreasing of testing time are also described. Python allows two ways of parallelization – multithreading and multiprocessing, both of them are considered. Due to the stochastic methods used in optical simulation calculation results may differ from time to time, which complicates regression testing. In this case, it is proposed to apply some (in each case - your own) threshold when comparing the simulation results. Separately automated testing of user interface which was elaborated basing on the AutoIt tool is described. The approach for testing the user interface of systems implemented in the form of plugins to existing CAD/PDM complexes, the source code of which is closed and not available to the authors of automatic tests, are described as well.

Keywords: software testing; automatic testing; computer graphics; lighting simulation; software product robustness.

For citation: Denisov E.Y., Voloboy A.G., Birukov E.D., Kopylov M.S., Kalugina I.A. Technology for automatic testing of a software package for realistic computer graphics. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 71–88. DOI: 10.15514/ISPRAS–2020–32(1)–4

1. Введение

Создание современных программных комплексов невозможно без автоматизации процесса их разработки и сопровождения [1]. Одним из важнейших этапов жизненного цикла программного продукта является тестирование. Полноценное тестирование требует проверки различных аспектов программы, таких как наличие требуемой функциональности, быстродействие имеющихся алгоритмов, их корректную работу при различных исходных данных, а также пользовательский интерфейс и его корректную реакцию на различные действия пользователя, в том числе ошибочные. Сложность современных программных комплексов реалистичной графики и оптического моделирования такова, что классическое тестирование [2] становится просто неэффективным. Оценки ресурсов и времени, необходимых для полного тестирования таких комплексов, очень высоки. Особенностями тестирования таких комплексов можно назвать использование общих компонент разнородными программными компонентами и комплексами. При необходимости выпускать частые новые версии ПО в жестких ограничениях на сроки их выпуска автоматизация процесса тестирования становится неизбежной

В последнее время широко распространяются системы автоматического тестирования программного обеспечения. Такие системы позволяют существенно увеличить объем тестирования при одновременном снижении усилий [3]. Особенно эффективно использование автоматического тестирования при большом количестве однотипных тестов, например, при тестировании работы одного алгоритма с разными наборами

исходных данных или тестировании поведения нескольких однотипных диалоговых окон [4]. В основном используется два типа тестирования: тестирование на основе аналитических расчётов, обеспечивающее сравнение результатов работы системы с ожидаемыми, и так называемое регрессионное тестирование, обеспечивающее совпадение поведения новой реализации системы и её предыдущей реализации.

Наш опыт показывает, что иногда ошибки, однажды уже исправленные в программе, могут опять возникнуть в будущих версиях программного продукта. Такие ситуации нечасты, но и «единичным случаем» их назвать нельзя. Надо заметить, что причиной повторного появления уже исправленных ошибок могут быть как ошибки интеграции исходных кодов, когда необходимая ревизия кода была просто утеряна или пропущена, так и совершенно новые независимые ошибки, которые приводят к такому же внешнему проявлению, что и ранее исправленная ошибка. Но для конечного пользователя программы, которому неизвестны истинные причины повторного появления ошибки, эта причина не важна. Факт повторного проявления уже однажды найденной и исправленной ошибки неизменно отрицательно влияет на репутацию программного продукта и на репутацию коллектива его разработчиков.

Вопрос создания мер по предотвращению подобных ситуаций выходит за рамки предлагаемой статьи. Однако здесь предлагается подход, позволяющий выявить такие повторные появления уже однажды исправленных ошибок на этапе тестирования. Подход этот состоит в проверке каждой версии программного продукта на отсутствие всех ранее исправленных ошибок. Такая проверка позволяет избежать повторного появления ошибочного поведения программы вне зависимости от его причины. Осуществление данного подхода при тестировании коллективом разработчиков требует много времени при выпуске каждой версии. Однако автоматизация этого процесса, как будет показано в статье ниже, делает его вполне осуществимым, и это позволяет значительно повысить надежность программного продукта.

2. Особенности автоматизации тестирования в системах реалистичной графики

Программные ошибки, выявляемые в системах моделирования освещенности и реалистичной компьютерной графики, можно подразделить на четыре основных класса.

Первый класс связан с ошибкой моделирования распространения света в виртуальной сцене. Ошибки проявляются в неверных вычислениях и результатах расчётов, которые можно представить в численном виде. Соответственно, тестирование данного класса ошибок строится на анализе или сравнении численных результатов расчёта. Так как моделирование освещенности часто требует значительного времени вычислений, то для этого класса ошибок важным аспектом является создание эффективных тестовых сцен.

Второй класс связан с ошибкой построения реалистичного изображения или ошибочной визуализацией результатов моделирования. Ошибки проявляются в некорректной картинке или неверном отображении результатов в графическом виде. Тестирование этого класса ошибок требует анализа и сравнения изображений или графических результатов расчёта.

Третий класс ошибок касается эффективности выполнения программы. Он включает в себя как случаи потери скорости расчётов (программа затрачивает на расчёт значительно большее время, чем раньше), так и потери при использовании памяти (программа необоснованно затрачивает при выполнении большее количество оперативной памяти, чем раньше или чем должно быть по оценке). Сюда же можно отнести ошибки «утечки» памяти, когда выполнение алгоритмов с выделением и последующим освобождением памяти приводит к неожидаемому уменьшению свободной памяти. Для автоматической

проверки отсутствия таких ошибок нам необходимо анализировать скорость работы программы и объем используемой памяти.

Наконец четвертый, последний класс ошибок включает в себя «зависания» программы в процессе работы или ее аварийное завершение, вызванные определённым набором входных данных или определёнными действиями пользователя. Проверка отсутствия таких ошибок возможна на основе слежения за состоянием расчётных процессов для определения их «зависания» или аварийного завершения.

Программные продукты, разрабатываемые нашим коллективом, предоставляют необходимую функциональность в нескольких формах. С точки зрения автоматизации тестирования важными являются:

- пакетная обработка данных (или выполнение командной строки);
- API, предоставляемый как пакет для языка программирования Python;
- Интерактивная программа с графическим интерфейсом пользователя;
- Дополнение (plug-in) к системе автоматизированного проектирования CATIA [5].

Все эти программы и приложения созданы на базе общего вычислительного ядра системы с различными модулями интерфейса. Программные ошибки могут проявляться как в одном из приложений, так и во всех формах сразу. Поэтому система тестирования должна поддерживать все указанные функциональности. Если ошибка воспроизводится только в каком-либо одном приложении, то тестирование вынужденно применяется именно для этой формы. Однако для случаев, когда ошибка воспроизводится в нескольких формах, ее тестирование целесообразно выполнять в форме, в которой тест может быть создан и выполнен наиболее эффективно. Например, ошибку в алгоритме трассировки лучей, допущенную внутри общего вычислительного ядра, можно будет выявить независимо от используемого интерфейса. Поэтому тест на отсутствие этой ошибки лучше написать с использованием самой простой формы, позволяющей запустить его на выполнение в фоновом режиме, – пакетной обработки данных.

Каждый автоматический тест выдает решение о том, пройдён ли тест успешно. Если тест не пройден, то возможна выдача дополнительной информации, призванной помочь в нахождении ошибки при анализе результатов тестирования. По окончании работы тестирующей системы выдаётся таблица со всеми результатами, что позволяет сделать вывод либо об успешном прохождении данного этапа автоматического тестирования, либо о необходимости исправления найденных ошибок. Например, для облегчения анализа результатов в случае неудачного теста на ошибку визуализации автоматически создаются два изображения: ожидаемое и полученное, с количественным и качественным описанием разницы между ними.

3. Тесты в фоновом режиме

3.1. Выполнение командной строки

Так как создание теста для командной строки является обычно простой задачей с технической точки зрения, то этот тип тестов является наиболее предпочтительным, если ошибка воспроизводится с его помощью, даже если изначально она была обнаружена при выполнении Python скрипта или в интерфейсной программе. Дополнительным преимуществом такой формы тестов является возможность выполнять их в фоновом режиме. Разработка теста состоит, как правило, в подготовке исходных данных, написании скрипта для выполнения необходимого расчёта и анализа результатов. Численные результаты теста проверяются через сравнение их с результатами, полученными либо аналитически, либо ранее вычисленными предыдущей, заведомо верной версией программы. Графические результаты

проверяются через сравнение полученных изображений с изображениями, полученными предыдущей корректной версией программы.

В наших комплексах в основе моделирования освещенности лежат стохастические методы трассировки лучей или фотонов света, поэтому полученные изображения могут различаться часто незаметно для глаза. Поэтому при сравнении изображений проверяется не полное их совпадение, а допускается отклонение с определённым уровнем точности, величина которого зависит от тестируемой подсистемы. Сравнением изображений занимается специальная программа, выдающая в результате разницу между пикселями сравниваемых изображений. Разница выдаётся в числовом формате (абсолютная и относительная разность между пикселями) и в графическом, в виде изображения, где большая яркость пикселя соответствует большей разности между сравниваемыми изображениями в этой точке.

Пакетный режим также можно использовать для контроля скорости работы программы путём вычисления времени, затраченного модулем на расчёт, и для проверки отсутствия «зависания» и аварийного завершения программы при определённых исходных данных путём контроля наличия и своевременного завершения соответствующих процессов. Однако для этого класса ошибок нельзя использовать выполнение тестов в фоновом режиме.

3.2. Использование Python API

Использование для тестирования скриптов на языке Python с вызовами функций API нашей системы моделирования является методом, следующим по предпочтительности после тестирования в пакетном режиме. Он применяется в тех случаях, когда командной строке не хватает функциональности, или необходимо задавать определенные настройки режима моделирования. Этот подход, за счёт расширенных возможностей программного API [6] и самого языка программирования Python, позволяет автоматизировать действия пользователя по работе с различными объектами комплекса, такими как объекты сцены, модули рендеринга, различные симуляторы и т.д. К настоящему времени общее число подобных объектов весьма велико, например, одних только типов объектов сцены существует более сотни. Также необходимо отметить, что многие из этих объектов имеют достаточно большое количество различных настроек, влияющих на режимы их работы и вычислений.

Использование программного API комплекса позволяет проверить все те же типы ошибок, что и модуль командной строки, а также дополнительно проверить количество памяти, занимаемое программой в процессе работы, отсутствие «зависания» и аварийного завершения программы в результате определённой последовательности команд. Для проверки отсутствия ошибок с «утечкой» памяти проводится определённое число одинаковых операций, захватывающих и потом освобождающих память, и сравнивается объём памяти, занимаемый программой после каждой операции. Если выявлено, что количество занятой памяти после каждой операции постоянно увеличивается, то это однозначно указывает на ошибку.

Сценарии можно запускать как в режиме командной строки, так и в режиме графического интерфейса. Как правило, при автоматическом тестировании используется режим командной строки. Запуск сценариев осуществляется с помощью консольной программы ipython.exe, которая представляет собой специальный модуль комплекса, содержащий в себе интерпретатор языка Python с возможностью доступа ко всем остальным объектам комплекса. Пример простейшего скрипта на языке Python приведён на листинге 1.

```
def Render(file):
  kernel = Kernel()
```

```
    kernel.LoadScene(file)
    rp = RenderParams()
    rp.res = (800, 600)
    rp.depth = 5
    res = kernel.Render(rp)
    res.SaveImageToFile(…)
  res.CompareImageWithEthalon(…)

files = [“file1”, “file2”, …, “file99”]
for f in files:
  Render(f)
```

Листинг 1. Пример простейшего скрипта на языке Python
Listing 1. An example of a simple Python script

В данном примере в ядро комплекса последовательно загружаются файлы, содержащие в себе различные модели (сцены). Затем производится рендеринг каждой модели и результат (как правило, это графическое изображение) сохраняется в новый файл. На заключительном шаге производится сравнение получившегося изображения с эталонным изображением с помощью специальной подпрограммы.

Работа со сценариями состоит из нескольких обязательных этапов - подготовка исходных данных, выполнение расчётов, сохранение и анализ результатов. В качестве необязательного этапа можно добавить установку или настройку различных параметров модулей, которые задействованы в вычислениях. Сценарии не имеют каких-либо ограничений по длине (количеству операторов), а также могут вызвать другие сценарии, быть вложенными, использовать рекурсию и т.д. При этом из любого сценария можно получить доступ как к одному, так и к нескольким модулям комплекса оптического моделирования одновременно.

Подобная широкая функциональность, несомненно, является весьма удобной для автоматизации вычислений. В рамках тестирования же, как правило, ограничиваются работой только с одним конкретным модулем в каждый момент времени, то есть для каждого модуля или компонента модуля пишется отдельный тестовый сценарий, который используется для тестирования работоспособности этого компонента или модуля. Укрупнение подобных сценариев тоже возможно, например, объединив их в пакет подпрограмм в рамках одного сценария.

Особенностью реализации Python API, разрабатываемой нами для написания тестовых сценариев, является строгая проверка на ошибки, возникающие по ходу выполнения сценариев. Допустим, если по каким-либо причинам исходный файл не сможет загрузиться, например, из-за ошибки в тестируемом модуле, то Python API немедленно вернёт код ошибки и выполнение сценария будет остановлено, что в свою очередь будет явно сигнализировать о возможных проблемах в данном модуле. Важно отметить, что даже успешное завершение тестового сценария не гарантирует правильности работы тестируемого модуля. Поэтому любой тестовый сценарий должен дополнительно выполнять анализ получившихся результатов. Обычно такой анализ представляет собой сравнение различных численных и графических данных, полученных при оптических симуляциях в загруженной сцене, например, результатов рендеринга.

Дополнительно требуется анализ скорости работы тестового сценария и количества используемой памяти в процессе работы. Выбор того или иного типа анализа зависит непосредственно от формата исходных данных, а также от теста (функциональный или регрессионный). Учитывая богатые возможности языка сценариев Python в плане обработки данных и в наличии множества дополнительных пакетов расширения самого разного назначения, таких как numpy, scipy, matplotlib, imageio, задачи анализа получившихся результатов могут быть реализованы непосредственно в сценарии

тестирования, обычно в виде отдельной подпрограммы. Например, если результатами некоторых симуляций являются графические изображения, то их можно сравнивать с помощью библиотеки scikit-image (часть пакета scipy) для поиска различающихся регионов в изображениях [7]. В некоторых случаях (например, при проведении трассировки лучей методом Path Tracing) в вычислениях используются случайные величины, поэтому результаты всегда будут немного отличаться от запуска к запуску. В этих случаях для успешного прохождения теста среднеквадратичное отклонение между сравниваемыми изображениями не должно превышать некоторого определённого значения.

3.3. Многопоточность в сценариях

При автоматическом тестировании компонентов комплекса оптического моделирования весьма важным критерием является эффективность данной процедуры с точки зрения затраченного времени. Поэтому в рамках работы с комплексом были выявлены задачи тестирования, выполняющиеся не оптимально с точки зрения использования доступных вычислительных ресурсов, характеризующиеся неполной загрузкой всех доступных процессоров или процессорных ядер. В основном это вызвано такими причинами как собственные ограничения алгоритмов обработки графических данных в многопроцессорных/многоядерных конфигурациях и конфигурациях с неоднородным доступом к памяти (NUMA), или же из-за использования тестовыми сценариями некоторых специфических методов ядра комплекса, которые являются однопоточными по своей природе.

К счастью, язык сценариев Python обладает достаточно продвинутой поддержкой многопоточного программирования, позволяющей значительно увеличить эффективность выполнения вышеперечисленных групп сценариев. Данным языком, например, поддерживаются оба базовых вида распределения вычислительной нагрузки: многопоточность и многопроцессность, которые реализованы в модулях threading и multiprocessing соответственно. При этом стоит отметить, что унификация интерфейсов данных модулей делает их весьма удобными в использовании. В некоторых случаях в исходном сценарии достаточно изменить лишь несколько строк кода, чтобы перейти от использования одного модуля к другому, и наоборот. Более того, оба модуля поддерживают один и тот же набор примитивов синхронизации, что упрощает разработку программ. Выбор того или иного способа реализации многопоточности в тестовом сценарии, как правило, зависит от конкретных задач, решаемых им, и выбирается разработчиком данного сценария.

Как показывает наш опыт, модуль threading имеет некоторые ограничения при работе с потоками в плане эффективности, если эти потоки в основном работают с общими данными, такими как массивы, словари или списки. Другое ограничение связано с возможностью ситуации, при которой из двух потоков одновременно будет вызвана одна и та же функция ядра комплекса. Учитывая, что многие функции ядра комплекса не являются потокобезопасными, существует реальная угроза аварийного завершения при выполнении многопоточных тестовых сценариев, или возникновения ошибок в результатах вычислений или тестирования. В связи с этим, использование модуля multiprocessing в тестовых сценариях является предпочтительным, так как при этом можно добиться большего параллелизма, особенно в задачах пакетной обработки данных. Стоит отметить, что модуль multiprocessing не имеет каких-либо ограничений [8].

Пример многопоточного тестового сценария, накладывающего фильтр на множество отдельных изображений, приведён на листинге 2.. В данном примере каждый экземпляр подпрограммы наложения фильтра на исходные изображения выполняется в отдельном

процессе благодаря использованию библиотеки multiprocessing, позволяющей вынести код любой функции сценария в отдельный процесс.

```
def ApplyFilter(name):
  pp = PostProcessor(name)
     res = pp.ApplyAverageFilter()
     res.SaveImageToFile(…)
  res.CompareImageWithEthalon(…)

if __name__ == '__main__':
  from multiprocessing import Process
  files = ["file1", "file2", … ]
     for f in files:
       p = Process(target = ApplyFilter, args = f)
       p.Start()
```

Листинг 2. Пример многопоточного тестового сценария
Listing 2. Sample multithreaded test script

4. Тестирование пользовательского интерфейса

4.1. Тестирование интерфейса автономного продукта

Все методы тестирования программного обеспечения можно разделить на две группы: метод "чёрного ящика", когда внутренние алгоритмы работы системы неизвестны тестировщику (или автору автоматического теста), и метод "белого ящика", когда такие алгоритмы известны, и их особенности учитываются при создании автоматического теста. Тесты пользовательского интерфейса в основном относятся к первой группе, так как для пользователя, который работает с интерфейсом, обычно важен только конечный результат работы программы при различных данных, которые он вводит с его помощью. Тестирование с использование графического интерфейса обычно наиболее трудозатратно как при создании теста, так при его выполнении. Поэтому его используют, когда другие тесты не позволяют выявить ошибку.

Много лет назад нашим коллективом была разработана система записи и воспроизведения интерактивных действий пользователя [9], которая позволяла полностью автоматизировать процесс тестирования пользовательского интерфейса для нашего комплекса оптического моделирования. Однако с переводом всех интерфейсных программ на платформу Qt использование этой системы без серьезной переработки стало невозможным. Поэтому мы сочли, что будет более эффективно взять один из существующих пакетов записи действий в интерактивной программе. Нами были сформулированы требования к такому пакету:

- способность распознавать элементы пользовательского интерфейса (окна, кнопки, поля ввода, диалоги, слайдеры и т.д.) и производить над ними действия (ввод символов, нажатия кнопками мыши и т.д.);
- поддержка языка сценариев, позволяющего производить математические операции над числами и операции над текстовыми строками;
- возможность модификации сценария без его полной перезаписи;
- чтение и запись текстовых файлов;
- работа с относительными координатами в окне тестируемой программы вместо работы с экранными координатами;
- отсутствие необходимости изменения кодов тестируемой программы (например, включение вспомогательных кодов для взаимодействия с внешним тестирующим пакетом);
- способность получать информацию о выполняемых процессах;

- наличие операций с таймерами.

Авторами были протестированы более семидесяти программных пакетов, позволяющих в том или ином виде записать и воспроизвести действия пользователя в интерактивной программе. В результате был выбран пакет AutoIt [10], удовлетворяющий большинству условий. Описание некоторых протестированных пакетов, включая AutoIt, приведены в [11]. Для тестирования с использованием пакета AutoIt создаются сценарии на проверку каждой функциональности тестируемой программы или известной ошибки. Сценарий выполняется, воздействуя на элементы пользовательского интерфейса этой программы, и результат выполнения сценария сравнивается с ожидаемым. Тест считается успешным, если результат моделирования или генерации изображения совпадают с корректным результатом (либо полученным теоретически, либо вычисленным более ранней и заведомо корректной версией программы), или если наступает ожидаемая реакция программы (например, нажатие кнопки интерфейса, сообщение об ошибке и т.д.) и отсутствует ошибочная реакция программы (например, аварийное завершение).

Неудачное завершение теста не всегда говорит об ошибке в программе, оно также бывает и при изменении функциональности. Нередко исправление одной ошибки отражается сразу на нескольких тестах. В каждом таком случае необходимо провести тщательный анализ результатов и по его итогам либо исправить выявленную ошибку, либо доработать тест (например, изменить корректное эталонное изображение или порядок действий пользователя).

После значительных изменений расположения элементов пользовательского интерфейса в окнах программы тест также может завершиться неудачно. В этом случае выдается диагностика «невозможно найти требуемый элемент интерфейса». Если элемент пользовательского интерфейса был удален, то тест становится бесполезным, и мы должны отказаться от его использования. Если же элемент был перемещен, то можно изменить только кусок теста с учётом изменённого расположения элемента. Такое локальное изменение занимает обычно несколько минут. Надо отметить, что в большинстве случаев такие изменения не требуются, т.к. интерактивные тесты создаются максимально гибко и обычно являются устойчивыми к перемещению и добавлению элементов интерфейса в пределах одного диалогового окна.

К сожалению, использовать пакет AutoIt не просто. В нём не полностью реализована поддержка пакета GUI SDK «Qt», что приводит к неудобствам в работе. Например, отсутствует поддержка элементов меню, нет возможности захвата изображения с области экрана для дальнейшей обработки или записи в файл, нет возможности узнать состояние некоторых элементов интерфейса (например, текущий элемент, выбранный в выпадающем списке). Симуляция интерфейсных операций, даже таких простых как, например, нажатий клавиш и движения мыши, потребовало дополнительных усилий, чтобы сделать их максимально точными и надёжными во всех поддерживаемых версиях Windows. Приведем примеры некоторых решений, разработанных под AutoIt.

4.2. Решения, разработанные под AutoIt

Модульный подход, применённый в процессе разработки программных комплексов, позволяет использовать одни и те же GUI элементы в различных местах пользовательского интерфейса. По этой причине оказалось целесообразным создание библиотеки функций на языке сценариев AutoIt для ускорения создания тестов. В таком случае модулям (диалогам) пользовательского интерфейса естественным образом соответствуют модули (функции) на языке сценариев AutoIt, такие как «открыть файл», сохранить результат расчёта» и другие. Например, на листинге 3 показано, как выглядит на этом языке сценариев функция для сохранения сцены – модели данных, состоящей из многих объектов различных типов, связанных определёнными иерархическими связями

(геометрические объекты, свойства их поверхностей, источники света, и т.д.), открытой в приложении в данный момент:

```
; Function to save current scene under given filename
Func SaveAs($file)
  ; Open menu File -> Save As
  Send("!fs{ENTER}")

  ; Wait for dialog to appear
  If WinWaitActive("Save Scene As...", "", $Timeout) = 0 Then
    Report("'Save scene As' window not found")
    Exit(1)
  EndIf

  ; Type filename in the dialog and press ENTER
  Send($file & "{ENTER}")

  ; Close dialog
  If WinWaitClose("Save Scene As...", "", $Timeout) = 0 Then
    Report("'Save scene As' window could not be closed")
    Exit(1)
  EndIf
EndFunc
```

Листинг 3. Функция для сохранения сцены на языке сценариев
Listing 3. Function to save a scene in a scripting language

Часто используемые сложные операции, такие как, например, присваивание геометрическому объекту сцены заданное оптическое свойство, также включены в библиотеку функций на языке сценариев AutoIt. В качестве примера можно привести загрузку из файла данных о дисторсии камеры. Эта функция в свою очередь использует другую функцию это библиотеки, `OpenCamProp()`,для открытия диалога свойств камеры.

```
; Function to load radial distortion data into camera
Func LoadCamDistr($file)
  ControlClick("[ACTIVE]", "", "[TEXT:BrowserWindow]")
  Sleep(500)
  ; Open “Camera properties” dialog
  OpenCamProp()

  ControlClick("Camera Properties", "", "[TEXT:rd_browse_pbtn]")
  ; Type filename
  Send($file & "{ENTER}")
  ControlClick("Camera Properties", "", "[TEXT:ok_btn]")
  Sleep(500)
EndFunc
```

Листинг 4. Функция загрузки из файла данных о дисторсии камеры
Listing 4. Function to load from file data about camera distortion

4.3. Тестирование пользовательского интерфейса в САПР CATIA

Тестирование дополнительного модуля (plugin) в системе автоматизации проектирования CATIA имеет свои отличия, связанные с особенностями пользовательского интерфейса системы. Благодаря наличию общего вычислительного ядра всех наших продуктов тестирование процесса моделирования выполняется автономно, методами описанными ранее.

Пользовательский интерфейс CATIA отличается от интерфейса других подобных систем тем, что дерево объектов сцены отображается в том же окне, что и сама сцена [12]. Стандартное окно системы CATIA с отображённым в нем деревом сцены показано на рис. 1. Ни само дерево, ни его отдельные элементы не имеют внешних идентификаторов. Поэтому в любом случае необходимы некоторые дополнительные действия для выделения в окне просмотра точки, соответствующей заданному объекту. Автоматизация тестирования пользовательского интерфейса в этом случае может быть произведена несколькими способами.

Первый способ предусматривает выделение мышью определённой точки на элементе интерфейса, это ведет к фиксации конкретных координат этой точки в сценарии теста. Второй способ заключается в использовании алгоритмов распознавания изображений. Оба этих способа имеют свои недостатки: в первом случае – необходимость использования строго определённого разрешения экрана и масштаба элементов управления, во втором случае – необходимость учитывать возможную ошибку распознавания, особенно, если на экране присутствуют несколько похожих друг на друга объектов.

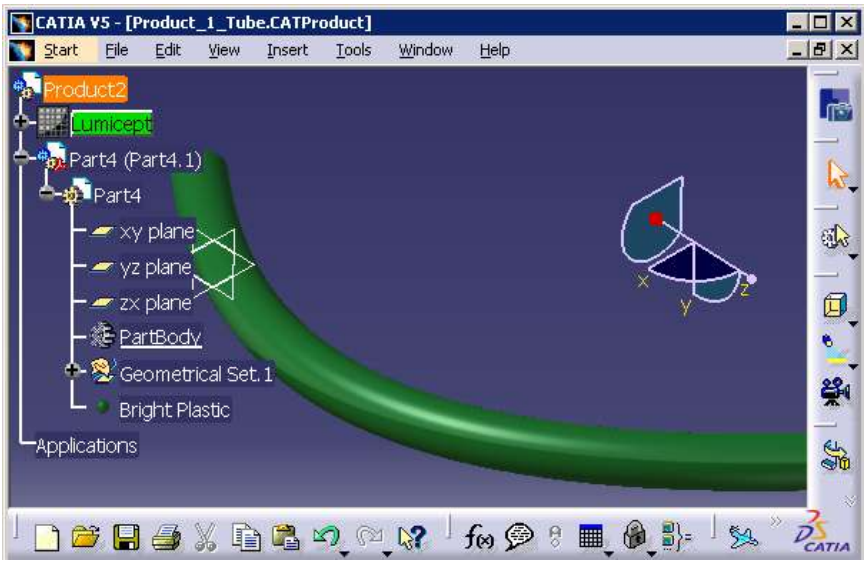


Рис. 1. Окно системы CATIA с загруженной сценой и отображённым деревом иерархии сцены
Fig. 1. CATIA system window with a loaded scene and the displayed scene hierarchy tree

Третий способ, выбранный авторами в качестве основного, представляет собой комбинацию использования сценариев, встроенных в тестируемую программу, и внешней программы автоматических тестов пользовательского интерфейса. Основу тестов составляет сценарий, выполняемый в самой тестируемой программе. Этот сценарий вызывает внешнюю программу тестирования пользовательского интерфейса (AutoIt в нашем случае) непосредственно в тех местах, где требуется тестирование отдельных элементов интерфейса.

Рассмотрим для примера тестирование диалогового окна редактирования параметров материала в системе CATIA. Данное диалоговое окно является частью самой системы, но при создании нашего plugin-а оптического моделирования к нему были добавлены дополнительные параметры. При тестировании используется сценарий на языке CATVBs (разновидность языка Visual Basic), который вызывается из самой системы, а также внешний инструмент AutoIt и его встроенный язык сценариев.

Тестирование состоит из следующих этапов.

1. Система CATIA запускается в стандартном режиме с графической оболочкой и с автоматическим запуском сценария. Для запуска используется пакетный командный файл, в котором заданы параметры – необходимая оболочка (Workbench) системы CATIA, а также путь к сценарию, который будет выполнен сразу после запуска. Следует обратить внимание на то, что в данном случае система CATIA будет открыта в стандартном графическом режиме, хотя, как правило, при автоматическом запуске сценариев используется консольный режим.

2. Выполняется первый сценарий на языке CATVBs, работающий с объектами сцены. Данный сценарий производит поиск материала с заданным именем, добавляет его в список выделенных объектов, вызывает стандартную команду Properties(), которая открывает диалог параметров выделенного объекта, после чего вызывает программу AutoIt для автоматического тестирования диалога параметров. Для вызова AutoIt используется команда SystemService.ExecuteProcessus() из набора системных процедур CATIA. Эта команда ждёт завершения работы программы AutoIt, на протяжении этого времени диалог параметров остаётся открытым. Его закрытие произойдёт уже далее, с помощью сценария AutoIt, который будет имитировать нажатие пользователем кнопки «ОК». После закрытия диалога параметров управление снова переходит к первому сценарию, который считывает значения требуемых параметров из объекта материала и проверяет, что они были правильно заданы с помощью диалога параметров.

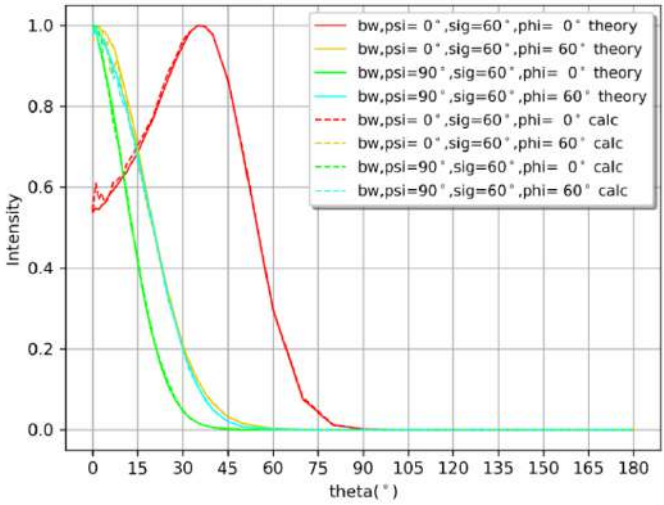
3. Выполнение сценария программы AutoIt. Этот сценарий находит диалоговое окно с заголовком "Properties" и проверяет, что оно принадлежит системе CATIA с помощью сравнения идентификаторов главного окна системы CATIA и данного диалогового окна. Затем AutoIt производит активацию этого окна, выполняет смысловые действия теста (например, записывает в определенное поле текст) и завершает работу с окном (например, инициирует нажатие кнопки «ОК»). Управление снова передаётся самой системе CATIA, в которой продолжает работу первый сценарий. Он проверяет правильность установленных данных.

Этот пример обеспечивает оптимальное сочетание принудительной активации тех элементов пользовательского интерфейса, тестирование которых требуется обязательно обеспечить, и воздействия на объекты сцены с помощью встроенных сценариев в тех случаях, когда активация элементов пользовательского интерфейса затруднена.

5. Примеры автоматизированных тестов

Ниже приведены примеры результатов тестирования с использованием различных критериев их правильности: корректность вычисленных величин, корректность визуализации, определяемая через разность эталонного и вычисленного изображений.

Тест 1. Результат теста на совпадение результатов расчёта с заранее рассчитанными аналитическими значениями выглядит так, как показано на *рис. 2*.




```
Psi     Sigma Phi      RMS
 0.0    60.0     0.0    0.22% passed
 0.0    60.0    60.0    0.34% passed
90.0    60.0     0.0    0.16% passed
90.0    60.0    60.0    0.17% passed
--------------------------------------
Status: passed
```

Рис. 2. Сравнение полученных результатов с аналитическими
Fig. 2. Comparison of the obtained results with analytical ones

Строятся графики аналитических и вычисленных значений, вычисляются среднеквадратичные отклонения (RMS), по которым и принимается решение о прохождении теста.

Тест 2. На рис. 3 приведены эталонное изображение наложения текстуры, полученное методом BRT (Backward Ray Tracing – обратная трассировка лучей) в предыдущей версии программы, и изображение, полученное тем же методом в текущей (проверяемой) версии.

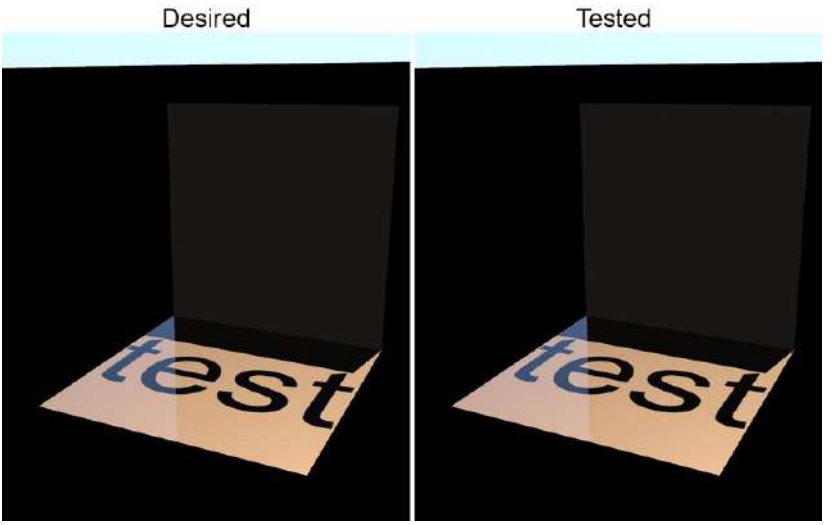


Рис. 3. BRT, наложение текстуры
Fig. 3. BRT, texture mapping

Для положительного прохождения теста разница между изображениями не должна превышать определённого значения:

```
RMS between desired and actual images = 0.0005%
-----------------------------------------------
Status: passed
```

Тест 3. В этом тесте проверяется правильность моделирования естественного освещения в методе Path Tracing (трассировка путей). На рис. 4 приведены эталонное изображение объекта при дневном свете, полученное в предыдущей версии программы, и изображение, полученное тем же методом в текущей (проверяемой) версии.

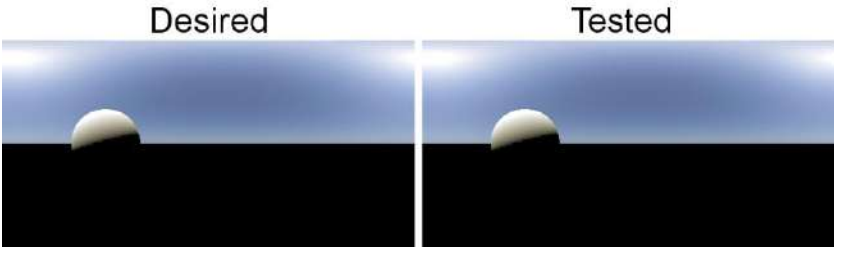


Рис. 4. Path Tracing, изображение при дневном освещении
Fig. 4. Path Tracing, daylight image

Так же, как и в предыдущем случае, разница изображений в виде среднеквадратичного отклонения используется для автоматического принятия решения о прохождении теста. Сами изображения сохраняются для возможного последующего анализа в случае отрицательного решения.

Тест 4. Следующий пример – сравнение двух изображений, полученных с использованием различных технологий трассировки лучей. На рис. 5 показано изображение, полученное методом BRT, на рис. 6 изображение получено методом PT.

Рис. 5. Метод Backward Ray Tracing
Fig. 5. Backward Ray Tracing Method

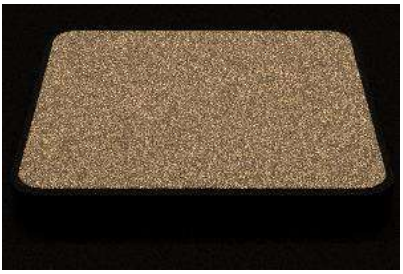


Рис. 6. Метод Path Tracing
Fig 6. Path Tracing Method

Визуально изображения различаются, это обусловлено особенностями использованных методов. Главное отличие в том, что BRT – детерминированная трассировка лучей, а PT – стохастическая, ее изображение содержит стохастический шум, неизбежный при малом времени вычислений. Однако среднее значения цвета и яркости должны быть одинаковы, вернее, не превышать определённого значения. Таким тестом производится перекрёстная проверка правильности разных методов трассировки.

```
Relative error between color of BRT and PT images = 0.2666%
-------------------------------------------------------------
Status: passed
```

Тест 5. Этот пример иллюстрирует подход, позволяющий одновременного тестировать нескольких функций программного продукта за один шаг (рис. 7). В специально созданной тестовой сцене проверяется правильность использования сложных атрибутов поверхностей, задание дневного освещения в различных методах моделирования (уже упомянутые BRT и PT, а также MCRT и BMCRT – прямая и обратная стохастическая трассировка лучей). Также здесь тестируется отображение значений яркостей в псевдоцветах. Для анализа прохождения теста, как и прежде, используется сравнение вновь полученного изображения с эталонным, полученным предыдущей версией программы, а решение принимается по среднеквадратичному отклонению.

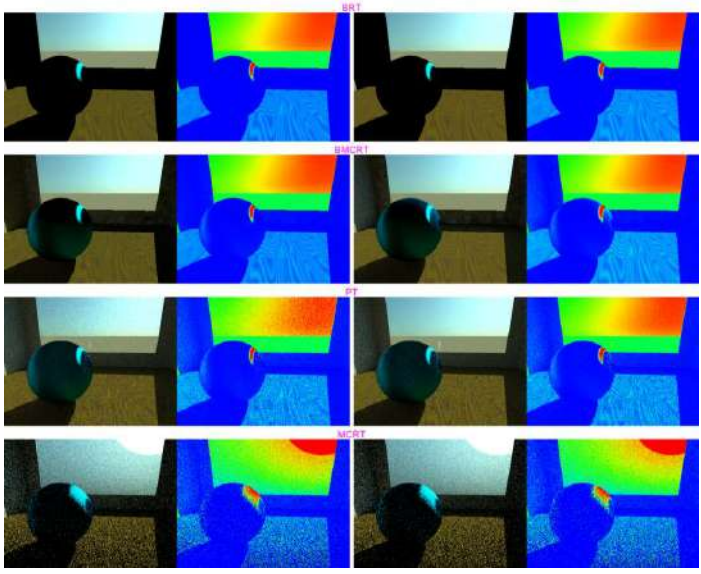


Рис. 7. Проверка корректности использования атрибутов поверхностей
Fig. 7. Verification of the use of surface attributes

Рис. 8. Эталонное изображение
Fig. 8. Reference image

Рис. 9. Изображение, полученное от тестируемой программы
Fig. 9. Image received from the tested program

Рис. 10. Разность изображений
Fig. 10. Image difference

Тест 6. Этот тест также сравнивает изображение, выдаваемое тестируемой версией программы, с эталонным изображением, полученным ранее. Тестируются самосветящиеся объекты сцены, используемые в качестве источников света. Визуально разница в изображениях незаметна: трудно сказать, насколько отличаются изображения на рис. 8 и рис. 9. Для анализа вычисляется разница между двумя изображениями и выдается в виде абсолютных значений и в виде изображения, удобном для восприятия (рис. 10).

На рис. 10 различными цветами отображается разность абсолютных значений яркости в соответствующих пикселях исходных изображений. Максимальное абсолютное значение разницы между эталонным и реальным изображениями для данной сцены составило 1.81%.

6. Заключение

Описанный подход позволяет выявить повторное появление в программных продуктах однажды уже исправленных ошибок. Важным фактором является правило, когда процесс исправления каждой обнаруженной и зарегистрированной ошибки обязательно завершается созданием очередного автоматического теста. Очевидно, что, помимо непосредственного выявления повторных ошибок, такое тестирование помогает заблаговременно выявить также и новые ошибки, которые оказывают влияние на результат моделирования или поведение программы, проверяемые имеющимися тестами. Это обусловлено тем, что в формировании таких результатов, как правило, используется работа сразу нескольких программных компонент, и наличие ошибки в любой из них может повлиять на конечный результат расчёта.

Автоматизация такого регрессионного тестирования открывает саму возможность интенсивного тестирования каждой версии выпускаемого продукта, что, безусловно, положительно влияет на его надежность. Выполнение каждого теста, даже если он тщательно разработан и записан, требует от человека прочитать и вспомнить, что и как необходимо проверить в данном тесте, а после выполнения – проанализировать результаты. В то же время автоматическая система выполняет тесты один за другим, без остановок, что позволяет значительно сократить время, затрачиваемое на тестирование программных продуктов. В случае необходимости время тестирования продукта можно уменьшить путём использования параллельно нескольких компьютеров, каждый из которых будет выполнять свой набор тестов независимо друг от друга. Таким образом, процесс интенсивного тестирования может быть включен в план выпуска версии продукта даже при условии жестких сроков его выпуска.

Описанная автоматическая система регрессионного тестирования используется при разработке и поддержке программного комплекса оптического моделирования и реалистичной компьютерной графики Lumicept [13, 14], существующего как в виде автономного продукта, так и дополнения (plugin) к широко используемой в автомобильной и авиационной промышленности САПР CATIA. Она приводит к существенной экономии времени и усилий разработчиков, позволяя повысить надежность продукта, минимизируя затраты. За несколько лет был создан набор из ~250 автоматизированных тестов. Автоматическое выполнение всего этого набора занимает около четырех часов на компьютере Xeon E3-1275 v5 3.60 GHz, в то время как оценка времени, необходимого для подобного тестирования человеком, составляет две-три недели.

Список литературы / References

[1]. Bouquet F., Grandpierre C., Legeard B., Peureux F. A test generation solution to automate software testing. In Proc. of the 3rd International Workshop on Automation of Software Test, 2008, pp.45-48.

[2]. Гленфорд Майерс, Том Баджетт, Кори Сандлер. Искусство тестирования программ, 3-е издание, М., «Диалектика», 2012 г., 272 стр. / Glenford J. Myers, Tom Badgett, Corey Sandler. The Art of Software Testing, 3rd Edition. Wiley, 2011, 256 p.

[3]. Tretmans J., Belinfante A. Automatic Testing with Formal Methods. In Proc. 7th of the European International Conference on Software Testing, Analysis and Review, 1999, pp.2012-2012

[4]. Huang Z., Carter L. Automated solutions: Improving the efficiency of software testing. Issues in Information Systems, vol. 4, 2003, pp. 171-177.

[5]. Барладян Б.Х., Волобой А.Г., Галактионов В.А., Шапиро Л.З. Интеграция реалистичной графики в системы автоматизированного проектирования и управления жизненным циклом изделия. Программирование, том 44, no. 4, 2018 г., стр. 26-35 / Barladian B.Kh., Voloboy A.G., Galaktionov V.A., Shapiro L.Z. Integration of Realistic Computer Graphics into Computer-Aided Design and Product Lifecycle Management Systems. Programming and Computer Software, vol. 44, no. 4, 2018, pp. 225–232. DOI: 10.1134/S0361768818040047

[6]. Дерябин Н.Б., Жданов Д.Д., Соколов В.Г. Внедрение языка сценариев в программные комплексы оптического моделирования. Программирование, vol. 43, no. 1, 2017 г., стр. 40-53 / Deryabin N.B., Zhdanov D.D., Sokolov V.G. Embedding the Script Language into Optical Simulation Software. Programming and Computer Software, vol. 43, no. 1, 2017, pp. 13-23. DOI: 10.1134/S0361768817010029.

[7]. Van der Walt S., Schönberger J. L., Nunez-Iglesias J., Boulogne F., Warner J. D., Yager N., Yu T. Scikit-image: image processing in Python. PeerJ, vol. 2, 2014, article e453.

[8]. Singh N., Browne L. M., Butler R. Parallel astronomical data processing with Python: Recipes for multicore machines. Astronomy and Computing, vol. 2, 2013, pp. 1-10.

[9]. Волобой А.Г., Денисов Е.Ю., Барладян Б.Х. Тестирование систем моделирования освещенности и синтеза реалистичных изображений. Программирование, vol. 40, no. 4, 2014 г., стр. 13-22 / Voloboi A. G., Denisov E. Yu., Barladyan B. Kh. Testing of Systems for Illumination Simulation and Synthesis of Realistic Images. Programming and Computer Software, vol. 40, no. 4, 2014, pp. 166–173. DOI:10.1134/S0361768814040094.

[10]. AutoIt. Available at: http://www.autoitscript.com, accessed: 20.02.2020.

[11]. Денисов Е.Ю., Волобой А.Г., Калугина И.А. Автоматизация тестирования интерактивной системы моделирования освещенности. Препринты ИПМ им. М.В. Келдыша, № 200, 2018, 19 стр. / Denisov E. Yu., Voloboi A. G., Kalugina I.A. Automatic testing of interactive lighting simulation software package. Keldysh Institute preprints, 2018, 200, 19 p. (in Russian).

[12]. Dassault Systemes, Inc. CATIA Version 5-6 R2015 Documentation. Available at: http://media.3ds.com, accessed 05.08.2018.

[13]. Жданов Д.Д., Потемин И.С., Галактионов В.А., Барладян Б.Х., Востряков К.А., Шапиро Л.З. Спектральная трассировка лучей в задачах построения фотореалистичных изображений. Программирование, vol. 37, no. 5, 2011 г., стр. 13-26. / Zhdanov D. D., Potemin I. S., Galaktionov V. A., Barladyan B. Kh., Vostryakov K. A., Shapiro L.Z. Spectral Ray Tracing in Problems of Photorealistic Imagery Construction. Programming and Computer Software, vol. 37, no. 5, 2011, pp. 236–244. DOI: 10.1134/S0361768811050069.

[14]. Жданов Д.Д., Гарбуль А.А., Потемин И.С., Волобой А.Г., Галактионов В.А., Ершов С.В., Соколов В.Г. Фотореалистичная модель объемного рассеивания в задаче двунаправленной стохастической трассировки лучей. Программирование, vol. 41. No. 5, 2015 г., стр. 66-75. / Zhdanov D.D., Garbul A.A., Potemin I.S., Voloboy A.G., Galaktionov V.A., Ershov S.V., Sokolov V.G. Photorealistic Volume Scattering Model in the Bidirectional Stochastic Ray Tracing Problem. Programming and Computer Software, vol. 41, no. 5, 2015, pp. 295–301, DOI: 10.1134/S0361768815050102.

Информация об авторах / Information about authors

Евгений Юрьевич ДЕНИСОВ – научный сотрудник. Сфера научных интересов: компьютерная графика, создание программных систем оптического моделирования, автоматическое тестирование, параллельные и распределённые вычисления.

Evgeniy Yuryevich DENISOV, researcher. Research interests: computer graphics, elaboration of optical simulation software systems, automatic testing, concurrent and distributed computing.

Алексей Геннадьевич ВОЛОБОЙ, ведущий научный сотрудник, доктор физико-математических наук, доцент. Научные интересы: компьютерная графика, оптика, оптическое моделирование.

Alexey Gennadievich VOLOBOY, Leading Researcher, Doctor of Physical and Mathematical Sciences, Associate Professor. Research interests: computer graphics, optics, optical modeling.

Елисей Дмитриевич БИРЮКОВ – младший научный сотрудник. Его научные интересы включают компьютерную графику, вычислительную оптику, трассировку лучей.

Elisey Dmitrievich BIRUKOV, junior researcher. His research interests include computer graphics, computing optics, ray tracing techniques.

Михаил Сергеевич КОПЫЛОВ – старший инженер. Его научные интересы включают компьютерную графику, вычислительную оптику, трассировку лучей.

Mikhail Sergeevich KOPYLOV, senior engineer. His research interests include computer graphics, computing optics, ray tracing techniques.

Ирина Александровна КАЛУГИНА является младшим научным сотрудником. Её научные интересы включают компьютерную графику, моделирование освещённости.

Irina Alexandrovna KALUGINA is a junior researcher. Her research interests include computer graphics, lighting simulation.

DOI: 10.15514/ISPRAS-2020-32(1)-5

BSQ-rate: новый подход к сравнению производительности видеокодеков и недостатки существующих решений

[1] *А.В. Звездакова, ORCID: 0000-0001-9253-1826 <azvezdakova@graphics.cs.msu.ru>*
[1,2] *Д.Л. Куликов, ORCID: 0000-0003-1264-2118 <dkulikov@graphics.cs.msu.ru>*
[1] *С.В. Звездаков, ORCID: 0000-0002-5320-0060 <szvezdakov@graphics.cs.msu.ru>*
[1] *Д.С. Ватолин, ORCID: 0000-0002-8893-9340 <dmitriy@graphics.cs.msu.ru>*
[1] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1
[2] *Государственный университет «Дубна»,*
141982, г. Дубна Московской обл., ул. Университетская, 19

Аннотация. В данной статье рассматриваются существующие подходы к сравнению видеокодеков, показываются недостатки некоторых популярных методов и предлагаются новые методики. С помощью анализа коллекции пользовательских видео показано, что один из популярных среди исследователей и разработчиков видеокодеков открытый набор видео не является репрезентативным с точки зрения соответствия реальным пользовательским видео. Также предложен метод создания репрезентативных наборов видео, покрывающих все сегменты пространственной и временной сложности пользовательских видеопоследовательностей. В разделе статьи, посвященном алгоритмам оценки качества видео, используемым в сравнениях видеокодеков, показаны недостатки популярных методов VMAF и NIQE. Также в статье показаны недостатки общепринятого метода финального ранжирования видеокодеков при проведении сравнений (BD-rate), и с учетом выявленных недостатков предложен алгоритм ранжирования, названный BSQ-rate. Данные результаты были получены в процессе исследований, проводимых в рамках ежегодных сравнений, организуемых видеогруппой лаборатории компьютерной графики и мультимедиа МГУ.

Ключевые слова: сравнение видеокодеков; кодирование видео; качество видео; нереференсная оценка качества; измерение качества видео; методология сравнения видеокодеков; bsq-rate

Для цитирования: Звездакова А.В., Куликов Д.Л., Звездаков С.В., Ватолин Д.С. BSQ-rate: новый подход к сравнению производительности видеокодеков и недостатки существующих решений. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 89-108. DOI: 10.15514/ISPRAS–2020–32(1)–5

Благодарности. Исследования выполнены при финансовой поддержке РФФИ в рамках научного проекта 19-01-00785а. Особая благодарность выражается Георгию Осипову и Денису Кондранину, которые помогли проанализировать обнаруженные проблемы и улучшили реализацию NIQE в программном комплексе MSU VQMT, Михаилу Ерофееву, который участвовал в улучшении методологии сравнения видеокодеков и провел субъективные сравнения, и команде Лаборатории компьютерной графики и мультимедиа МГУ за ценные советы и поддержку наших проектов.

BSQ-rate: a new approach for video-codec performance comparison and drawbacks of current solutions

[1] *A.V. Zvezdakova, ORCID: 0000-0001-9253-1826 <azvezdakova@graphics.cs.msu.ru>*
[1,2] *D.L. Kulikov, ORCID: 0000-0003-1264-2118 <dkulikov@graphics.cs.msu.ru>*
[1] *S.V. Zvezdakov, ORCID: 0000-0002-5320-0060 <szvezdakov@graphics.cs.msu.ru>*
[1] *D.S. Vatolin, ORCID: 0000-0002-8893-9340 <dmitriy@graphics.cs.msu.ru>*

Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[2] *Dubna State University,*
19, Universitetskaya, Dubna, Moscow region, 141982, Russia

Abstract. This paper is dedicated to the analysis of the existing approaches to video codecs comparisons. It includes the revealed drawbacks of popular comparison methods and proposes new techniques. The performed analysis of user-generated videos collection showed that two of the most popular open video collections from media.xiph.org which are widely used for video-codecs analysis and development do not cover real-life videos complexity distribution. A method for creating representative video sets covering all segments of user videos the spatial and temporal complexity is also proposed. One of the sections discusses video quality estimation algorithms used for video codec comparisons and shows the disadvantages of popular methods VMAF and NIQE. Also, the paper describes the drawbacks of the BD-rate – generally used method for video codecs final ranking during comparisons. A new ranking method called BSQ-rate which considers the identified issues is proposed. The results of this investigation were obtained during the series of research conducted as part of the annual video-codecs comparisons organized by video group of computer graphics and multimedia laboratory at Moscow State University.

Keywords: video quality; no-reference metric; quality measuring; video-codec comparison; comparison methodology; bsq-rate

For citation: Zvezdakova A.V., Kulikov D.L., Zvezdakov S.V., Vatolin D.S. BSQ-rate: a new approach for video-codec performance comparison and drawbacks of current solutions. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 89-108 (in Russian). DOI: 10.15514/ISPRAS–2020–32(1)–5

Acknowledgments. The studies were carried out with the financial support of RFBR in the framework of the scientific project 19-01-00785a. Special thanks go to George Osipov and Denis Kondranin, who helped analyze the problems found and improved the implementation of NIQE in the MSU VQMT software package, to Mikhail Erofeev, who participated in improving the video codec comparison methodology and made subjective comparisons, and the team of the Moscow State University of Computer Graphics and Multimedia Laboratory for valuable advice and support for our projects.

1. Введение

На сегодняшний день видео занимает более 70% всемирного интернет-трафика. По прогнозам Cisco [1], к 2022 году доля видеотрафика вырастет до 82%. Снижение затрат на передачу видео становится одним из важнейших направлений исследований и разработок во многих университетах и компаниях. В связи с ростом стоимости лицензирования решений для кодирования видео, многие крупные компании проводят разработку собственных видеокодеков и объединяются для сотрудничества, чтобы внести вклад в разработку новых стандартов кодирования. Так как реализации существующих стандартов кодирования видео непрерывно совершенствуются, а также разрабатываются новые стандарты как в открытых, так и в коммерческих решениях, технологии в этой области развиваются очень быстро. Для того, чтобы помочь пользователям выбрать кодек для различных задач (например, облачное кодирование, проведение онлайн-трансляций, видеонаблюдение или любое другое практическое применение), независимые лаборатории и специалисты проводят аналитические сравнения различных решений для кодирования видео. Организация таких сравнений

требует решения нескольких фундаментальных задач, которые влияют на репрезентативность получаемых результатов.

В данной статье рассматриваются общие подходы для сравнения видеокодеков, приведены выводы о преимуществах и недостатках различных методов, а также дано описание предлагаемых новых методов выбора видео, применения алгоритмов оценки качества видео и вычисления общих результатов сравнения. Данная статья организована следующим образом: в разделе 2 описаны общие этапы сравнения видеокодеков; в разделе 3 подробно представлены особенности построения наборов видео для применения в сравнениях видеокодеков, а также описаны недостатки открытых коллекций видео и предлагаемый подход к выбору видео; в разделе 4 рассмотрены особенности применения различных видов алгоритмов измерения качества видео в задаче сравнения видеокодеков; в разделе 5 описаны недостатки общепринятого подхода для вычисления итогового рейтинга видеокодеков и предлагается новый метод для решения данной задачи; раздел 6 содержит общие выводы.

2. Этапы сравнения видеокодеков

Существует множество подходов для проведения сравнений видеокодеков, которые отличаются используемыми алгоритмами измерения качества, форматами видео и кодеков, способом финального ранжирования участников сравнения. Однако все они имеют следующие общие этапы: выбор видео, выбор значений параметров кодирования, непосредственно кодирование тестовых видео, измерение качества закодированных видео, подведение итогов и ранжирование участников. Подготовка к проведению аналитического сравнения видеокодеков включает в себя разработку методологии, резервирование вычислительных ресурсов, выбор видео, которые будут использоваться для тестирования кодеков, поиск и привлечение участников, ведение переговоров с участниками. Получение исполняемого файла видеокодека от участников для тестирования не означает возможность проведения его тестирования, так как современные видеокодеки имеют много параметров, влияющих на производительность кодирования, и требуют специальной настройки. Базовый способ установки параметров – использование стандартных конфигураций, но не все видеокодеки имеют такую опцию. Даже если видеокодек имеет предустановленные наборы параметров, эти наборы могут быть слишком разными и не сопоставимыми в рамках проводимого анализа.

Более адекватный способ установки параметров заключается в их запросе у разработчиков видеокодека, при этом обычно удается получить параметры, удовлетворяющие заданным ограничениям сравнения. Наконец, если нет возможности получить параметры кодирования от разработчиков, можно найти оптимальные конфигурации с использованием различных алгоритмов оптимизации. Для устранения различий в производительности используемого для измерений времени работы видеокодеков оборудования, запуски проводятся на одном или нескольких одинаковых серверах. Для устранения влияния других внешних факторов, таких как перегрев оборудования или работы системных процессов, необходимо выполнить несколько запусков и выбрать среднее или лучшее время кодирования каждого видеофайла. В масштабных сравнениях запуск видеокодеков занимает значительное время, так как иногда кодеки дают сбои во время кодирования, и разработчикам требуется время для внесения необходимых исправлений.

Видеогруппа лаборатории компьютерной графики и мультимедиа МГУ на протяжении 16 лет проводит международные сравнения видеокодеков. За эти годы наша команда выпустила 44 отчета с более чем 230 видеокодеками. В следующих разделах описаны результаты проведенных исследований о некоторых недостатках широко распространенных подходов и описываются новые предлагаемые подходы.

3. Выбор видео

Существует ряд общедоступных коллекций видео, которые широко используются для разработки и сравнения алгоритмов обработки видео. Например, наборы видео на ресурсе media.xiph.org [2] часто используются для сравнения и разработки видеокодеков. Другие наборы видео включают в себя тестовые последовательности Video Quality Experts Group [3], MPEG-2 Transport Stream Test Patterns and Tools [4], Sveriges Television: The SVT High Definition Multi-Format Test Set [5], Columbia Consumer Video (CCV) Database [6], CDVL The Consumer Digital Video Library [7], база данных субъективного качества видео объектов LIVE [8], примеры из KODI Wiki [9], тестовые последовательности Ultra Video Group [10]. В данном разделе показывается, что одна из самых популярных открытых коллекций, используемых для оценки качества кодирования видео, значительно отличается от реальных видео, производимых пользователями, и предлагается метод создания репрезентативных наборов видео.

3.1 Недостатки открытых наборов видео

Многие из популярных и доступных наборов видео были сформированы таким образом, чтобы быть репрезентативными с точки зрения сложности видео. Однако реальные пользовательские видео имеют неравномерное распределение пространственной и временной сложности. Для оценки данного распределения был собран набор реальных видеопоследовательностей, созданных обычными пользователями и размещенных на видеохостинге Vimeo. Набор состоит из 18418 коротких видеороликов длиною примерно 30-60 секунд каждый, все видеоролики имеют высокий битрейт и разрешение FullHD/4K. Пространственная и временная сложности были вычислены для каждого видео с использованием алгоритмов, описанных в [11]. Пространственная сложность представляет собой нормализованный средний размер I-кадра в байтах, а временная сложность представляет собой средний размер P-кадра, деленный на средний размер I-кадра.

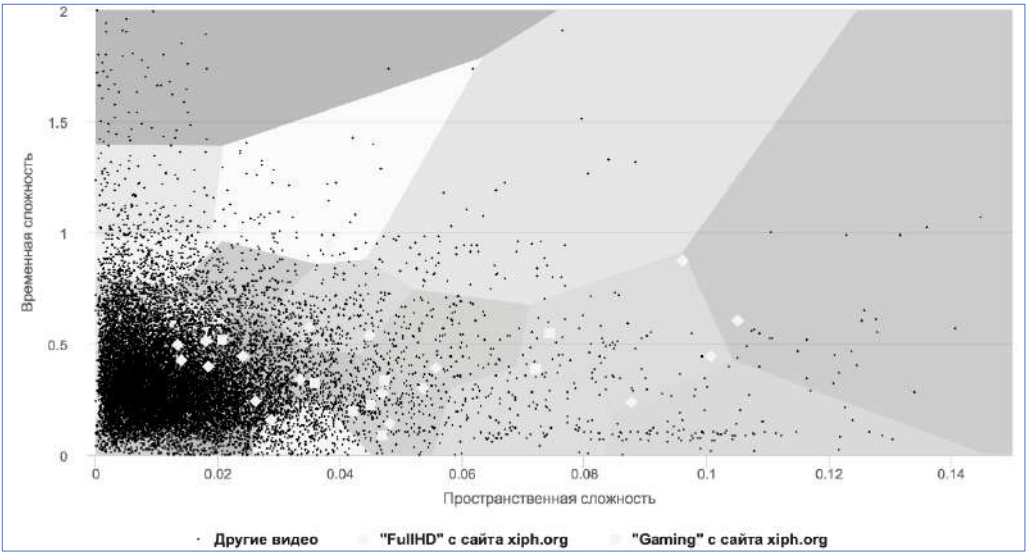


Рис. 1. Сравнение 18 418 пользовательских видео с популярными открытыми коллекциями видео ресурса media.xiph.org

Fig. 1. Comparison of 18,418 user videos with popular open collections of the video resource media.xiph.org

Полученное распределение было сопоставлено с двумя наборами видео «Gaming» и «FullHD», расположенными на ресурсе media.xiph.org [2] – одними из самых

популярных видео коллекций, используемых для измерения качества кодирования видео. Пространство из всех 18418 видео было разбито в соответствии с размером коллекций на 30 кластеров. Результат кластеризации представлен на рис. 1. На нем показано, что большинство видеопоследовательностей из коллекции xiph.org имеют высокую пространственную и временную сложность и в целом высокую репрезентативность среди сложных видео, но не охватывают область наиболее часто встречаемых видео (ближе к началу координат). Эта область представляет видео, с которыми обычно сталкиваются видеокодеки в повседневных задачах кодирования.

Согласно результатам сопоставления, только 14 из 30 полученных сегментов включали в себя видео с сайта media.xiph.org, а другие сегменты (в том числе наиболее популярная область ближе к нулю) не охватывались виделпоследовательностями из данных наборов. В то же время в некоторые кластеры попало много видео, например, один из кластеров содержит 7 видео и еще один – 5 видео. Такое неравномерное распределение по пространственно-временной сложности, а также отсутствие примеров простых видео, которые чаще всего встречаются в реальных данных, показывает, что данные коллекции не являются репрезентативными для анализа производительности видеокодеков.

3.2. Предлагаемый метод выбора видео

В данном подразделе предлагается методика создания репрезентативных наборов видеопоследовательностей для оценки качества кодирования. Описываемых подход использовался в нескольких аналитических сравнениях видеокодеков, проведенных видеогруппой лаборатории компьютерной графики и мультимедиа в МГУ. Для того, чтобы повысить достоверность сравнений, необходимо иметь возможность менять тестовые наборы видео в каждом новом сравнении. Для создания коллекции, из которой можно составлять наборы тестовых видео, было проанализировано 384946 видео на видеохостинге Vimeo с целью получения 4K и FullHD видео с битрейтом выше 50 Мбит/с. Выбрав видео с лицензией, разрешающей их использование, было получено 2091 видео в формате 4K и 4945 видео в формате FullHD. На рис. 2 показано распределение битрейта видео полученной коллекции по годам в процессе ее пополнения.

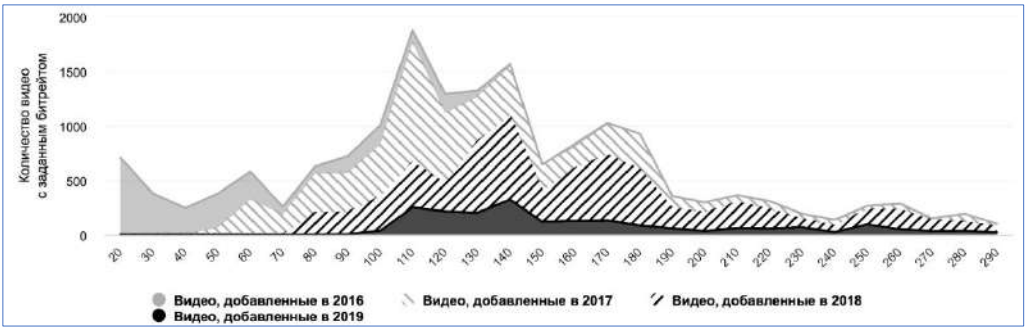


Рис. 2. Изменение распределения битрейта видео в используемой коллекции с 2016 по 2019 год
Fig. 2. Change in the distribution of video bitrate in the used collection from 2016 to 2019

Большинство среди загруженных видеопоследовательностей были длиною 10-15 минут, в то время как для тестирования качества кодирования обычно достаточно 30-60 секунд. Поэтому полученные видеопоследовательности были разбиты на фрагменты длиной около 1000 кадров. Из всех FullHD видео было получено 12402 фрагмента, а из видео 4K – 6016 фрагментов. Общий размер нашей коллекции видеофрагментов достиг 18418.

Для оценки пространственной и временной сложности полученных фрагментов они были закодированы видеокодеком x264 с фиксированным параметром квантования (QP),

а затем для них были вычислены временная и пространственная сложности по методу, описанному в предыдущем подразделе. Кроме того, был добавлен дополнительный этап предварительной обработки видео для унификации его цветности, так как это влияет на оценку сложности видео: все видео были преобразованы в цветовой формат YUV 4:2:0. Следующий шаг для построения репрезентативного набора видео заключается в разделении (кластеризации) всего пространства видео на группы, содержащие похожие видео с точки зрения пространственной и временной сложности. На данном этапе может использоваться любой метод кластеризации, при этом количество кластеров определяется необходимым размером составляемого набора видео. После кластеризации видео, находящиеся ближе всего к центру каждого кластера, помечаются как кандидаты для финального набора. Последний шаг заключается в проверке отмеченных кандидатов. Этот шаг выполняется при наличии каких-либо ограничений на используемые видео, таких как тип содержания.

При подготовке первой части аналитического отчета о производительности видеокодеков на FullHD видео [12] были привлечены эксперты из видеоиндустрии для участия в отборе видео с помощью голосования за предложенных видео-кандидатов в финальный набор. Этот шаг потребовал дополнительного времени, но позволил создать более репрезентативный набор, учесть предпочтения разработчиков кодеков и помог сделать отчет о проведенном анализе более репрезентативным. На рис. 3 набор видео, созданный автоматически, сравнивается с финальным набором, созданным после получения голосов экспертов.

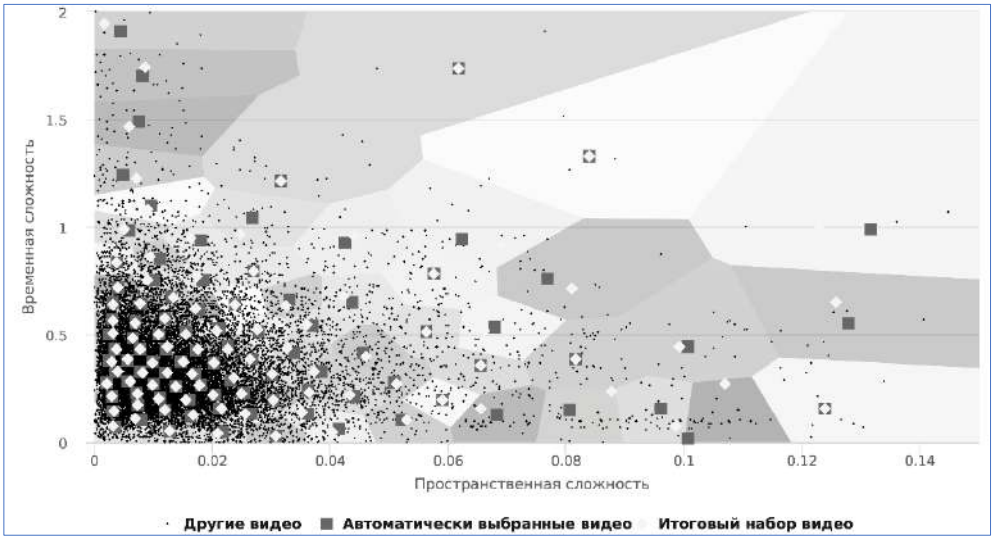


Рис. 3. Финальные видео после голосования экспертов и автоматически выбранные видео на фоне всей видеоколлекции

Fig. 3. Final videos after expert voting and automatically selected videos against the background of the entire video collection

Созданная коллекция из более чем 18000 видео занимает около 15,4 Тб. Ссылки на видео из предыдущих тестовых наборов, использованных в сравнениях, доступны в отчетах [13].

4. Измерение качества закодированного видео

Методы оценки качества видео можно разделить на 3 категории [14]: полные референсные, сокращенные референсные и нереференсные методы. Полные референсные методы наиболее распространены, поскольку их результаты легко

интерпретируются. Как правило, такие методы используются для оценки степени искажений в видео и их заметности для зрителя. Основным недостатком данного подхода по сравнению с сокращенным референсным и нереференсным подходами является необходимость наличия оригинального видео для сопоставления со сравниваемым, что часто неосуществимо.

Наиболее популярные алгоритмы для оценки качества кодирования имеют полный референсный тип: PSNR, SSIM [15], VIF [16], VMAF [17]. В этом разделе обсуждается проблема повышения результатов работы алгоритмов оценки качества с помощью изменения видео для получения более высоких объективых показателей при ухудшении визуального качества. В качестве примера продемонстрирован подход изменения видео, который позволяет фальсифицировать результаты метода VMAF, популярного полного референсного метода оценки качества видео. Кроме того, в разделе описаны проблемы применения нереференсных подходов измерения качества видео для аналитических сравнений видеокодеков на примере популярной метрики NIQE [18] и предложена методика фильтрации ее значений, которая позволяет повысить релевантность результатов NIQE для оценки качества видеопоследовательностей.

4.1. Модификация видео для увеличения показателей качества

Многие видеокодеки имеют возможность модифицировать качество выходного видеопотока для повышения показаний определенного алгоритма объективного качества. Например, видеокодеки x265 и x264 имеют опции для увеличения значений PSNR и SSIM. Эти опции обычно используются для достижения более высоких позиций в рейтингах объективных сравнениях, в то время как в повседневных сценариях кодирования чаще используются параметры психовизуальной адаптации. Однако недавнее исследование, проведенное в лаборатории компьютерной графики и мультимедиа посвященное субъективному сравнению видеокодеков [19], показало, что опция «--tune ssim» также увеличивает и субъективное качество закодированного видео.

В данном подразделе описаны результаты проведенного исследования по модификации цвета и контрастности видеопоследовательностей, которые повышают значения алгоритма оценки качества VMAF. VMAF – это полный референсный метод для оценки качества изображений, разработанный компанией Netflix, который быстро завоевал популярность благодаря техническому блогу Netflix на Medium [17] и большому количеству публикаций в других блогах. Согласно заявлениям авторов, метод показывает высокую корреляцию с субъективным качеством по результатам множества аналитических сравнений и исследований. Несмотря на популярность алгоритма VMAF, ни один из видеокодеков на данный момент еще не имеет возможности модификации выходного потока для увеличения показаний VMAF.

VMAF имеет открытый исходный код и имеет возможность подключения пользовательских моделей, обученных на различных наборах данных для оценки визуального качества. Недостаток возможности настройки видео для VMAF заключается в том, что для полных референсных алгоритмов оценки качества видео любые искажения видео должны приводить к ухудшению значений качества, прогнозируемых алгоритмом. Эту особенность следует учитывать в задаче сравнения видеокодеков, так как преобразования выходного видеопотока для повышения показаний алгоритмов качества может привести к неверным выводам о победителях сравнения. Чтобы доказать возможность «обмана» алгоритма VMAF, был проведен анализ способов искажения видео, которые сохраняют показатели других алгоритмов оценки качества на том же уровне и в то же время увеличивают VMAF. В качестве контрольного алгоритма качества видео был использован метод SSIM.

Для исследования преобразований видео, которые могут увеличить показания VMAF, было выбрано четыре видеофрагмента различного содержания и с разной пространственной и временной сложностью, яркостью и контрастностью. Все видефрагменты имели разрешение FullHD и высокий битрейт. Три видеопоследовательности (*Crowd Run*, *Red Kayak* и *Speed Bag*) были взяты из открытой коллекции видео media.xiph.org и еще одна (*Bay timelapse*, сцена залива с волнами и колышущейся травой) взята из созданной коллекции, описанной в предыдущем разделе. Для преобразования цвета и контрастности были выбраны два известных и широко распространенных алгоритма обработки изображений: unsharp mask и выравнивание гистограммы. Для нахождения оптимальных конфигураций параметров данных методов использовался многоцелевой алгоритм оптимизации NSGA-II [20].

Согласно результатам проведенного исследования, для некоторых конфигураций метода выравнивания гистограммы оценка качества алгоритмом VMAF становится выше (увеличивается с 68 до 74), при этом SSIM изменяется слабо (уменьшается с 0,88 до 0,86). Результаты немного отличаются для некоторых протестированных видеопоследовательностей. Для видеопоследовательности *Crowd Run* значение VMAF не было увеличено при помощи алгоритма unsharp mask, однако оно было увеличено при помощи выравнивания гистограммы. Для видеофрагментов *Red Kayak* и *Speed Bag* с помошью unsharp mask были значительно увеличены значения VMAF и при этом значения SSIM уменьшились незначительно.

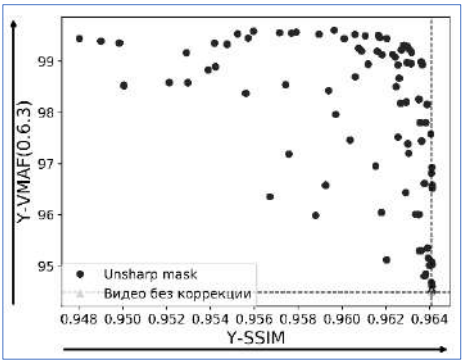


Рис. 4. Сравнение результатов измерения качества видео, измененного алгоритмом unsharp mask с разными параметрами для видео Speed bag. Каждая точка представляет собой качество одной конфигурации

Fig. 4. Comparison of the results of measuring the video quality, modified by the unsharp mask algorithm with different parameters for the video Speed bag. Each point represents the quality of one configuration.

На рис. 4 представлены результаты применения unsharp mask с разными настройками на одном из тестовых видеофайлов. Применение unsharp mask к кадрам видеопоследовательности отдаляет его от оригинала, поэтому снижение качества должно отражаться на снижении значений VMAF и SSIM аналогичным образом. На графике же показано, что существуют конфигурации, которые сохраняют значение SSIM и увеличивают VMAF. Примеры кадров после применения unsharp mask визуально не различимы, но есть небольшое различие при применении выравнивания гистограммы.

На Рис. 5 показана разница между исходным и измененным кадрами (до и после выравнивания гистограммы) видеопоследовательности *Crowd Run*, в этом примере VMAF был повышен с 51,005 до 53,083 при незначительном снижении SSIM (с 0,715 до 0,712). Более подробные результаты данного исследования описаны в [21]. Хотя VMAF стал значимым алгоритмом для оценки качества видео, широко используемым разработчиками видеокодеков, в его применении все еще имеется ряд нерешенных

вопросов. Именно поэтому SSIM до сих пор используется во многих сравнениях видеокодеков как основной объективный показатель качества видео.

Рис. 5. Сравнение в виде шахматной доски кадра №1 из видеопоследовательности Crowd run до и после выравнивания гистограммы с параметрами kernelsize = 8 и cliplimit = 0.00419. Обведены области, где разница наиболее заметна

Fig. 5. Comparison in the form of a chessboard of frame No. 1 from the Crowd run video sequence before and after aligning the histogram with the parameters kernelsize = 8 and cliplimit = 0.00419. The areas where the difference is most noticeable are circled

4.2. Проблемы применения нереференсных метрик

Нереференсные методы оценки качества видео незаменимы для задач, требующих визуальной оценки качества, в которых невозможно получить исходную версию видеофайла для контроля степени его искажения. Примерами таких задач являются облачное кодирование видео, восстановление фона видео, преобразование частоты кадров [22] и многие другие. Алгоритмы оценки качества видео и изображений данного типа часто создаются специально для определенной задачи (например, определения степени зашумленности, размытия или наличия блочных искажений) и нацелены на оценку восприятия зрителями данного искажения. Однако в задаче оценки качества кодирования видео данный тип алгоритмов может использоваться только как дополнение к полным референсным методам.

Нереференсные методы не могут стать основным показателем для анализа видеокодеков, так как иначе видеокодеки смогут быть настроены на выдачу визуально идеального результата, но при этом выдаваемое видео будет иметь мало общего с входным видео. В данной работе проанализировано применение нереференсного метода NIQE (Natural Quality Quality Evaluator) [18] к задаче аналитического сравнения видеокодеков. Данный метод популярен в настоящее время и показывает высокую корреляцию с визуальными оценками качества для изображений. Основная идея метода NIQE основана на построении набора функций качества видео и их использовании с помощью многомерной Гауссовой модели. Значение NIQE отражает степень искажений в кадре, и чем ниже значение, тем выше качество кадра.

Для оценки применимости метода NIQE для анализа видеокодеков было использовано 28 различных видеопоследовательностей разрешения FullHD с разной частотой кадров (от 24 до 60). Данные видеофрагменты были выбраны с использованием предложенной методики выбора видео, описанной во второй секции статьи. Каждый видеофрагмент был закодирован с помощью видеокодеков x264 и x265 с тремя разными предустановками («medium», «slow» и «placebo» для x264; «ultrafast», «medium» и «veryslow» для x265) и с 7 различными битрейтами (от 1 Мбит/с до 12 Мбит/с). Общее количество закодированных видеопотоков, использованных для анализа NIQE, составило 1176.

Для большинства закодированных видео алгоритм NIQE показал адекватные результаты, сопоставимые с другими алгоритмами оценки качества и с обычным распределением зависимости качества от величины битрейта. Но были случаи, когда NIQE показал неверные результаты. Ряд ситуаций, когда метод выдавал ошибочные значения, появлялся в видео с шумом или множеством мелких и текстурированных деталей, таких как песок, волны и трава. Например, на видеопоследовательности *Bay timelapse*, закодированной видеокодеком x265 с предустановкой «medium», оценка качества более высоких битрейтов согласно NIQE была хуже, чем низких битрейтов. Данный видеофрагмент содержал сцену морской бухты с движущейся водой и травой, снятую в режиме замедленной съемки

В другом похожем примере – видеопоследовательности *Playground* – для результатов кодирования обоими видеокодеками оценка качества NIQE была хуже на битрейте 4000 кбит/с, чем на 2000 кбит/с. Это видео содержало высокодетализированные фрагменты, такие, как яркая трава и песок. Подобные текстуры сложны для кодирования, и поэтому на низких битрейтах присутствовали видимые артефакты компрессии. При этом оценка качества NIQE для них оказалась выше, чем для высоких битрейтов, лишенных видимых артефактов компрессии (рис. 6). Это произошло из-за того, что алгоритм NIQE воспринял области с мелкой текстурой как шум, в то время как при кодировании трава стала размытой и менее детализированной. Другим примером ошибок данного алгоритма стали результаты оценки качества видеопоследовательностей с полностью черными кадрами (например, в начале).

Рис. 6. Кадр 58 из видеопоследовательности Playground. Более низкие оценки NIQE означают более высокое качество (пример ошибки метода)

Рис. 6. Кадр 58 из видеопоследовательности Playground. Более низкие оценки NIQE означают более высокое качество (пример ошибки метода)

Подобные кадры имели низкие показатели качества согласно NIQE. Данная особенность могла возникнуть из-за отсутствия подобных примеров в обучающих данных, используемых для создания NIQE. В некоторых случаях NIQE показывал корректный результат, несмотря на ошибки самих видеокодеков. На рис. 7 показаны кадры из видеопоследовательности *Hera*, где на более высоком битрейте видны артефакты компрессии, отсутствующие на низком битрейте. При анализе алгоритма была обнаружена еще одна особенность, которая не совпадала с данными, заявленными в оригинальной статье про NIQE. Авторы утверждали, что алгоритм не применим к

кадрам, созданным с помощью компьютерной графики, но в проведенном исследовании было обнаружено, что метод работает для некоторых типов реалистичной анимации (особенно для записей видеоигр).

Рис. 7. Кадр 208 из видеопоследовательности Hera. Согласно NIQE, левое изображение визуально имеет лучшее качество (пример правильной работы метода)
Fig. 7. Frame 208 from the Hera video sequence. According to NIQE, the left image visually has the best quality (an example of the correct operation of the method)

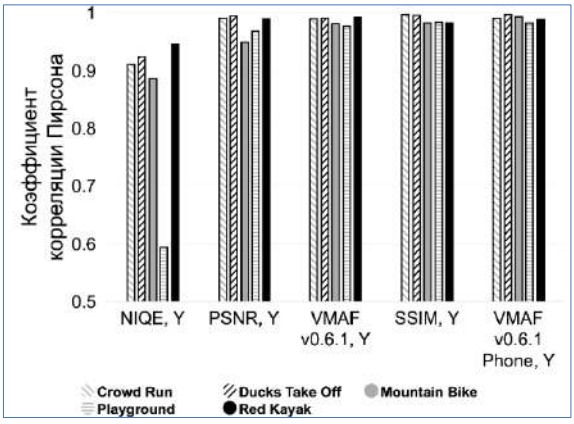


Рис. 8. Корреляция между объективными показателями качества и субъективными оценками
Fig. 8. The correlation between objective quality indicators and subjective assessments

Полученные показатели качества NIQE были сопоставлены с субъективными оценками на части протестированных видеопоследовательностей (рис. 8). Попарное субъективное сравнение качества закодированных видеофрагментов было проведено на платформе Subjectify.us, где в общей сложности было получено 22542 оценок от 473 экспертов. Подробное описание и методологию можно найти в отчете [23]. Усредненные по всем видео значения корреляции Пирсона показывают, что NIQE имеет самую низкую корреляцию с субъективными оценками (0,85), в то время как версия для смартфонов метрики VMAF v.0.6.1 имеет самую высокую корреляцию (0,99). Следует также отметить, что в настоящее время NIQE имеет даже более низкую корреляцию с субъективным качеством, чем PSNR (0,98), который, как считается, имеет низкую схожесть с субъективным качеством видео в задаче сравнения алгоритмов кодирования.

Некоторые из перечисленных проблем с ошибочными значениями NIQE могут быть решены с помощью фильтрации. В ходе покадрового анализа результатов NIQE было замечено, что значения, превышающие 40, встречаются крайне редко. Высокие значения результатов алгоритма часто встречаются в зашумленных или черных кадрах. Для повышения релевантности результатов оценки качества видео значения алгоритма NIQE на подобных кадрах не учитывались при получении усредненной оценки качества видеопоследовательностей. Таким образом, с учетом особенностей метода оценка NIQE для видео V была рассчитана следующим образом:

$$Score_i = \frac{\sum_i m_i * k_i}{\sum_i k_i}, i \in [0, N], k_i = \begin{cases} 1, m_i \in [0,15), \\ -0.04 * m_i + 1.6, m_i \in [15, 40), \\ 0, m_i \in [40, +\infty), \end{cases}$$

где m_i – значение NIQE для кадра i, k_i – весовой коэффициент для m_i, N – количество кадров.

Предложенный метод взвешенного усреднения оценок повысил релевантность результатов NIQE для некоторых видеопоследовательностей. Рис. 9 и 10 содержат примеры скорректированных RD-кривых (кривых битрейт-качество), где качество было измерено с помощью NIQE до и после фильтрации. С предложенной техникой усреднения RD-кривая для видеопоследовательности *Forest Dog* была избавлена от выбросов (рис. 9). Другим примером, где значения NIQE для обоих видеокодеков были исправлены предлагаемым усреднением, является видеопоследовательность *Music Clip* (рис. 10). Немонотонность RD-кривой кодека x264 вызвана высокой временной и пространственной сложностью этого видео. В статье [24] содержится подробное описание проведенного исследования.

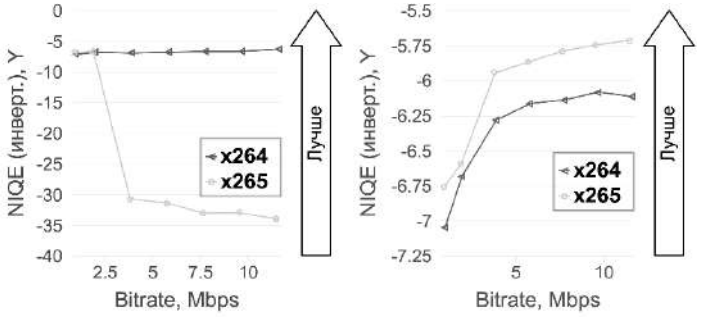


Рис. 9. График битрейт-качество видеопоследовательности Forest Dog до и после предложенного усреднения

Fig. 9. Bitrate-quality graph of Forest Dog video sequence before and after the proposed averaging

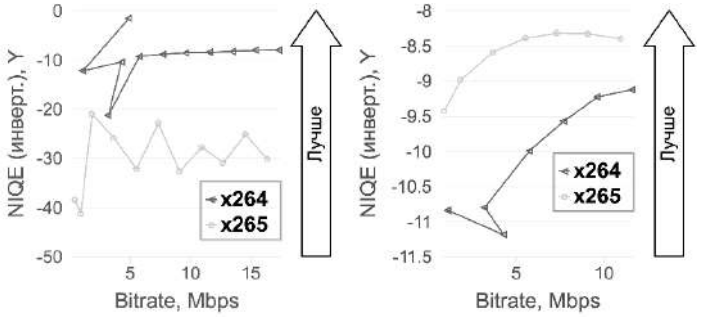


Рис. 10. График битрейт-качество для видеопоследовательности Music Clip до и после предложенного усреднения

Fig. 10. Bitrate-quality graph for the Music Clip video sequence before and after the proposed averaging

В проведенном исследовании алгоритм NIQE показал адекватные результаты для большинства видео. Однако все же был обнаружен ряд случаев, для которых алгоритм не применим. Результаты проведенного сравнения помогли выявить недостатки NIQE, которые необходимо учитывать при его использовании, например, неприменимость к анимационным роликам, к видео с полностью черными кадрами, шумным и высокодетализированным/текстурированным кадрам видео. По этим причинам алгоритм NIQE в настоящее время не является универсальным и показывает низкую корреляцию с субъективными оценками в задаче анализа качества кодирования видео. При необходимости использования данного алгоритма для анализа и сравнения видеокодеков или в других задачах, для повышения достоверности результатов алгоритма в описанных проблемных случаях может быть использован предложенный метод постобработки результатов NIQE.

5. Интерпретация результатов сравнения видеокодеков

Под интерпретацией результатов сравнения в данной статье подразумевается способ получения финальных оценок производительности видеокодеков с точки зрения соотношения битрейта, скорости кодирования и качества закодированных видео. После вычисления оценок качества для всех закодированных видео проводится сравнение степени сжатия и обеспеченное им качество видео. Существует несколько подходов для выполнения этого сравнения. Базовый способ заключается в вычислении значений качества для каждого видео и усреднении результатов по всем видео для каждого видеокодека. Однако данный метод не отражает производительности видеокодека при кодировании с разными битрейтами: производительность может быть высокой на всех битрейтах, кроме какого-либо определенного диапазона.

Более точный способ, называемый BD-rate (Bjontegaard Metric) [25], заключается в сравнении результатов кодирования по всем битрейтам. Метод основан на построении кривой полученных значений качества в зависимости от полученных значений битрейта. Данные кривые называются RD-кривыми (кривыми битрейт-качество). Значение BD-rate представляет собой среднее расстояние между двумя RD-кривыми сравниваемых кодеков, и сравнение обычно выполняется попарно между одним кодеком, выбранным в качестве референсного, и другими, называемыми тестовыми. В настоящее время BD-rate является наиболее популярным показателем, используемым для комплексной оценки производительности видеокодеков. Он может быть интерпретирован как процент снижения битрейта, обеспечиваемый тестовым видеокодеком по сравнению с референсным видеокодеком при сохранении качества закодированного видео.

5.1. Недостатки BD-Rate – метода ранжирования видеокодеков в сравнениях

В данном подразделе показаны недостатки подхода BD-rate на примерах ситуаций, возникающих в сравнениях видеокодеков, а другой подход для оценки производительности кодеков, учитывающий обнаруженные особенные случаи, описан в следующем подразделе.

Вычисление BD-rate включает в себя несколько этапов, каждый из которых может быть использован в разных вариациях. Алгоритм состоит из следующих шагов.

1. Вычисляются RD-значения (точки на графике битрейт-качество) для референсного и тестового видеокодеков. Эти точки представляют результаты кодирования тестового видео с разными целевыми битрейтами. Должно быть вычислено как минимум четыре точки, а дополнительные точки, не попавшие в определенный правилами диапазон битрейтов, должны быть удалены из рассмотрения.

2. Для полученных показателей вычисляется логарифм.
3. Проводится интерполяция кубическим полиномом для получения RD-кривых из полученных на предыдущем шаге точек.
4. Определяется диапазон значений метрики качества как диапазон пересечения полученных двух кривых, ограниченный выбранными значениями битрейта (также в качестве границ могут использоваться значения метрики третьего видеокодека с фиксированными значениями параметра квантизации).
5. Полученные кривые численно интегрируются в диапазоне значений метрики качества
6. Результирующие интегрированные логарифмические значения преобразуются обратно в линейные с помощью взятия экспоненты, а затем вычисляется разница в процентах между референсным и тестовым кодеками.

Во время проведения сравнений видеокодеков было обнаружено несколько случаев, в которых BD-rate не может быть вычислен. Первый случай довольно распространен в масштабных сравнениях кодеков, особенно при сравнении представителей разных стандартов. Он заключается в ситуации, в которой для очень разных кодеков две RD-кривые не имеют пересечения по показателям качества в диапазоне битрейтов, используемом для сравнения. BD-rate не может быть вычислен для этих кодеков, в то время как возможное решение заключается в добавлении результатов на дополнительных битрейтах, необходимых для получения требуемого пересечения.

Другая проблема связана со стратегией интерполяции кривых, используемой в BD-rate. Графики ниже на рис. 11 показывают, как кубический полином неверно определяется полученными точками, в то время как обычная линейная интерполяция строит более реалистичную RD-кривую. Обычно RD-кривая монотонна и имеет форму параболы: качество видео плавно увеличивается с увеличением битрейта.

Еще одна проблема с BD-rate заключается в том, что в данном методе не рассматривается случай наличия немонотонных RD-кривых и его обработка. Подобная ситуация возникает, когда получившийся битрейт закодированного видеофрагмента сильно отличается от целевого или качество видео с большим битрейтом оказывается хуже, чем качество видео с меньшим битрейтом, из-за особенностей работы видеокодека. В методологии BD-rate не упоминается, какой порядок точек (по возрастанию целевого битрейта или отсортированный по фактическому битрейту) следует использовать, или такие точки в таком случае должны быть удалены из рассмотрения.

BD-rate – более показательный метод сравнения производительности видеокодеков, чем, например, получение простого среднего значения качества. Однако у него существует ряд недостатков, описанных выше. Кроме того, непрофессионалам может быть сложно интерпретировать оценки BD-rate: он показывает процентную разницу двух интегралов, разделенную на диапазон интегрирования, и отражает битрейт, скорость или качество для одной «единицы» соответствующего показателя. Такое соотношение не всегда верно отражает прирост производительности, так как многие объективные алгоритмы качества имеют нелинейное соответствие значений с субъективным качеством. Например, небольшая разница в высоких значениях метрики SSIM обычно менее заметна, чем в средних значениях. Что еще более важно, все проблемы с интерпретацией BD-rate, описанные выше, могут повлиять на результаты сравнений видеокодеков и изменить распределение победителей. По этой причине предлагается иной алгоритм вычисления рейтинга видеокодеков, который описан в следующем подразделе.

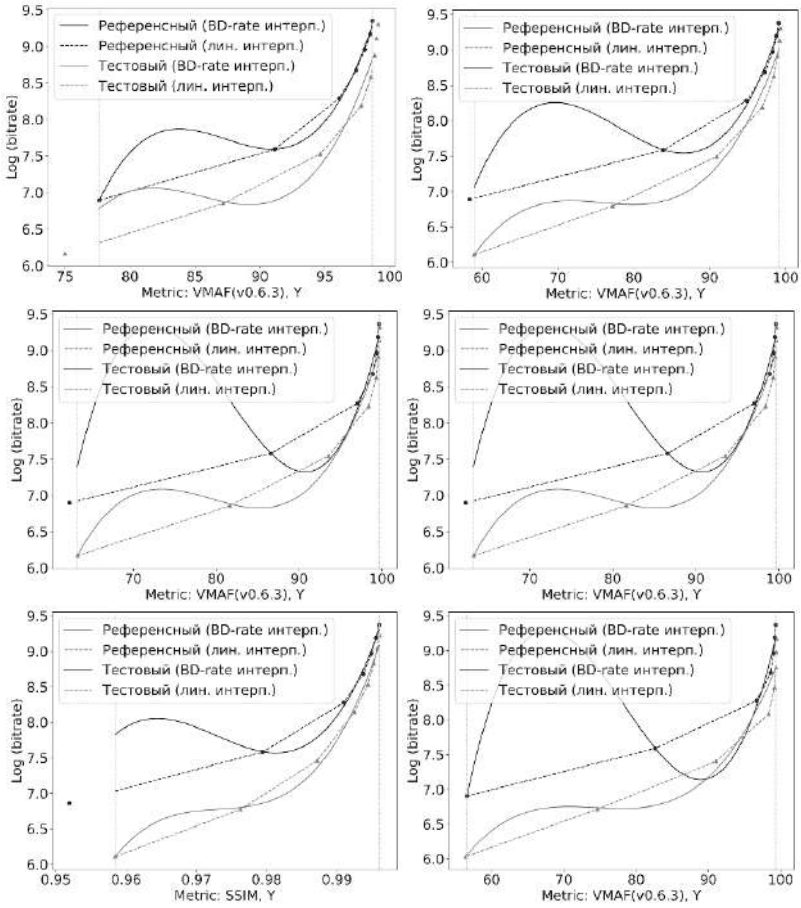


Рис. 11. Сравнение линейной (пунктирная линия) и кубической полиномиальной (сплошная линия) интерполяции по заданным точкам битрейта/качества. Линии ограничены границами сравниваемых областей референсного и тестового кодеков (учитывается только область пересечения)

Fig. 11. Comparison of linear (dashed line) and cubic polynomial (solid line) interpolation at given bitrate / quality points. The lines are limited by the boundaries of the compared areas of the reference and test codecs (only the intersection area is taken into account)

5.2. BSQ-rate – предлагаемый метод построения общего рейтинга видеокодеков

С учетом недостатков BD-rate предлагается новый алгоритм вычисления рейтинга видеокодеков по соотношению битрейта и качества закодированных видео. В новом подходе к сравнению видеокодеков также учитывается скорость кодирования на этапе определения правил аналитического сравнения: все видеокодеки сравниваются в определенных сценариях, имеющих ограничения на скорость кодирования. Предлагаемый подход для ранжирования видеокодеков имеет несколько шагов, аналогичных с BD-rate, таких как интеграция RD-кривых референсного и тестового видеокодеков, но отличается стратегией интерполяции и определением разницы между кривыми.

Кроме того, предлагаемый алгоритм учитывает ряд особых случаев, которые встречаются в реальных сравнениях, таких как немонотонные и непересекающиеся RD-

кривые видеокодеков. Данный алгоритм использовался для проведения исследований и подготовки аналитических отчетов о производительности видеокодеков, проводимых видеогруппой лаборатории компьютерной графики и мультимедиа МГУ ежегодно в течение нескольких лет. Предлагаемый метод был назван «BSQ-rate» (bitrate for the same quality, битрейт для одинакового качества), и его вычисление состоит из следующих этапов.

1. Вычисляются RD-значения (точки на графике битрейт-качество) для референсного и тестового видеокодеков
2. (опционально) При наличии значительных выбросов, которые приводят к образованию немонотонной RD-кривой, они удаляются из рассмотрения
3. Оси битрейта и качества видео инвертируются
4. Применяется линейная интерполяция к полученным точкам. Точки должны быть связаны в порядке реального (не целевого) битрейта видео
5. Устанавливается интервал для интегрирования как граница пересечения сегментов полученных кривых
6. (опционально) Если пересечение отсутствует или точки референсного кодека не полностью покрыты значениями тестового кодека, выполняются дополнительные измерения для получения необходимых точек для пересечения
7. Вычисляются площади под кривыми в выбранном интервале интегрирования и определяется их соотношение: BSQ-rate = S1/S2 (см. рис. 12). Результатом является среднее соотношение битрейта при фиксированном качестве для двух видеокодеков. При сравнении более, чем двух кодеков, один из них определяется как референсный кодек, а качество остальных сравнивается с его качеством.

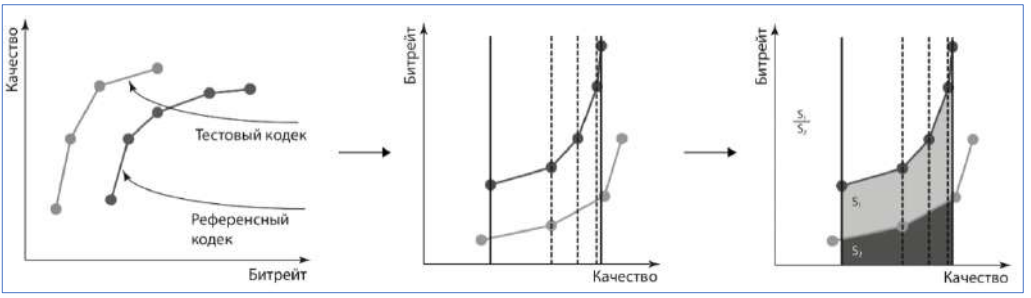


Рис. 12. Основные этапы вычисления BSQ-rate
Fig. 12. The main stages of calculating the BSQ-rate

В данном алгоритме описано два дополнительных шага, которые учитывают реальные ситуации, возникающие при сравнении видеокодеков. Первый случай - когда не существует общего сегмента качества видео, для которого два видеокодека имеют вычисленные результаты. В этом случае проводятся дополнительные измерения видеокодеков для дополнительных более высоких и/или более низких битрейтов. Схематический пример (рис. 13) показывает, что, когда результаты для дополнительных битрейтов (пунктирная часть линии) пересекаются с референсным кодеком, они позволяют провести сравнение двух кривых. В зависимости от ситуации дополнительные измерения могут проводиться как для референсного, так и для тестовых видеокодеков.

Другой опциональный шаг описывает случай, когда результаты кодирования для соседних битрейтов сильно различаются из-за особенностей работы кодека. Такая ситуация иногда возникает при перегреве серверов или влиянии внешних процессов, что решается проведением и усреднением нескольких измерений. Однако существуют случаи, когда немонотонная RD-кривая была построена по результатам серии измерений.

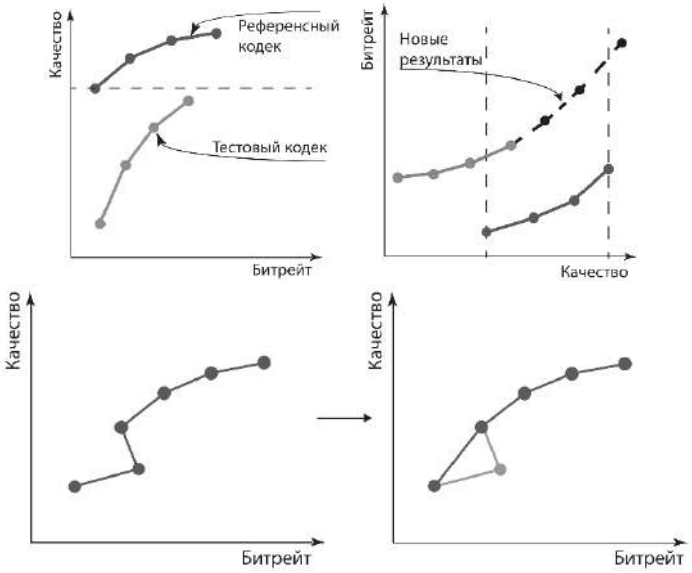


Рис. 13. Измерение кодека на дополнительных битрейтах для достижения пересечения с другими кодеками по оси качества
Fig. 13. Measurement of the codec at additional bitrates to achieve intersection with other codecs along the quality axis

Подобные случаи возникают из-за сбоев и особенностей работы видеокодеков и часто случаются при обработке какого-либо сложного видео. Данные ситуации обрабатываются следующим образом: при интерполяции для каждой точки используется следующая точка, соответствующая целевому битрейту, которая имеет большее или равное качество. Предложенный метод дает возможность получить монотонную кривую (рис. 14).

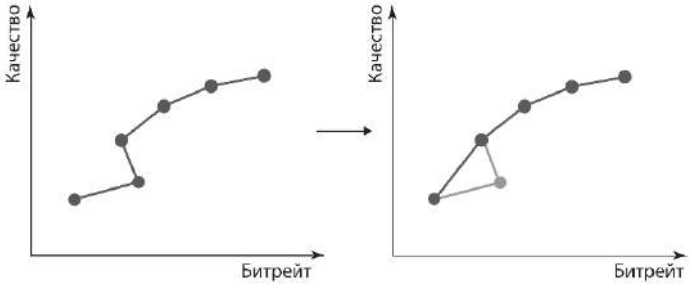


Рис. 14. Обработка немонотонных RD-кривых
Fig. 14. Processing nonmonotonic RD-curves

6. Заключение

В данной статье рассмотрены основные этапы подготовки и проведения аналитических сравнений видеокодеков, показаны недостатки некоторых распространенных подходов и предложены новые решения для сравнения видеокодеков. Собрана коллекция из 18418 фрагментов пользовательских видео на видеохостинге Vimeo и показано, что открытые наборы видео “FullHD” и “Gaming” ресурса media.xiph.org, популярные среди исследователей и разработчиков видеокодеков, не является репрезентативным с точки зрения соответствия реальным пользовательским видео. Большинство видео данных наборов являются сложными и наборы не покрывают область с самым распространенным среди видео соотношением пространственно-временной сложности.

Предложен метод создания репрезентативных наборов видео, покрывающих все сегменты пространственной и временной сложности пользовательских видео, основанный на кластеризации большой коллекции видео.

В разделе, посвященном алгоритмам оценки качества закодированных видео, показаны недостатки популярных методов VMAF и NIQE. Для первого из них в результате исследований выявлены способы нечестного повышения показателей, а для второго обнаружены случаи, приводящие к получению нерелевантных оценок и предложен способ их устранения. Также в статье показаны недостатки общепринятого метода BD-rate ранжирования результатов сравнения видеокодеков. С учетом выявленных недостатков предложен алгоритм ранжирования, названный BSQ-rate, включающий в себя корректный для получения RD-кривых способ интерполяции кривых и учитывающий обработку необычных результатов, полученных в ходе сравнения.

Данные результаты были получены в процессе исследований производительности видеокодеков и подготовки аналитических отчетов, организуемых видеогруппой лаборатории компьютерной графики и мультимедиа МГУ, и предлагаемая в данной статье методология валидировалась и дорабатывалась и на протяжении нескольких лет. Хотя существует множество способов сравнения видеокодеков и предлагаемые методы не являются единственно правильными, любая методология должна подвергаться всевозможным разнообразным проверкам и улучшаться с учетом обнаруженных недостатков. Таким образом мы будем стремиться к повышению качества и репрезентативности проводимых исследований производительности видеокодеков.

Список литературы / References

[1] Cisco VNI Report 2017-2022, 2018 update. Available at: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.
[2] Xiph.org Test Media, 2019. Available at: https://media.xiph.org/.
[3] Video Quality Experts Group Test Sequences, 2019. Available at: ftp://ftp.crc.ca/crc/vqeg/TestSequences/.
[4] MPEG-2 Transport Stream Test Patterns and Tools, 2019. Available at: http://www.w6rz.net/.
[5] Sveriges Television: The SVT High Definition Multi-Format Test Set, 2019. Available at: ftp://vqeg.its.bldrdoc.gov/HDTV/SVT_MultiFormat.
[6] Columbia Consumer Video (CCV) Database, 2019. Available at: .http://www.ee.columbia.edu/ln/dvmm/CCV.
[7] CDVL The Consumer Digital Video Library, 2019. Available at: https://www.cdvl.org/.
[8] LIVE Public-Domain Subjective Video Quality Database, 2019. Available at: http://live.ece.utexas.edu/research/quality/live_video.html.
[9] Video samples from KODI Wiki, 2019. Available at: https://kodi.wiki/view/Samples.
[10] Ultra Video Group test sequences, 2019. Available at: http://ultravideo.cs.tut.fi/#testsequences
[11] C. Chen, S. Inguva, A. Rankin, A. Kokaram. A subjective study for the design of multi-resolution ABR video streams with the vp9 codec. In Proc. of Electronic Imaging Symposium. Visual Information Processing and Communication VII, 2016, pp. 1-5.
[12] D. Vatolin, D. Kulikov, M. Erofeev, A. Antsiferova, S. Zvezdakov, D. Kondranin, S. Grokholsky. 2019. MSU FullHD Video Codec Comparison 2019. Available at: http://compression.ru/video/codec_comparison/hevc_2019/#main_report.
[13] MSU Video Codecs Comparisons, 2019. Available at: http://compression.ru/video/codec_comparison/index_en.html.
[14] S. Chikkerur, V. Sundaram, M. Reisslein, L. J. Karam. Objective video quality assessment methods: A classification, review, and performance comparison. IEEE Transactions on Broadcasting, vol. 57, issue 2, 2011, pp. 165–182.
[15] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, vol. 13, issue 4, 2004, pp. 600–612.
[16] H.R. Sheikh, A.C. Bovik. Image information and visual quality. IEEE Transactions on image processing, vol.15, issue 5, 2006, pp. 430-444.

[17] VMAF: Perceptual video quality assessment based on multi-method fusion, 2016. Available at: https://medium.com/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652

[18] A. Mittal, R. Soundararajan, A. C. Bovik. Making a «completely blind» image quality analyzer. IEEE Signal Processing Letters, vol. 20, issue 3, 2012, p. 209-212.

[19] D. Vatolin, D. Kulikov, M. Erofeev. MSU Video Codec Comparison 2017 Part III: Full HD Content, Subjective Evaluation, http://www.compression.ru/video/codec_comparison/hevc_2017/MSU_HEVC_comparison_2017_P3_subjective.pdf

[20] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, vol. 6, issue 2, 2002, pp.182-197.

[21] A. Zvezdakova, S. Zvezdakov, D. Kulikov, D. Vatolin. Hacking VMAF with Video Color and Contrast Distortion. In Proc. of the 29th International Conference on Computer Graphics and Vision (GraphiCon 2019). CEUR Workshop Proceedings, vol. 2485, 2019, pp. 53-57.

[22] Ватолин Д.С., Гришин С.В. Двукратное увеличение частоты кадров видео на основе двунаправленной компенсации движения. Программирование, том 35, no. 6, 2009, стр. 67-80 / D.S. Vatolin, S. V. Grishin. Double up-conversion of video frame rate based on bidirectional motion compensation. Programming and Computer Software, vol. 35, no. 6, 2009, pp. 351-364.

[23] D. Vatolin, D. Kulikov, M. Erofeev, A. Antsiferova, S. Zvezdakov, D. Kondranin. 2018. MSU Video Codec Comparison 2018, Subjective Report. Available at: http://compression.ru/video/codec_comparison/hevc_2018/#subjective_report.

[24] A. Zvezdakova, D. Kulikov, D. Kondranin, D. Vatolin. Barriers Towards No-reference Metrics Application to Compressed Video Quality Analysis: on the Example of No-reference Metric NIQE. In Proc. of the 29th International Conference on Computer Graphics and Vision (GraphiCon 2019). CEUR Workshow Proceedings, 2019, Vol. 2485, p. 22-27.

[25] G. Bjontegaard. Calculation of average PSNR differences between RD-curves. ITU-T VCEG, Document VCEG-M33, 2001.

Информация об авторах / Information about authors

Анастасия Всеволодовна ЗВЕЗДАКОВА (Анциферова), аспирантка факультета ВМК, участник видеогруппы лаборатории компьютерной графики и мультимедиа. Сфера научных интересов включает анализ и оптимизацию видеокодеков, оценку субъективного качества стереоскопического видео.

Anastasia Vsevolodovna ZVEZDAKOVA (Antsiferova), postgraduate student of the CMC faculty, a member of the video group of the computer graphics and multimedia laboratory. Her research interests include analysis and optimization of video codecs, assessment of the subjective quality of stereoscopic video.

Дмитрий Леонидович КУЛИКОВ, кандидат физико-математических наук, доцент Института системного анализа и управления Государственного университета «Дубна», участник видеогруппы лаборатории компьютерной графики и мультимедиа МГУ. Научные интересы: проблемы обработки видео: удаление шумов, улучшение яркости, контрастности и улучшение цвета и т.д.; анализ видеокодеков.

Dmitry Leonidovich KULIKOV, Candidate of Physics and Mathematics, Associate Professor of the Institute for System Analysis and Management of Dubna State University, member of the video group of the Laboratory for Computer Graphics and Multimedia, Moscow State University. Research interests: video processing problems: noise removal, brightness and contrast enhancement, color enhancement, etc.; video codec analysis.

Сергей Васильевич ЗВЕЗДАКОВ, аспирант факультета ВМК, участник видеогруппы лаборатории компьютерной графики и мультимедиа. Его научные интересы связаны с анализом и оптимизацией работы видеокодеков, анализом качества видео, построением математических моделей видеокодеков.

Sergey Vasilievitch ZVEZDAKOV, postgraduate student of the faculty of the VMK, a member of the video group of the laboratory of computer graphics and multimedia. His research interests are related to the analysis and optimization of video codecs, analysis of video quality, the construction of mathematical models of video codecs.

Дмитрий Сергеевич ВАТОЛИН, кандидат физико-математических наук, старший научный сотрудник лаборатории компьютерной графики и мультимедиа МГУ. Научные интересы: сжатие видео и данных, обработка видео, 3D-видео.

Dmitry Sergeevich VATOLIN, Candidate of Physics and Mathematics, Senior Researcher, Laboratory of Computer Graphics and Multimedia, Moscow State University. Research interests: video and data compression, video processing, 3D-video.

DOI: 10.15514/ISPRAS-2020-32(1)-6

Дополненная реальность при визуализации данных с использованием свойств «золотого» сечения

А.В. Воронин, ORCID: 0000-0002-0956-0413 <aleksey.v.v@mail.ru>

Федеральный исследовательский центр «Информатика и управление» РАН, 119333, Россия, г. Москва, ул. Вавилова, д. 44, к. 2

Аннотация. Дополненная реальность, как результат введения в поле восприятия данных, обеспечивающих лучшую визуализации информации, все больше привлекает внимание специалистов в области программного обеспечения для демонстрационных комплексов и геоинформационных систем. Наглядность визуализируемой информации имеет важное значение как для работы оператора, так и пользователей информационной системы. Использование закономерностей зрительного восприятия объектов, связанных со свойствами «золотого» сечения, позволяет сформулировать критерий наглядности визуализируемых данных, характеризующий комплексное восприятия информации, отображаемой на экране видеомонитора или проекционной панели. Цель представленного в статье исследования заключается в определении критерия наглядности визуализируемых данных на основе свойств «золотого» сечения и исследования условий ее обеспечения на примере отображения метаданных на экране монитора и проекционной панели. Критерий наглядности визуализации данных определяется через коэффициент покрытия площади экрана информацией. Оптимальное значение коэффициента соответствует математическому определению «золотого» сечения. В качестве результата исследования следует выделить анализ свойств «золотого» сечения при отображении информации и определение критерия наглядности визуализации данных, что позволяет операторам и потребителям комплексно воспринимать видеоданные на электронных средствах проекции. Разработаны итерационные алгоритмы выбора масштаба отображения данных по критерию наглядности: алгоритм анализа отображаемых данных слоя на примере электронной карты и алгоритм последовательного анализа слоев. Исследовано влияние масштаба отображаемых данных на наглядность их визуализации на экранах различных размеров. Практическая ценность результатов состоит в том, что предложенный критерий представляет математическую трактовку свойства «золотого» сечения для визуализации информации на современных электронных средствах отображения данных.

Ключевые слова: геоинформационная система; электронная карта; наглядность визуализации данных; «золотое» сечение

Для цитирования: Воронин А.В. Дополненная реальность при визуализации данных с использованием свойств «золотого» сечения. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 109-120. DOI: 10.15514/ISPRAS–2020–32(1)–6

Augmented reality when visualizing data using «golden» section properties

A.V. Voronin, ORCID: 0000-0002-0956-0413 <aleksey.v.v@mail.ru>

The Federal Research Center «Computer Science and Control» of the RAS,
44, Cor. 2, Vavilova St., Moscow, 119333, Russian Federation

Abstract. Augmented reality, as a result of introducing into the field of perception data providing the best visualization of information, is increasingly attracting the attention of specialists in the field of software for demonstration complexes and geographic information systems. The visibility of the visualized information is important both for the work of the operator and users of the information system. Using the laws of visual perception of objects associated with the properties of the «golden» section, it is possible to formulate a visualization criterion for visualized data that characterizes the integrated perception of information displayed on the screen of a video monitor or projection panel. The purpose of the study presented in the article is to determine the visualization criterion for visualized data based on the properties of the «golden» section and study the conditions for its provision by the example of displaying of metadata on a monitor screen and projection panel. The visualization criterion is determined through the coverage coefficient of the screen area with information. The optimal value of the coefficient corresponds to the mathematical definition of the «golden» section. As a result of the study, it is necessary to highlight the analysis of the properties of the «golden» section when displaying information and the definition of the visualization criterion for data visualization, which allows operators and consumers to comprehensively perceive video data on electronic projection tools. Iterative algorithms have been developed for selecting the scale of data display by the criterion of visibility: an algorithm for analyzing the displayed layer data using an electronic map as an example and an algorithm for sequential layer analysis. The influence of the scale of the displayed data on the visibility of their visualization on screens of various sizes is investigated. The practical value of the results lies in the fact that the proposed criterion represents a mathematical interpretation of the property of the «golden» section for the visualization of information on modern electronic means of displaying data.

Keywords: geoinformation system; electronic card; demonstrativeness of data visualization» «golden» section

For citation: Voronin A.V. Augmented reality when visualizing data using «golden» section properties. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 109-120 (in Russian). DOI: 10.15514/ISPRAS–2020–32(1)–6

1. Введение

Качество и наглядность визуализации данных определяют удобство работы оператора и потребителей услуг с современными электронными сервисами, отображающими данные на экранах портативных устройств, мониторов и проекционных панелей [1-4]. В общем случае дополненная реальность, являющаяся результатом введения в поле восприятия сервисов, улучшающих визуализацию информации, является важной частью развития инфокоммуникационного мира и, как правило, имеет дело с динамически изменяющейся отображаемой ситуацией, большим числом объектов и метаданных, которые при отображении на экране формируют сложные представления. Эти изображения не должны затруднять зрительного восприятия информации. Поэтому визуализация данных должна быть информативна и наглядна.

В широком смысле под наглядностью понимается свойство психических образов объектов познания, выражающее степень доступности и понятности этих образов для познающего субъекта [5,6]. Применительно к отображаемым видеоданным под наглядностью следует понимать возможность легкого восприятия оператором или потребителем информации на экране системы. Поскольку данные представляют изображения, формируемые расположением и размерами объектов, блоков сопроводительной и другой информации, которые отображаются на экране устройства в

виде знаков соответствующего масштаба, наглядность визуализации может быть формализована в рамках моделей зрительного восприятия и распознавания объектов по изображениям [7, 8].

В литературе по вопросам построения информационных систем основное внимание уделяется архитектуре этих систем, вопросам хранения и программным решениям обработки информации. Визуализация же данных рассматривается применительно к отдельным видам отображаемой информации [9-12].

В статье на основе анализа зрительного восприятия и распознавания объектов сформулирован подход к повышению наглядности визуализации данных – выбор параметров и структуры отображаемых видеоданных с использованием свойств «золотого» сечения. Предложен и определен критерий наглядности визуализации данных. Разработаны рекомендации по отображению большого числа объектов с разнородной информацией.

2. Анализ особенностей зрительного восприятия данных и свойств «золотого» сечения при визуализации информации

Визуализация данных на современном этапе их развития предполагает отображение информации преимущественно с метаданными – сопроводительной текстовой информацией и графическими объектами. При этом сопроводительная информация чаще всего занимает значительное пространство экрана визуализационного устройства. Большинство имеющихся решений предполагают использование прозрачных или «всплывающих» форм. Отображаемые объекты и метаданные на экране формируют сложные представления разнородных данных, затрудняют восприятие отображаемой информации в целом. Для повышения наглядности отображения видеоданных необходимо учитывать объективные и субъективные закономерности зрительного восприятия объектов.

При зрительном восприятии оператор реализует структурное распознавание отображаемых объектов [6, 7]. Основными структурными признаками являются: размер, форма, цвет, тон. При структурном распознавании объекты характеризуются описанием их образа – совокупностью нескольких структурных признаков. При этом любой сложный по структуре объект можно представить в виде совокупности более простых объектов [13].

Для выполнения условий зрительного восприятия объектов и сопроводительной информации, определяемых моделью визуального распознавания, должны быть учтены особенности совместного зрительного восприятия простых объектов в составе сложных. К числу таких особенностей относится наличие «золотого» сечения, определяющего пропорции между размерами элементов объектов, при котором их изображения наилучшим образом воспринимаются человеком.

«Золотое» сечение – инструмент гармоничного восприятия образов, которое Человечество определило за историю своего развития. В качестве примеров использования пропорций «золотого» сечения в архитектуре и живописи следует привести элементы фасада Парфенона, картины Леонардо «Тайная Вечеря» и Микеланджело «Святое семейство», портрет Мона Лизы. Пример современного HiTech-дизайна – iPod фирмы Apple, банковские пластиковые карты – размеры устройства и карт пропорциональны параметрам «золотого» сечения.

Макс Билл (швейцарский скульптор, художник абстракционист, графический дизайнер) дал следующее определение искусства в математической трактовке, отражающее сущность визуализации информации с использованием свойств золотого сечения: «Математические концепции искусства не являются математикой в строгом смысле этого слова. Можно даже сказать, что было бы трудно в этом методе применить то, что мы понимаем под точной математикой. Это, скорее, сочетание ритмов и связей, законов,

имеющих личную природу в том же смысле, что и математика имеет свои инновационные элементы, рожденные умами ее первопроходцев».

Предлагается, рассмотрев «ключ к пониманию секретов совершенства в природе и искусстве», использовать свойства золотого сечения при отображении информации как решение задачи дополненной реальности при визуализации данных.

«Золотое» сечение с математической точки зрения – это иррациональное бесконечное число $\Phi = (1 + \sqrt{5})/2 = 1{,}618$ которое определяется из отношения $1/x = x/(1-x)$, где x – длина большего отрезка в геометрической трактовке «золотого» сечения [14]. Геометрическая трактовка «золотого» сечения состоит в делении отрезка AB единичной длины ($|AB| = 1$) точкой C на две части, при этом длина большего отрезка составляет $|BC| = x$, длина меньшего отрезка составляет $|AC| = x$, и выполняется соотношение $|AB|/|BC| = |BC|/|AC|$. В процентном значении величина «золотого» сечения Φ есть отношение величин $61{,}8/38{,}2$.

«Золотое» сечение присутствует в большинстве шедевров изобразительного искусства и архитектуры и гармонизирует восприятие изображений сложных объектов, чему посвящен ряд специальных исследований [15, 16]. Поэтому связанные с «золотым» сечением закономерности зрительного восприятия изображений, являясь субъективными, находят многочисленные проявления, и их целесообразно использовать при визуализации данных на электронных средствах проекции. На рис. 1 приведен «золотой» прямоугольник, как пример фигуры с параметрами «золотого» сечения в геометрии.

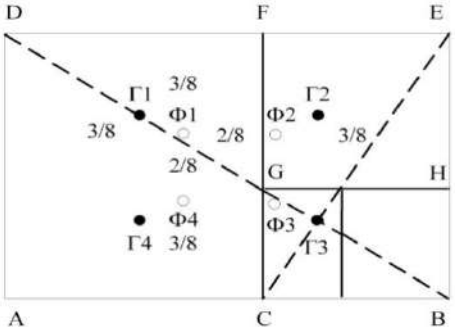


Рис. 1. «Золотой» прямоугольник
Fig. 1. «Golden» rectangle

В «золотом» прямоугольнике (рис. 1) отношение сторон (форматное отношение) равно числу Φ. Если отделить от «золотого» прямоугольника $ABDE$ квадрат $ADFC$ со стороной, равной меньшей стороне «золотого» прямоугольника, то оставшийся прямоугольник $CFEB$ тоже будет «золотым». То же самое произойдет, если от «золотого» прямоугольника $CFEB$ отделить квадрат $GFEH$, и т.д. Если провести диагонали в двух «золотых» прямоугольниках $CFEB$ и $CGHB$, то они всегда пересекаются под прямым углом. Точка пересечения диагоналей является геометрической точкой притяжения, куда уходит бесконечная последовательность получаемых «золотых» прямоугольников. В «золотом» прямоугольнике точками притяжения могут быть четыре точки $Г1, Г2, Г3, Г4$ – в зависимости от того, каким образом выделяются более мелкие «золотые» прямоугольники. Кроме того, в «золотом» прямоугольнике существуют так называемые фокусные точки или зрительные центры, расположенные на удалении $|AB|/\Phi$ и $|AD|/\Phi$ от его сторон. Таких точек также четыре: $Ф1, Ф2, Ф3, Ф4$ – они являются центрами зрительного восприятия.

Из наиболее популярных известны также «золотой» треугольник и правильный пятиугольник. «Золотой» треугольник – равносторонний треугольник, длины сторон которого равны длине стороны правильного десятиугольника, вписанного в круг, и радиусу

этого круга. Правильный пятиугольник – отношение длины диагонали к длине стороны равно числу Φ, диагонали правильного пятиугольника образуют правильную пятиконечную звезду, у которой каждый отрезок делится пересекающим его отрезком в «золотом» сечении.

Свойства «золотого» сечения могут быть положены в основу критерия наглядности визуализации данных, использование которого при отображении видеоданных позволяет улучшить субъективное восприятие информации и, как следствие, создать комфортные условия при решении прикладных задач оператору или пользователям.

3. Определение критерия наглядности визуализации данных

Визуализация информации и метаданных [17, 18] осуществляется согласно отображению числа объектов и блоков сопроводительной информации, соотнесенных к каждому объекту. Для наглядного восприятия пространственной и сопроводительной информации в виде, удобном для понимания и визуального анализа, требуется отображение объектов и сопроводительной информации без затемнения (закрытия) существенных для визуального анализа элементов изображения.

Общая площадь объектов, отображаемых на экране проекционного устройства, определяется выражением

$$S_{\Sigma_{\text{Об}}} = \sum_{1}^{I} X_i Y_i, \#(1)$$

где X_i– длина i-го объекта;

Y_i– ширина i-го объекта;

I – число объектов.

Общая площадь, занимаемая на экране проекционного устройства метаданными (сопроводительной информацией) определяется как

$$S_{\Sigma_{\text{МД}}} = \sum_{1}^{J} U_j V_j\,, \#(2)$$

где U_j– длина j-го блока метаданных;

V_j – ширина j-го блока метаданных;

J – число блоков метаданных.

Наиболее распространенным видом метаданных являются текстовые данные. Они выводятся на экран в виде текстовых фрагментов прямоугольной формы или выносок, отнесенных к соответствующим объектам.

Определяемые выражениями (1) и (2) общие площади, занимаемые на экране устройства отображаемыми объектами и блоками метаданных, суммируются и образуют суммарную площадь, занимаемую на экране объектами и метаданными:

$$S_{\Sigma} = S_{\Sigma_{\text{Об}}} + S_{\Sigma_{\text{МД}}}. \#(3)$$

Отношение суммарной площади, занимаемой на экране объектами и метаданными, к площади экрана назовем коэффициентом покрытия площади экрана информацией:

$$k = S_{\Sigma}/S_{\text{Э}}\,, \#(4)$$

где $S_{\text{Э}}$ – площадь экрана, определяемая его линейными размерами H_X и H_Y: $S_{\text{Э}} = H_X H_Y$.

В соответствии с определением, коэффициент k нормирован: $0 < k < 1$.

Коэффициент покрытия площади экрана информацией k, определяемый выражением (4), влияет на визуальное восприятие (наглядность) отображаемого на экране сложного изображения, представляющего собой совокупность пространственной и сопроводительной информации. Чем коэффициент k больше и ближе к единице, тем больше закрытие объектами и метаданными подложки и насыщенность изображения в целом. Чем коэффициент k меньше и ближе к нулю, тем меньше вероятность

распознавания малоразмерных объектов на изображении и ниже информативность изображения в целом. В обоих случаях ухудшается качество зрительного восприятия изображения и снижается наглядность визуализации отображаемых данных.

Исходя из качественного анализа влияния коэффициента покрытия площади экрана информацией k на комплексное восприятие видеоданных, предлагается критерий наглядности визуализации данных с использованием свойств «золотого» сечения. Пропорции, определяемые «золотым» сечением, сопоставляются с долей площади экрана визуализирующего устройства, занимаемой отображаемыми данными. В качестве критерия наглядности визуализации данных целесообразно использовать значение коэффициента покрытия площади экрана информацией (k), близкое к 0,382. Данное значение коэффициента соответствует значению знаменателя дроби, определяющей величину Φ. В этом случае покрытие площади экрана выводимыми на него объектами и метаданными менее половины всей площади экрана и наглядность визуализации отображаемой информации высокая – имеет место наилучшее комплексное восприятие видеоданных оператором или потребителем услуги. При этом значение коэффициента, близкое к значению $0{,}618$ числителя дроби, определяющей величину Φ, полагается предельно допустимым. В этом случае покрытие площади экрана выводимыми на него объектами и метаданными более половины всей площади экрана, комплексное восприятие видеоданных оператором или потребителями услуги затруднено. В диапазоне значений коэффициента покрытия площади экрана информацией $0{,}382 < k < 0{,}618$ в соответствии с субъективными данными качества зрительного восприятия изображений комплексное восприятие видеоданных возможно, но не наглядно. Введенный критерий наглядности отображения данных может быть использован при решении различных визуализационных задач.

В общем случае наилучшая наглядность визуализации данных в соответствии с предложенным критерием зависит от числа и геометрических размеров выводимых на экран объектов и блоков метаданных, а также размеров экрана. Коэффициент k является функцией ряда параметров:

$$k\left(I, X_1, \dots, X_I, Y_1, \dots, Y_I, J, U_1, \dots, U_J, V_1, \dots, V_J, H_X, H_Y\right) = \frac{S_{\Sigma_0}(I, X_1, \dots, X_I, Y_1, \dots, Y_I) + S_{\Sigma_{\text{МД}}}\left(J, U_1, \dots, U_J, V_1, \dots, V_J\right)}{S_{\text{Э}}(H_X, H_Y)} \#(5)$$

Для наглядности на рис. 2 представлена визуализация данных на экране монитора в соответствии с предложенным критерием при $k > 0{,}618$ (рис. 2а) и $k < 0{,}382$ (рис. 2б).

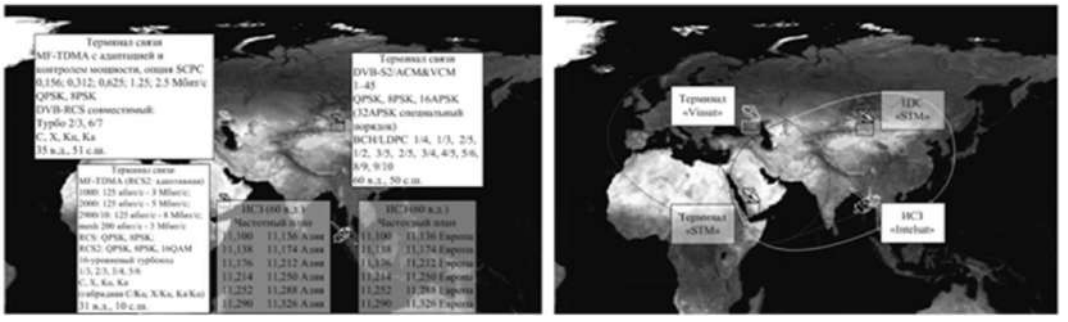


Рис. 2. Визуализация данных на экране монитора: а) k>0,618; б) k<0,382
Fig. 2. Visualization on the monitor screen: a) k>0,618; б) k<0,382

Определение коэффициента k в соответствии с его зависимостью от параметров отображаемых на экране объектов, блоков метаданных и размеров экрана (выражение

(5)) и обеспечение наилучшей наглядности визуализации данных в соответствии с пропорциями «золотого» сечения ($k \approx 0{,}382$) целесообразно алгоритмизировать.

4. Алгоритмы выбора масштаба отображения данных по критерию наглядности

На основе проведенного анализа свойств «золотого» сечения и введенного критерия наглядности визуализации данных с использованием свойств «золотого» сечения разработаем последовательность действий и счетную процедуру, определяющие алгоритм выбора масштаба отображения данных по критерию наглядности на примере отображения информации на устройстве визуализации в геоинформационной системе, как наиболее сложной с визуализационной точки зрения среди информационных систем.

В основе выбора масштаба отображения видеоданных лежит анализ зависимости коэффициента покрытия площади экрана информацией от геометрических параметров отображаемых на экране геообъектов, блоков метаданных и размеров экрана, определяемого выражением (5). Совокупность параметров, от которых зависит коэффициент k, может быть объединена понятиями структуры геообъектов и метаданных.

Вследствие зависимости коэффициента покрытия площади экрана информацией k от большого числа параметров $(I, X_1, \dots, X_I, Y_1, \dots, Y_I, J, U_1, \dots, U_J, V_1, \dots, V_J, H_X, H_Y)$ выбор масштаба отображения видеоданных осуществляется итерационно в результате последовательной трансформации структуры геообъектов и метаданных с поиском сочетания параметров, обеспечивающих достижение выбранного критерия наглядности визуализации данных. Основными этапами выбора масштаба отображения видеоданных являются следующие действия.

1. Определение слоя (слоёв), выводимых на экран данных и перечень отображаемых геообъектов.
2. Определение отображаемых метаданных (сопроводительной информации), соотнесенной к каждому отображаемому геообъекту.
3. Расчет суммарной площади геообъектов, выводимых на экран с учетом масштаба карты.
4. Расчет суммарной площади блоков метаданных, выводимых на экран, с учетом их пространственных размеров.
5. Определение суммарной площади, занимаемой на экране геообъектами и метаданными.
6. Расчет коэффициента покрытия площади экрана информацией для заданного размера экрана.
7. Сравнение рассчитанного значения коэффициента с критериальным значением, соответствующим выполнению критерия наглядности визуализации данных.
8. Трансформация структуры геообъектов и метаданных, если значение коэффициента k отличается от критериального значения, соответствующего выполнению критерия наглядности визуализации данных. Повторяется выполнение этапов 3–7 до выполнения критерия наглядности визуализации данных.
9. Соответствие структуры геообъектов и метаданных выбранному масштабу наглядного отображения, если значение коэффициента k совпадает с критериальным значением, соответствующим выполнению критерия наглядности визуализации данных.
10. Использование выбранной структуры геообъектов и метаданных, определяющей масштаб отображения данных для работы оператора или потребителей ГИС-услуги.

Практически для выполнения алгоритма целесообразно задавать диапазон значений коэффициента k. Это может быть либо диапазон значений $0{,}382 - \Delta k < k < 0{,}382 + \Delta k$ в окрестности «золотого» сечения, где Δk – допустимое отклонение от оптимального с

точки зрения комплексного восприятия данных, либо диапазон значений $0{,}382 < k < 0{,}618$, в котором комплексное восприятие возможно, но недостаточно наглядно.

Рассмотренный алгоритм является базовым. На практике целесообразно его применять последовательно в несколько проходов для каждого слоя отображаемых геообъектов, причем каждый последующий слой рассматривается совместно с уже рассмотренными слоями, и оптимальная структура геообъектов и метаданных выбирается для совокупности слоев, рассматриваемых на данном проходе. При использовании для отображения геообъектов L-слоев общее число проходов составляет L. На первом проходе рассматривается один слой, на втором проходе – два слоя и т.д., на последнем проходе – L слоев. Основными этапами выбора масштаба отображения видеоданных с последовательным рассмотрением слоев на каждом l-ом проходе $(l = 1, 2, \dots, L)$ являются следующие действия.

1. Определение слоёв выводимых данных и перечень отображаемых геообъектов.
2. Определение отображаемых геообъектов l-го слоя.
3. Определение отображаемых метаданных (сопроводительной информации), соотнесенных к каждому отображаемому геообъекту l-го слоя.
4. Анализ геообъектов и метаданных, относящихся к l-му слою, если $l = 1$.
5. Анализ совокупности геообъектов и метаданных l слоев, если $1 < l \leq L$ (геообъекты и метаданные, относящиеся к l-му слою, объединяются с геообъектами и метаданными, которые анализировались, и для которых была выбрана структура на предыдущем $(l-1)$-ом проходе).
6. Расчет суммарной площади совокупности анализируемых на l-ом проходе геообъектов, выводимых на экран.
7. Расчет суммарной площади совокупности анализируемых на l-ом проходе блоков метаданных, выводимых на экран.
8. Определение суммарной площади, занимаемой на экране совокупностью анализируемых на l-ом проходе геообъектов и метаданных.
9. Расчет коэффициента покрытия площади экрана информацией для заданного размера экрана.
10. Сравнение полученного значения коэффициента k для совокупности анализируемых на l-ом проходе геообъектов и метаданных с критериальным значением, соответствующим выполнению критерия наглядности визуализации данных.
11. Трансформация структура совокупности анализируемых на l-ом проходе геообъектов и метаданных, если значение коэффициента k отличается от критериального значения, соответствующего выполнению критерия наглядности визуализации данных. Повторяется выполнение этапов 5–10 до выполнения критерия наглядности визуализации данных.
12. Соответствие выбранной совокупности анализируемых на l-ом проходе геообъектов и метаданных масштабу наглядного отображения видеоданных, если значение коэффициента k совпадает с критериальным значением, соответствующим выполнению критерия наглядности визуализации данных.
13. Повтор выполнение этапов 4–12 с совместным анализом найденной совокупности геообъектов и метаданных для l слоев и геообъектов и метаданных, относящихся к $(l + 1)$-му слою, если $1 \leq l < L$.
14. Соответствие выбранной структуры геообъектов и метаданных L слоев масштабу наглядного отображения видеоданных, если $l = L$.
15. Использование выбранной структуры геообъектов и метаданных, определяющей масштаб отображения данных для работы оператора или потребителей ГИС-услуги.

Преимущества данного алгоритма с последовательным рассмотрением на каждом проходе различных слоев отображаемых данных по сравнению с базовым алгоритмом, на основе которого он построен, заключаются в ускорении поиска оптимальной структуры геообъектов и метаданных, в оперативной результативности промежуточной и итоговой визуализации.

При введении дополнительных условий выполнения каждого прохода и промежуточной визуализации результатов выполнения алгоритма на каждом проходе алгоритм позволяет выполнять оптимизацию послойной и поэлементной визуализации данных. Это позволяет оператору (потребителю) оперативно получать представление об интересующих его объектах с наглядным отображением данных в удобном для комплексного восприятия виде.

Как и при выполнении базового алгоритма, для выполнения алгоритма с последовательным рассмотрением различных слоев отображаемых данных целесообразно задавать диапазон значений коэффициента k либо в виде диапазона допустимых отклонений в окрестности «золотого» сечения, либо в виде диапазона значений, в котором комплексное восприятие видеоданных возможно, но недостаточно наглядно.

5. Исследование условий обеспечения наглядности визуализации данных с использованием свойств «золотого» сечения на экранах различных размеров

В соответствии с разработанным критерием наглядности визуализации данных при использовании свойств «золотого» сечения проведено исследование условий обеспечения наглядности визуализации данных. Анализировались зависимости коэффициента покрытия площади экрана информацией от числа и геометрических параметров отображаемых на экране объектов и блоков метаданных, а также размеров экрана. Расчеты коэффициента k проводились в соответствии с выражениями (1)–(5) для различных сочетаний параметров объектов и блоков метаданных.

На рис. 3 представлены графики зависимости коэффициента покрытия площади экрана информацией от размера экрана визуализирующего устройства, характеризуемого номером варианта размера экрана n, при различных значениях объектов I и блоков метаданных J.

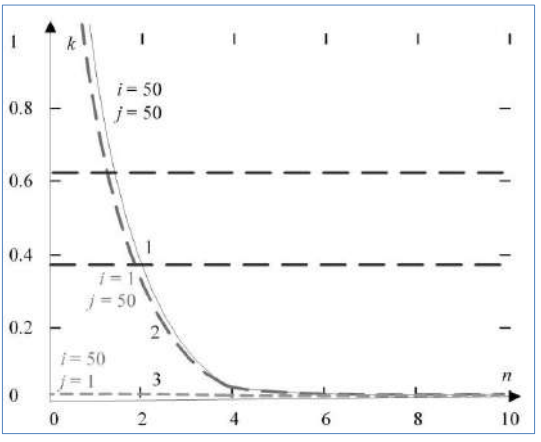


Рис. 3. Зависимости коэффициента покрытия площади экрана информацией от размера экрана визуализирующего устройства
Fig. 3. Dependences of the information coverage coefficient of the screen area of the screen size of the visualizing device

Значения линейных размеров экрана H_X и H_Y для рассмотренных вариантов $n = 1, \dots, 10$ приведены в табл. 1.

Табл. 1. Размеры экрана визуализирующего устройства
Table 1. Screen sizes of the visualizing device

Вариант, *n*	1	2	3	4	5	6	7	8	9	10
H_X, м	0,2	0,4	0,6	0,8	1,0	1,2	1,4	1,6	1,8	2,0
H_Y, м	0,4	0,8	1,2	1,6	2,0	2,4	2,8	3,2	3,6	4,0

Полагалось, что площадь изображения каждого объекта $S_{0i} = 2$ мм^2, а площадь изображения каждого блока метаданных $S_{\text{МД}i}$=25 см^2 (размер блока метаданных 5х5 см). Кривая 1 соответствует $I = 50$, J=50; кривая 2: $I = 1$, $J = 50$; кривая 3: $I = 50$, $J = 1$, штриховыми линиями показаны уровни, соответствующие значениям коэффициента покрытия площади экрана информацией $k = 0{,}382$ и $k = 0{,}618$.

Доминирующее влияние на значения коэффициента k оказывает число блоков метаданных вследствие достаточно большого размера площади блоков по сравнению с размером площади объектов. В результате значения коэффициента k в случаях, когда $I = 50$, J=50 и $I = 1$, $J = 50$, отличаются незначительно. С увеличением площади экрана $S_{\text{Э}} = H_X H_Y$ величина коэффициента k уменьшается, что свидетельствует о нецелесообразности увеличения размеров экрана, на которых отображаемые объекты и блоки метаданных занимают небольшую площадь. В рассмотренном случае при малом объеме метаданных ($I = 50$, $J = 1$) значения коэффициента k невелики и существенно меньше оптимального значения $k = 0{,}382$, соответствующего «золотому» сечению, а при большом объеме метаданных ($I = 50$, J=50 и $I = 1$, $J = 50$) значения коэффициента k, близкие к оптимальному значению $k = 0{,}382$, имеют место для варианта n=2 ($S_{\text{Э}} =$ 0,4х0,8 м), близкого к параметрам экрана монитора ПЭВМ. В то же время для варианта $n = 1$ ($S_{\text{Э}} = 0{,}2$х0,4 м), близкого к параметрам экрана портативного ноутбука, при большом объеме метаданных имеет место $k > 0{,}618$, что соответствует условиям, когда комплексное восприятие видеоданных затруднено.

Таким образом, с точки зрения наглядности визуализации данных и комплексного восприятия видеоданных во всех рассмотренных случаях нецелесообразно использование визуализационных экранов больших размеров, которым соответствуют варианты $n = 5, \dots, 10$. Они могут использоваться лишь как средство отображения информации для коллективного пользования. При большом объеме метаданных также нецелесообразно использование портативных ноутбуков с экранами малых размеров, которым соответствует вариант $n = 1$, поскольку при этом существенно затрудняется восприятие видеоданных. Следует отметить, что при анализе не учитывались форматы (виды) отображения изображений на экране отображающего устройства, характеризуемые соотношением числа пикселов по каждому линейному размеру экрана [19]. При задании формата отображения изображения и размера пиксела визуализирующего устройства достижимые значения коэффициента покрытия площади экрана информацией k могут быть уточнены, однако общие рекомендации о выборе размера экрана и структуры отображаемых объектов и метаданных следуют уже из результатов проведенных исследований.

6. Заключение

Для оценки наглядности визуализации данных (дополненная реальность в информационных системах) предложен критерий, использующий свойства «золотого» сечения и характеризующий возможности комплексного восприятия данных, отображаемых на экране визуализирующего устройства. Связанные с «золотым» сечением закономерности зрительного восприятия изображений, являясь

субъективными, находят многочисленные проявления как в искусстве так и точных науках, и их целесообразно использовать при визуализации данных.

В качестве показателя наглядности визуализации определен коэффициент покрытия площади экрана информацией, значение которого в соответствии с выведенным критерием целесообразно выбирать близким к 0,382, что соответствует математическому определению «золотого» сечения. Практически следует задаваться диапазоном значений коэффициента как в виде допустимых отклонений в окрестности «золотого» сечения так и в виде значений, при которых комплексное восприятие информации возможно, но уже недостаточно наглядно.

Величина коэффициента покрытия площади экрана информацией зависит от геометрических параметров отображаемых объектов, блоков метаданных и размеров экрана. Условию наилучшей наглядности визуализации соответствует определенный масштаб отображения видеоданных и соответствующая ему структура объектов и метаданных.

Разработан базовый алгоритм выбора масштаба отображения видеоданных по критерию наглядности, использующий свойства «золотого» сечения. На его основе сформирован алгоритм выбора масштаба отображения видеоданных с последовательным анализом слоев отображаемых данных.

Исследованы условия обеспечения наглядности визуализации данных в соответствии с предложенным критерием. Для различного числа отображаемых объектов и метаданных проанализирована наглядность их визуализации на экранах различных размеров. Сделаны выводы и рекомендации по выбору размера экрана визуализирующего устройства и структуры отображаемых объектов и метаданных.

Полученные результаты свидетельствуют о выполнении целевой установки исследования – определении критерия наглядности визуализируемых данных на основе свойств «золотого» сечения и исследования условий ее обеспечения на примере отображения данных на экране монитора и проекционной панели. Полученные результаты целесообразно использовать при проектировании и разработке дополненной реальности в информационных системах, а также при создании и сопровождении программных средств прикладного назначения.

Список литературы / References

[1]. Геоинформатика: в 2-х кн. Под ред. В.С. Тикунова. М., Издательский центр «Академия», 2010 г., кн. 1, 400 стр., кн.2, 432 стр. / Geoinformatics: in 2 vol. Tikunov V.S., ed. M., «Academy», 2010, vol. 1, 400 p., vol. 2., 432 p. (in Russian).

[2]. Журкин И.Г., Шайтура С.В. Геоинформационные системы. М., Кудиц-Пресс, 2009 г., 272 стр. / Zhurkin I.G., Shaitura S.V. Geographic information systems. M., Kudic-Press, 2009, 272 p. (in Russian).

[3]. Шокин Ю.И., Потапов В.П. ГИС сегодня: состояние, перспективы, решения. Вычислительные технологии, № 5, 2015 г., стр. 175–213 / Shokin Y.I., Potapov V.P. GIS today: state, prospects, solutions. Computing technology, № 5, 2015, pp. 175–213 (in Russian).

[4]. Воронин А.В. Результаты анализа перспектив развития геоинформационных систем // Системы высокой доступности, №4, 2017 г., стр. 68–75 / Voronin A.V. Results of analyzing geoinformation systems perspectives. High availability systems, 2017, № 4, pp. 68–75 (in Russian).

[5]. Михайлов А.Ю. Принцип наглядности. От традиции к инновации в обучении. Lambert Academic Publishing, 2012 г., 84 стр. / Mikhailov A.Y. Principles of visibility. From tradition to innovation in education. Lambert Academic Publishing, 2012, 84 p. (in Russian).

[6]. Трофимов Е.А. Эргономика зрительного восприятия. М., Актуальные издательские решения, 2013 г., 192 стр. / Trofimov E.A. Ergonomics of visual perception. M., Actual publishing solutions, 2013, 192 p. (in Russian).

[7]. Marr D. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. MIT press, 2010, 428 p.

[8]. Красильников Н.Н. Цифровая обработка изображений. М., Вузовская книга, 2001 г., 320 стр. / Krasilnikov N.N. Digital image processing. M., University book, 2001, 320 p. (in Russian).

[9]. Лурье И.К. Геоинформационное картографирование. Методы геоинформатики и цифровой обработки космических снимков. М., КДУ, 2008 г., 422 стр. / Lurie I.K. Geographic information mapping. M., KDU, 2008, 422 p. (in Russian).
[10]. Интеллектуальные географические информационные системы для мониторинга морской обстановки. Под ред. Р.М. Юсупова и В.В. Поповича. СПб., Наука, 2013 г., 284 стр. / Intelligent geographic information systems for marine monitoring. Yusupov R.M., Popovich V.V., eds. SPb., Science, 2013, 284 p. (in Russian).
[11]. Матерухин А. В. Проблематика создания ГИС на основе систем управления потоками данных. Геодезия и картография, №4, 2017 г., стр.44–47 / Materukhin A.V. Problems of creating GIS based on data flow control systems. Geodesy and cartography, № 4, 2017, pp. 44–47. (in Russian).
[12]. Воронин А.В., Зацаринный А.А., Ионенков Ю.С. Особенности оценки эффективности геоинформационной системы как элемента ситуационного центра. Системы и средства информатики, № 2, 2018 г., стр. 75–87 / Voronin A.V., Zatsarinny A.A., Ionenkov Y.S. The features of efficiency evaluation of a geoinformation system as element of a situational center. Systems and means of informatics, № 2, 2018, pp. 75–87 (in Russian).
[13]. Воронин А.В., Мальцев Г.Н., Сохен М.Ю. Наглядность визуализации данных в геоинформационной системе при использовании свойств золотого сечения. Информационно-управляющие системы, № 6, 2018 г., стр. 46–57 / Voronin A.V., Maltsev G.N., Sokhen M.Y. Data visualization quality in a geographic information system using golden ratio properties // Information management systems, № 6, 2018, pp. 46–57 (in Russian).
[14]. Гулина Ю.С., Колючкин В.Я. Методика расчета вероятности распознавания изображений человеком-оператором. Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение, № 1, 2012 г., стр. 100–107 / Gulina Y.S., Koluchkin V.J. Methodology for calculating the probability of image recognition by a human-operator // Herald of the Bauman Moscow State Technical University. Series Instrumentation, 2012, № 1, pp. 100–107 (in Russian).
[15]. Короленко П. В., Грушина Н. В. Золотое сечение и самоподобные структуры в оптике. М., УРСС, 2010 г., 136 стр. / Korolenko P.V., Grushina N.V. Golden section and self-similar structures in optics. M., URSS, 2010, 136 p. (in Russian).
[16]. Ковалев Ф.В. Золотое сечение в живописи. М., РИП-Холдинг, 2016 г., 192 стр. / Kovalev F.V. Golden ratio in painting. M., RIP-Holding, 2016, 192 p. (in Russian).
[17]. Косиков А.Г., Ушакова Л.А. Виртуальные геоизображения пространственно-временных моделей окружающей среды. Геодезия и картография, № 5, 2016 г., стр. 43–51 / Kostikov A.G., Ushakova L.A. Virtual geoimages of spatio-temporal models of the environment. Geodesy and cartography, № 5, 2016, pp. 43–51 (in Russian).
[18]. Мироненко А.Н., Радионов В.А. Структура и основные свойства цифровой модели местности с координатной идентификацией топографической информации. Геодезия и картография, № 9, стр. 37–41 / Mironenko A.N., Radionov V.A. Structure and basic properties of a digital terrain model Geodesy and cartography, with coordinate identification of topographic information. Geodesy and cartography, № 9, 2017, pp. 37–41 (in Russian).
[19]. Мальцев Г.Н., Сазонов К.В., Панкратов А.В. Метод обнаружения начальных кадров видеопотока. Вопросы радиоэлектроники. Серия Техника телевидения, вып. 1, 2016 г., стр. 31–37 / Maltsev G.N., Sazonov K.V., Pankratov A.V. Method for detecting the initial frames of a video stream. Issues of radio electronics. Series Television technique, issue 1, 2016, pp. 31–37 (in Russian).

Информация об авторе/ Information about author

Алексей Владимирович ВОРОНИН – кандидат технических наук, доцент. Сфера научных интересов: алгебраический и интеллектуальный анализ, алгоритмизация процессов, нейрокомпьютерные технологии, цифровая обработка сигналов, методы защиты информации, дополненная реальность в геоинформационных системах.

Aleksey Vladimirovich VORONIN – Candidate of Technical Sciences, Associate Professor. Research interests: algebraic and intellectual analysis, process algorithms, neurocomputer technologies, digital signal processing, information protection methods, augmented reality in geographic information systems.

DOI: 10.15514/ISPRAS-2020-32(1)-7

Разработка алгоритма распознавания движений человека методами компьютерного зрения в задаче нормирования рабочего времени

С.Е. Штехин, ORCID: 0000-0003-2866-4864 <sergei.shtekhin@ocrv.ru>
Д.К. Карачёв, ORCID: 0000-0002-1008-2535 <denis.karachev@ocrv.ru>
Ю.А. Иванова, ORCID: 0000-0003-1575-6882 <yustina.ivanova@ocrv.ru>

ОАО Отраслевой центр разработки и внедрения,
Россия, г. Сочи, Триумфальный проезд, д.1

Аннотация. Цель исследования заключается в разработке и тестировании алгоритмов для распознавания по видео людей и инструментов, с которыми они работают в конкретный момент времени. В рамках исследования в качестве базового решения был предложен и реализован алгоритм, состоящий из нескольких этапов: распознавание в видео-кадрах людей и определение координат краевых точек прямоугольника, в котором находится человек; определение в видео кадрах координат ключевых точек обнаруженных людей; распознавание в видео-кадрах инструментов и определение координат их краевых точек; определение инструментов, с которыми человек работает в конкретный момент времени (время считается по номеру кадра из видео). Для реализации алгоритма было проведено исследование, в ходе которого было протестировано дообучение существующих моделей компьютерного зрения для следующих задач компьютерного зрения: детекция объектов (Object detection) и людей, в частности, определение ключевых точек людей (Pose estimation), наложение объектов (Object Overlaying). В качестве метрики для мультиклассификационной задачи определения инструментов, которые находятся в руках у человека в каждом кадре (Object Overlaying), использовались следующие показатели: точность, чувствительность и f1-мера. Алгоритм запущен на web-сервисе и протестирован специалистами.

Ключевые слова: нейронные сети; компьютерное зрение; распознавание движений человека; машинное зрение; машинное обучение; наложение объектов; детектирование объектов

Для цитирования: Штехин С.Е., Карачёв Д.К., Иванова Ю.К. Разработка алгоритма распознавания движений человека методами компьютерного зрения в задаче нормирования рабочего времени. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 121-136. DOI: 10.15514/ISPRAS–2020–32(1)–7

Благодарности: Авторы благодарны ОАО РЖД.

Computer vision system for Working time estimation by Human Activities detection in video frames

S.E. Shtekhin, ORCID: 0000-0003-2866-4864 <ivanov@ispras.ru>
D.K. Karachev, ORCID: 0000-0002-1008-2535 <denis.karachev@ocrv.ru>
Yu.A. Ivanova, ORCID: 0000-0003-1575-6882 <yustina.ivanova@ocrv.ru>
Industry Center for Information Systems' Development and Deployment, 1, Triumphalny, Sochi, 109004, Russia

Abstract. The goal of the research is to develop and to test methods for detecting people, parametric points for their hands and their current working tools in the video frames. The following algorithms are implemented: humans bounding boxes coordinates detection in video frames; human pose estimation: parametric points detection for each person in video frames; detection of the bounding boxes coordinates of the defined tools in video frames; estimation of which instrument the person is using at the certain moment. To implement algorithms, the existing computer vision models are used for the following tasks: Object detection, Pose estimation, Object overlaying. Machine learning system for working time detection based on computer vision is developed and deployed as a web-service. Recall, precision and f1-score are used as a metric for multi-classification problem. This problem is defined as what type of tool the person uses in a certain frame of video (Object Overlaying). Problem solution for action detection for the railway industry is new in terms of work activity estimation from video and working time optimization (based on human action detection). As the videos are recorded with a certain positioning of cameras and a certain light, the system has some limitations on how video should be filmed. Another limitation is the number of working tools (pliers, wrench, hammer, chisel). Further developments of the work might be connected with the algorithms for 3D modeling, modeling the activity as a sequence of frames (RNN, LSTM models), Action Detection model development, time optimization for the working process, recommendation system for working process from video activity detection.

Keywords: neural networks; computer vision; pose estimation; computer vision; machine learning; object overlaying; object detection; work optimization.

For citation: Shtekhin S.E., Karachev D.K., Ivanova Yu.A. Computer vision system for working time estimation by human activities detection in video frames. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 121-136 (in Russian). DOI: 10.15514/ISPRAS-2020-32(1)-7

Acknowledgements: The authors are graceful to Russian Railway Company.

1. Введение

Для изучения и распространения передовых методов труда, затрат рабочего времени и установления нормативных величин, а также при исследовании трудовых процессов с быстрыми движениями и малыми отрезками времени, которые трудно или невозможно охватить методом визуальных наблюдений, используется видеонаблюдение [1].

В среднем, в год по всей сети железных дорог инженерами по организации и нормированию труда структурных предприятий функциональных филиалов ОАО «РЖД» пересматривается или разрабатывается вновь порядка 837 производственных процессов (порядка 18-22 нормативных сборников норм времени), на которые должна быть проведена видеосъемка. Видеосъемка рабочего времени является методом исследования производственных процессов, трудовых операций и фактических затрат рабочего времени. Этот метод не только обеспечивает высокую точность измерения всех фактических затрат рабочего времени, любых трудовых операций, движений, действий, но и позволяет фиксировать и демонстрировать их содержание.

Отличие видеосъемки от классических методов исследования рабочего времени заключается в том, что процесс замеров времени и анализ полученных результатов отделены друг от друга. Рабочая операция анализируется после съемки. С помощью видео можно наблюдать не только за ходом выполнения рабочего задания, но и за

используемыми средствами производства и материалами, приемами труда, рабочим местом. Результаты видеосъемки служат основой для проектирования рациональных трудовых процессов, нормативов на подготовительно-заключительные действия, обслуживание рабочего места, регламентированных перерывов для отдыха и питания, уточнения (проверки) или разработки норм времени, получения данных для проведения специальной оценки условий труда. Видеосъемка позволяет проводить обучение работников предприятий железнодорожной отрасли передовым приемам и методам труда.

Прогнозируется в среднем в год более 100 тысяч видеосъемок, с учетом проведения 3-х замеров по каждому производственному процессу со всех железных дорог и с учетом того, что процессом видеосъемки будет охвачено только 25-30% всех работ.

В связи с таким большим количеством видеосъемок в год, является актуальной задача автоматической разметки и анализа видео средствами машинного обучения (компьютерного зрения). На первом этапе были выделены технологические операции, в которых производится работа сотрудника с инструментами. Для таких технологических операций задача была поставлена следующим образом: «Произвести автоматическую разметку видеосъемки, определяя алгоритмами компьютерного зрения кадры, на которых сотрудник, выполняющий определенные работы, держит конкретные виды инструментов в руках».

Задача настоящего исследования относится к задаче распознавания движений человека (Human Action Detection). Задача решалась на данных по железнодорожной тематике, которая заключается в определении события, происходящего с человеком на обрабатываемом кадре (изображении). Всего использовалось 10 видео длительностью около 10 минут, 25 кадров на каждую секунду времени. Размер видео 1920 px & 1040 px.

2. Распознавание в видео кадре людей и определение их координат

Для исследования алгоритмов детекции людей проведено тестирование Object Detection на датасете COCO [2]. В основном, модели детекции людей показывают высокую точность (более 80%). Наибольшую точность показывает RetinaNet (0.99).

Первым этапом в рамках данной задачи является определение наличия сотрудника в видео кадре и его координаты. В дальнейшем, если сотрудник определен в видео кадре, то данный кадр передается на следующие этапы алгоритма, если сотрудника в кадре нет, то кадр дальше не обрабатывается. Это условие показывает важность точного определения наличия человека в кадре для следующего этапа – поиск ключевых точек сотрудника.

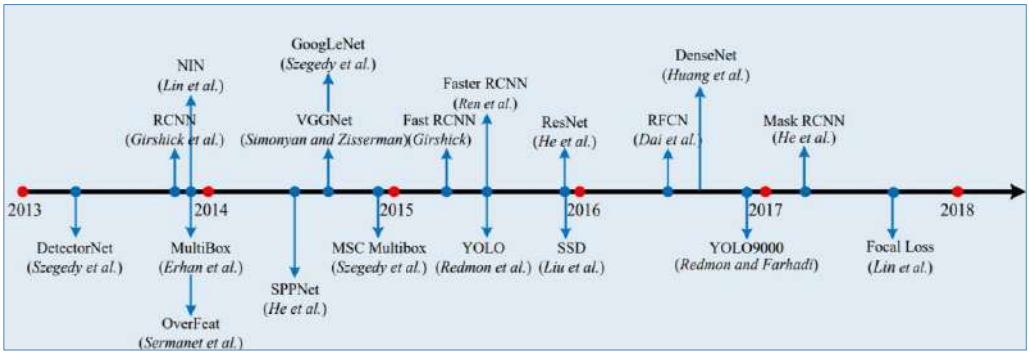


Рис. 1. Основные вехи развития моделей сверточных нейронных сетей, решающих задачу распознавания объектов [3]

Fig. 1. The main milestones of the development of convolutional neural network models that solve the problem of object recognition [3]

Данная задача относится к следующему классу задач компьютерного зрения – распознавание объектов (object detection). **Object detection** – обнаружение всех объектов указанных классов и определение охватывающей рамки (bounding box) для каждого из них. Современные методы решения задач данного класса представляют собой глубокие нейронные сети (Deep Learning). Основные вехи развития моделей сверточных нейронных сетей, решающих задачу распознавания объектов показаны на рис. 1.

Табл. 1. Сравнение современных сверточных нейросетей для методов Object Detection на COCO датасете [12]. Были проведены измерения при различных коэффициентах Жаккара (AP50 — при коэффициенте 50%, AP75 - при коэффициенте 75%, APS — для объектов площадью менее 32 квадратных пикселя, APM — для объектов площадью более 32, но менее 96 квадратных пикселя, APL — для объектов большей площади)

Table 1. Comparison of modern convolutional neural networks for Object Detection methods on COCO dataset [12]. Measurements were taken at various Jacquard coefficients (AP50 - at a coefficient of 50%, AP75 - at a coefficient of 75%, APS - for objects with an area of less than 32 square pixels, APM - for objects with an area of more than 32 but less than 96 square pixels, APL - for objects larger area)

	backbone	*AP*	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [4]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FIN [5]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [14]	34.7	55,5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [7]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [8]	DarkNet-19 [8]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9-10]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD523 [10]	ResNet-101-SSD DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [11]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [11]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608x608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9
COCO for YO:Ov3							

Были рассмотрены двухуровневые (Faster R-CNN+++ [4], Faster R-CNN w FPN [5], Faster R-CNN by G-RMI [6], Faster R-CNN w TDM [7] и одноуровневые модели сверточных нейронных сетей (YOLOv2 [8], SSD513 [9-10], DSSD513 [10], RetinaNet [11], YOLOv3

[12]). Результаты сравнения работы этих моделей показаны на датасете COCO (табл. 1). В качестве метрики для измерения точности использовалась Mean Average Precision – средняя точность детекции всех объектов [13], которая вычисляется как среднее значение детекций для всех классов объектов. Для каждого класса вычисляется точность предсказания (Average Precision – AP). Данная метрика связана с подсчетом коэффициента Жаккара (intersection over union), где для найденного объекта подсчитывается площадь совпадающей ограничивающей рамки.

Из результатов исследования видно, что максимальная точность показана моделью RetinaNet.

Был подготовлен датасет из 1000 кадров из видеофайлов, предоставленных РЖД, из которых 500 изображений с человеком и 500 изображений без человека (в основном, эти изображения содержат железную дорогу, рельсы и поезд). На данном датасете были исследованы вышеупомянутые модели. В результате данного исследования была выбрана модель RetinaNet для определения людей на кадре. В табл. 2 показаны результаты метрик для отобранной модели при решении классификационной задачи нахождения людей.

Среднее время обработки моделью RetinaNet одного кадра 0,076 с, что также в среднем быстрее чем другие модели.

В датасете COCO существует категория «person» (человек), поэтому предобученная модель подходит для задачи нахождения человека в кадре. Модель была реализована с помощью библиотек Keras и Tensorflow (python3.6).

Результатом данного этапа является json-файлы для каждого кадра видео, в которых имеется информация о наличии или отсутствии людей в кадре и координаты ограничивающей их рамки при их присутствии.

Табл. 2. Результаты предсказания модели RetinaNet на датасете из 1000 кадров. Класс 0 – изображение не содержит в себе человека. Класс 1 – изображение содержит в себе человека. Всего было отобрано 1000 изображений, 500 из них – с человеком, 500 – без. Support – количество людей в каждом классе. Precision, recall, f1-score — точность, полнота, f1-мера для каждого из классов.

Tab. 2. Results of the RetinaNet model predicting on a dataset of 1000 frames. Class 0 – the image does not contain a person. Class 1 – the image contains a person. In total, 1000 images were selected, 500 of them with a person, 500 without. Support – the number of people in each class.

	Precision	**Recall**	**F1-score**	**Support**
0	0.96	0.99	098	500
1	0.99	0.96	0.98	500
accuracy				1000
macro avg	0.98	0.98	0.98	1000
weighted avg	0.98	098	0.98	1000

3. Определение в видеокадре ключевых точек людей

Следующим этапом решения задачи является определение координат рук, локтей и плеч сотрудников.

Эта задача относится к следующему классу задач машинного зрения: определение ключевых параметрических точек человека и оценка позы человека по изображению (Pose estimation).

В рамках исследования были рассмотрены следующие модели, обученные на COCO датасете: PAFs [15], OpenVino [16], CPN [17], Simple [18]. Также был исследован метод постобработки PoseFix [19] на всех моделях.

Модели исследовались на датасете из 500 кадров, содержащих изображения людей, на которых вручную были размечены ключевые точки (всего размечено 6 точек для каждого человека, по две точки на кисти, локти, плечи). Модели сравнивались по двум параметрам: точность и время обработки. Точность оценивалась по метрике Object Keypoint Similarity (OKS) [2].

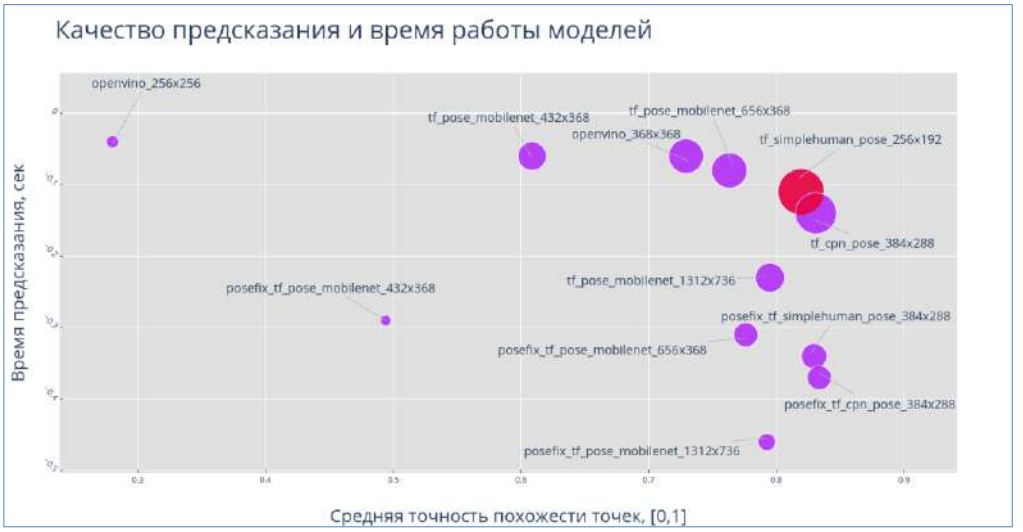


Рис 2. Сравнение моделей Pose Estimation по метрике OKS (средняя точность похожести точек) и времени обработки кадра. Модель tf_simplehuman_pose_256x192 выбрана в качестве оптимальной по детекции и по времени по сравнению с другими моделями

Fig 2. Comparison of Pose Estimation models by the OKS metric (average accuracy of similarity of points) and frame processing time. The tf_simplehuman_pose_256x192 model was selected as optimal in detection and in time compared to other models

Результаты сравнения моделей представлены на рис. 2, названия моделей показаны как tf_pose_*, openvino_*, tf_cpn_pose* и tf_simplehumanpose_* с различными входными данными соответственно (цифры после названия означают размер исходного изображения). По результатам сравнения была выбрана tf_simplehuman_pose_256_192, которая показала время обработки одного кадра 0,11с и среднюю метрику OKS = 0,82.

Модель была реализована с помощью библиотеки Tensorflow. На вход в данную модель подается кадр видео и координаты сотрудника, полученные на предыдущем этапе.

Результатом данного алгоритма является json-файлы, в которых хранятся координаты 6 ключевых точек (или меньше, если алгоритм не нашел всех точек) – плеча, локтя и кисти (левой и правой руки соответственно) для каждого кадра видео, на которых есть человек.

4. Распознавание в видеокадре инструментов и определение координат их ограничивающей рамки (Object Detection)

Следующим этапом решения задачи является определение координат инструментов. Данная задача относится к проблеме распознавания объектов (object detection). Для её решения выбрана модель RetinaNet (она же была использована на первом этапе). Так как в датасете COCO нет категорий инструментов, используемых в РЖД, для дообучения

нейросети была необходима подготовка датасета с изображениями, на которых были размечены инструменты (пример разметки инструментов показан на рис. 3).

Рис. 3. Исходный кадр видео с отмеченными инструментами. Каждый инструмент в кадре помечен соответствующими координатами ограничивающей его рамки
Fig. 3. The original frame of the video with the marked tools. Each tool in the frame is marked with the corresponding coordinates of the bounding box

Первоначально из четырёх видео были получены кадры, в которых содержатся инструменты (часть кадров показана на рис. 4). Видео были отобраны таким образом, что человек может различить инструмент (хорошее освещение, и съемка не далее, чем с расстояния 7 метров).

Рис. 4. Датасет №1, собранный из 4-ех видео. Примеры некоторых кадров
Fig. 4. Dataset number 1, assembled from 4 ex video. Examples of some frames

Был создан первый тестовый датасет для обучения: в датасете №1 содержатся все инструменты каждого кадра без аугментации, всего 25198 изображений молотков, 21805 изображений инструмента зубило, 133663 изображений инструмента ключ, 48551 изображений инструмента плоскогубцы. В качестве тестовых данных было использовано одно из видео «Rakurs4», которое содержит 10450 изображений инструмента зубило, 9461 изображений инструмента молоток, 10179 изображений инструмента плоскогубцы, 16798 изображений инструмента ключ.

Была обучена нейросеть RetinaNet [8] в течение 20 эпох (batch size=4). Нейросеть реализована с помощью библиотеки Tensorflow.

В результате дообучения нейросети были получены результаты точности детекции, представленные в табл. 3.

Табл. 3. Средняя точность детекции инструментов на тестовой выборке, созданной из видео «Rakurs4», нейросетью, обученной на датасете 1
Tab. 3. The average accuracy of detection of instruments on a test sample created from the video "Rakurs4", a neural network trained on dataset 1

	Средняя точность детекции объектов (инструментов)				
	Зубило	Молоток	Плоскогубцы	Ключ	Все инструменты
Датасэт 1	0.9788	0.1245	0.8034	0.5412	0.6120

5. Эксперименты с аугментацией

Было произведено несколько экспериментов для улучшения детекции инструментов нейросетью. Исходные данные (картинки с объектами инструментов: зубило, плоскогубцы, молоток, ключ, полученные из датасета №1) были увеличены с помощью различных аугментаций, и полученные изображения использовались в качестве тренировочных данных для обучения нейросети RetinaNet [9]. Исследователи доказали, что увеличение изображений в тренировочном датасете с помощью различных аугментаций приводит к увеличению качества предсказания нейросети [20-21], проводимые эксперименты также показали улучшение детекции объектов на тестовых данных.

Датасет №1

Описан в предыдущем разделе (раздел 4).

Датасет №2

Было увеличено количество объектов с поворотами: все исходные изображения (датасет №1) были повернуты произвольно на угол от -30 до 30 градусов. 20 эпох обучения.

Датасет №3

К исходному датасету (датасету №1) было применено масштабирование в непропорциональном соотношении (случайно от 0 до 30% от исходной длины и ширины изображения).

Датасет №4

Исходные изображения (датасет №1) были отражены по горизонтали.

Датасет №5

Объединены датасеты №1, №2, №3 и №4 и создан датасет №5.

Датасеты №6, №7, №8

Изображения из датасетов №2, №3 и №4 были перемешаны в случайном порядке

Датасет №9

Решено применить масштабирования в большем интервале совместно со смещениями (в пределах 20% по горизонтали и по вертикали),

Датасет №10

В десятом эксперименте к исходным данным (датасет №1) были применены повороты (случайное значение от -30 до 30 градусов), масштабирование (пропорционально по длине и ширине в случайном значении от 0.2 до 1.2) и смещение (случайное значение в пределах 20% относительно длины и ширины).

Датасет №11

Так как количество элементов в каждом классе инструмента неуравновешенно (несбалансированные данные), была выдвинута гипотеза, что применение аугментации для выравнивания количества классов может привести к улучшению качества детекции. Была удалена часть инструментов (ключ) и добавлен за счет аугментаций инструмент молоток. После балансировки количество изображений для тренировки нейросети увеличилось и пропорции классов выровнялись (рис. 5).

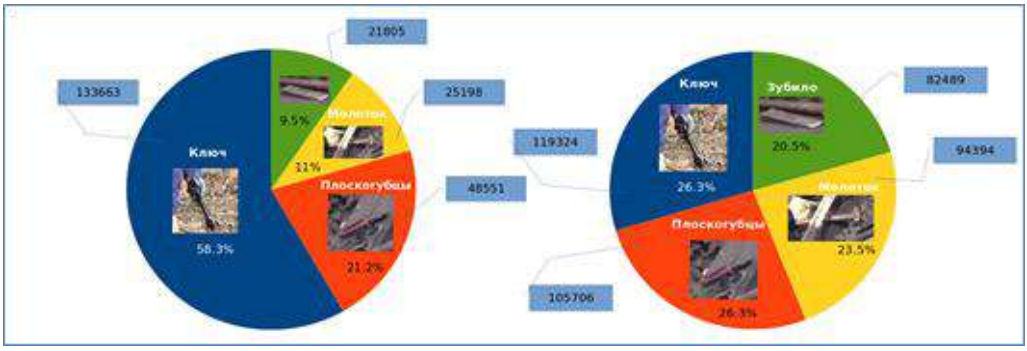


Рис. 5. Первоначальная пропорция количества объектов в датасете №1 (слева) и аугментированного датасета №11 (справа)
Fig. 5. The initial proportion of the number of objects in dataset number 1 (left) and augmented dataset number 11 (right)

Датасет №12

Было увеличено количество ключей (за счет добавления тех изображений, которые были удалены в датасете №11).

Датасет №13

Был составлен датасет №13 из 10 размеченных видео с аугментацией (датасет №1 – часть этих 10 видео).

Датасет №14

Для улучшения точности детекции нейросети было решено использовать дополнительные изображения из открытых источников. Был найден датасет KTH Handool Dataset [22-23], который представляет собой изображения для трех инструментов – молоток, плоскогубцы, отвертка – на разном фоне, под различным светом и сделанными под разными положениями камеры. Всего в датасете KTH Handtool 4500 изображений, по 1500 изображений на каждый объект. Для каждого изображения имеется xml-файл, где указано название объекта и краевые точки прямоугольника, в котором находится объект в изображении.

Датасет №14 состоит из изображений, полученных из датасета №13 и датасета KTH Handool Dataset. Так как некоторые инструменты из датасета KTH Handtool похожи на инструменты, которые детектируются в видео РЖД, была выдвинута гипотеза, что при добавлении дополнительных данных будет улучшение детекции инструментов.

На рис. 6 приводятся изображения плоскогубцев, используемых в компании РЖД и представленных в данных KTH Handtool.

На основе результата тренировки нейросети сделан вывод, что добавление дополнительных изображений для тренировки нейросети может привести к улучшению детекции, если инструменты добавляемого датасета схожи с инструментами РЖД.

Рис. 6. Плоскогубцы. Сверху – инструменты из датасета KTH Tool, внизу – инструменты, полученные из видео РЖД. Можно увидеть сходство инструментов
Fig. 6. Pliers. At the top are the tools from the KTH Tool dataset, at the bottom are the tools obtained from the Russian Railways video. You can see the similarity of tools

Все результаты приведены в табл. 4. По результатам проведенных экспериментов можно сделать следующие выводы. Использование аугментации при обучении нейросети RetinaNet улучшает детекцию объектов, но необходимо также делать балансировку объектов в каждом классе. Самыми эффективными преобразованиями являются повороты в случайном порядке от 0 до 360 градусов и масштабирование в интервале от 20% до 120% от исходного размера.

Табл. 4. Средняя точность детекции инструментов нейросетью на тестовой выборке, полученной из видео «Rakurs4»
Table 4. The average accuracy of detection of instruments by a neural network in a test sample obtained from the video «Rakurs4»

	Средняя точность детекции объектов				
	Зубило	**Молоток**	**Плоскогубцы**	**Ключ**	**Все инструменты**
Датасет №1	0.9788	0.1245	0.8034	0.5412	0.6120
Датасет №2	0.9765	0.0918	0.8880	0.6169	0.6433
Датасет №3	0.9725	0.1679	0.9461	0.6140	0.6751
Датасет №4	0.9163	0.3747	0.9296	0.5659	0.6966
Датасет №5	0.9062	0.1106	0.9554	0.5944	0.6417
Датасет №6	0.9978	0.3590	0.9475	0.6951	0.7498
Датасет №7	0.9961	0.4379	0.9314	0.7626	0.7820
Датасет №8	0.9532	0.0333	0.6913	0.5339	0.5529
Датасет №9	0.9987	0.4714	0.7768	0.7509	0.7495
Датасет №10	0.9992	0.4644	0.9471	0.6028	0.7534
Датасет №11	0.9981	0.4899	0.9621	0.4471	0.7200
Датасет №12, 24 эпохи	0.9963	0.4525	0.9491	0.5983	0.7490
Датасет №12, 40 эпох	0.9861	0.4380	0.9594	0.6252	0.7522
Датасет №13	0.9184	0.0618	0.8011	0.7642	0.6364
Датасет №14	0.9414	0.0618	0.9612	0.8280	0.6998

Можно заметить, что датасет №13 и датасет №14 дают низкие показатели детектирования для инструмента молоток, это связано с несбалансированностью данных датасетов. Датасет KTH Handtool Dataset имеет потенциал для улучшения алгоритма детекции, так как детекция плоскогубцев улучшилась на 16% при добавлении данного датасета в обучающую выборку (датасеты №13 и №14 для плоскогубцев в табл. 4).

Результатом данного этапа является json-файл, в котором хранится список распознанных инструментов с их координатами для каждого кадра видео.

6. Определение инструментов, которые находятся в руках у человека

Последним этапом в рамках данной задачи является определение инструментов, которые находятся в руках человека. Для этого используются все объединенные результаты предыдущих алгоритмов для каждого кадра видео. Сложность обусловлена тем, что на изображении находятся несколько различных инструментов, могут присутствовать несколько людей, и при расчёте расстояний до объектов необходимо учитывать их масштаб и возможные комбинации инструмент-человек. Кроме того, нельзя не учитывать тот факт, что с некоторых ракурсов инструменты могут не определяться, так как могут быть перекрыты другими объектами (рельсами, самим человеком и т.д.).

Метод заключается в построении модели, которая предскажет, находится ли данный инструмент в руках у человека или нет. Каждый инструмент проверяется для каждого человека; таким образом, алгоритм работает даже в случае, когда в кадре присутствует большое количество инструментов и людей, что говорит о гибкости данного подхода.

Данный этап является интеграционным, так как производится расчёт на основе результатов предсказаний моделей на предыдущих этапах. Предсказания на данном этапе происходят на основе подсчета расстояний между точками:

- координаты точки кисти;
- координаты центра инструментов.

Однако для обучения модели этих данных недостаточно, так как видео имеет различный масштаб и, соответственно, расстояние на различных видео между точками для находящегося в руке инструмента будет сильно отличаться. Для решения данной проблемы были созданы дополнительные признаки, такие как отношение размера инструмента к расстоянию между точками (от плеча до локтя, от локтя до кисти) каждой руки, а также квадраты и логарифмы этих значений.

При обучении модели важную роль играет баланс классов. Сгенерированные тренировочные данные с видео показаны в табл. 5.

Табл. 5. Сгенерированные тренировочные данные для обучения классификационной модели детектирования инструмента в руке человека (класс 1) или не в руке (класс 0)
Tab. 5. Generated data for training the classification model of detecting an instrument in a person's hand (class 1) or not in a hand (class 0)

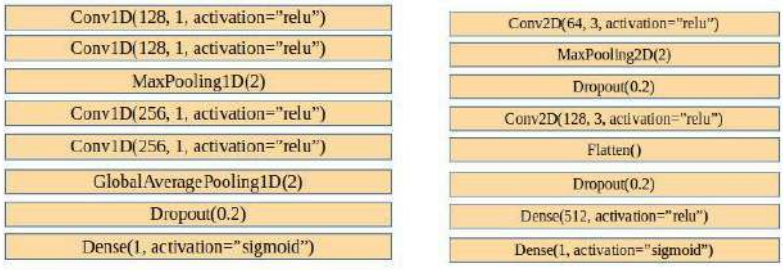
Инструмент	В руке (1)	Не в руке (0)
Плоскогубцы	2892	24221
Молоток	862	10752
Гаечный ключ	24568	40621

Исходя из таблицы 5, можно сделать вывод о том, что нужна балансировка классов. Сделать ее можно двумя способами:

- удалить лишние нули и привести количество всех классов к одному;

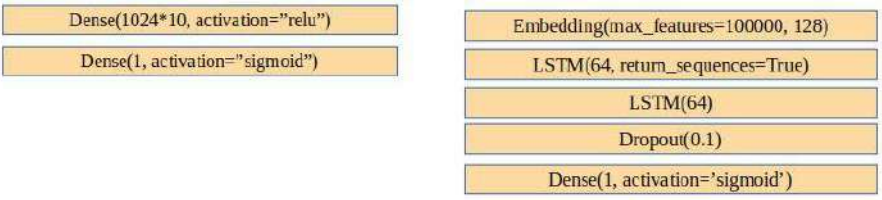
– случайным образом продублировать значения классов, примеров которых недостаточно, до определенного количества и удалить лишние примеры классов, число значений которых велико.

В процессе исследования были оценены различные типы нейронных сетей (рис. 7 и 8): полносвязные, свёрточные (с 1D и 2D свёртками), сети с кратковременной памятью и LSTM (Long short-term memory). Кроме того, дополнительно был протестирован подход с дообучением для сети ResNet50 [24]. Результат предсказания получается в результате сравнения ответа модели с порогом (если значение ниже порога, значит инструмент в руках, иначе – не в руках), который подбирается эмпирически.



a)1D свертка b) Нейросеть с 2D свертками

Рис. 7. Архитектура моделей с 1D свертками (a) и 2D свертками (b)
Fig. 7. Architecture of models with 1D convolution (a) and 2D convolution (b)



a) Полносвязная нейросеть b) Нейросеть с LSTM слоями

Рис. 8. Архитектура модели полносвязной нейросети (a) и нейросети с LSTM слоями (b)
Fig. 8. Architecture of a fully connected neural network model (a) and a network with LSTM layers (b)

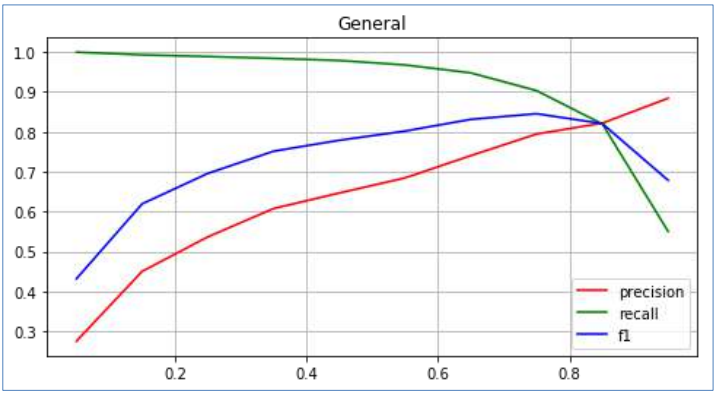


Рис. 9. Предсказания сети с 1D свёртками на тестовых данных. Показаны зависимости точности (precision), чувствительности (recall) и их гармоническое среднее (f1) от порога
Fig. 9. Predictions of a network with 1D convolutions on test data. Shown are the dependences of accuracy (sensitivity), sensitivity (recall) and their harmonic mean (f1) on the threshold

В результате сравнения, наилучшим образом себя показала модель с 1D свёртками [25], архитектура которой представлена на рис. 7; на рис. 9 представлены показаны ее точность, чувствительность и среднее гармоническое в зависимости от порога. Основные параметры для тренировки сети: оптимизатор – Adadelta [26], размер батча (выборки) – 500, количество эпох – от 5 до 10.

Для всех типов нейронных сетей, кроме полносвязной, данные группировались таким образом, что получалась одна таблица (фрейм) для предсказания, в которой каждая строчка – это информация об инструменте в определенный момент времени (каждый момент времени представлен кадром видео, в 1 секунде 45 кадров видео), таким образом, учитывается временная составляющая.

В табл. 6 представлены результаты при наивысшем f1 – score, при пороге равном 0.7, выше которого считается, что объект найден.

Табл. 6. Общие результаты лучшей модели с 1D свёртками
Tab. 6. General results of the best model with 1D convolution

Класс	**Точность**	**Полнота**	**f1-score**
0	0.97	0.94	0.96
1	0.79	0.90	0.85

Табл. 7. Агрегирующая таблица результатов для классов 1
Tab. 7. Aggregate results table for grades 1

Модель	**Точность**	**Полнота**	**f1-score**
Нейросеть с 1D свертками [27]	0.79	0.90	0.85
Нейросеть с 2D свертками [27]	0.66	0.95	0.78
Полносвязная нейросеть [28]	0.25	0.81	0.39
Нейросеть с LSTM слоями [29]	0.25	0.81	0.39

Результатом данного этапа и всей задачи в целом является json-файлы – список сотрудников с их координатами и для каждого сотрудника список инструментов, которые находятся у них в руках для каждого кадра видео (если модель задетектировала инструмент) с координатами.

7. Заключение

Описанные алгоритмы показали высокую точность детекции инструментов в руках человека, благодаря чему могут быть использованы для дальнейшей разработки. Задача детектирования действий человека по видео является не новой для науки, но методы, предложенные в данной статье, опираются на новейшие разработки в области компьютерного зрения, с применением методов, не использованных ранее. Данное исследование может быть использовано для оптимизации труда (определение норм времени по видео), а также в качестве рекомендательных показаний сотрудникам РЖД о том, сколько времени необходимо на выполнение операции, и оптимизации рабочего труда по видео.

Предполагается вести дальнейшие исследования по следующим направлениям.

1. Детектирование людей и их действий по видео:
   1.1. настройка и постобработка RetinaNet.
2. Определение ключевых точек:
   2.1. модель 3D pose estimation;
3. Определение инструментов:
   3.1. дообучение моделей с разными ракурсами;
   3.2. исследование различных аугментаций;
   3.3. детектирование новых инструментов.
4. Определение человека с инструментом:
   4.1. исследование различных атрибутов для классификационной модели;
   4.2. исследование моделей, работающих с последовательностями кадров (RNN, LSTM) для детектирования действий человека по видео.
5. Исследование моделей Action Detection.
6. Оптимизация рабочего труда:
   6.1. отбор видео, в которых сотрудник затрачивает минимальное количество времени на выполнение операции;
   6.2. дальнейшее использование данного видео в качестве рекомендации другим сотрудникам для улучшения производительности.

Список литературы / References

[1]. Методические рекомендации по изучению затрат рабочего времени в структурных подразделениях ОАО «РЖД». Утверждены распоряжением ОАО «РЖД» от 10 апреля 2018 / Guidelines for the study of the costs of working time in structural divisions of JSC Russian Railways. Approved by the order of Russian Railways on April 10, 2018

[2]. Keypoint evaluation metrics used by COCO. Available at: http://cocodataset.org/#keypoints-eval, accessed 05.01.2020.

[3]. Andrea Gaetano Tramontano. Deep Learning Networks for Real-time Object Detection on Mobile Devices. Master's Degree Thesis, University of Padova, Italy, 2018/2019.

[4]. C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. arXiv:1701.06659, 2017.

[5]. J. Huang, V. Rathod et al. Speed/accuracy trade-offs for modern convolutional object detectors. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 3296-3297.

[6]. D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. Iqa: Visual question answering in interactive environments. arXiv:1712.03316, 2017.

[7]. O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2121–2131.

[8]. J. Parham, J. Crall, C. Stewart, T. Berger-Wolf, and D. Rubenstein. Animal population censusing at scale with citizen science and photographic identification. In Proc. of the AAAI 2017 Spring Symposium on Artificial Intelligence for the Social Good, 2017, pp. 37-44.

[9]. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. ´ Focal loss for dense object detection. arXiv:1708.02002, 2017.

[10]. M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. International Journal of Computer Vision, vol. 88, no. 2, 2010, pp. 303– 338.

[11]. I. Krasin, T. Duerig et al. Openimages: A public dataset for large-scale multi-label and multi-class image classification, 2017. Available at: https://github.com/openimages, accessed 05.01.2020.

[12]. Joseph Redmon, Ali Farhadi: YOLOv3: An Incremental Improvement. arXiv:1804.02767, 2018

[13]. Jonathan Hui. mAP (mean Average Precision) for Object Detection. Available at: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173, accessed 05.01.2020.

[14]. M. Scott. Smart camera gimbal bot scanlime:027, Dec 2017. 4

[15]. Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multiperson 2d pose estimation using part affinity fields. CVPR, 2017.
[16]. D. Osokin, Real-time 2d multi-person pose estimation on CPU: Lightweight OpenPose arXiv:1811.12004
[17]. Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun. Cascaded pyramid network for multi-person pose estimation. In Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7103-7112.
[18]. Xiao, Bin, Haiping Wu, and Yichen Wei. Simple Baselines for Human Pose Estimation and Tracking. Lecture Notes in Computer Science, vol. 11210, 2018, pp. 472-487.
[19]. G. Moon, J.Y. Chang and K.M. Lee. PoseFix: Model-Agnostic General Human Pose Refinement Network. In Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 7765-7773.
[20]. Arun Gandhi. Data Augmentation. How to use Deep Learning when you have Limited Data – Part 2. Available at: https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/, accessed 05.01.2020.
[21]. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM, vol. 60, no. 6, 2017, pp. 84-90.
[22]. Massimiliano Mancini, Hakan Karaoguz, Elisa Ricci, Patric Jensfelt, Barbara Caputo. Kitting in the Wild through Online Domain Adaptation. arXiv:1807.01028, 2018.
[23]. Hakan Karaoguz, Patric Jensfelt. Fusing Saliency Maps with Region Proposals for Unsupervised Object Localization, arXiv:1804.03905, 2018.
[24]. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv:1506.01497, 2015.
[25]. Kiranyaz S., Ince T. & Gabbouj M. Real-Time Patient-Specific ECG Classification by 1D Convolutional Neural Networks. IEEE Transactions on Biomedical Engineering, vol. 63, issue 3, 2016, pp.664–675.
[26]. M.D. Zeiler. ADADELTA: an adaptive learning rate method. arXiv:1212.5701, 2012.
[27]. Zha, Xuefan. (2018). A Comparison of 1-D and 2-D Deep Convolutional Neural Networks in ECG Classification. arXiv:1810.07088, 2018.
[28]. G. Huang, Z. Liu, and K.Q. Weinberger. Densely connected convolutional networks. arXiv:1608.06993, 2017..
[29]. A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. arXiv:1808.03314, 2018.

Информация об авторах / Information about authors

Сергей Евгеньевич ШТЕХИН – старший специалист по анализу данных. Сфера научных интересов: компьютерное зрение.

Sergey Evgenievich SHTEKHIN – Senior Data Analyst. Research interests: computer vision.

Юстина Алексеевна ИВАНОВА – специалист по анализу данных. Сфера научных интересов: компьютерное зрение, временные ряды, рекомендательные системы.

Justina Alekseevna IVANOVA – Data Analyst. Research interests: computer vision, time series, recommendation systems.

Денис Константинович КАРАЧЕВ – специалист по анализу данных. Сфера научных интересов: компьютерное зрение, беспилотный транспорт.

Denis Konstantinovich KARACHEV – Data Analyst. Research interests: computer vision, unmanned vehicles.

DOI: 10.15514/ISPRAS-2020-32(1)-8

Эффективные реализации алгоритмов тематического моделирования

М.А. Апишев, ORCID: 0000-0002-9023-3670 <mel-lain@yandex.ru>
Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1

Аннотация. Представлен обзор эффективных алгоритмов вероятностного тематического моделирования больших текстовых коллекций. Рассматриваются алгоритмы обучения моделей латентного размещения Дирихле (LDA) и аддитивно регуляризованных тематических моделей (ARTM) для многопроцессорных систем. Предложена систематизация технических приёмов для организации параллельных вычислений, распределённого хранения данных, потоковой обработки, уменьшения потребления оперативной памяти, повышения отказоустойчивости. Проведён сравнительный анализ доступных реализаций.

Ключевые слова: параллельные алгоритмы; распределённое хранение данных; обработка потоковых данных; отказоустойчивость; тематическое моделирование; EM-алгоритм; латентное размещение Дирихле; аддитивная регуляризация тематических моделей.

Для цитирования: Апишев М.А. Эффективные реализации алгоритмов тематического моделирования. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 137–152. DOI: 10.15514/ISPRAS–2020–32(1)–8

Благодарности. Данная работа поддержана Российским фондом фундаментальных исследований, проект 20-07-00936.

Effective implementations of topic modeling algorithms

M.A. Apishev, ORCID: 0000-0002-9023-3670 <mel-lain@uandex.ru>
Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia

Abstract. Topic modeling is an area of natural language processing that has been actively developed in the last 15 years. A probabilistic topic model extracts a set of hidden topics from a collection of text documents. It defines each topic by a probability distribution over words and describes each document with a probability distribution over topics. The exploding volume of text data motivates the community to constantly upgrade topic modeling algorithms for multiprocessor systems. In this paper, we provide an overview of effective EM-like algorithms for learning latent Dirichlet allocation (LDA) and additively regularized topic models (ARTM). Firstly, we review 11 techniques for efficient topic modeling based on synchronous and asynchronous parallel computing, distributed data storage, streaming, batch processing, RAM optimization, and fault tolerance improvements. Secondly, we review 14 effective implementations of topic modeling algorithms proposed in the literature over the past 10 years, which use different combinations of the techniques above. Their comparison shows the lack of a perfect universal solution. All improvements described are applicable to all kinds of topic modeling algorithms: PLSA, LDA, MAP, VB, GS, and ARTM.

Keywords: parallel algorithms; distributed data storage; stream data processing; fault tolerance; topic modeling; EM algorithm; latent Dirichlet allocation; additive regularization of topic models.

For citation: Apishev M.A. Effective implementations of topic modeling algorithms. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 137–152 (in Russian). DOI: 10.15514/ISPRAS–2020–32(1)–8

Acknowledgements. This work is supported by Russian Foundation for Basic Research, grant 20-07-00936.

1. Введение

Тематическое моделирование – одно из современных направлений машинного обучения (machine learning, ML) и обработки естественного языка (natural language processing, NLP), активно развивающееся с конца 90-х годов [1, 2, 3]. Вероятностная тематическая модель (probabilistic topic model, PTM) коллекции текстовых документов описывает каждый документ дискретным распределением вероятностей тем, каждую тему – дискретным распределением вероятностей слов. Тематическое моделирование преследует несколько целей: выявить тематическую кластерную структуру коллекции; построить тематические векторные представления документов; описать каждую тему словами или фразами естественного языка. В отличие от обычной «жёсткой» кластеризации тематическая модель распределяет каждый документ по многим кластерам-темам «мягко», с различными вероятностями.

Современные приложения текстовой аналитики сталкиваются с коллекциями огромных объёмов. Это требует от алгоритмов обучения тематических моделей линейной вычислительной сложности как по объёму входных данных, так и по числу тем. Данному требованию удовлетворяют EM-подобные алгоритмы, рассматриваемые в данном обзоре. Мы анализируем особенности их программной реализации в 14 инструментальных средствах тематического моделирования для многопроцессорных систем. Рассматриваются особенности параллельных вычислений, распределённого хранения данных, пакетной обработки данных, экономии оперативной памяти, обеспечения отказоустойчивости.

Цель обзора – помочь академическим исследователям и прикладным разработчикам определиться с выбором инструментария или обосновать собственную разработку.

Изложение имеет следующую структуру: в разд. 2 вводятся основные обозначения и теоретические основы EM-подобных алгоритмов тематического моделирования; разд. 3 систематизирует технические приёмы повышения их производительности; в разд. 4 перечисляются реализации, выполненные за последние 10 лет и использующие эти приёмы в различных сочетаниях; в разд. 5 реализации и приёмы сопоставляются в виде таблицы и приводятся заключительные выводы.

2. Задача тематического моделирования и EM-подобные алгоритмы

Пусть заданы три конечных множества: коллекция D текстовых документов, словарь W всех употребляемых в них термов, и множество тем T. В роли термов могут выступать исходные слова, лемматизированные слова, словосочетания, термины – в зависимости от того, какие методы были использованы на стадии предварительной обработки текста. Последовательность термов всех документов описывается вектором наблюдаемых переменных $X = (d_i, w_i)_{i=1}^n$, где n – суммарная длина всех документов коллекции. С каждым i-м термом связана неизвестная тема t_i. Последовательность $Z = (t_i)_{i=1}^n$ называется вектором скрытых переменных.

Вероятностная тематическая модель описывает распределение термов в документе $p(w|d)$ вероятностной смесью распределений термов в темах $p(w|t) = \varphi_{wt}$, причём каждая тема имеет свою условную вероятность в документе $p(t|d) = \theta_{td}$:

$$p(w|d) = \sum_{t\in T} p(w|t)p(t|d) = \sum_{t\in T} \varphi_{wt}\theta_{td}\,.\#(1)$$

Задача тематического моделирования состоит в том, чтобы по наблюдаемым данным X найти матрицы параметров модели $\Phi = (\varphi_{wt})_{W\times T}$ и $\Theta = (\theta_{td})_{T\times D}$.

Вероятностный латентный семантический анализ PLSA [1] основан на максимизации правдоподобия вероятностной модели данных:

$$\ln p(X|\Phi,\Theta) = \sum_{d\in D}\sum_{w\in d} n_{dw} \ln \sum_{t\in T} \varphi_{wt}\theta_{td} \to \max_{\Phi,\Theta}.$$

Это задача низкорангового неотрицательного матричного разложения. Она имеет бесконечно много решений, то есть является некорректно поставленной. Чтобы доопределить постановку задачи и сделать решение устойчивым, можно ввести дополнительный критерий регуляризации $R(\Phi,\Theta) \to \max$.

Аддитивная регуляризация тематических моделей ARTM обобщает эту идею введением взвешенной суммы нескольких регуляризаторов [4]:

$$\ln p(X|\Phi,\Theta) + R(\Phi,\Theta) \to \max_{\Phi,\Theta};$$
$$R(\Phi,\Theta) = \sum_k \tau_k R_k(\Phi,\Theta).$$

Как показано в [5], решение данной оптимизационной задачи удовлетворяет следующей системе уравнений относительно искомых параметров модели φ_{wt} и θ_{td} и неизвестных вероятностей скрытых переменных $p_{tdw} = p(t|d,w)$:

$$p_{tdw} = \underset{t\in T}{\mathrm{norm}}(\varphi_{wt}\theta_{td})\,;\#(2)$$

$$\varphi_{wt} = \underset{w\in W}{\mathrm{norm}}\left(n_{wt} + \varphi_{wt}\frac{\partial R}{\partial \varphi_{wt}}\right), \quad n_{wt} = \sum_{d\in D} n_{dw}p_{tdw}\,;\#(3)$$

$$\theta_{td} = \underset{t\in T}{\mathrm{norm}}\left(n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}\right), \quad n_{td} = \sum_{w\in d} n_{dw}p_{tdw}\,.\#(4)$$

где переменная n_{td} оценивает число термов темы t в документе d; переменная n_{wt} оценивает, сколько раз терм w относился к теме t во всей коллекции; оператор norm преобразует произвольный заданный вектор $(x_i)_{i\in I}$ в вектор вероятностей $(p_i)_{i\in I}$ дискретного распределения путём обнуления отрицательных элементов и нормировки:

$$p_i = \underset{i\in I}{\mathrm{norm}}(x_i) = \frac{\max\{0, x_i\}}{\sum_{j\in I}\max\{0, x_j\}}, \forall i \in I.$$

Модель PLSA соответствует тривиальному частному случаю $R(\Phi,\Theta) = 0$.

Для решения системы уравнений (2)–(4) применяется метод простых итераций: сначала выбираются начальные приближения параметров φ_{wt} и θ_{td}, по ним вычисляются вспомогательные переменные p_{tdw} и следующее приближение параметров φ_{wt} и θ_{td}. Вычисления продолжаются в цикле до сходимости. Этот итерационный процесс называется EM-алгоритмом [6]. Вычисление условных распределений скрытых переменных (2) называется E-шагом (expectation), оценивание параметров модели (3) и (4) – M-шагом (maximization).

В байесовском подходе для задания ограничений на параметры модели вводится априорное распределение $p(\Phi,\Theta|\gamma)$ с вектором гиперпараметров γ. Принцип максимума апостериорной вероятности (maximum a posteriori probability, MAP) эквивалентен введению регуляризатора, равного логарифму априорного распределения:

$$\ln p(X|\Phi,\Theta) + \ln p(\Phi,\Theta|\gamma) \to \max_{\Phi,\Theta}.\#(5)$$

Таким образом, вероятностные предположения о параметрах модели, задаваемые через априорное распределение, можно переводить в вероятностный регуляризатор, и применять для решения задачи всё тот же EM-алгоритм (2)–(4).

Модель латентного размещения Дирихле LDA [2] является наиболее известной в тематическом моделировании. Она основана на априорном предположении, что столбцы матрицы Φ порождаются $|W|$-мерным распределением Дирихле $\mathrm{Dir}(\varphi_t|\beta)$ с вектором гиперпараметров $\beta = (\beta_w)_{w\in W}$, а столбцы матрицы Θ – $|T|$-мерным распределением Дирихле $\mathrm{Dir}(\theta_d|\alpha)$ с вектором гиперпараметров $\alpha = (\alpha_t)_{t\in T}$. Таким образом, в модели LDA $\gamma = (\beta, \alpha)$.

Общая система уравнений ARTM (2)–(4) после подстановки в неё вероятностного регуляризатора модели LDA принимает вид

$$p_{tdw} = \underset{t\in T}{\mathrm{norm}}(\varphi_{wt}\theta_{td}) \#(6)$$

$$\varphi_{wt} = \underset{w\in W}{\mathrm{norm}}(n_{wt} + \beta_w - 1), \quad n_{wt} = \sum_{d\in D} n_{dw}p_{tdw} \#(7)$$

$$\theta_{td} = \underset{t\in T}{\mathrm{norm}}(n_{td} + \alpha_t - 1), \quad n_{td} = \sum_{w\in d} n_{dw}p_{tdw} \#(8)$$

В тематическом моделировании наибольшее распространение получили методы байесовского обучения. Чтобы оценить параметры Φ и Θ, выводят их апостериорные распределения $p(\Phi, \Theta|X, \gamma)$, затем берут по ним оценки математического ожидания. Заметим, что это более трудный путь по сравнению с ARTM.

Среди методов байесовского обучения в тематическом моделировании наиболее популярны вариационный байесовский вывод и сэмплирование Гиббса.

Вариационный байесовский вывод (Variational Bayes, VB) основан на вычислении совместного апостериорного распределения параметров модели и скрытых переменных $p(\Phi, \Theta, \mathrm{Z}|\mathrm{X}, \gamma)$. Точное выражение для него получить не удаётся, поэтому ищется его приближённое представление в виде произведения независимых распределений по переменным t_i, φ_t, θ_d. Для модели LDA вариационный байесовский вывод приводит к системе уравнений, похожей на EM-алгоритм [2, 7]:

$$p_{tdw} = \underset{t\in T}{\mathrm{norm}}\left(\frac{E(n_{wt} + \beta_w)}{E(\sum_w(n_{wt} + \beta_w))} \cdot \frac{E(n_{td} + \alpha_t)}{E(\sum_t(n_{td} + \alpha_t))}\right); \#(9)$$

$$\varphi_{wt} = \underset{w\in W}{\mathrm{norm}}(n_{wt} + \beta_w), \quad n_{wt} = \sum_{d\in D} n_{dw}p_{tdw}\,; \#(10)$$

$$\theta_{td} = \underset{t\in T}{\mathrm{norm}}(n_{td} + \alpha_t), \quad n_{td} = \sum_{w\in d} n_{dw}p_{tdw}\,, \#(11)$$

где $E(x) = \exp\big(\psi(x)\big) \approx x - \frac{1}{2}$ – экспонента от дигамма-функции $\psi(x) = \frac{\Gamma\prime(x)}{\Gamma(x)}$.

Сэмплирование Гиббса (Gibbs Sampling, GS) – это байесовский вывод апостериорного распределения скрытых переменных $p(Z|X, \gamma) = \int_\Phi \int_\Theta p(\Phi, \Theta, Z|X, \gamma) d\Phi d\Theta$, из которого сэмплируются значения $Z \sim p(Z|X, \gamma)$. Для этих Z вычисляется апостериорное распределение параметров модели $p(\Phi, \Theta|X, Z, \gamma)$, и по нему находятся оценки математического ожидания параметров Φ, Θ. Для модели LDA сэмплирование Гиббса приводит к системе уравнений, снова похожей на EM-алгоритм [8]:

$$t_i \sim p_{td_iw_i} = \underset{t\in T}{\mathrm{norm}}\left(\frac{n_{w_it} + \beta_{w_i} - 1}{\sum_w(n_{wt} + \beta_w) - 1} \cdot \frac{n_{td_i} + \alpha_t - 1}{\sum_t(n_{td_i} + \alpha_t) - 1}\right); \#(12)$$

$$\varphi_{wt} = \underset{w\in W}{\mathrm{norm}}(n_{wt} + \beta_w), \quad n_{wt} = \sum_{i=1}^{n} [w_i = w][t_i = t]\,; \#(13)$$

$$\theta_{td} = \underset{t \in T}{\text{norm}}(n_{td} + \alpha_t), \quad n_{td} = \sum_{i=1}^{n} [d_i = d][t_i = t] \,. \# \text{ (14)}$$

Главное отличие этого алгоритма от предыдущих в том, что для каждого терма (d_i, w_i) на каждой итерации сэмплируется единственная тема t_i, которая и участвует в аккумулировании счётчиков n_{wt} и n_{td}. Фактически, на М-шаге суммируются не сами распределения $p(t|d_i, w_i)$, а их вырожденные эмпирические оценки $[t = t_i]$, сделанные каждый раз по единственной сэмплированной теме t_i. Сумма таких оценок сходится к сумме исходных распределений, согласно закону больших чисел.

В работе [9] сэмплирование рассматривалось как отдельная эвристика, которую можно использовать в любом ЕМ-подобном алгоритме тематического моделирования, начиная с PLSA. Эксперименты показали, что сэмплирование практически не влияет на сходимость и качество модели. В ARTM оно может свободно сочетаться с любыми регуляризаторами.

ЕМ-подобные алгоритмы (2)–(4), (6)–(8), (9)–(11), (12)–(14) отличаются небольшими поправками к частотным оценкам условных вероятностей. При $n_{wt} \gg 1$, $n_{td} \gg 1$ эти поправки пренебрежимо малы. Они влияют лишь на близкие к нулю условные вероятности φ_{wt} и θ_{td}, которые не являются значимыми для тематической модели. Сходство ЕМ-подобных алгоритмов PLSA, MAP, VB, GS для модели LDA и ещё нескольких их вариантов было замечено в [10].

Во всех рассмотренных алгоритмах вычисление переменных-счётчиков n_{wt} и n_{td} на М-шаге требует однократного прохода коллекции в цикле по всем термам $w \in d$ всех документов $d \in D$. Внутри этого цикла значение p_{tdw} вычисляется только один раз при обработке терма w в документе d, после чего его можно забыть. Это позволяет выделить итерационный процесс обработки одного документа при фиксированной матрице Φ.

В этом процессе Е-шаг чередуется с М-шагом для одного столбца матрицы Θ, соответствующего данному документу. По завершении обработки документа матрица Φ обновляется. Такая организация вычислений не требует дополнительных вычислительных затрат и обходится без хранения трёхмерной матрицы p_{tdw}. Алгоритмы подокументной обработки коллекции линейно масштабируются по длине коллекции и числу тем, допуская возможность параллельной обработки документов и распределённого хранения коллекции.

Практическая эффективность ЕМ-подобных алгоритмов тематического моделирования определяется не столько математическими различиями, сколько особенностями программной реализации. Не столь важно, используется ли байесовский вывод или аддитивная регуляризация, используется ли сэмплирование или нет. Важно, в каком порядке организованы вычисления по формулам Е- и М-шагов, как хранятся исходные данные и параметры модели, и как используются механизмы параллельных вычислений. Анализу этих различий посвящены следующие разделы данной статьи.

3. Техники эффективного обучения тематических моделей

Представленные в литературе эффективные реализации алгоритмов тематического моделирования используют различные технические приёмы для параллельных вычислений, распределённого хранения данных, ускорения процесса обучения, уменьшения потребления ресурсов, обеспечения отказоустойчивости системы. В данном разделе систематически описываются отдельные приёмы. Они могут быть полезны как по отдельности, так и в сочетаниях; как для тематического моделирования, так и для других задач матричного разложения больших разреженных матриц. Обзор реализаций, представляющих собой различные сочетания этих приёмов, будет дан в следующем разделе.

3.1 Распределённые хранение и обработка коллекции

В современных приложениях анализа текстов объём коллекции может быть настолько большим, что либо её не удастся разместить во внешней памяти одного вычислительного узла, либо время её обработки на одном узле окажется неприемлемым.

Возможным решением данной проблемы является использование вычислительного кластера [11–21]. Документы d вместе с соответствующими им счётчиками n_{td} распределяются равномерно по ядрам узлов кластера и обрабатываются параллельно. Распределённо храниться могут также параметры, зависящие от документов, например θ_{td} или сэмплированные скрытые темы Z. Увеличение числа машин и ядер может обеспечивать рост производительности до определённого момента. Однако обработка сверхбольших или динамически пополняемых коллекций может потребовать дополнительной оптимизации.

Параллельное выполнение сэмплирования или E-шага для ускорения обработки документов может использоваться и в рамках одного вычислительного узла, если он имеет достаточный объём оперативной памяти. Такой подход используется в [16, 22, 23].

3.2 Синхронная параллельная обработка

Параллельная обработка документов, вне зависимости от того, выполняется она на кластере или на одной машине, может быть организована различными способами. Отличаются они, главным образом, методами накопления счётчиков n_{wt} и обновления параметров ϕ_{wt}. Проблема в том, что для любого параллельного алгоритма эти величины являются разделяемыми ресурсами, к которым все рабочие процессы должны иметь доступ на этапе обработки документов.

Один из способов организации параллельных вычислений, представленный в работах [11, 12, 16–19, 21, 22], можно условно назвать синхронным. В нём текущая версия матрицы n_{wt} или нужная её часть копируется в начале итерации обработки коллекции в память каждого рабочего процесса и доступна ему на чтение, а получаемые в результате обработки приращения счётчиков n_{wt} аккумулируются локально. После того, как все параллельные процессы завершают работу над своими порциями документов, они складывают приращения в глобальную матрицу n_{wt}, которая копируется и используется для обработки текстов во время следующей итерации.

Синхронный подход к аккумулированию счётчиков прост в реализации, но плох тем, что нагрузка на вычислительные и сетевые ресурсы распределена неравномерно: попеременно то сеть, то процессоры или простаивают, или перегружены. Кроме того, скорость параллельной обработки во время одной итерации определяется самым медленным из обработчиков, что может приводить к существенной деградации производительности.

3.3 Асинхронная параллельная обработка

В отличие от синхронной параллельной обработки документов, асинхронный вариант [13, 14, 15, 20, 23] не имеет выделенного шага синхронизации и обновляет счётчики n_{wt} (а при необходимости и матрицу Ф) одновременно с обработкой документов. Архитектуры на его основе сложнее в реализации и настройке. Кроме того, асинхронность часто увеличивает потребление оперативной памяти. Однако производительность и масштабируемость у асинхронных архитектур обычно выше, чем у синхронных.

Способы реализации асинхронных архитектур разнообразны. В [13] случайным образом выбранные пары вычислительных узлов обмениваются своими приращениями n_{wt} после завершения обработки документов (асинхронность заключается в том, что все прочие узлы могут продолжать работать независимо от других). В реализации [15] обновления

счётчиков передаются в глобальное хранилище модели в фоновом режиме, без остановки процесса обработки документов. Схожим образом процесс организован в [23], где асинхронность обеспечивается наличием нескольких версий матриц n_{wt} и Ф. Новая версия Ф на основе обработанных ранее документов подготавливается одновременно с обработкой следующей порции документов, для которой используется предыдущая версия параметров.

3.4 Внешнее хранение параметров документов

Очевидным узким местом с точки зрения используемой памяти является хранение счётчиков документов n_{td}, значений переменных Z или параметров Θ в памяти отдельного компьютера или кластера. В реализации Light LDA [20] предлагается хранить все связанные с документами счётчики и параметры на диске, подгружая их в память по мере необходимости.

Иной подход реализован в библиотеке BigARTM [22, 23]. Заметим, что счётчики и параметры, связанные с документом, необходимы только на E-шаге при обработке данного документа. Вместо того, чтобы хранить их между итерациями обработки коллекции, можно на каждой итерации вычислять их заново при обработке документа и удалять после её завершения. Обработка документа d начинается с инициализации $\theta_{td} = \frac{1}{|T|}$. Затем делается несколько итераций по документу, в которых чередуются E-шаг (2) и M-шаг (4) при фиксированных параметрах φ_{wt}. Схожим образом производится обработка данных в онлайновых алгоритмах, которые рассматриваются ниже.

3.5 Онлайновая (потоковая) обработка

Оффлайновые алгоритмы делают на каждой итерации полный проход коллекции, накапливая счётчики n_{wt}, после этого обновляют параметры φ_{wt}. Такие алгоритмы удобны, когда коллекция небольшая и не пополняется. В случае большой коллекции может потребоваться слишком много проходов для сходимости матрицы Ф.

Онлайновые алгоритмы обновляют матрицу Ф после обработки каждого документа [24] или (в пакетном варианте) после каждого пакета документов [16, 23, 25, 26]. Это ускоряет сходимость итерационного процесса. На большой коллекции матрица Ф может сойтись и перестать меняться задолго до окончания первой итерации. В таких случаях одного прохода по коллекции может оказаться достаточно для построения модели. Поэтому онлайновые алгоритмы предпочтительны для обработки больших или потоковых данных. При отказе от хранения параметров θ_{td} и счётчиков n_{td} онлайновый алгоритм позволяет тематизировать потенциально бесконечное число документов при фиксированном потреблении памяти.

3.6 Распределённое или оптимизированное хранение модели

При обработке больших текстовых коллекций в моделях с большим числом тем размер матрицы счётчиков n_{wt} и матрицы параметров Ф может превысить размер оперативной памяти одного компьютера. Эту проблему можно решать двумя способами: либо хранением значительной части модели во внешней памяти с подгрузкой нужных её фрагментов в оперативную память [24], либо распределённым хранением модели в памяти машин в кластере [14, 18–21].

Первый вариант позволяет строить большие модели на одной машине, но представляет меньший интерес, поскольку операции с внешней памятью существенно медленнее, чем с оперативной.

Во втором случае предполагается, что суммарный объём оперативной памяти на машинах кластера достаточно велик для хранения всей модели, но в память одной

машины вся модель не поместится. В этой ситуации и коллекция, и модель разбиваются некоторым образом на части, которые хранятся и обрабатываются на различных вычислительных узлах (конкретный способ разбиения и взаимодействия частей зависит от реализации). Это обеспечивает и параллельную обработку коллекции, и размещение большой модели в памяти кластера без увеличения объёма оперативной памяти на отдельных узлах.

3.7 Разреженное хранение модели

Ещё одним способом работы с большой моделью является её представление в разреженном виде. В ходе итераций матрицы n_{wt} и Φ в моделях LDA и ARTM становятся всё более разреженными [9, 5], что открывает возможность для их более экономного хранения. Для этого могут использоваться форматы хранения разреженных данных типа CSR (Compressed Sparse Row) или хэш-таблицы.

Обратной стороной разреженного хранения может стать снижение производительности, вызываемое заменой прямого доступа к элементам матриц на последовательный. По этой причине в [20] и [21] используется гибридный подход: параметры и счётчики, связанные с наиболее часто встречающимися термами, хранятся в плотном виде для ускорения вычислений, а оставшиеся термы хранятся разреженно в виде хэш-таблицы для экономии памяти.

3.8 Разреженная инициализация модели

Как было отмечено выше, разреженные матрицы n_{wt} и Φ позволяют ощутимо снизить потребление оперативной памяти. Чтобы этот подход давал реальную экономию, матрицы должны быть разреженными на протяжении всего итерационного процесса, а не только начиная с некоторого момента. Стандартная процедура инициализации параметров случайными числами и сэмплирование тем на стартовой итерации дают в результате плотную матрицу, что нивелирует все усилия по снижению пикового потребления памяти (по крайней мере, для оффлайновых алгоритмов).

Одним из возможных решений этой проблемы является разреженная инициализация. В зависимости от алгоритма обучения она может заключаться в обнулении части значений в матрице Φ при случайной генерации значений параметров или в сужении допустимого множества присваиваемых термам тем при сэмплировании.

Несмотря на кажущуюся грубость такого решения, эксперименты в [20, 21] показывают, что использование разреженной инициализации не сильно ухудшает качество тематической модели, существенно снижая при этом потребление оперативной памяти.

3.9 Динамическое изменение размера модели

Другой способ экономии памяти при построении разреженных моделей заключается в постепенном добавлении новых тем по мере увеличения числа документов в коллекции. В этом случае сначала строится предварительная модель с относительно небольшим числом тем по имеющейся части коллекции. Затем в модель добавляются новые темы по мере поступления новых документов. Возможность изменения числа тем в матрицах Φ и Θ доступна в библиотеке BigARTM [22, 23].

3.10 Разреженная обработка термов

Термы могут иметь существенно различную частоту в коллекции. Термы, которые часто встречаются в документах, обрабатываются чаще, поэтому связанные с ними параметры φ_{wt} сходятся быстрее. С определённого момента приращения счётчиков n_{wt}

высокочастотных термов перестают добавлять в модель новую информацию, продолжая потреблять значительные ресурсы на своё вычисление.

Для решения данной проблемы в [21] и [24] предлагается хранить для каждого терма предшествующие результаты сэмплирования или E-шага и оценивать, насколько сильно они изменились. Термы, для которых изменения систематически оказываются незначительными, исключаются из дальнейшей обработки либо безусловно, либо с некоторой вероятностью.

3.11 Обеспечение отказоустойчивости

Обеспечение отказоустойчивости важно для любого длительного процесса, выполняемого в сложной вычислительной среде, где возможны сбои в сети, аппаратном или программном обеспечении. Обучение тематических моделей в этом случае не является исключением.

Сложность реализации отказоустойчивого алгоритма на кластере во многом зависит от базового фреймворка, поверх которого строится реализация. Так, промышленные системы Hadoop [27] и Spark [28] обеспечивают высокую устойчивость кластера к сбоям в процессе выполнения вычислительной задачи, в то время как MPI [29] такой возможности не предоставляет, и обеспечение отказоустойчивости целиком возлагается на разработчиков.

Основным методом повышения отказоустойчивости EM-подобных алгоритмов является периодическое сохранение на диск текущего состояния модели и счётчиков. Такой подход позволяет восстановить состояние системы и продолжить обучение с места остановки при сбоях, не затрагивающих диск. Он может использоваться в реализациях алгоритма как на кластере, так и на отдельной машине.

4. Обзор реализаций алгоритмов

В этом разделе рассматриваются (в основном, в хронологическом порядке) реализации алгоритмов тематического моделирования и описываются детали использования технических приёмов, описанных в предыдущем разделе

4.1 AD-LDA

AD-LDA [11] – одна из первых параллельных реализаций обучения модели LDA. За основу взят алгоритм сэмплирования Гиббса, параллелизм реализован на кластере на уровне ядер с помощью технологии MPI. Каждое ядро обрабатывает свою порцию документов и получает локальную копию счётчиков n_{wt} перед началом очередной итерации. Используется синхронная архитектура, то есть после того, как обработка документов завершается на всех ядрах, запускается процедура добавления всех полученных приращений счётчиков в общую глобальную матрицу счётчиков. Отказоустойчивость и дополнительные оптимизации в AD-LDA отсутствуют.

4.2 PLDA

PLDA [12] является ещё одной реализацией идей AD-LDA с помощью технологии MPI. Никаких существенных алгоритмических или архитектурных улучшений алгоритма не предлагается. В отличие от своего предшественника, PLDA поддерживает отказоустойчивость между блоками итераций сэмплирования путём сохранения промежуточных данных на диск.

4.3 AS-LDA

AS-LDA [13] – одна из первых асинхронных многопроцессорных архитектур для параллельного выполнения алгоритма сэмплирования Гиббса. Как и в AD-LDA, все процессоры получают при старте свою порцию документов и вычисляют приращения счётчиков. Синхронизация организована следующим образом: каждый процессор по завершении очередного этапа обработки обменивается результатом (коммутирует) со случайным процессором, также завершившим свой этап.

Содержательной частью реализации является процедура агрегации данных при таком обмене. В ситуации первой коммутации процессоры просто прибавляют полученные счётчики к собственным. При повторном обмене аппроксимированные значения счётчиков n_{wt}, полученных в предыдущей коммутации, вычитаются из текущих значений счётчиков процессоров, после чего результат складывается с новыми значениями от текущей коммутации. Тривиальное решение в виде хранения предыдущих счётчиков авторы посчитали неприемлемым из-за существенных затрат оперативной памяти.

Алгоритм реализован на кластере с помощью MPI без дополнительных усовершенствований.

4.4 PLDA+

PLDA+ [14] – асинхронная кластерная реализация на основе сэмплирования Гиббса. Ключевой её особенностью является разделение процессоров на две группы: рабочие, которые обрабатывают документы и выполняют сэмплирование тем, и транспортные, отвечающие за обновление и доставку глобальных параметров. Такой подход позволяет производить обмен данными в фоновом режиме, без прерывания обработки документов.

Жизненный цикл группы транспортных процессоров состоит из этапа размещения матрицы n_{wt} и этапа обработки запросов. Каждый процессор получает свою часть счётчиков и далее занимается их обновлением и доставкой рабочим процессорам.

Перед началом работы документы коллекции распределяются между рабочими процессорами случайным образом. Необычным решением является одновременная обработка вхождений одного и того же терма во всех документах данного процессора. Для удобства реализации этой идеи термы из словаря процессора группируются в блоки для выполнения итераций сэмплирования Гиббса и отправки запросов. На данном этапе редкие термы сливаются с частыми для сокращения времени сэмплирования. Для минимизации вероятности одновременной обработки одного терма двумя рабочими процессорами создаётся циклическая очередь обрабатываемых термов и расписание на ней. Это позволяет избежать коллизий при отправке запросов к транспортным процессорам.

4.5 Y!LDA

Y!LDA [15] – это кластерная версия алгоритма сэмплирования Гиббса, в которой параллелизм реализован на уровне вычислительных узлов. На каждом узле запускается многопоточный процесс, занимающийся сэмплированием тем для термов документов, размещённых на узле перед стартом обучения. Общая для всех потоков память позволяет хранить в пределах одного узла всего две локальные матрицы n_{wt} вместо $O(N)$, как в AD-LDA и PLDA, где N – число ядер в процессорах узла. При этом на каждом узле имеется выделенный транспортный поток, который асинхронно получает от рабочих потоков приращения счётчиков и обновляет локальную матрицу n_{wt}.

Глобальная матрица размещается в распределённом хранилище. Каждый вычислительный узел работает с ней (отправляет свои приращения и обновляет

локальные значения), блокируя по одному терму за раз вне зависимости от действий других узлов, что обеспечивает асинхронность и на кластерном уровне.

Между итерациями сэмплирования потоки в узлах сохраняют текущие значения тем Z для термов своих документов на диск, что обеспечивает возможность рестарта обработки с последней итерации в случае сбоя.

4.6 Vowpal Wabbit LDA

Vowpal Wabbit [26] – это библиотека онлайновых алгоритмов машинного обучения, которая включает в себя реализацию онлайнового вариационного EM-алгоритма для модели LDA, предложенного в [25]. Данная реализация алгоритма является пакетной, однопоточной и рассчитана на использование в рамках одного компьютера.

4.7 Gensim

Gensim [16] – это программный пакет для тематического моделирования и векторных представлений текста. В нём реализованы две версии онлайнового вариационного EM-алгоритма [25]: однопоточная и многопоточная. Однопоточный вариант представляет собой обычный пакетный онлайновый алгоритм. В многопоточном используется модель параллелизма, схожая с Y!LDA на одном вычислительном узле: коллекция разбивается на пакеты, обрабатываемые в несколько потоков, выделенный поток асинхронно собирает счётчики, полученные рабочими потоками, и обновляет матрицу n_{wt} этого узла.

4.8 FOEM-LDA

FOEM-LDA [24] предоставляет однопоточную реализацию онлайнового EM-алгоритма для критерия MAP в модели LDA (формулы (6)–(8)) для работы на одном вычислительном узле. Алгоритм не является пакетным (обновление параметров производится после каждого обработанного документа) и не хранит матрицу Θ целиком.

В данной реализации решается проблема хранения модели, не помещающейся в оперативной памяти. Для этого матрицы Φ и n_{wt} разделяются на две части, соответствующие более и менее важным термам. Первые хранятся в оперативной памяти постоянно, вторые подгружаются по мере необходимости. Важность термов на первой итерации оценивается по их частоте, далее – по скорости сходимости, рассчитываемой на основе текущих и предшествующих значений p_{tdw}.

Для минимизации количества обновлений n_{wt} на текущей итерации (и, соответственно, числа загрузок данных с диска) расчёт скорости сходимости производится сразу после обработки пакета документов. Приращения счётчиков для термов, у которых значения p_{tdw} изменились слабо, игнорируются.

4.9 Mr.LDA

Mr.LDA [17] представляет собой реализацию вариационного EM-алгоритма в рамках парадигмы MapReduce [30] на базе фреймворка Hadoop.

Каждой итерации алгоритма соответствует один запуск процедуры MapReduce. На этапе Map для каждого документа коллекции независимо выполняется E-шаг (9). Полученные результаты агрегируются по ключам-номерам тем и отправляются на этап Reduce, где для каждой темы выполняется M-шаг (10)–(11). После этого производится обновление параметров модели, которые хранятся в распределённом кэше Hadoop-кластера – специальной, доступной только для чтения и общей для всех компонентов системы памяти.

4.10 Spark-LDA

Spark-LDA [18] – одна из первых библиотек для обучения модели LDA с помощью сэмплирования Гиббса в рамках фреймворка Spark. И данные, и параметры модели в Spark-LDA распределяются между вычислительными узлами таким образом, чтобы множества документов для разных машин не пересекались, а множества термов не пересекались для частей данных, обрабатываемых узлами в текущий момент времени. Для этого матрицы исходных данных n_{dw} и параметров n_{wt} нарезаются по термам на P частей. После на столько же частей производится нарезка данных и счётчиков n_{td} по документам, в результате получается $P \times P$ блоков данных. Из этих блоков набирается P наборов по P блоков, после чего для каждого набора выполняется параллельное сэмплирование и обновление локальных счётчиков n_{wt} и n_{td}. Документные счётчики локальны для данного вычислительного узла, поэтому их не нужно передавать по сети, счётчики же n_{wt} перемещаются между машинами при обработке очередного блока данных.

4.11 Peacock

Peacock [19] предназначена для обучения модели LDA на кластере с помощью сэмплирования Гиббса, с возможностью распределённого хранения как данных, так и модели.

В Peacock все ядра разделяются на три группы: серверы сэмплирования, серверы данных и серверы синхронизации. Матрица входных данных n_{dw} и счётчики n_{wt} разбиваются на M блоков: первая матрица разбивается по документам, каждый блок связывается с некоторым сервером данных; вторая – по термам, каждый блок связывается с некоторым сервером сэмплирования.

Каждый сервер данных посылает каждому серверу сэмплирования те части своих документов, в которых содержатся термы, связанные с этим сервером сэмплирования. После обработки очередного блока сервер сэмплирования обновляет свои счётчики n_{wt} и отправляет приращения счётчиков на соответствующий сервер данных.

В обычной для текстовых коллекций ситуации $|D| \gg |W|$ разделение словаря и коллекции на множество одинаковых частей может привести к перегрузке серверов данных. Для решения этой проблемы коллекция предварительно делится на C частей, для каждой из которых производится своё разбиение на блоки и, соответственно, серверы. Для синхронизации счётчиков между различными разбиениями используются серверы синхронизации. Системе требуется M таких серверов (по числу блоков в разбиении), каждый работает с соответствующим блоком во всех разбиениях. В конце каждого набора итераций сэмплирования Гиббса все серверы сэмплирования отправляют обновления счётчиков n_{wt} соответствующим серверам агрегирования. После сбора всех обновлений каждый сервер агрегирования пересылает соответствующим серверам сэмплирования новую версию части матрицы n_{wt}, за которую они ответственны.

4.12 Light LDA

Light LDA [20] – это обучение модели LDA на кластере. Вместо сэмплирования Гиббса используется более общий алгоритм Метрополиса-Гастингса. Как данные, так и модель распределяются между узлами кластера. При этом данные хранятся статично, а фрагменты модели пересылаются между обработчиками по мере необходимости, как в PLDA+ и Spark-LDA.

Текстовая коллекция нарезается на блоки данных, каждый из которых является единицей обработки вычислительного узла. При загрузке блока данных в оперативную память узла происходит отбор небольшого количества термов из словаря этого блока.

Выбранное множество достаточно мало, чтобы соответствующее его элементам множество строк матрицы n_{wt}, называемое срезом, могло поместиться в оперативной памяти узла. Система загружает указанный срез из распределённого хранилища, содержащего все параметры n_{wt}, после чего на узле обрабатываются только те термы блока, которые покрываются текущим срезом модели. Остальные игнорируются. Как только сэмплирование для термов текущего среза завершается, система загружает следующий срез и возобновляет сэмплирование.

Сетевые коммуникация производятся в фоновом режиме. Обновления счётчиков n_{wt} вычисляются локально и отправляются в хранилище асинхронно, как в Y!LDA. Для хранения счётчиков используется гибридное решение – для 10% самых частых термов строки матриц хранятся в плотном виде, для остальных – в разреженном.

Light LDA реализован поверх распределённой системы Petuum [31], представляющей собой фреймворк для реализации параллельных алгоритмов машинного обучения. Каждый обработчик на вычислительном узле работает в многопоточном режиме, для чего обрабатываемый блок данных разделяется на непересекающиеся подблоки, которые обрабатываются отдельными потоками. Срез модели в этой схеме является общим для всех потоков и доступен только для чтения.

4.13 BigARTM

BigARTM [22, 23] реализует две версии параллельного EM-алгоритма для обучения моделей ARTM на одной машине. Первый алгоритм – оффлайновый и синхронный, второй – пакетный онлайновый и асинхронный.

Онлайновый алгоритм параллельно обрабатывает несколько пакетов документов (по одному пакету на поток) и одновременно с этим занимается агрегированием счётчиков n_{wt} и вычислением матрицы Φ на основании результатов обработки предыдущего набора пакетов. Таким образом, происходит обновление параметров с запаздыванием. Система содержит в каждый момент времени не менее двух матриц счётчиков и параметров – текущих активных и формируемых новых.

Обе версии алгоритма допускают построение модели без хранения счётчиков n_{td} и матрицы Θ. Кроме того, BigARTM позволяет динамически изменять размер модели, а также даёт возможность сохранять состояние обучающего процесса между итерациями на диск и загружать его обратно.

BigARTM – единственная параллельная реализация тематического моделирования, выходящая далеко за рамки модели LDA. Благодаря механизму аддитивной регуляризации BigARTM позволяет строить модели с заданными свойствами путём комбинирования готовых модулей-регуляризаторов. В частности, модели ARTM могут быть одновременно мультимодальными, мультиязычными, n-граммными, иерархическими, темпоральными, сегментирующими, разреженными, декоррелированными, и т.д. [3].

4.14 ZenLDA

ZenLDA [21] – ещё одна реализация параллельного алгоритма сэмплирования Гиббса для модели LDA на Spark. Её ключевой особенностью является представление данных и модели в виде двудольного ненаправленного графа. Граф имеет два типа вершин – термы и документы. Ребро между вершиной-документом и вершиной-термом означает, что данный терм встречается в данном документе. Счётчики n_{wt} хранятся раздельно в виде разреженных векторов, каждый из которых прикреплён к своей вершине-терму. Для счётчиков документов n_{td} всё аналогично. Значения Z привязаны к рёбрам между соответствующими вершинами-термами и вершинами-документами. С каждым ребром

связан вектор таких значений, поскольку каждый терм может встретиться в документе более одного раза.

Параллельность вычислений достигается разделением графа на части, обрабатываемые вычислительными узлами одновременно и независимо, что обеспечивает распределённость как данных, так и модели. В реализации используется модификация алгоритма «degree-based hashing», производящего разделение графа по вершинам [21]. ZenLDA является синхронным итерационным алгоритмом.

Для ускорения алгоритма и уменьшения потребления ресурсов используется ряд оптимизаций. Во-первых, это разреженная инициализация вектора Z путём случайного сужения множества возможных тем для сэмплирования, которая приводит к получению менее плотной матрицы n_{wt} на первых итерациях. Во-вторых – исключение из обработки с некоторой вероятностью термов в документах, для которых значение присвоенной темы в векторе Z не изменилось с прошлой итерации. В-третьих, это использование гибридного метода хранения параметров для частых и редких термов (такого же, как в Light LDA). Кроме того, ZenLDA переопределяет ряд стандартных функций библиотеки GraphX [32] для использования всех ядер машины в параллельной обработке одной части графа.

5. Заключение

В данном обзоре описаны 11 технических приёмов для повышения эффективности алгоритмов тематического моделирования и 14 инструментальных средств, в которых эти приёмы реализуются в различных сочетаниях. В табл. 1 сведены все приёмы и реализации. Не существует идеального решения, объединяющего достоинства всех подходов, поскольку оптимизация одних характеристик может приводить к ухудшению других. Выбор компромиссного решения зависит от требований по времени адаптации под конкретную задачу, времени обучения моделей, вычислительным ресурсам, гибкости, масштабируемости и отказоустойчивости.

Описанные приёмы одинаково применимы к EM-подобным алгоритмам тематического моделирования PLSA, MAP, VB, GS, ARTM.

Большинство реализаций ограничиваются моделью латентного размещения Дирихле LDA. Исключение составляет BigARTM с библиотекой готовых модулей-регуляризаторов, из которых можно конструировать модели с требуемыми свойствами.

Табл. 1. Сравнение особенностей представленных в обзоре реализаций.
Table 1. Comparison of the implementations presented in the review.

Реализация	A	B	C	D	E	F	G	H	I	J	K	L	M
AD-LDA	MPI	C++	+	+									
PLDA	MPI	C++	+	+									+
AS-LDA	MPI	C	+		+								
PLDA+	RPC	C++	+		+			+					
Y!LDA	Memcached + многопоточность	C++	+		+								+
VW LDA	Однопоточность	C++				+	+						
Gensim	Многопоточность	Python			+	+	+						
FOEM-LDA	Однопоточность	C					+	+				+	
Mr.LDA	Hadoop + MapReduce	Java	+	+									+
Spark-LDA	Spark	Scala	+	+				+					+

Peacock	RPC + OpenMP	Go	+	+				+					
Light LDA	Petuum + многопоточность	C++	+		+			+	+	+			+
BigARTM offline	Многопоточность	C++		+		+					+		+
BigARTM online	Многопоточность	C++			+	+	+				+		+
ZenLDA	Spark	Scala	+	+				+	+	+		+	+

Расшифровка столбцов: A – фреймворк или способ организации вычислений; B – использованный язык программирования (основной, без учёта языков интерфейсов); C – распределённые хранение или обработка коллекции; D – синхронная параллельная обработка; E – асинхронная параллельная обработка; F – хранение параметров документов во внешней памяти; G – онлайновая обработка; H – распределённое или оптимизированное хранение модели; I – разреженное хранение модели; J – разреженная инициализация модели; K – динамическое изменение размера модели; L – разреженная обработка термов в документах; M – обеспечение отказоустойчивости.

Column names explanation: A – framework or computation strategy; B – core programming language; C – distributed storage or processing of text collections; D – synchronous parallel processing; E – asynchronous parallel processing; F – storage of document parameters in external memory;G – online processing; H – distributed or optimized model storage; I – sparse model storage; J – model sparse initialization; K – dynamic model resizing; L – sparse processing of terms in documents; M – fault tolerance.

Список литературы / References

[1]. Hofmann T. Probabilistic Latent Semantic Indexing. In Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999, pp. 50-57.

[2]. Blei D., Ng A., Jordan M. Latent Dirichlet Allocation. Journal of Machine Learning Research, vol. 3, 2003, pp. 993-1022.

[3]. Kochedykov D., Apishev M., Golitsyn L., Vorontsov K. Fast and Modular Regularized Topic Modeling. In Proc. of the 21st Conference of Open Innovations Association (FRUCT), 2017, pp. 182-193.

[4]. Воронцов К.В. Аддитивная регуляризация тематических моделей коллекций текстовых документов. Доклады РАН, том 455, №3, 2014, стр. 268–271 // Vorontsov K.V. Additive regularization for topic models of text collection. Doklady Mathematics. vol. 89, №3, 2014, pp. 301-304.

[5]. Vorontsov K.V., Potapenko A.A. Additive regularization of topic models. Machine Learning, Special Issue on Data Analysis and Intelligent Optimization with Applications, vol. 101, no. 1, 2015, pp. 303-323.

[6]. Dempster A.P., Laird N.M., Rubin D.B. Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society. Series B (Methodological), vol. 39, no. 1, 1977, pp. 1-38.

[7]. Teh Y.W., Newman D., Welling M. A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation. In Proc. of the 19th International Conference on Neural Information Processing, 2006, pp. 1353-1360.

[8]. Steyvers M., Griffiths T. Finding scientific topics. Proceedings of the National Academy of Sciences of the United States of America, vol. 101, Suppl. 1, 2004, pp. 5228-5235.

[9]. Воронцов К.В., Потапенко А.А. Модификации EM-алгоритма для вероятностного тематического моделирования. Машинное обучение и анализ данных, том 1, № 6, 2013 г., стр. 657–686 / Vorontsov K.V., Potapenko A.A. EM-like Algorithms for Probabilistic Topic Modeling. Machine Learning and Data Analysis, vol. 1, no. 6, 2013, pp. 657-686 (in Russian).

[10]. Asuncion A., Wekking M., Smyth P., Teh Y. W. On Smoothing and Inference for Topic Models. In Proc. of the International Conference on Uncertainty in Artificial Intelligence, 2009, pp. 27-34.

[11]. Newman D., Asuncion A., Smyth P., Welling M. Distributed Algorithms for Topic Models. The Journal of Machine Learning Research, vol. 10, 2009, pp. 1801-1828.

[12]. Wang Y., Bai H., Stanton M., Chen W., Chang E. PLDA: Parallel Latent Dirichlet Allocation for Large-Scale Applications. Lecture Notes in Computer Science, vol. 5564, 2009, pp. 301-314.

[13]. Asuncion A., Smyth P., Welling M. Asynchronous Distributed Estimation of Topic Models for Document Analysis. Statistical Methodology, vol. 8, no. 1, 2010, pp. 3-17.
[14]. Liu Z., Zhang Y., Chang E., Sun M. PLDA+: Parallel Latent Dirichlet Allocation withData Placement and Pipeline Processing. ACM Transactions on Intelligent Systems and Technology, vol. 2, issue 3, article no. 26.
[15]. Smola A., Narayanamurthy S. An architecture for parallel topic models // Proceedings of the VLDB Endowment, 2010. Vol. 3, Issue 1-2. Pp. 703-710.
[16]. Řehůřek R., Sojka P. Software Framework for Topic Modelling with Large Corpora. In Proc. of the LREC 2010 Workshop on New Challenges for NLP Frameworks, 2010, pp. 45-50.
[17]. Zhai K., Boyd-Graber J., Asadi N., Alkhouja M. Mr. LDA: A Flexible Large Scale Topic Modeling Package using Variational Inference in MapReduce. In Proc, of the 21st International Conference on World Wide Web, 2012, pp. 879-888.
[18]. Qiu Z., Wu B., Wang B., Yu L. Collapsed Gibbs Sampling for Latent Dirichlet Allocation on Spark. Proceedings of Machine Learning Research, vol. 36, 2014, pp. 17-28.
[19]. Wang Y., Zhao X., Sun Z., Yan H., Wang L., Jin Z., Wang L., Gao Y., Law C., Zeng J. Peacock: Learning Long-Tail Topic Features for Industrial Applications. ACM Transactions on Intelligent Systems and Technology, 2015, vol. 6, no. 4, article no. 47.
[20]. Yuan J., Gao F., Ho Q., Dai W., Wei J., Zheng X., Xing E., Liu T., Ma W. LightLDA: Big Topic Models on Modest Computer Clusters. In Proc. of the 24th International Conference on World Wide Web, 2015, pp. 1351-1361.
[21]. Zhao B., Zhou H., Li G., Huang Y. ZenLDA: An Efficient and Scalable Topic Model Training System on Distributed Data-Parallel Platform. Big Data Mining and Analytics, vol. 1, issue 1, 2018, pp. 57-74.
[22]. Vorontsov K., Frei O., Apishev M., Romov P., Suvorova M., Yanina A. Non-Bayesian Additive Regularization for Multimodal Topic Modeling of Large Collections. In Proc. of the 2015 Workshop on Topic Models: Post-Processing and Applications, 2015, pp. 29-37.
[23]. Frei O., Apishev M. Parallel Non-blocking Deterministic Algorithm for Online Topic Modeling. Communications in Computer and Information Science, vol. 661, 2016, pp. 132-144.
[24]. Zeng J., Liu Z., Cao X. Fast Online EM for Big Topic Modeling. IEEE Transactions on Knowledge and Data Engineering, vol. 28, issue 3, 2016, pp. 675-688.
[25]. Hoffman M., Blei D., Bach F. Online Learning for Latent Dirichlet Allocation. In Proc. of the 23rd International Conference on Neural Information Processing Systems, vol. 1, 2010, pp. 856-864.
[26]. Vowpal Wabbit ML library. Available at: https://github.com/JohnLangford/vowpal_wabbit, accessed 15.01.2020.
[27]. White T. Hadoop: The definitive guide. O'Reilly Media, Inc., 2012, 688 p.
[28]. Zaharia M., Chowdhury M., Franklin M. Spark: Cluster Computing with Working Sets. In Proc. of the 2nd USENIX Conference on Hot Topics in Cloud Computing, 2010, 10 p.
[29]. MPI: A Message-Passing Interface Standard, Version 3.0. Forum Message Passing Interface, 2012. Available at: https://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf, accessed 15.01.2020.
[30]. Dean J., Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. In Proc. of the 6th Symposium on Operating Systems Design and Implementation, 2004, pp. 137-149.
[31]. Petuum ML platform. Available at: http://www.petuum.com, accessed 15.01.2020.
[32]. Spark GraphX library. Available at: https://spark.apache.org/graphx, accessed 15.01.2020.

Информация об авторах / Information about authors

Мурат Азаматович АПИШЕВ – аспирант факультета вычислительной математики и кибернетики, кафедра математических методов прогнозирования. Сфера научных интересов: машинное обучение, обработка естественного языка, семантические векторные представления текста, тематическое моделирование, параллельные алгоритмы тематического моделирования.

Murat Azamatovich APISHEV – postgraduate student at Faculty of Computational Mathematics and Cybernetics, Department of Mathematical Methods of Forecasting. Research interests: machine learning, natural language processing, text embeddings, topic modeling, parallel algorithms of topic modeling.

DOI: 10.15514/ISPRAS-2020-32(1)-9

В ожидании нативных архитектур СУБД на основе энергонезависимой основной памяти

С.Д. Кузнецов, ORCID: 0000-0002-8257-028X <kuzloc@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, Москва, ул. А. Солженицына, д. 25
Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1
Московский физико-технический институт,
141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9
НИУ «Высшая школа экономики»,
101978, Россия, Москва, ул. Мясницкая, д. 20
Российский экономический университет имени Г.В. Плеханова,
117997, Москва, Стремянный пер., 36

Аннотация. Многие специалисты в области управления данными считают, что появление доступной для практического использования энергонезависимой байт-адресуемой основной памяти (Non-Volatile Main Memory, NVM) приведет к появлению нового вида сверхскоростных систем управления базами данных (СУБД) с одноуровневым хранением данных (NVM-ориентированных СУБД). Однако, как отмечается во введении данной статьи, число исследователей, активно занимающихся исследованиями архитектур NVM-ориентированных СУБД, за последние годы не возросло. Наибольшую активность проявляют молодые исследователи, не боящиеся рисков, которые, безусловно, имеются в этой новой области. Во втором разделе статьи рассматривается состояние дел в области аппаратных средств NVM. Анализ показывает, что NVM в форм-факторе DIMM уже стали реальностью, и что в ближайшем будущем можно ожидать появления на рынке NVM-DIMM со скоростью обычной DRAM и стойкостью, близкой к стойкости жестких дисков. Третий раздел посвящен обзору родственных работ, среди которых наиболее продвинутыми представляются работы молодых исследователей Джоя Арулраджа и Исмаила Укида. В четвертом разделе мы утверждаем и обосновываем, что работы, выполненные к настоящему времени в области NVM-ориентированных СУБД, не привели к появлению чистой архитектуры. Этому мешает анализируемый нами набор ограничивающих факторов. В связи с этим, в пятом разделе мы приводим набросок «чистой» архитектуры NVM-ориентированной СУБД, на выбор которой влияют только стремление к простоте и эффективности. В заключение подводятся итоги статьи и утверждается потребность в дополнительном исследовании многих аспектов «чистой» архитектуры NVM-ориентированной СУБД.

Ключевые слова: энергонезависимая байт-адресуемая основная памяти; система управления базами данных; одноуровневое хранение данных; управление транзакциями; индексация; оптимизация запросов.

Для цитирования: Кузнецов С.Д В ожидании нативных архитектур СУБД на основе энергонезависимой основной памяти. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 153-180. DOI: 10.15514/ISPRAS–2020–32(1)–9

In anticipation of native DBMS architectures based on non-volatile main memory

S.D. Kuznetsov, ORCID: 0000-0002-8257-028X <kuzloc@ispras.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia
Moscow Institute of Physics and Technology (State University),
9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia
National Research University, Higher School of Economics
20, Myasnitskaya Ulitsa, Moscow, 101978, Russia
Plekhanov Russian University of Economics,
36, Stremyanny lane, Moscow, 117997, Russia

Abstract. Many experts in the field of data management believe that the emergence of non-volatile byte-addressable main memory (NVM) available for practical use will lead to the development of a new type of ultra-high-speed database management systems (DBMS) with single-level data storage (native in-NVM DBMS). However, the number of researchers who are actively engaged in research of architectures of native in-NVM DBMS has not increased in recent years. The most active researchers are PhD students that are not afraid of the risks, which, of course, exist in this new area. The second section of the article discusses the state of the art in NVM hardware. The analysis shows that NVM in the DIMM form factor has already become a reality, and that in the near future we can expect the appearance on the market NVM-DIMMs with the speed of conventional DRAM and endurance close to that of hard drives. The third section is devoted to the review of related works, among which the works of young researchers are the most advanced. In the fourth section, we state and justify that the work performed so far in the field of in-NVM DBMS, did not lead to the emergence of a native architecture. This is hampered by the set of limiting factors analyzed by us. In this regard, in the fifth section, we present a sketch of the native architecture of an in-NVM DBMS, the choice of which is influenced only by the goals of simplicity and efficiency. In conclusion, we summarize the article and argues the need for additional research into many aspects of the native architecture of an in-NVM DBMS.

Keywords: non-volatile byte-addressable main memory, storage-class memory, database management system, single-level storage, transaction management, indexing, query optimization

For citation: Kuznetsov S.D. In anticipation of native DBMS architectures based on non-volatile main memory. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 153-180 (in Russian). DOI: 10.15514/ISPRAS–2020–32(1)–9

1. Введение

Энергонезависимая основная память (Non-Volatile Main Memory, NVM, Persistent Main Memory, Storage-Class Memory) становится все более реальной. К массовому выпуску соответствующих чипов уже приступили гиганты электронной промышленности, такие как Intel и Samsung [1-3]. Расширяется набор технологий, используемых даже внутри одной компании (наиболее яркий пример представляет Intel [1, 3]) для производства чипов NVM.

Однако, как это ни странно, число исследователей, активно занимающихся исследованиями архитектур NVM-ориентированных СУБД, структур данных, методов и алгоритмов, обеспечивающих построение таких систем, за последние два-три года не увеличилось. На общем фоне выделяются два молодых исследователя: Джой Арулрадж (Joy Arulraj) и Исмаил Укид (Ismail Oukid). Первый из них получил степень PhD [4] в 2018 г. в университете Карнеги-Меллон, где он занимался исследованиями в группе баз данных вместе с Эндрю Павло (Andrew Pavlo), который был его руководителем в

аспирантуре, и в соавторстве, с которым сделаны основные публикации Арулраджа по тематике NVM-ориентированных СУБД (в частности, [5,6]). Стоит заметить, что Э. Павло, чрезвычайно заметный в настоящее время в сообществе баз данных, сам защитил диссертацию всего лишь в 2013 г. С 2018 г. Дж. Арулрадж работает в Технологическом институте Джорджии. В конце апреля 2019 г. ACM SIGMOD наградил Арулраджа премией им. Джима Грея за лучшую диссертационную работу; премия была вручена Джою на Международной конференции ACM SIGMOD по управлению данными в июле 2019 г. [7].

Исмаил Укид защитил диссертацию на степень PhD [8] в 2017 г. в Дрезденском техническом университете (Германия), где он учился и работал на кафедре баз данных (Дрезденская группа систем баз данных). Одновременно с этим он работал с компаниями Intel и SAP, и после защиты диссертации полностью перешел в SAP. Стоит заметить, что И. Усид сам пробился на крупнейшие конференции в области баз данных и в самые рейтинговые журналы (например, [9, 10]). Заслуживает внимания также тот интересный факт, что И. Укид (как и Дж. Арулрадж) вообще не занимался исследованиями в области баз данных до начала работы над диссертацией.

Чем же можно объяснить это сравнительно равнодушное отношение «взрослых» представителей сообщества баз данных к актуальной и перспективной теме NVM-ориентированных СУБД? Ответить на этот вопрос можно по-разному.

Во-первых, данные о реальных характеристиках промышленных образцов MVM недостаточны и противоречивы. Производители соответствующих аппаратных средств недостаточно информируют открытым образом научную общественность об их характеристиках, а достоверность цифр, публикуемых популярных Интернет-изданиях, невозможно проверить.

Во-вторых, видимо, из сиюминутных маркетинговых соображений, вендоры NVM на первых порах предпочитают предлагать продукты, основанные на технологии NVM, в форм-факторе блочных устройств внешней памяти, а не чипов энергонезависимой байт-адресуемой основной памяти. (Текущему состоянию аппаратных средств NVM посвящается следующий раздел статьи.)

В-третьих, известно, что на выбор направлений научных исследований (по крайней мере, в области компьютерных наук) в значительной степени влияет возможность получения финансирования (чаще всего в виде грантов). Во всем мире фонды, поддерживающие научные исследования, требуют гарантий получения при выполнении проекта практических результатов, близких к возможности реального использования. В настоящее время относительно проектов в направлении NVM-ориентированных СУБД таких гарантий никто дать не может.

Наконец, следует заметить, что заслуженные члены сообщества баз данных на самом деле в своих исследованиях затрагивают вопросы, прямо относящиеся к архитектуре и алгоритмам NVM-ориентированных СУБД. Формально эти исследования относятся к области СУБД с хранением баз данных в обычной основной памяти (in-memory СУБД), но при этом исследователи абстрагируются от того обстоятельства, что содержимое традиционной основной памяти утрачивается при отключении питания.

Во втором разделе статьи мы кратко обсудим состояние дел в области аппаратных средств NVM. Краткий обзор работ, прямо или косвенно связанных с архитектурой, структурами данных и алгоритмами NVM-ориентированных СУБД приводится в разд. 3.

По мнению автора этой статьи, все публикации, посвященные NVM-ориентированным СУБД, обладают одним общим свойством, которое можно считать достоинством или недостатком: их авторы не решаются замахнуться на полноценную разработку СУБД с одноуровневой системой хранения, когда базы данных целиком и полностью хранятся в энергонезависимой основной памяти, и никакая внешняя блочная память вообще не

используется. Причины того, что «чистую» архитектуру NVM-ориентированной СУБД до сих пор никому предложить не удалось, рассматриваются в разд. 4.

С одной стороны, расчет на использование иерархически организованной системы хранения с включением NVM в слой между энергозависимой основной памятью и блочной флэш-памятью (solid-state disks, SSD) ближе к сегодняшней реальности. С другой стороны, именно полный переход к одноуровневой энергонезависимой системе хранения может позволить упростить организацию СУБД и значительно увеличить их эффективность. В разд. 5 приводится набросок «чистой» архитектуры такой СУБД.

2. Аппаратные средства NVM

За прошедшие пару лет основные технологические тренды NVM практически не изменились. Как и раньше (см., например, [11]), основные компании, заинтересованные в производстве NVM, делают ставки на методы энергонезависимого хранения данных с побайтовой адресацией на основе физических эффектов

- фазовых переходов (Phase-Change Memory, PCM),
- изменения сопротивления диэлектриков (Resistive Random-Access Memory, ReRAM) и
- переноса спинового момента (Spin-Torque-Transfer Memory, STT-RAM).

Последняя в этом списке технология относится к более общему классу магнитно-резистивных решений (magnetoresistive random-access memory, MRAM). В конечном счете, все разновидности MRAM, как и ReRAM, основаны на изменении сопротивления; в случае MRAM сопротивление изменяется при изменении ориентации намагниченности.

В табл. 1 приведены примерные характеристики NVM разновидностей PCM, ReRAM и STT-RAM. К сожалению, в открытом доступе не удается найти значения таких характеристик от производителей и/или разработчиков NVM. В таблице собраны неофициальные показатели, найденные в различных публикациях ([3, 5, 12, 13]) за период с 2016 по 2019 гг. Местами они довольно сильно различаются, но общий вывод сделать можно: на сегодняшний день ближе всех к DRAM по скоростным характеристикам находится STT-RAM.

Табл. 1. Приблизительные характеристики разных видов NVM
Table 1. Approximate characteristics of different types of NVM

	Источник	Год публикации	PCM	STT-RAM	ReRAM	DRAM
Время чтения (нс)	[3]	2019	20-70	10-30	10	10-50
	[10]	2018	250	20	100?	100
	[5]	2017	50	20	100	60
	[13]	2016	50	10	10	50
Время записи (нс)	[3]	2019	50-500	13-95	1-100	10-50
	[10]	2018	250	20	100?	100
	[5]	2017	150	20	100	60
	[13]	2016	500	50	50	50
Стойкость	[3] *	2019	$1\text{-}10^8$	10^{15}	10^{10}-10^{12}	$> 10^{17}$
	[10] DWPD **	2018	100	↑	40?	↑
	[5] *	2017	10^{10}	10^{15}	10^8	$> 10^{16}$
	[13] *	2016	10^8-10^9	$> 10^{15}$	10^{11}	$> 10^{15}$
* Число циклов записи						
** DWPD, Drive Writes Per Day, допустимый объем суточной записи относительно емкости самого устройства						

Немного смущает указываемое одним из авторов различное время чтения и записи у STT-RAM. Нигде в литературе не удается найти какие-либо сведения об обязательности наличия этого явления. Поскольку двое других авторов указывают для STT-RAM одинаковое время чтения и записи (и практически такое же, как у DRAM), далее в этой статье будем считать, что так оно и есть. Скорее всего, ближайшее будущее покажет, насколько оправдан этот оптимизм.

С разработкой разных видов NVM связан ряд компаний. В частности, лидером в области технологии NVM на основе фазовых переходов принято считать компанию Micron Technology [14]. Технологией ReRAM наиболее активно занимаются компании Panasonic [15] и Crossbar [16]. Что касается разработки STT-RAM, можно выделить компании Everspin Technologies [17] и Crocus Technology [18] (и ее российскую «дочку» Крокус Наноэлектроника [19]). Однако, несмотря на значительный вклад этих компаний в развитие технологий NVM, ни одна из них не является вендором мирового уровня, способным (или хотя бы желающим) производить доступные для широкого использования модули энергонезависимой памяти.

Реальным прорывом в этом направлении является начало массового производства продуктов категории NVM компаниями Intel и Samsung [1]. Еще в 2015 г. компании Intel и Micron объявили о создании технологии 3D XPoint – NVM на основе PCM (для разработки технологии NVM на основе фазовых переходов в 2006 г. компании Intel и Micron создали специальное совместное предприятие IM Flash Technologies; 3D XPoint – не первая, но самая успешная разработка IM Flash Technologies).

В марте 2017 г. Intel объявила о промышленном производстве модулей 3D XPoint под брендом Optane. По маркетинговым соображениям (а также, по-видимому, из-за того, что по своим скоростным характеристикам NVM Optane занимает нишу между DRAM и флэш-памятью) на первых порах для продвижения Optane Intel использовала два решения: кэш внутри традиционных дисковых устройств и твердотельные диски целиком на основе Optane. Оба решения оказались весьма эффектными, и о SSD на основе Optane мы еще поговорим ниже в этом разделе.

В августе 2018 г. компания Intel объявила о начале производства DIMM на основе Optane. С одной стороны, DIMM является общепринятым форм-фактором современных энергозависимых модулей основной памяти. С другой стороны (увы!), DIMM на основе Optane невозможно использовать вместе с обычными процессорами Intel. Одновременно с объявлением DIMM на основе Optane Intel анонсировала новую линейку процессоров Cascade Lake AP (Advanced Processor) Xeon CPU, в которых эта возможность будет обеспечена. Так что использовать Intel Optane NVM в качестве замены RAM пока проблематично.

Со своей стороны, компания Samsung в начале 2018 г. объявила о начале производства SSD на основе собственной технологии Z-NAND (многослойная NAND, или 3D NAND). Мы не беремся судить об особенностях этой технологии, но утверждается, что она позволила повысить скорость чтения из внешней памяти чуть ли не в 10 раз по сравнению с обычными SSD на основе флэш-памяти. В [1] приводятся следующие характеристики SSD на основе Z-NAND от Samsung и SSD на основе Optane от Intel. При емкости SSD в 800 Гб Samsung SZ985 Z-NAND поддерживает в секунду 750,000/170,000 произвольных блочных обменов с внешней памяти по чтению или записи соответственно. При той же емкости Intel Optane P4800X поддерживает в секунду 550,000/500,000 таких обменов. Вполне конкурентоспособные результаты, несмотря на принципиальное различие применяемых технологий.

Тему быстрых устройств внешней памяти и связанную с их появлением потребность в пересмотре архитектуры и алгоритмов организации СУБД мы кратко затрагивали в [11]. Понятно, что эта потребность только усиливается в связи с новыми достижениями Intel и

Samsung, но в данной статье это обсуждаться не будет. Основные же события, которые к этой статье имеют непосредственное отношение, произошли в 2019 г.

На конференции International Solid-State Circuits в феврале 2019 г. разработчики из компании Intel заявили о готовности компании начать промышленный выпуск модулей энергонезависимой памяти STT-MRAM [20]. Интересно, что на той же конференции в выступлении другой группы разработчиков было заявлено о готовности Intel к разработке моделей памяти и на основе технологии ReRAM [21]. Intel планирует применять эту более дешевую и менее быструю память в областях Интернета вещей и автомобилестроения. Хотя ко времени написания этой статьи официальных заявлений компании Intel, анонсирующих соответствующие продукты, еще не было, думается, что указанные выступления на конференции могли состояться только с полного благословения руководства компании. Так что совсем скоро мы сможем получить от Intel высокопроизводительные модули памяти STT-MRAM.

А может быть, и не от Intel, потому что буквально через две недели после выступления разработчиков из Intel компания Samsung официально объявила [2] о начале поставок модулей энергонезависимой памяти на основе своей технологии eMRAM на основе STT. Пока говорится, что модули eMRAM будут использоваться в микроконтроллерах, устройствах Интернета вещей и почему-то в системах искусственного интеллекта, но если технология успешно себя зарекомендует, то Samsung сможет обеспечит и выпуск моделей памяти в формате DIMM.

Общим выводом этого раздела является то, что высокопроизводительная энергонезависимая основная память стала реальностью, и работа в области NVM-ориентированных СУБД приобрела еще большую актуальность.

3. Родственные работы

Как отмечалось во введении, в мире проводится недостаточно активная исследовательская работа по поиску архитектур, структур данных, алгоритмов и методов, необходимых и достаточных для построения полноценных NVM-ориентированных СУБД. Несмотря на это, число публикаций, относящихся к специфике управления данными в энергонезависимой основной памяти, достаточно велико. Однако в подавляющем большинстве этих публикаций рассматриваются частные вопросы, в отрыве от общей задачи построения СУБД нового поколения.

Возможно, для полноты картины стоило бы написать обзор всех доступных работ, так или иначе касающихся использования NVM в системах управления данными. Но это выходит за пределы текущих интересов автора. В этой статье нас интересуют архитектурные и алгоритмические черты NVM-ориентированных СУБД. Работ, посвященных решению задачи именно в этой постановке, удивительно мало. Именно их обзору посвящен этот раздел.

3.1 Использование NVM в существующих СУБД

Стратис Виглас (Stratis D. Viglas) из Эдинбургского университета в своей краткой статье 2015 г. [22] опирается на предположение о том, что доступная энергонезависимая основная память обеспечивает скорость доступа, близкую к возможностям современной DRAM, но при этом ассиметричная: операции чтения выполняются быстрее операций записи. Поэтому, по мнению автора, NVM не сможет заменить ни DRAM, ни HDD. Предлагаются два пути интеграции NVM в иерархию сред хранения данных: путем использования в качестве устройства внешней памяти и на основе подключения таким же способом, что и основная память. Автор отмечает недостаточную исследованность области управления данными с использованием NVM и призывает активизировать работу в этом направлении.

Навид Уль Мустафа (Naveed Ul Mustafa) и др. из университета Билькент (Анкара, Турция) и Барселонского суперкомпьютерного центра в [23] описывают свою работу по замене подсистемы хранения данных в СУБД PostgreSQL, изначально ориентированной на использование HDD, на разработанную ими подсистему, ориентированную на применение NVM. Для обеспечения доступа к NVM авторы использовали файловую систему PMFS (Persistent Memory File System)[1], которая отображает файлы, хранящиеся в энергонезависимой основной памяти, в виртуальную память работающего с ними процесса. Тем самым, хотя обеспечивался «прямой» доступ к NVM без потребности переписи данных из NVM в RAM и обратно, на уровне файловой системы поддерживается блочная структура NVM. Кроме изменений подсистемы хранения данных, в PostgreSQL были ликвидированы за ненадобностью кэш в основной энергозависимой памяти и журнал REDO.

Авторы сравнивали производительность исходного варианта PostgreSQL с использованием SSD и измененного варианта системы с использованием эмулируемой NVM на основе бенчмарка TPC-H (моделирующего рабочую нагрузку, свойственную системам поддержки принятия решений). Результаты не слишком вдохновляют: время выполнения запросов сократилось в среднем всего на 20%. По-видимому, это связано с тем, что фактически сохраняется иерархическая система хранения данных, и для доступа к очередной порции данных из базы данных системе всегда требуется обращение к файловой системе.

В [24] Михня Андрей (Mihnea Andrei) и др. из компании SAP SE и немецкого подразделения компании Intel представляют первую попытку внедрить использование NVM в СУБД HANA. HANA – это заново разработанная в SAP производственная СУБД с поколоночным хранением таблиц в основной памяти (in-memory columnar DBMS). HANA поддерживает как транзакционную, так и аналитическую рабочую нагрузку, причем в приложениях SAP даже в транзакционной рабочей нагрузке более 80% операций составляет чтение данных, так что поколоночное хранение таблиц в основной памяти является оправданным.

В HANA поддерживается высокий уровень доступности и надежности данных на основе резервного копирования, репликации и журнализации в режиме REDO. Для всего этого используется дисковая внешняя память. Для полного перехода на NVM потребовалась бы серьезная переделка системы, и в своей первой попытки внедрения NVM разработчики SAP на это не решились. По техническим соображениям в NVM, используемой в форм-факторе DIMM, размещаются лишь редко изменяемые крупные фрагменты описателей таблиц.

В статье не приводятся какие-либо убедительные показатели преимуществ полученного варианта системы, но отмечается, что, во-первых, общая организация HANA хорошо подходит для перехода к полноценному использованию NVM. Во-вторых, для этого предстоит решить массу технических задач. В-третьих, пути к будущему переводу HANA на NVM намечены в статьях Исмаила Укида и др. (авторы, в число которых сам Укид не входит, ссылаются на статьи [30, 31, 33]). На работах Укида мы остановимся в конце этого раздела.

Александр ван Ренен и др. из Мюнхенского технического университета (Германия) и компании Fujitsu в [25] описывают свою разработку подсистемы хранения данных с использованием NVM, которую они оправдывают следующей цитатой из недавнего интервью Майкла Стоунбрейкера (Michael Stonebraker)[2]: модели памяти NVM «недостаточно быстры, чтобы заменить основную память, и они недостаточно дешевы, чтобы заменить диски, и они недостаточно дешевы, чтобы заменить флэш-память».

[1] https://github.com/linux-pmfs/pmfs

[2] https://www.nextplatform.com/2017/08/15/hardware-drives-shape-databases-come/

Поскольку в интервью Стоунбрейкер отвечает на вопрос о 3D XPoint или ReRAM, с ним нельзя не согласиться, но, что бы он сказал о технологии STT-RAM 2019 года?

Тем не менее, следуя этому указанию, авторы статьи предлагают архитектуру системы хранения данных, в иерархии которой NVM занимает место между DRAM и SSD. Утверждается, что в результате ряда технических приемов, описываемых в статье, авторам удалось получить решение, конкурентоспособное по отношению к in-memory СУБД и СУБД, использующим NVM в качестве единственного уровня хранения данными. Возможно, подход авторов действительно соответствует соотношению скоростей DRAM, ReRAM и SSD, имевшихся в 2018 г. Будет ли он хорош, например, при наличии быстрых SSD на основе Intel Optane или Samsung Z-NAND, совершенно непонятно.

Заканчивают авторы статью цитатой из книги еще одного классика баз данных покойного Джима Грея[3], которой мы тоже руководствуемся в своей работе: «Не позволяйте себя дурачить книгам о сложности или всяким сложным и малопонятным алгоритмам, которые вы найдете в этой книге или где-то еще. Хотя нет учебников по простоте, простые системы работают, а сложные нет».

На конец обзора мы оставили работы Джоя Арулраджа и Исмаила Укида, которые, по нашему мнению, в настоящее время больше других исследователей продвинулись в разработке архитектур, алгоритмов и методов, которые пригодны для построения NVM-ориентированных СУБД.

3.2 Использование NVM в СУБД Peloton

Джой Арулрадж выполнил все работы, которые составили его диссертацию, в рамках проекта Peloton[4], лидером которого является молодой и чрезвычайно энергичный исследователь и преподаватель Энди Павло (он же был научным руководителем Джоя). В проекте участвуют как представители старшего поколения исследователей университета Карнеги-Меллон (Тодд Моури (Todd C. Mowry) и Энтони Томашич (Anthony Tomasic)), так и около 20 студентов и аспирантов, в число которых до защиты диссертации входил и Джой Арулрадж. Как уже отмечалось, в настоящее время Джой работает доцентом в Технологическом институте Джорджии, но во всех его публикациях, кроме [7], он аффилирован с университетом Карнеги-Меллон.

Peloton – это новая «самоуправляемая» in-memory СУБД, в которой все аспекты поведения системы контролируются интегрированным планировщиком, который не только оптимизирует систему для выполнения текущей рабочей нагрузки, но и прогнозирует будущую рабочую нагрузку, чтобы система к ней подготовиться. Для этого в систему интегрированы черты искусственного интеллекта, в частности, машинного обучения.

Но кроме этого, в Peloton реализована поддержка энергонезависимой основной памяти. Судя по публикациям, это сделано, главным образом, стараниями Джоя Арулраджа при содействии и под руководством Энди Павло. Результаты, полученные Арулраджем, опубликованы в многочисленных статьях [5-6, 26-29], но окончательным текстом, в котором эти результаты четко изложены и упорядочены, является его диссертация [4], которой мы и будем следовать в своем обзоре.

Как отмечает автор, в его диссертации представлены разработка и реализация СУБД Peloton, ориентированной на использование NVM. Полученная архитектура системы

[3] Jim Gray and Andreas Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, 1993

[4] https://pelotondb.io/

обладает несколькими важными преимуществами перед другими существующими системами, из которых в данной статье нам наиболее интересны следующие:

- в архитектуре подсистемы хранения данных используются свойства долговечности и байтовой адресации NVM; обеспечивается экономное использование энергонезависимой памяти и увеличивается срок ее эксплуатации за счет уменьшения числа записей;
- в ней применяется новый протокол журнализации и восстановления «журнализация с отложенной записью в журнал» (Write-Behind Logging, WBL), позволяющий обеспечить высокий уровень доступности;

Обсудим эти особенности архитектуры более подробно.

3.2.1 Подсистемы хранения данных

Для случая, когда для хранения данных используется только NVM, а внешняя память вовсе не используется, были разработаны и оптимизированы три разных подсистемы хранения данных:

- подсистема с обновлением данных прямо на текущем месте их хранения (In-Place Updates Engine, NVM-InP);
- подсистема с созданием новой копии данных при их обновлении (Copy-on-Write Updates Engine, NVM-CoW) и
- подсистема с журнально-структурированными обновлениями (Log-structured Updates Engine, NVM-Log).

Если не вдаваться в технические детали, при разработке подсистемы NVM-InP применялись достаточно традиционные методы с учетом специфики NVM – на основе журнализации с упреждающей записью изменений (Write-Ahead Log, WAL). Основным отличием от случая, когда данные хранятся во внешней памяти, является отсутствие потребности в повторном выполнении операций при перезапуске системы после сбоя, поскольку результаты любой зафиксированной транзакции гарантированно находятся в долговременной NVM. Журнал поддерживается в виде списка в NVM, и «верхняя» часть списка, соответствующая уже зафиксированным транзакциям, освобождается.

Поэтому при выполнении операции INSERT в журнал записывается только указатель на заново вставляемую строку, позволяющий удалить ее при откате транзакции. При выполнении операции DELETE в журнал тоже записывается только указатель за удаляемую строку; память, занимаемая этой строкой, освобождается только после фиксации соответствующей транзакции, что позволяет при откате транзакции восстановить все индексы на основе сохраненного в журнале указателя. Понятно, что при выполнении операции UPDATE приходится сохранять в журнале предыдущие значения изменяемых полей строки.

Подсистема NVM-CoW, фактически, является системой хранения с поддержкой не более двух копий всех элементов данных, включая строки таблиц и узлы B+-деревьев (индексов). Для каждой таблицы и каждого B+-дерева в NVM поддерживаются два справочника – текущий (current), сохраняющий ссылки на копии строк таблиц или узлов B+-дерева, созданные зафиксированными транзакциями, и измененный (dirty), который содержит ссылки на соответствующие объекты, созданные незафиксированными транзакциями.

При фиксации транзакции все измененные справочники, содержащие ссылки на копии объектов, которые были созданы при выполнении данной транзакции, делаются текущими. Для этого насильственно выталкиваются в NVM все кэши процессора, и справочник объявляется текущим. Для этого для каждого справочника в NVM поддерживается специальная «мастер-запись», содержимое которой показывает, какой

справочник является текущим. Мастер-запись изменяется атомарной командой записи. Кстати, для экономии операций выталкивания кэша и обновления мастер-записей в [4] транзакции фиксируются не по одной, а пакетами.

Понятно, что при таком управлении хранением данных при перезапуске СУБД после сбоя не требуется откатывать какие-либо транзакции, а значит, не требуется и поддерживать журнал UNDO в NVM. Для отката транзакции при наличии работоспособной системы достаточно запоминать в локальной памяти транзакции список координат в измененной справочнике копий объектов, созданных данной транзакцией.

Заметим еще по поводу NVM-CoW, что Арулрадж проводит аналогию между используемым подходом и «теневым» механизмом, который использовался в System R для установки контрольных точек базы данных [30]. По нашему мнению, этот подход в большей степени напоминает двухверсионный двухфазный метод синхронизационных блокировок (Two Version 2PL, 2V2PL), описанный в пятой главе [31]. Поддержку двух копий на уровне строк таблицы трудно представить без соответствующей поддержки изолированности транзакций: если некоторая транзакция изменяет некоторую строку таблицы, то до конца этой транзакции никакая другая транзакция не должна иметь возможности изменять эту строку, но должна иметь возможность прочитать копию этой строки, указатель на которую содержится в текущем справочнике. К сожалению, в [4] и других публикациях Арулраджа ничего не говорится об этом аспекте NVM-CoW, из-за чего реализация этой подсистемы остается непонятной.

Третья подсистема NVM-Log основана на использовании LSM-деревьев (Log-Structured Merge-tree), введенных в 1996 г. Патриком О'Нейлом (Patrick O'Neil) и др.[5] Вся база данных поддерживается в виде последовательности специальных хранилищ – MemTable. Изменения (строка таблицы при выполнении операции INSERT, новое содержимое изменяемых столбцов строки при выполнении UPDATE, пометка об удалении строки для DELETE) заносятся в текущую изменяемую MemTable. Когда ее заполнение достигает установленного порога, создается новая текущая MemTable, а предыдущая становится неизменяемой. Все MemTable хранятся в NVM, неизменяемые – сливаются. Для доступа к базе данных поддерживаются индексы – B+-деревья и фильтры Блюма. Для восстановления базы данных поддерживается журнал UNDO WAL, в котором сохраняются не значения, а только ссылки с места их хранения в MemTable.

Для проведения экспериментов была реализована специальная «легковесная» СУБД с использованием эмулятора NVM. Для сравнения характеристик подсистем применялись тестовые наборы YCSB (Yahoo! Cloud Serving Benchmark)[6] и TPC-C[7]. Основным выводом является то, что для рабочих нагрузок с большим числом изменений лучше работает NVM-InP, и для рабочих нагрузок с преобладание операций чтения – NVM-CoW. Эти результаты согласуются с интуицией.

3.2.2 Откладываемая запись в журнад

Исторически в СУБД, использующих журнализацию, применяются протоколы журнализации с упреждающей записью в журнал (Write Ahead Log). В СУБД категории in-memory, к которым относится Peloton, восстановление базы данных происходит путем загрузки в основную память из внешней памяти копии базы данных, соответствующей последней контрольной точке, с последующим откатом изменений, выполненных транзакциями, которые не завершились к моменту сбоя (UNDO), и повторным применением операций, выполненных транзакциями, которые не зафиксировались до

[5] The Log-Structured Merge-Tree (LSM-Tree). Acta Informatica, vol. 33, issue 4, 1996, pp. 351-385.
[6] https://en.wikipedia.org/wiki/YCSB.
[7] http://www.tpc.org/tpcc/.

установки последней контрольной точки, но успели зафиксироваться до точки сбоя (REDO). Если для управления транзакциями используется какая-либо разновидность многоверсионного протокола (Multi-Version Concurrency Control, MVCC)[8], то можно обойтись без UNDO, но REDO требуется, а значит, в журнал в долговременной памяти требуется помещать записи по поводу каждой операции изменения базы данных.

Придуманный Арулраджем протокол журнализации с откладываемой записью в журнал (Write-Behind Logging) позволяет значительно сократить расходы на журнализацию и ускорить восстановление базы данных после сбоев. Протокол рассчитан на использование в СУБД, использующих для хранения баз данных только NVM, но применяющих для служебных целей и обычную энергозависимую основную память (DRAM). На рабочей фазе выполнения транзакции результаты всех операций изменения базы данных сохраняются в DRAM в таблице измененных кортежей (Dirty Tuple Table, DTT). Эта таблица применяется для индивидуальных откатов транзакции.

При выполнении операции COMMIT (а фиксация выполняется сразу для группы транзакций) сначала все изменения каждой фиксируемой транзакции записываются в постоянное место хранения таблиц в NVM, а затем в журнал пишется запись, содержащая временную метку фиксации транзакции из данной группы, которая последней закончила запись в NVM элементов своей DTT, а также временную метку со значением, заведомо большим временной метки фиксации транзакций из следующей группы.

После сбоя в NVM заведомо содержатся все результаты зафиксированных транзакций, но, возможно, еще и некоторые результаты транзакций, не зафиксированных до момента сбоя. При использовании любого варианта MVCC эти результаты не были до сбоя видны другим транзакциям, и работу системы можно возобновить немедленно, одновременно запустив служебный процесс сборки мусора.

3.3 Исследования в Дрезденской группе систем баз данных

Дрезденская группа систем баз данных (Database Systems Group Dresden) сформировалась на основе кафедры баз данных Института системных архитектур факультета компьютерных наук Дрезденского технического университета[9]. С момента основания кафедры ее заведующим и лидером группы систем баз данных является профессор Вольфганг Лехнер (Wolfgang Lehner). Он защитил диссертацию PhD в 1998 г. в Университете Эрлангена-Нюрнберга им. Фридриха-Александра, работал в США в компании IBM, а с 2002 г. работает в Дрезденском техническом университете.

В Дрезденской группе баз данных выполняется много проектов с участием как штатных сотрудников, так и студентов, но судя по всему, напрямую с тематикой NVM-ориентированных СУБД была связана только работа Исмаила Укида, которую он выполнял во время своего пребывания в аспирантуре с 2013 по 2017 гг., где его научным руководителем являлся Лехнер. Под его же руководством Укид подготовил диссертационную работу, которую защитил в самом конце 2017 г. При выполнении своего диссертационного проекта Укид, по всей видимости, пользовался инфраструктурой группы, а также сотрудничал с исследователями компаний SAP и Intel. В частности, для проведения экспериментов на основе инфраструктуры Дрезденской группы и при активном участии студентов был реализован прототип NVM-ориентированной СУБД SOFORT [32].

[8] В [4] (а также в [5, 6]) говорится, что в Peloton используется MVCC, но нигде не уточняется, какой вариант.

[9] https://wwwdb.inf.tu-dresden.de/

После защиты диссертации Исмаил Укид сразу перешел на работу в компанию SAP (скорее всего, в группу, занимающуюся разработкой СУБД HANA). Информация о его деятельности с новом качестве пока недоступна, но за время учебы и работы в Дрезденской группе он опубликовал свои результаты в ряде интересных статей ([9-10, 32-36]), а также систематизировал их в тексте диссертационной работы [8]. Этим текстом мы и воспользуемся в данном подразделе обзора родственных работ.

Как уже отмечалось, Укидом был реализован прототип NVM-ориентированной СУБД SOFORT. Это система с поколоночным хранением таблиц, квалифицируемая автором как СУБД категории HTAP (Hybrid Transactional and Analytical Processing). Как пишет Укид, SOFORT обеспечивает эффективное выполнение аналитических рабочих нагрузок, обеспечивая конкурентоспособную поддержку транзакционных приложений. По всей видимости, выбор такого неочевидного подхода (об этой неочевидности мы поговорим в начале разд. объясняется теснымиями связями Дрезденской группы вообще и ее руководителя Лехнера с проектом гибридной поколоночной un-memory СУБД HANA компании SAP. Кстати, влияние HANA прослеживается во всей работе Укида.

В целом, SOFORT – это СУБД с одноуровневой системой хранения, использующей как NVM, так и традиционную энергозависимую основную память. Применялся эмулятор NVM компании Intel. В своей работе Исмаил Укид последовательно описывает

- свою модель программирования использованием NVM;
- методы управления энергонезависимой памятью и ее распределения для нужд СУБД;
- реализованные в SOFORT методы управления транзакциями и восстановления баз данных после сбоев, а также
- разработанный фреймворк для тестирования NVM-ориентированного программного обеспечения.

В своем обзоре мы кратко обсудим только аспекты управления транзакциями.

Для управления транзакциями в SOFORT был адаптирован, оптимизирован и реализован многоверсионный оптимистический протокол (Multi-Version Optimistic Concurrency Control, MVOCC), впервые описанный в статье Пер-Оке Ларсона (Per-Ake Larson) и др.[10]

Основными отличиями подхода, используемого в SOFORT, от этого «традиционного» MVOCC являются (1) откат любой транзакции, пытающейся заблокировать строку таблицы, уже заблокированную другой транзакцией в несовместимом режиме и (2) особая, очень дешевая обработка фиксации транзакций, которые в своей рабочей фазе выполняли только операции чтения (для только читающих транзакций обеспечивается уровень snapshot isolation[11]).

В SOFORT каждая транзакция жестко ассоциируется с аппаратным потоком управления. Поэтому в системе поддерживается столько описателей транзакций, сколько потоков поддерживает используемая аппаратура. У каждого описателя транзакции имеются две части, одна из которых (persistent) располагается в энергонезависимой памяти, а другая (transient) – в обычной основной памяти. В первой части сохраняются признак фиксации транзакции, временная метка фиксации, а также наборы ссылок на строки таблиц, созданные и удаленные данной транзакцией. Информация из «постоянной» части описателя транзакции используется для выполнения индивидуальных откатов

[10] Per-Åke Larson, Spyros Blanas, Cristian Diaconu, Craig Freedman, Jignesh M. Patel, Mike Zwilling. High-performance concurrency control mechanisms for main-memory databases. Proceedings of the VLDB Endowment, vol. 5, issue 4, 2011, pp. 298-309.

[11] https://en.wikipedia.org/wiki/Snapshot_isolation

транзакции, а также заменяет журнал UNDO при восстановлении базы данных после сбоев.

Стоит заметить, что поколоночное хранение приводит к значительному усложнению выполнения всех транзакционных операций обновления таблиц и порождает дополнительные накладные расходы. Но нужно иметь в виду, что в компании SAP, под влиянием которой выполнялась работа Укида, ориентируются на транзакционную рабочую нагрузку с очень небольшим числом операций обновления баз данных (см. подраздел 3.1).

4. Факторы, ограничивающие «чистоту» архитектуры NVM-ориентированной СУБД

Стараниями, главным образом, молодежи мы получили более ясное представление о способах использования энергонезависимой памяти в архитектуре СУБД, о возникающих проблемах и возможных способах их решения. Однако все предпринятые попытки построения архитектуры NVM-ориентированных СУБД производились в условиях ряда разного рода ограничений, мешающих получить «чистую» архитектуру, на которую не влияли бы какие-либо внешние факторы.

По нашему мнению, на разработку архитектуры NVM-ориентированных СУБД влияют

- инфраструктурные ограничивающие факторы;
- недостаточная развитость аппаратных технологий NVM и/или недостаточная информированность сообщества баз данных со стороны вендоров;
- влияние решений, наиболее активно используемых в близкой, но принципиально отличающейся области in-memory СУБД.

4.1 Инфраструктурные ограничивающие факторы

Для разработки архитектуры любой программной системы, тем более такой сложной, как СУБД, тем более СУБД, основанной на новых принципах, требуется наличие инфраструктуры. Инфраструктура должна включать подготовленных и заинтересованных специалистов, с которыми можно было бы обсуждать общий облик будущей системы и/или альтернативные частные решения. Инфраструктура должна обеспечивать возможность быстрого построения прототипов системы, поддерживать необходимые процессы тестирования и отладки. Для этого необходимо наличие как готовых для использования аппаратно-программных инструментальных средств, так и программистов, способных быстро и достаточно качественно участвовать в разработке. Наконец, инфраструктура должна помогать находить источники финансирования новых проектов.

Как показывает обзор родственных работ в предыдущем разделе, почти все успешные проекты, связанные с использованием NVM в СУБД, опирались на наличие подобной инфраструктуры. Лучше всего ее обеспечивают исследовательские лаборатории компаний или университетов, постоянно выполняющие различные проекты по тематике, связанной с базами данных, обладающие необходимыми инструментальными средствами, содержащие высококвалифицированных специалистов и привлекающие к исследовательской работе студентов и аспирантов. Очевидными примерами являются группа баз данных в университете Карнеги-Меллон[12] и Дрезденская группа систем баз данных.

Но наличие инфраструктуры и ее особенности накладывают на разработку новой архитектуры свои ограничения. Главным образом, они связаны с тем, что трудно

[12] https://db.cs.cmu.edu/

решиться на открытие совсем нового проекта по теме, которая не гарантирует абсолютного успеха. Это в полной мере соответствует актуальной, но не бесспорной тематике NVM-ориентированных СУБД. Понятно, что выполнение такого проекта в любом случае принесло бы пользу, но практическая значимость результатов не может быть известна заранее. Не говоря уже о том, что получить финансирование подобных проектов совсем непросто, недостаточно убедительные результаты могут навредить имиджу лаборатории.

Поэтому естественно возникает желание «пристегнуть» новый проект к какому-то уже существующему, зарекомендовавшему себя и устойчиво финансируемому проекту. В нашем случае мы наблюдали привязку проекта Арулраджа к проекту СУБД Peloton в группе Карнеги-Меллон и привязку проекта Укида к проекту HANA в Дрезденской группе. Это приводит к нарушению чистоты архитектуры.

В группе Карнеги-Меллон основной целью проекта Peloton, в рамках которого работал Джой Арулрадж, является разработка самоуправляемой (self-driven) СУБД нового поколения, способной самостоятельно, без потребности во вмешательстве людей настраиваться на поддержку эффективного выполнения гибридных рабочих нагрузок. Под руководством явно увлеченного этой темой Энди Павло разработчики системы активно используют методы искусственного интеллекта и машинного обучения. При построении системы явно учитывается то, что она должна быть в состоянии настраиваться и на транзакционную, и на аналитическую рабочие нагрузки.

Нельзя не согласиться с важностью и актуальностью тематики самоуправляемых СУБД. Но, с другой стороны, по нашему мнению, Peloton отчасти связывает с тематикой NVM-ориентированных СУБД только то, что в ней базы данных хранятся в основной памяти. Самоуправляемость и поддержка гибридных рабочих нагрузок противоречат основной цели NVM-ориентированной СУБД – очень быстрая обработка транзакций и очень быстрое восстановление базы данных и работоспособности системы после сбоев. При совмещении обеих целей в одной разработке, фактически, происходит возврат к эпохе универсальных «безразмерных» систем, закат которой был убедительно показан Майклом Стоунбрейкером более 10 лет тому назад [37].

В работе Исмаила Укида предусловием, по-видимому, являлась ориентация на совместимость с производственной СУБД HANA компании SAP. С одной стороны, HANA является современной in-memory СУБД, основанных на современных, но уже проверенных технологиях. Но с другой стороны, эта система является (a) производственной, активно используемой внутри и вне SAP и (b) гибридной, рассчитанной на поддержку как аналитических, так и транзакционных рабочих нагрузок. Другими словами, это тоже универсальная in-memory СУБД. На мой взгляд, близость в HANA сильно повлияла на работу Укида, построенную им архитектуру NVM-ориентированной СУБД, на выбор алгоритмов и методов. Видимо, это оказалось полезным для профессиональной карьеры Укида как сотрудника SAP, но повредило «чистоте» его разработки.

Мы считаем, что нативную архитектуру одноуровневой СУБД можно разработать только с нуля, не подвергаясь влиянию особенностей инфраструктуры.

4.2 Ограничивающие предположения о характеристиках аппаратных средств NVM

Как показывает разд. 2 этой статьи, достоверную информацию о характеристиках не только ожидаемых аппаратных решениях NVM, но и уже производимых продуктах получить весьма затруднительно. Компании, разрабатывающие или уже производящие NVM в форм-факторе DIMM, не хотя преждевременно раскрывать их точные

характеристики. Показатели разных видов NVM, публикуемые в открытых источниках, называют приблизительными сами авторы публикаций.

Данные, приводимые в табл. 1, относительно убедительно показывают только то, что (1) NVM на основе STT-RAM обладает показателями скорости доступа и стойкости, близкими к показателям DRAM и (2) NVM на основе PCM и ReRAM показывает более слабые характеристики. Однако судя по данным отдельных источников, скорость записи STT-RAM в пять раз меньше скорости чтения ([3, 13]), а стойкость этого же вида NVM на один-два десятичных порядка уступает стойкости DRAM ([3, 5]).

Исходя из подобных недостаточно достоверных сведений, различные исследователи принимают разные пессимистические предположения о характеристиках ожидаемой энергонезависимой памяти.

- Некоторые из них полагают, что NVM является и навсегда останется существенно более медленной, чем DRAM (и, тем более, SSD) (возможно, так оно и есть для PCM), и поэтому следует использовать NVM в качестве еще одного слоя в иерархии систем хранения данных между DRAM и DDR.
- Другие исследователи полагают, что неустранимой особенностью всех видов байт-адресуемой NVM является существенная разница в скорости выполнения операций чтения и записи: запись медленнее чтения. Понятно, что это предположение сильно влияет на методы использования NVM в СУБД.
- Наконец, еще одним распространенным предположением является меньшая стойкость (endurance) NVM по сравнению как с DRAM, так и с DDR. Конечно, это предположение сдерживает энтузиазм по отношению к разработке одноуровневых NVM-ориентированных СУБД.

С одной стороны, поскольку в будущем, вполне возможно, на рынке станут доступными разные виды NVM в форм-факторе DIMM с разными характеристиками, и менее быстрая и стойкая энергонезависимая память будет стоить дешевле энергонезависимая память будет стоить дешевле быстрой и стойкой памяти (еще одно предположение!), могут оказаться востребованными различные архитектуры СУБД с использованием NVM.

- При наличии не слишком быстрой, но достаточно стойкой NVM может оказаться достаточно эффективная многоуровневая организация системы хранения DRAM-NVM. Это является естественной эволюцией технологии in-memory СУБД.
- При аналогичных показателях NVM вполне перспективен и путь к использованию иерархии DRAM-NVM-DDR (или SSD). Понятно, что этот подход является эволюцией традиционной технологией дисковых СУБД. В зависимости от показателей стойкости NVM эта память может использоваться в виде долговременно хранимого кэша между DRAM и DDR(SSD) или в виде части постоянного хранилища данных.
- Наконец, еще один путь к использованию NVM в форм-факторе DIMM состоит в том, чтобы вообще не использовать байт-адресуемую энергонезависимую память в архитектуре будущей СУБД. Как мы отмечали в разд. 2, в последнее время появились SSD (компании Intel на основе технологии Optane-3D XPoint и компании Samsung на основе технологии Z-NAND), которые обеспечивают очень высокую скорость блочных обменов. Поскольку в SSD компании Intel используется NVM типа PCM, в этих дисковых устройствах время произвольной записи блока почти не отличается от времени чтения. Конечно, при использовании таких SSD для хранения

баз данных требуется пересмотр архитектуры СУБД (см., например, [11]), и в этой архитектуре может не оказаться места медленной байт-адресуемой СУБД.

Это все верно, но нас интересует чистая одноуровневая архитектура NVM-ориентированной СУБД, в которой применяются только байт-адресуемые DRAM и NVM. Чтобы обеспечить «чистоту» (nativeness) этой архитектуры, мы должны сделать оптимистические предположения относительно характеристик NVM:

- скорость произвольного доступа к NVM одинакова для операций чтения и записи и не уступает скорости доступа к DRAM;
- стойкость NVM не уступает стойкости DDR (SSD).

Возможно, эти предположения слишком оптимистичны (особенно второе), но они соответствуют наблюдаемым тенденциями развития технологии NVM.

4.3 Влияние методов, используемых в близких областях

NVM обладает чертами как устройств хранения данных во внешней памяти (энергонезависимость и следующая из этого долговременность хранения), так и традиционной основной памяти (прямая байтовая адресация). Это побуждает разработчиков архитектуры NVM-ориентированной СУБД пытаться использовать решения, успешно себя зарекомендовавшие в архитектурах дисковых и/или in-memory СУБД, даже если разработка новой архитектуры ведется с нуля. Однако NVM обладает важным отличием от устройств внешней памяти: это байт-адресуемая память прямого доступа. NVM принципиально отличается и от традиционной основной памяти: она энергонезависима и потому обеспечивает долговременное хранение данных.

Наверное, можно найти много примеров необоснованных заимствований решений, но в этой статье мы обсудим только два из них, относящихся к важным компонентам СУБД: индексации и управления транзакциями.

4.3.1 Индексация

Для организации индексов в дисковых СУБД наибольшей популярностью пользуются B+-деревья [38]. Узлами B+-деревьев являются блоки дисковой памяти, причем в промежуточных узлах хранятся линейные списки пар <значение ключа, номер дочернего блока>, таких что в соответствующем дочернем блоке (если он все еще не листовой) хранится список аналогичных пар со значением ключа, большим или равным значению ключа из родительской пары и меньшим значения ключа из следующей пары родительского списка. Листовые узлы B+-дерева хранят упорядоченные по возрастанию значений ключа пары <значение ключа, линейный список идентификаторов строк таблицы>.

В каждом промежуточном узле B+-дерева сохраняется, вообще говоря, много пар <значение ключа, номер дочернего блока>, поэтому дерево является сильно ветвистым и потому обладает небольшой глубиной. Из-за этого при поиске строк таблицы по заданному значению ключа, который всегда начинается с корневого узла, а заканчивается при достижении листового узла, содержащего требуемое значение ключа, приходится прочитать из дисковой в основную память небольшое число блоков.

Кроме того, для B+-деревьев поддерживается свойство полной сбалансированности, т.е. длина пути от корня до любого листа всегда одна и та же. Для этого используются операции расщепления (splitting) узла B+-дерева при переполнении узла и слияния (merging) соседних узлов (на одном уровне) при опустошении узла. Расщепление и слияние всегда начинаются от некоторого листового узла и двигаются вверх по дереву по направлению к корню, иногда, но редко приводя к глобальной перестройке дерева. Сбалансированность индекса на основе B+-дерева означает, что независимо от значения

ключа, по которому производится поиск, стоимость поиска (оцениваемая как требуемое число обменов с внешней памятью) всегда одна и та же.

Листовые узлы B+-дерева связываются в двунаправленный список, что позволяет использовать соответствующие индексы для поиска строк таблицы с заданным диапазоном значений ключа по возрастанию или убыванию этих значений. Иногда для дополнительного ускорения поиска по диапазону значений ключа пары, хранимые в листовых узлах, упорядочивают по возрастанию или убыванию ключа.

Фундаментальные свойства B+-дерева – сильная ветвистость и полная сбалансированность – играют на пользу индексации, если индекс хранится во внешней памяти. При оценках стоимости операций над такими индексами учитывается только требуемое число обменов с внешней памятью, а временные расходы на обработку содержимого узлов после их считывания в основную память не принимаются во внимание.

Во что превратится B+-дерево, если прямо перенести эту структуру в основную память? Легко видеть, что по своей сути это упорядоченный по возрастанию значений ключа список ссылок на строки индексируемой таблицы. Этот список хранится в листовых узлах. Верхняя часть поддерева служит только для того, чтобы можно было найти в этом списке ключ с заданным значением в этом листовом списке, не перебирая его последовательно. Фактически, это тоже иерархически организованный упорядоченный список упорядоченных подсписков. То, что каждый подсписок хранится внутри блока с последовательными адресами в линейной форме, никак не влияет на требуемое число обращений в памяти и операций сравнения.

Как справедливо отмечалось в [39], наличие поисковой подструктуры в B+-дереве, отделенной от списка ключей и ссылок на индексируемые строки таблицы, при хранении B+-деревьев в основной памяти приводит только к дополнительным накладным расходам памяти и процессорных ресурсов. Если уж применять B-деревья для организации индексов в основной памяти, то это должны быть классические B-деревья [40], в которых каждое существующее в индексируемой таблице значение ключа хранится только один раз (в каком-то промежуточном или листовом узле) и при нем сохраняется соответствующий набор ссылок на строки таблицы.

Можно сказать, что B-дерево в основной памяти – это один рекурсивно организованный упорядоченный список упорядоченных подсписков, или дерево, узлы которого являются списками. Если все подсписки содержат примерно одно и то же число значений ключа k, и глубина дерева равна n, то в худшем случае для поиска ключа с заданным значением придется потратить $O(k \times n)$ операций обращения к памяти и сравнения. Но блочная структура узлов дерева, сохраняющих подсписки, здесь не имеет никакого смысла. Каждый элемент любого подсписка может располагаться в отдельной области памяти, и все они могут связываться ссылками.

Тем не менее, как это не странно, индексы на основе именно B+-деревьев получили широкое распространение в in-memory СУБД. Изобретено большое число разновидностей B+-деревьев в основной памяти, и они активно заимствуются в новые архитектуры NVM-ориентированных СУБД именно в блочной форме. В частности, свои варианты B+-деревьев предлагают и Джой Арулрадж и Исмаил Укуд. По нашему мнению, подобные подходы к индексации таблиц в NVM-ориентированных СУБД не обоснованы, и методы индексации требуют дополнительного изучения (см. разд. 5).

4.3.2 Управление транзакциями

В последние десятилетия в сообществе разработчиков СУБД резко вырос интерес в многоверсионным методам управления транзакциями (MultiVersion Concurrence Control, MVCC). При применении протоколов MVCC каждое изменение объекта базы данных

(как правило, строки таблицы) приводит к образованию новой версии этого объекта, а его предыдущая версия остается доступной по чтению в других транзакциях.

Основной причиной возросшей популярности MVCC является то, что при использовании классических двухфазных блокировочных протоколов (Two Phase Locking, 2PL) даже вполне безобидные транзакции, которые только читают данные, могут попасть в синхронизационный тупик, и из-за этого система может их насильственно откатить. Примером может служить следующая последовательность операций над объектами базы данных o_1 и o_2, выполняемых в транзакциях T_1 и T_2: $R_{T_1}(o_1), W_{T_2}(o_2), R_{T_1}(o_2), W_{T_2}(o_1)$. Если используется какой-либо протокол категории MVCC, то первая транзакция прочитает без всякой задержки на синхронизацию то состояние объекта o_2, в котором он находился до изменения транзакцией T_2, т.е. план выполнения смеси транзакций T_1 и T_2 изменится на допустимый сериализацией план $R_{T_1}(o_1), R_{T_1}(o_2), W_{T_2}(o_2), W_{T_2}(o_1)$.

Мы отложим более подробное обсуждение протоколов MVCC и связанных с ними преимуществ и проблем до разд. 5, а здесь поговорим об их применении в in-memory и NVM-ориентированных СУБД. Похоже, что во всех современных in-memory СУБД используется какой-либо вариант протокола MVCC. Наиболее известными являются базовые подходы многоверсионного протокола с упорядочиванием по временным меткам (MultiVersion Timestamp Ordering, MVTO), двухверсионный двухфазный блокировочный протокол (Two-Version 2PL, 2V2PL) и оптимистический многоверсионный протокол (MultiVersion Optimistic Concurrency Control, MVOCC).

Краткий обзор этих протоколов можно найти, например, в [41][13]. В этой же статье говорится о том, что все эти протоколы были реализованы в СУБД Peloton и был произведен их экспериментальный анализ с использованием бенчмарков YCSB и TPC-C. На основе содержимого статьи можно сделать следующие наблюдения:

- разработчики Peloton так и не выбрали базовый протокол управления транзакциями для своей системы;
- у всех испытанных протоколов имеются свои достоинства и недостатки, проявляемые при различных параметрах рабочей нагрузки;
- видимо, поэтому в работах [4-6] расплывчато говорится об использовании в архитектуре NVM-ориентированной СУБД протокола MVCC без уточнения его разновидности.

По нашему мнению, это означает, что Аруладж и его научный руководитель Павло в действительности не представили полную архитектуру NVM-ориентированной СУБД, поскольку специфические черты конкретного механизма управления транзакциями накладывают отпечаток на многие другие компоненты СУБД.

И это видно на примере работы Исмаила Укуда [8]. В своей архитектуре он выбрал протокол MVOCC. Выбор именно этого протокола серьезно не оправдывается. Протокол сложный, реализация технически трудная. Возможно, для диссертации это только плюс, а для «чистой» архитектуры NVM-ориентированной СУБД – минус, по нашему мнению. Сошлемся на цитату из книги Джема Грея, полностью приведенную в конце подраздела 3.1: «…простые системы работают, а сложные нет».

5. Набросок архитектуры «чистой» NVM-ориентированной СУБД

В настоящее время у автора отсутствует полная архитектура «чистой» NVM-ориентированной СУБД. Даже эскизы архитектуры недостаточно технически проработана. Полностью отсутствует какая-либо реализация. Однако в условиях

[13] Мы не включали эту статью в число ранее указанных публикаций Аруладжа, потому что в ней ничего не говорится о NVM-ориентированных СУБД.

недостаточного внимания исследователей к общим архитектурным вопросам организации NVM с использованием одноуровневой энергонезависимой памяти приводимые в этом разделе наблюдения и рассуждения могут оказаться полезными при будущей полномасштабной разработке.

Начнем раздел со сводки базовых предположений, которыми руководствуется автор.

- NVM обладает всеми «оптимистическими» характеристиками: скорость чтения и записи одна и та же и не меньше, чем у DRAM; стойкость не меньше, чем у DDR (или хотя бы SSD).
- Целевой компьютер содержит требуемое число ядер или потоков управления, поддерживаемых аппаратурой, а также достаточные объемы энергозависимой и энергонезависимой основной памяти.
- Внешняя память совсем не используется.
- Вся серверная часть системы выполняется на одном компьютере, в энергонезависимой основной памяти которого сохраняется баз данных. Сеть используется только для взаимодействия с сервером клиентских частей приложений.
- Архитектура рассчитана на поддержку только транзакционных рабочих нагрузок.
- При использовании аппаратных средств не допускается никакая виртуализация. СУБД сама следит за распределением энергонезависимой основной памяти. Каждой транзакции жестко соответствует отдельное ядро процессора или поток управления, поддерживаемый аппаратурой.

С учетом этих предположений рассмотрим предлагаемые архитектурные решения.

5.1 Распределение энергонезависимой памяти

Для упрощения решения, в том числе, преодоления проблемы внешней фрагментации памяти за счет допущения ограниченной внутренней фрагментации предлагается основывать распределение памяти на классическом методе двоичных близнецов (buddy system) [42]. При применении этого метода при запросе k байт памяти всегда выделяется участок размером в 2^nбайт, где 2^n – ближайшая к k сверху степень двойки.

По нашему мнению, для распределения энергонезависимой памяти в контексте NVM-ориентированной СУБД больше подходит именно метод двоичных близнецов, а не, например, фибоначчиевых близнецов (Fibonacci system)[14], когда при запросе k байт памяти выделяется участок, размер которого равен ближайшему сверху числу Фибоначчи. Это связано с тем, что фактическим стандартом современной индустрии микропроцессоров общего назначение является использование в кэш-памяти блоков (cache line) размером 64 байта. Поэтому мы предполагаем, что размер динамически запрашиваемой энергонезависимой в NVM-ориентированной СУБД никогда не будет меньше 2^6, а в основном будут запрашиваться участки памяти именно этого размера.

Понятно, что память может запрашиваться в любой выполняемой транзакции, которой соответствует ядро процессора или поток управления, поддерживаемый аппаратурой. Чтобы избежать потребности в синхронизации параллельно выполняемых потоков управления, свободная энергонезависимая память заранее делится поровну между всеми рабочими потоками управления (worker), и в каждом потоке работает собственный компонент распределения памяти в своей части общей кучи.

Представляется, что в основном свободные участки памяти будут запрашиваться явно путем обращения к «системным» вызовам типа malloc. Но потребуются и неявные запросы памяти («по требованию») при расширении сегментов, требуемых для поддержки индексов (см. 5.3). Нужно, чтобы в случае, когда прерывание по отсутствию

[14] Daniel S. Hirschberg. A class of dynamic memory allocation algorithms. Communications of the ACM, vol. 16, issue 10, 1973, pp. 615-618.

в NVM страницы виртуальной памяти соответствующего сегмента возникает при выполнении транзакции в некотором ядре процессора, операционная система брала соответствующий фрейм NVM из «подкучи» этого же ядра (аппаратного потока управления).

5.2 Хранение таблиц и управление транзакциями

Для упрощения архитектуры и по другим соображениям, обсуждаемым в п. 5.2.2, мы предлагаем использовать в «чистой» архитектуре NVM-ориентированной СУБД разновидность протокола 2V2PL. Напомним, что 2V2PL предполагает хранение двух копий каждой строки каждой таблицы базы данных: текущей версии, созданной последней зафиксированной транзакцией, и измененной версии, созданной еще не зафиксированной транзакцией. Естественно, это влияет на организацию хранения таблиц в NVM.

5.2.1 Хранение таблиц

Каждая таблица сохраняется в энергонезависимой основной памяти в виде списка строк, т.е. таблицы хранятся, вообще говоря, по строкам (с оговоркой, приводимой ниже). Выборку строк их таблицы можно выполнять либо путем последовательного просмотра этого списка, либо через индекс (см. под

Транзакционные базы данных по определению являются изменчивыми, поэтому память, запрошенная при вставке строки или при создании первой копии, не освобождается до выполнения операции удаления этой строки. В первом приближении схема выглядит следующим образом. При вставке строки всегда запрашивается участок памяти размером не менее 64 байта. Этот участок представляет новую строку в списке строк таблицы. Точный размер запрашиваемого участка памяти определяется тем, что в нем должны поместиться две копии строки таблицы и служебная информация (ссылки по списку строк и информация, требуемая для управления транзакциями (см. п. 5.2.2)).

Если таблица содержит столбцы большого размера (например, длинные varchar) для их хранения используются отдельные участки памяти требуемого размера, для доступа к которым в основном месте хранения строк, вместо самих данных записываются указатели на области их расположения. Возможно, потребуется оптимизация при таком отдельном хранении столбцов, свойственная поколоночным СУБД (устранение дубликатов и/или сжатие), но для целей данной статьи это несущественно.

5.2.2 Управление транзакциями

На решение использовать для управления транзакциями в «чистой» архитектуре протокол 2V2PL во многом повлияли статьи [41, 43], в которых на основе убедительных экспериментов было показано, что ни один из известных протоколов управления транзакциями даже на простых транзакционных бенчмарках не демонстрирует преимуществ. Видимо, это так и есть, и с использованием имеющихся подходов невозможно обеспечить горизонтальное масштабирование как in-memory, так и NVM-ориентированных СУБД при значительном возрастании числа ядер процессора. С другой стороны, основной задачей разработки «чистой» архитектуры NVM-ориентированной СУБД является обеспечение значительного повышения скорости обработки транзакций при сохранении максимальной простоты решений, а 2V2PL, по нашему мнению, гораздо проще, чем MVTO и, тем более, MVOCC.

Замечание. После публикаций очень эффектных статей о проекте H-Store[15] и столь же эффектном появлении коммерческой СУБД VoltDB[16] у автора данной статьи создалось впечатление, что удалось найти путь к горизонтальному масштабированию транзакционных СУБД путем использования массивно-параллельных компьютерных архитектур и разделения данных между узлами в духе shared-nothing [44]. Но даже и в этой чрезмерно оптимистической статье высказывались сомнения о возможности эффективной и масштабируемой поддержке распределенных транзакций.

Эти сомнения подтвердила очень полезная статья [45], в которой на основе экспериментов показано, что в настоящее время отсутствует метод построения СУБД, обеспечивающий горизонтальное масштабирование системы при наличии транзакционных рабочих нагрузок, содержащих распределенные транзакции. Одновременно эта статья укрепила наше мнение, высказанное в [11], о том, что «чистая» архитектура NVM-ориентированной СУБД должна основываться на одномашинной многоядерной аппаратной архитектуре. Попытки добиться горизонтальной масштабируемости оставим на будущее.

В общих чертах предлагаемый вариант 2V2PL выглядит следующем образом. Поскольку в каждом ядре (аппаратном потоке) в каждый момент времени выполняется только одна транзакция, можно содержать в энергонезависимой памяти по одному описателю транзакции для каждого ядра (потока). У каждой транзакции имеется уникальный идентификатор, не используемый повторно для будущих транзакций. Идентификатор транзакции сохраняется в описателе транзакции. Когда транзакция создает новую версию некоторой строки некоторой таблицы (вставляет новую или изменяет существующую строку), она записывает в служебную часть хранилища этой копии свой идентификатор транзакции и ссылку на свой описатель, а также обнуляет поле идентификатора транзакции в предыдущей копии строки, если она существует. Последним действием при фиксации транзакции, после которого ее уже нельзя откатить, является обнуление идентификатора в описателе транзакции. Другими словами, признаком наличия измененной (незафиксированной) копии строки является совпадение сохраненного при ней идентификатора транзакции с идентификатором, находящимся в описателе транзакции. В общей служебной части хранилища строки содержится неотрицательный счетчик, значение соответствует числу незафиксированных транзакций, прочитавших эту копию, и признак выполнения операции записи.

Тогда в рабочей фазе любой транзакции при выполнении операции чтения строки таблицы увеличивается на единицу значение счетчика последней зафиксированной копии (эта операция атомарная) и считывается строка. В описателе транзакции сохраняется эта строка, а также порядковый номер ее версии. В конце операции чтения счетчик уменьшается на единицу. Если в той же транзакции повторно выполняется чтение той же строки, проверяется, что у нее не изменился номер версии. Если он изменился, то чтение все равно производится, но логика приложения об этом оповещается и принимает решение о продолжении или откате транзакции.

При выполнении операции записи проверяется признак выполнения операции записи той же строки другой транзакцией. Если признак установлен, транзакция откатывается, иначе признак устанавливается (проверка и установка признака – атомарная операция). Далее проверяется значение счетчика последней зафиксированной копии. Если он равен нулю, проверяется, не существует ли копия, образованная еще не зафиксированной

[15] Robert Kallman, Hideaki Kimura, Jonathan Natkins, Andrew Pavlo, Alexander Rasin, Stanley B. Zdonik, Evan P. C. Jones, Samuel Madden, Michael Stonebraker, Yang Zhang, John Hugg, Daniel J. Abadi. H-store: a high-performance, distributed main memory transaction processing system. Proceedings of the VLDB Endowment, vol. 1, issue 2, 2008, pp. 1496-1499.

[16] https://en.wikipedia.org/wiki/VoltDB

транзакцией. Если такая копия не существует, запись производится на место предпоследней зафиксированной копии, а в служебную область этой копии пишутся ссылка на описатель и идентификатор транзакции, в которой выполняется операция записи (образуется незафиксированная, измененная копия. Иначе снимается признак выполнения операции записи и транзакция откатывается. При благополучном исходе в описателе транзакции запоминается предыдущее состояние строки, номер образованной версии и адрес для обратной записи при восстановлении. В конце операции признак выполнения операции записи снимается.

Фиксация транзакции происходит мгновенно: обнуляется идентификатор транзакции в ее описателе. После этого в рабочем потоке можно выполнять новую транзакцию.

Индивидуальные откаты транзакций выполняются путем обратной записи изменений. Поскольку это касается только операций записи, обратную запись при восстановлении можно выполнять напрямую по сохраненным в описателях транзакции указателям.

Восстановление после сбоя, приведшего к перезапуску системы можно производить, пользуясь информацией, сохраненной в описателях транзакций, в которых к моменту сбоя содержатся ненулевые идентификаторы. Хронологически порядок обратных записей для какой-либо строки поддерживать не обязательно: если номер версии очередной копии строки для восстановления оказывается меньше номера копии, содержащегося в области хранения строки, обратную запись выполнять не нужно.

5.3 Организация индексов

Нам неизвестны точные характеристики запросов к базам данных, встречающиеся в типичных современных транзакционных рабочих нагрузках. Но практически во всех известных случаях для измерения и/или сравнения производительности транзакционных СУБД используется бенчмарк TPC-C. Мы принимаем это как свидетельство о реалистичности этого бенчмарка.

Почти все запросы к базе данных в транзакциях TPC-C опираются на условия вида $a = b$, где a – имя столбца одной таблицы из раздела FROM запроса, а b – это либо константа, либо имя столбца другой таблицы из раздела FROM (для запросов с соединениями). При этом чаще всего в условиях используются имена столбцов первичных ключей таблиц.

Поэтому прежде всего в транзакционной NVM-ориентированной СУБД требуются индексы, обеспечивающие быстрый прямой доступ к строкам таблиц по значению указанного столбца. Очевидно, что для этого лучше всего приспособлены методы, основанные на хешировании значений ключа поиска, и по нашему мнению наиболее просто и эффективно можно реализовать индекс на основе варианта метода линейного хеширования [46]. Основная часть таблицы должна храниться в расширяемом сегменте виртуальной памяти, страницы которого отображаются на NVM.[17]

Представляется, что для каждого значения свертки будет поддерживаться основной бакет размером 64 байта. Если до потребности в расщеплении элемента таблицы список элементов <ключ, ссылка на строку таблицы> перестанет помещаться в уже выделенных бакетах, то будет запрашиваться еще один бакет, и он будет привязываться к списку уже существующих бакетов (так может быть при индексировании таблицы по столбцам с длинными значениями, которые, тем не менее, невыгодно хранить отдельно). Критерием расщепления является достижение установленного предела длины списка с одинаковым значением свертки.

[17] По мнению автора, для NVM-ориентированной СУБД более подходящей была бы виртуальная память с небольшими страницами (например, 512 байт). В настоящее время ОС Linux допускает минимальный размер страницы в 4K байт.

Поскольку трудно отказаться от поисковых операций уровня SQL с указанием условий поиска вида $c_1 < a < c_2$, где a – имя столбца таблицы из раздела FROM запроса, а c_1 и c_2 – константы (запрос по диапазону, range query), придется обеспечивать другой вид индексов, скорее всего, на основе B-деревьев. На наш взгляд, стоит рассмотреть два варианта. В первом варианте все узлы дерева представляют списки бакетов по 64 байта, а критерием расщепления или слияния соседних узлов является длина списка. Во втором варианте каждое B-дерево располагается в отдельном расширяемом сегменте виртуальной памяти, а узел дерева – линейный список элементов, располагающийся в пределах одной страницы сегмента. Для выбора варианта требуются эксперименты, измерения и сравнения.

Дополнительного изучения требуют вопросы организации параллельного изменения индексов, а также обратного изменения индексов при откате транзакций.

5.4 Оптимизация запросов

В этом подразделе мы ограничимся лишь несколькими замечаниями. Во-первых, аспекты оптимизации SQL-запросов в NVM-ориентированных СУБД к настоящему времени исследованы явно недостаточно. Публикации, посвященные этой теме, практически отсутствуют, хотя, например, в заключении статьи [47] уже отмечалась потребность в таких исследованиях.

Насколько можно судить по текущим публикациям (например, обзорной статье [48]), в существующих in-memory СУБД используются методы оптимизации запросов, не отличающиеся от применяемых в традиционных дисковых СУБД. Обоснований этого консерватизма найти не удается. Хотелось бы при разработке «чистой» архитектуры NVM-ориентированной СУБД хорошо разобраться с требуемыми методами оптимизации запросов.

Во-вторых, мы ориентируемся на транзакционный режим использования СУБД. Очевидно, что в NVM-ориентированной СУБД транзакции должны обрабатываться очень быстро, поскольку в принципе отсутствуют обмены с устройствами внешней памяти. Чтобы обеспечить потенциально возможную пропускную способность NVM-ориентированной транзакционной СУБД, нужно избегать любых задержек при выполнении транзакций, в том числе, и коммуникаций с клиентскими рабочими станциями. Поэтому традиционным способом организации транзакционных приложений баз данных является перенос всей логики приложения на сторону сервера баз данных обычно в виде хранимых процедур.

Здесь мы не будем подробно останавливаться на методах программирования хранимых процедур в «чистой» NVM-ориентированной СУБД. Заметим лишь, что код хранимой процедуры, в которой содержится логика транзакции, должен выполняться в той же ядре (или аппаратном потоке управления), в котором выполняются все остальные системные программы, нужные для поддержки выполнения транзакции. Для всей СУБД поддерживается одно адресное пространства. Поэтому для программирования хранимых процедур должны использоваться безопасные, компилируемые в машинные коды (для поддержки эффективности), языки программирования (например, язык Rust[18]).

В этих хранимых процедурах, безусловно, должно допускаться использование языка SQL. Но компиляция и оптимизация декларативных оператором SQL должна производиться тогда же, когда выполняется компиляция кода хранимой процедуры. Это должно делаться раньше того времени, когда система начнет выполнять реальную транзакционную рабочую нагрузку. Поэтому должна производиться полноценная оптимизация с учетом специфики среды NVM.

[18] https://www.rust-lang.org/

Конечно, при оптимизации нужно учитывать стоимость генерируемых планов запросов, и для этого придется использовать статистику базы данных на момент оптимизации. Эта статистика может измениться при выполнении транзакционной рабочей нагрузки, и выбранный план может стать далеким от оптимального. NVM-ориентированная СУБД, видимо, сможет обнаружить этот факт в рабочем режиме, но выполнить повторную оптимизацию, не нарушая установившийся режим обработки транзакций, невозможно. Известные приемы, в том числе, адаптивная (самообучаемая) оптимизация запросов[19] здесь неприменимы. Требуется дополнительное исследование.

В-третьих, важно понять, какой SQL нужно поддерживать в NVM-ориентированных СУБД. С одной стороны, очевидно, что полный набор возможностей SQL, специфицированных в стандарте, здесь не потребуется. С другой стороны, волевым усилием ограничить некоторое подмножество стандарта явно невозможно. Требуется тщательная и кропотливая работа для выбора подмножества стандарта, которое бы действительно требовалось в специализированной транзакционной СУБД.

6. Заключение

Энергонезависимая байт-адресуемая основная память становится (или уже стала) реальностью. Характеристики разных видов NVM, доступные в открытом доступе, позволяют рассчитывать, что доступная в ближайшем будущем энергонезависимая основная память будет обладать скоростью, не меньше, чем DRAM, и стойкостью, сопоставимой со стойкостью HDD (или хотя бы SSD). Поэтому исследования архитектур СУБД с одноуровневой системой хранения становятся более чем актуальными.

Однако, как показывает анализ литературных источников, в сообществе баз данных этой тематике все еще уделяется недостаточное внимание. Даже наиболее удачные опубликованные исследования выполнялись при наличии ряда ограничений, не позволяющих получить «чистую» архитектуру транзакционной NVM-ориентированной СУБД, при разработке которой принималось бы во внимание только стремление к достижению максимально высоких показателей скорости выполнения транзакций и пропускной способности системы.

В статье приводится набросок такой архитектуры, в котором выделяются важные аспекты распределения памяти, организации хранения таблиц и индексов, управления

[19] Идеи адаптивной оптимизации запросов можно наблюдать уже в ранних публикациях о System R, но первой хорошо проработанной работой мы считаем статью Volker Markl, Guy M. Lohman, Vijayshankar Raman. LEO: An autonomic query optimizer for DB2. IBM Systems Journal, vol. 42 , issue 1, 2003, pp. 98 – 106.https://www.researchgate.net/profile/Guy_Lohman?_sg%5B0%5D=ELH-E-Yzi3fszkO_yVedJ0V5Q4VZbRqh0EqUso3I24Dq8Yo5DhuLV8GhxOab1pV5f9XXZj4.1wMCbguPAFxh_5XeNp6fsBpGv3GBAxuMCJQ9Vz08Hg2Cug_ctuZ6IXjmHbLaiBG2kiAoC6LlH9TIJur-yZtQkg&_sg%5B1%5D=Kx4GFWwumiraq3g_xHMpBcpPZfhcBtRgu_ZDJEpkRMACZRa3cHMOgdJaXgeOevsCea8Q89aO3cR2G5Sm.zmu4srFrjmhU83RgVHkVe3tx_rToku5gR6iszYUFosSXkAxYHc5UYgyw9MmO7EsAlazljiA5lr4oCnu3GV07gQhttps://www.researchgate.net/profile/Vijayshankar_Raman?_sg%5B0%5D=ELH-E-Yzi3fszkO_yVedJ0V5Q4VZbRqh0EqUso3I24Dq8Yo5DhuLV8GhxOab1pV5f9XXZj4.1wMCbguPAFxh_5XeNp6fsBpGv3GBAxuMCJQ9Vz08Hg2Cug_ctuZ6IXjmHbLaiBG2kiAoC6LlH9TIJur-yZtQkg&_sg%5B1%5D=Kx4GFWwumiraq3g_xHMpBcpPZfhcBtRgu_ZDJEpkRMACZRa3cHMOgdJaXgeOevsCea8Q89aO3cR2G5Sm.zmu4srFrjmhU83RgVHkVe3tx_rToku5gR6iszYUFosSXkAxYHc5UYgyw9MmO7EsAlazljiA5lr4oCnu3GV07gQ

транзакциями и оптимизации запросов. Почти во всех предложениях автора отмечается потребность в дополнительных исследованиях.

Список литературы / References

[1] Chris Mellor. Escaping the DRAM price trap: Storage Class Memory, what it is and why it matters. Blocks & Files, 2019. Available at: https://blocksandfiles.com/2018/11/28/2019-the-year-of-storage-class-memory/, accessed 02-06-2019

[2] Anton Shilov. Samsung Ships First Commercial Embedded MRAM (eMRAM) Product. AnandTech, March 6, 2019. Available at: https://www.anandtech.com/show/14056/samsung-ships-first-commercial-emram-product, accessed 02-06-2019.

[3] Lucian Armasu. Intel STT-MRAM Technology Is Ready for Mass Production. Tom's Hardware, February 21, 2019. Available at: https://www.tomshardware.com/news/intel-stt-mram-mass-production,38665.html, accessed 02-06-2019.

[4] Joy Arulraj. The Design and Implementation of a Non-Volatile Memory Database Management System. PhD Thesis, Computer Science Department, School of Computer Science, Carnegie Mellon University, 2018, 51 p.

[5] Joy Arulraj, Andrew Pavlo. How to Build a Non-Volatile Memory Database Management System. In Proc. of the 2017 ACM International Conference on Management of Data, 2017, pp. 1753-1758.

[6] Joy Arulraj, Andrew Pavlo. Non-Volatile Memory Database Management Systems. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2019, 173 p.

[7] Joy Arulraj. Data Management on Non-Volatile Memory. Award Talk. In Proc. of the 2019 International Conference on Management of Data, 2019, p. 1114.

[8] Ismail Oukid. Architectural principles for database systems on storage-class memory. PhD.thesis, Technische Universität Dresden, 2017, 189 p.

[9] Oukid, D. Booss, A. Lespinasse, W. Lehner, T. Willhalm, and G. Gomes. Memory management techniques for large-scale persistent-main-memory systems. Proceedings of the VLDB Endowment, Vol. 10, Issue 11, 2017, pp. 1166-1177.

[10] Oukid and W. Lehner. Data structure engineering for byte-addressable non-volatile memory. In Proc. of the 2017 ACM International Conference on Management of Data, 2017, pp. 1759-1764.

[11] С.Д. Кузнецов. Новые устройства хранения данных и их влияние на технологию баз данных. Программная инженерия, no. 4. 2018, стр. 147–155 / Sergey D. Kuznetsov. New Storage Devices and the Future of Database Management. Baltic Journal of Modern Computing, Vol. 6, No. 1. 2018, pp. 1-12. DOI: 10.22364/bjmc.2018.6.1.01.

[12] Байтоадресуемая энергонезависимая память и СУБД. Презентация Андрея Николаенко на Tarantool Conference 2018 / Byte-addressable non-volatile memory and DBMS. Presentation by Andrey Nikolayenko at the Tarantool Conference 2018. Available at: https://ibs.ru/media/media/baytoadresuemaya-energonezavisimaya-pamyat-i-subd/, accessed 09.06.2019 (in Russian).

[13] Sparsh Mittal, and Jeffrey S. Vetter A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems. IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 5, 2016, pp. 1537-1550.

[14] Micron Technology. Available at: https://www.micron.com/, accessed 18-06-2019.

[15] Panasonic and United Microelectronics Corporation Agreed to Develop Mass Production Process for Next Generation ReRAM, Feb 01, 2017. Available at: https://news.panasonic.com/global/press/data/2017/02/en170201-3/en170201-3.html, accessed 18-06-2019.

[16] Crossbar. Available at: https://www.crossbar-inc.com/company/about-crossbar/, accessed 18-06-2019.

[17] Everspin Technologies Available at: https://www.everspin.com/, accessed 18-06-2019.

[18] Crocus Technology. Available at: https://crocus-technology.com, accessed 22-06-2019.

[19] Crocus Nano Electronics. Available at: http://crocusnano.com, acccessed 22-06-2019.

[20] L. Wei, J. G. Alzate, U. Arslan, J. Brockman, N. Das, K. Fischer, T. Ghani,O. Golonzka, P. Hentges, R. Jahan, P. Jain, B. Lin, M. Meterelliyoz, J. O'Donnell,C. Puls, P. Quintero, T. Sahu, M. Sekhar, A. Vangapaty, C. Wiegand, F. Hamzaoglu. 7Mb STT-MRAM in 22FFL FinFET Technology with 4ns Read Sensing Time at 0.9V Using Write-Verify-Write Scheme and Offset-Cancellation Sensing

Technique. In Proc. of the IEEE International Solid-State Circuits Conference (ISSCC), 2019, pp. 214-216.

[21] P. Jain, U. Arslan, M. Sekhar, B. C. Lin, L. Wei, T. Sahu, J. Alzate-vinasco,A. Vangapaty, M. Meterelliyoz, N. Strutt, A. B. Chen, P. Hentges, P. A. Quintero,C. Connor, O. Golonzka, K. Fischer, F. Hamzaogl. 3.6Mb 10.1Mb/mm2Embedded Non-Volatile ReRAM Macro in 22nm FinFET Technology with Adaptive Forming/Set/Reset Schemes Yielding Down to 0.5V with Sensing Time of 5ns at 0.7. In Proc. of the IEEE International Solid-State Circuits Conference (ISSCC), 2019, pp. 212-214.

[22] Stratis D. Viglas. Data Management in Non-Volatile Memory. In Proc. of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp. 1707-1711

[23] Naveed Ul Mustafa, Adrià Armejach, Ozcan Ozturk, Adrián Cristal, Osman S. Unsal. Implications of non-volatile memory as primary storage for database management systems. In Proc. of the 2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), 2016, pp. 164-171

[24] Mihnea Andrei, Christian Lemke, Günter Radestock, Robert Schulze, Carsten Thiel, Rolando Blanco, Akanksha Meghlan, Muhammad Sharique, Sebastian Seifert, Surendra Vishnoi, Daniel Booss, Thomas Peh, Ivan Schreter, Werner Thesing, Mehul Wagle, Thomas Willhalm. SAP HANA Adoption of Non-Volatile Memory. Proceedings of the VLDB Endowment, vol. 10, no. 12, 2017, pp. 1754-1765.

[25] Alexander van Renen, Viktor Leis, Alfons Kemper, Thomas Neumann, Takushi Hashida, Kazuichi Oe, Yoshiyasu Doi, Lilian Harada, Mitsuru Sato. Managing non-volatile memory in database systems. In Proc. of the 2018 ACM SIGMOD International Conference on Management of Data, 2018, pp. 1541-1555.

[26] Joy Arulraj, Andrew Pavlo, Subramanya R. Dulloor. Let's Talk About Storage & Recovery Methods for Non-Volatile Memory Database Systems. In Proc. of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 707-722

[27] Joy Arulraj, Matthew Perron, Andrew Pavlo. Write-Behind Logging. Proceedings of the VLDB Endowment, vol. 10, no. 4, pp. 337-348, 2017.

[28] J. Arulraj, J. Levandoski, U. F. Minhas, and P.-A. Larson. BzTree: A high-performance latch-free range index for non-volatile memory. Proceedings of the VLDB Endowment, vol. 11, no. 5, pp. 553-565, 2018.

[29] Joy Arulraj, Andrew Pavlo, Krishna Teja Malladi. Multi-Tier Buffer Management and Storage System Design for Non-Volatile Memory. arXiv:1901.10938, 2019.

[30] J. Gray, P. McJones, M. Blasgen, B. Lindsay, R. Lorie, T. Price, F. Putzolu, and I. Traiger. The recovery manager of the system R database manager. ACM Computing Surveys, vol. 13, no. 2, 1981, pp. 223-242.

[31] Philip A. Bernstein, Vassos Hadzilacos, Nathan Goodman. Concurrency control and recovery in database systems. Addison-Wesley, 1987, 370 p.

[32] Ismail Oukid, Daniel Booss, Wolfgang Lehner, Peter Bumbulis, Thomas Willhalm. SOFORT: A Hybrid SCM-DRAM Storage Engine for Fast Data Recovery. In Proc. of the Tenth International Workshop on Data Management on New Hardware (DaMoN'14), 2014.

[33] Ismail Oukid, Wolfgang Lehner, Thomas Kissinger, Thomas Willhalm, Peter Bumbulis. Instant Recovery for Main-Memory Databases. In Proc. of the 7th Biennial Conference on Innovative Data Systems Research (CIDR'15), 2015.

[34] Ismail Oukid, Johan Lasperas, Anisoara Nica, Thomas Willhalm, Wolfgang Lehner. FPTree: A Hybrid SCM-DRAM Persistent and Concurrent B-Tree for Storage Class Memory. In Proc. of the 2016 ACM SIGMOD International Conference on Management of Data, 2016, pp. 371-386.

[35] Ismail Oukid, Wolfgang Lehner. Towards a Single-Level Database Architecture on NVM. Presentation abstract. In Proc. of the 8th Annual Non-Volatile Memories Workshop, 2017.

[36] Ismail Oukid. Architectural Principles for Database Systems on Storage-Class Memory. This paper is a summary of the author's PhD thesis of the same title. Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn, 2019.

[37] Michael Stonebraker, Ugur Cetintemel. «One Size Fits All»: An Idea Whose Time Has Come and Gone. In Proc. of the 21st International Conference on Data Engineering, 2005, pp. 2-11.

[38] Douglas Comer. Ubiquitous B-Tree. ACM Computing Surveys, vol, 11, issue 2, 1979, pp. 121-137.

[39] Tobin J. Lehman, Michael J. Carey. A Study of Index Structures for Main Memory Database Management Systems. In Proc. of the 1986 ACM SIGMOD international conference on Management of data, 1986, pp. 239-250.
[40] Rudolf Bayer. R. Bayer, E. McCreight. Organization and Maintenance of Large Ordered Indexes. Acta Informatica, vol. 1, issue 3, 1972, pp. 173-189.
[41] Yingjun Wu, Joy Arulraj, Jiexi Lin, Ran Xian, Andrew Pavlo. An empirical evaluation of in-memory multi-version concurrency control. Proceedings of the VLDB Endowment, vol. 10, issue 7, 2017, pp. 781-792.
[42] Kenneth C. Knowlton. A fast storage allocator. Communications of the ACM, vol. 8, issue 10, 1965, pp. 623-624.
[43] Xiangyao Yu, George Bezerra, Srinivas Devadas, Michael Stonebraker, Andrew Pavlo. Staring into the Abyss: An Evaluation of Concurrency Control with One Thousand Cores. Proceedings of the VLDB Endowment, vol. 8, issue 3, 2014, pp. 209-220.
[44] Кузнецов С.Д. Транзакционные параллельные СУБД: новая волна. Труды ИСП РАН, том 20, 2011, стр. 189-251 / Sergey D. Kuznetsov. Transactional Massive-Parallel DBMSs: A New Wave. Trudy ISP RAS/Proc. ISP RAS, vol. 20, 2011, pp. 189-251 (in Russian).
[45] Rachael Harding, Dana Van Aken, Andrew Pavlo, Michael Stonebraker. An Evaluation of Distributed Concurrency Control. Proceedings of the VLDB Endowment, vol. 10, issue 5, 2017, pp. 553-564.
[46] Witold Litwin. Linear Hashing: A New Tool for File and Table Addressing. In Proc. of the 6th International Conference on Very Large Data Bases, 1980, pp. 212-223.
[47] S. Pelley, T. F. Wenisch, and K. LeFevre. Do query optimizers need to be SSD-aware? In Proc. of the 2nd International Workshop on Accelerating Data Management Systems using Modern Processor and Storage Architecture, 2011.
[48] F. Faerber, A. Kemper, P. Å Larson, J. Levandoski, T. Neumann, and A. Pavlo. Main Memory Database Systems. Foundations and Trends in Databases, vol. 8, No. 1-2, 2016, pp. 1–130.

Информация об авторе / Information about author

Сергей Дмитриевич КУЗНЕЦОВ – доктор технических наук, профессор, главный научный сотрудник ИСП РАН, профессор кафедр системного программирования МГУ, МФТИ и ВШЭ, старший научный сотрудник РЭУ им. Г.В. Плеханова. Научные интересы: управление данными, архитектуры систем управления данными, модели и языки данных, управление транзакциями, оптимизация запросов.

Sergey Dmitrievich KUZNETSOV - Doctor of Technical Sciences, Professor, Chief Researcher at ISP RAS, Professor at the Departments of System Programming of MSU, MIPT, and HSE, Senior Researcher at REU. Research interests: data management, architectures of data management systems, data models and languages, transaction management, query optimization.

DOI: 10.15514/ISPRAS-2020-32(1)-10

Big Data: Analytical Solutions, Research Challenges and Trends

[1,3] *N.M. Ali, ORCID: 0000-0002-3922-7136 <no3man_mohamed@himc.psu.edu.eg>*
[2] *B.A. Novikov, ORCID: 0000-0003-4657-0757 <borisnov@acm.org>*

[1] *Port Said University,*
Port Fuad, Port Said 42526, Egypt
[2] *National Research University Higher School of Economics,*
3, bldg. 1, ul. Kantemirovskaya, St. Petersburg, 194100, Russia
[3] *Saint Petersburg State University,*
7-9 Universitetskaya Emb., St Petersburg 199034, Russia

Abstract. The term Big Data refers to an extensive collections of digital data generating every second. Produced datasets come in structured, semi-structured, and unstructured formats throughout the world, which is difficult for the traditional database management systems to analyze. Recently, big data analytics emerges as an essential research area due to the popularity of the Internet and the advent of new Web technologies. This growing area of research represents a multi-disciplinary that attracts researchers from various research fields. Interested researchers are invited to design, develop, and implement several tools, technologies, architecture, and platforms for analyzing these large volumes of data. This paper begins with a brief introduction to big data and related concepts, including the main characteristics of big data, followed by discussions of the most significant open research challenges and emerging trends. Next, we review a study of big data analytics, the advantages of using big data solutions, and the preliminary assessments required before migrating from traditional solutions. Finally, we present a review of the recent main applications to obtain a broad perspective of big data analytics.

Keywords: Big Data; Big Data Analytics; Decision Making; Big Data Applications

For citation: Ali N.M., Novikov B.A. Big Data: Analytical Solutions, Research Challenges and Trends. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 181-204. DOI: 10.15514/ISPRAS-2020-32(1)-10

Большие данные: аналитические решения, исследовательские задачи и тенденции

[1,3] *Н.М. Али, ORCID: 0000-0002-3922-7136 <no3man_mohamed@himc.psu.edu.eg>*
[2] *Б.А. Новиков, ORCID: 0000-0003-4657-0757 <borisnov@acm.org>*

[1] *Университет Порт-Саида,*
Египет, 42526, Порт-Саид, Порт-Фуад
[2] *Национальный исследовательский университет «Высшая школа экономики»,*
194100, Россия, Санкт-Петербург, ул. Кантемировская, д. 3, корп. 1
[3] *Санкт-Петербургский государственный университет,*
199034, Россия, Санкт-Петербург, Университетская набережная, д. 7–9

Аннотация. Термин «большие данные» относится к объемным коллекциям цифровых данных, генерируемых каждую секунду. Производимые наборы данных представлены в структурированном, полуструктурированном и неструктурированном форматах по всему миру, и

их трудно анализировать с применением традиционных систем управления базами данных. В последнее время аналитика больших данных становится важной областью исследований из-за популярности Интернета и появления новых веб-технологий. Эта растущая область исследований представляет собой междисциплинарную деятельность, которая привлекает исследователей из различных областей. Исследователи проектируют, разрабатывают и внедряют инструменты, технологии, архитектуры и платформв для анализа этих больших объемов данных. Эта статья начинается с краткого введения в проблематику большие данные и связанные с ними концепции, включая основные характеристики больших данных, после чего обсуждаются наиболее важные открытые исследовательские проблемы и возникающие тенденции. Далее приводится обзор исследований в области аналитики больших данных, обсуждаются преимущества использования решений для больших данных и обсуждаются виды оценок, требуемых перед переходом с традиционных решений. Наконец, представлен обзор основных существующих приложений, обеспечивающий общую панораму аналитики больших данных.

Ключевые слова: большие данные; аналитика больших данных; принятие решений; приложения больших данных

Для цитирования: Али Н.М., Новиков Б.А. Большие данные: аналитические решения, исследовательские задачи и тенденции. Труды ИСП РАН, том 32, вып. 1, 2020 г., стр. 181–204 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(1)–10

1. Introduction

The era of big data is now coming. This fact due to the popularity of the Internet and the advent of Web 2.0 technologies, also the increase of utilizing digital sensors, communications, computation, and storage that create massive collections of data [1]. Now, the volume of data available on the internet measured in exabytes (10^{18}) and zettabytes (10^{21}). Accordingly, expectations refer to that, in the next few years, the volume of data on the internet will exceed the storage capacity of living people's brains around the world [2].

Digital data generated every second throughout the world is producing in a structured, semi-structured, and unstructured format. This massive accumulation of generated data known as «Big Data». Moreover, the generation and adoption of specialized applications related to Social Media, Marketing, E-commerce, etc., provides extensive opportunities and challenges for researchers and practitioners. The erroneous volume of data generated by users using these platforms is the result of the integration between their experience and daily activities [3].

Recently, Big Data analytics has emerged as an important research area and intensively researched. Unfortunately, traditional data analytic techniques may not be able to handle such large quantities of data [4]. Such data consist of data sets that are difficult for legacy database management system to analyze [5]. Therefore, this emerging field has attracted researchers around the world to design, develop, and implement various tools, techniques, architecture, and platforms to analyze this growing volume of generated data [6-9].

Interested researchers are invited to handle the following challenges: how to design and develop a high-performance framework for efficiently analyzing big data; and how to design a suitable algorithm for mining and extracting useful information from big data [4]. To deeply discuss this issue, the structure of the paper is depicted below.

This section is an introductory section about the subjects and motivations for this paper. Section 2 describes topic foundations and the most significant aspects of big data, including the main characteristics of big data, the most significant research challenges, as well as the identification of emerging trends regarding big data. Section 3 presents a study of big data analytics and state the advantages of using big data solutions. Additionally, it involves asserting the necessary preliminary assessments that are required to perform successful migration to the new technologies. Next, Section 4 presents a review of the most significant fields that employ big data applications to obtain a broad perspective regarding big data analytics. Finally, in section 5, we provide some conclusions.

2. Big Data: Concepts, Characteristics, and Challenges

In this section, theoretical conceptualizations of big data and its characteristics are presented to reveal the challenges of tackling big data analytics in various fields of applications. Uncovering these challenges helps to determine, at a high level, essential functional and non-functional requirements that should put into consideration while the process of designing and developing big data analytics frameworks.

2.1 Concepts and Definitions

Proceeding from the fact concerning the existence of many types of modern digital technologies, that have permeated our daily lives like mobiles, sensors, and social media networks as a result of the expansion of using advanced digital artifacts. The proliferation of these technologies in everyday life enhances human-to-human, human-to-machine, and machine-to-machine interaction at unprecedented levels, resulting in vast amounts of data known as Big Data.

Several proposals appeared to describe this phenomenon and to give a definition of the term Big Data, which invented by Roger Magoulas from O'Reilly Media in 2005 [10].

James Manyika et al. [11] define Big Data as «datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze». According to this definition, there are no standard limits for considering dataset as big data (e.g., describe big data in terms of being larger than a certain number of terabytes). Also, we can notice that the volume of data is not the only factor in considering a dataset as big data. Therefore, it is significant to distinguish big data from massive data.

The analyst of Gartner [12] introduces a definition of big data, which considers one of the most comprehensive and widely use in this context, «Big Data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation».

The challenge with Gartner's definition is that, in addition to state the main characteristics of big data, also, it focuses on describing how the benefit of big data can achieve and state the desired outcome. According to a clear understanding of these perspectives, organizations could determine whether they are using big data solutions or even if they have problems that need a big data solution, regarding the difficulties in scoping what is intended to design, developed and delivered, and what the result means to the organization.

In accommodation with Gartner's definition, David Loshin states in [13] that, «Big Data is fundamentally about applying innovative and cost-effective techniques for solving existing and future business problems whose resources requirements exceed the capabilities of traditional computing environments». Furthermore, Krish Krishnan [14] defines Big Data as «volumes of data available in varying degrees of complexity, generated at different velocities and varying degrees of ambiguity, that cannot be processed using traditional technologies, processing methods, algorithms, or any commercial off-the-shelf solutions».

Likewise, based on the distinctions between the capabilities of legacy database technologies and new data storage and processing techniques and tools (e.g., Hadoop clusters, Bloom filters, and R data analysis tools), Davis and Patterson [15] states that big data refers to «data too big to be handled and analyzed by traditional database protocols such as SQL». Also, Paul C. Zikopoulos et al. [16] says, «Big Data applies to information that can't be processed or analyzed using traditional processes or tools».

2.2 The Four V's Characteristics

For the most part, in the popularization of the big data concepts, the group of authors mentioned previously proceeding away from recognizing the size aspect of data only while the

process of defining Big Data! Therefore, there are other meaningful characteristics of big data to be considered in addition to the volume of data. The research community looks sticking with the attraction that appears in the common parts of presented definitions, which focus heavily on what referred to as the 3, 4 or even 9 V's as depicted in [17].

Although the definition of V's is ubiquitous, it should note that the origin of the concept is not entirely new, it provided by the analyst Doug Laney in a research note published by Meta Group (Now Gartner Group), from 2001 concerning «3-D Data Management» [18]. The author noted that the changing of economic conditions, affects the efforts done by companies as they struggle to standardize systems and fold redundant databases to enable greater operational, analytical and collaborative coherence, it also made this task more difficult. Also, he identifies e-commerce as the reason for raising data management challenges across three dimensions: Volumes, Velocity, and Variety. Finally, the author advised information technology organizations to assemble a variety of methods at their disposal to deal with each.

Commonly, big data characterizes by four V's characteristics, as mentioned in Fig. 1: Volume, Velocity, Variety, and Veracity. Other researchers have built upon that trend to include additional V's such as Visualization or Validity, intended to capitalize on an apparent improvement to the definition. As follows, a brief discussion of the fundamental characteristics of big data.

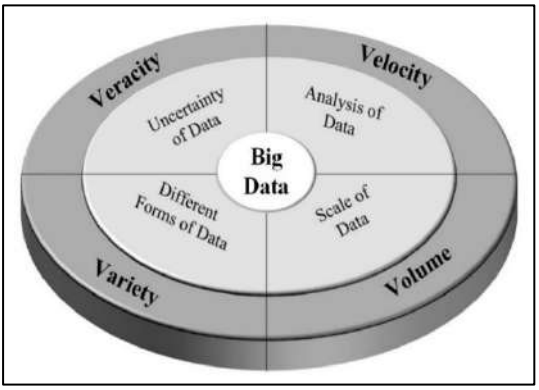


Fig. 1. Big Data Characteristics

- **Volume:** As the name implies, the size of data exceeds the capacity of traditional operational databases or data warehouses. In 2019, Hootsuite & We Are Social, published the Global Digital Statshot report regarding Internet Trends in Q3 [19]. The report displays the continuous growth of digital connectivity at an extraordinary rate around the world. Authors say that every day over the past year, almost 900,000 people came online for the first time.

  Also, many factors participate in increasing the volume of data like streaming data, storing different types of data from social networks, and other resources. Moreover, the Internet of Things (IoT), and scattered sensors all over the world in all devices that generate data every second represents a major grantor to the expanding digital universe [20].

  Consequence, International Data Corporation (IDC), expects that by 2025, the Global Datasphere will grow around 61% from 33 Zettabytes (ZB) in 2018 to 175 ZB. It noted that as much of the data residing in the cloud as in data centers [21].

  One of the primary goals is to make the volume of data useful for users and consumers and optimize future results. Nowadays, with decreasing storage costs, better storage solutions like Hadoop and the algorithms, the processing of large data sets, and creating

meaning from all of the data are not a problem at all. Thus, companies are required to accommodate the new volumes by improving archiving and data importance strategies.

- **Velocity:** Denotes the speed of generating, storing, analyzing, and visualizing the data. It notes the high rate of data streaming into hosting platforms. Currently, the speed of data generation is almost unimaginable. For example, users upload more than 720,000 hours of new content per day on YouTube, which means 500 hours of fresh video per minute [22]. Moreover, there are 500 million tweets sent every day on averages of more than 20,000 tweets per minute [23]. Also, in 2014, the Facebook research center reported that over 4 new petabytes of data generate and run of 600,000 queries per day [24].

  This characteristic, in addition to the high rate of data generation, imposes an essential concern on data aging, and the lifetime of data. How long the data will be valuable is a big challenge that organizations have to cope with, regarding the high rate of data generation and use in real-time. The speed of data generation requires keeping up with processing tasks to meet the demand. Sometimes, the speed of applying the analysis of streaming data is critical [25].

- **Variety:** Big data refers to the large volumes of data generated in different formats. The complexity of Big data formats requires different approaches and techniques to store all raw data. Several different types of data differ in the way of creation and store. These types require various types of analysis to apply or use different tools. According to nature and characteristics, data categorized into three types.

  Structured Data: In the past, most created digital data was structured data, but today it constitutes around 10% of the total digital data. This type of data concerns all data which could be neatly fitted in columns and rows and stored into a spreadsheet or database. Systematic data refers to highly organized information, with relative simplicity in entering, storing, querying, and even analyzing, but a strict pre-definition of the field name and type is indispensable, besides having a relational key to be easily mapped into pre-designed fields. Relational data (e. g. SQL database), Meta-data (e. g. time and date creation), Library catalogs (e. g. date and author), Census records (income and employment), and Economic data (e. g. GDP) are various examples of this type.

  Semi-structured Data: This type refers to data that have some organizational properties, such format could help in the analysis process, also known as a self-describing structure. Unlike structured data, it couldn't establish in a rational database directly among the formal structure of data models associated with relational databases or other forms of spreadsheets. However, this format implies the involving of tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data. Special processing may help to neatly fit, and store data into a spreadsheet or database. Examples of semi-structured data exist like XML (e. g. stored personal data), JSON (e. g. script documents), and NoSQL (e. g. databases like MongoDB).

  Unstructured Data: This type refers to, as the name implies, non-systematic data. That may include every kind of data that carries an unknown form or structure. It may have an internal structure but does not conform neatly into a spreadsheet or database. Today more than 80% of the data that is generated by the organization is unstructured data. There exist many examples of unstructured data like sensor data, social media streams, images, videos, mobile data, text files, etc.

- **Veracity:** Refers to the reliability of the data. Many parts of data consider Not useful such parts involve noise, biases, anomalies, and abnormal data. Wrong data will lead to misleading results and incorrect decisions. So, organizations must decide before beginning the analysis, whether data meaningful or not. Accordingly, ensure the trustworthiness of data sources, in addition to the correctness of data, are mandatory. Today's developers become involves and more willing to invest in the effort to clean up the data at the source.

Consequently, the term Big data exceed the reference to datasets only, and expand to cover space problems, technologies, and opportunities to enhance business value, so it considers a general term. Specifically, the realization of the tremendous business value of data is the main reason for using big data. However, the expectation that the adjective big will fade over time still considered meaningful, as the explicit meaning of the data will be intuitively expanding to include all data types [26].

2.3 Big Data Challenges and Trends

Big Data is a general term for large and complex datasets where traditional techniques and applications for data processing become inadequate. The main objective of processing big data is to help organizations make appropriate decisions on various important issues. Therefore, before moving on to use big data solutions, organizations require to understand the nature and risks involves [27].

Accordingly, we can identify the main challenges of Big Data that should be taken into consideration while handling a big data solution [28-30]: Understanding of Big Data, Data Storage, Data Quality, Data Integration, Data Processing, Data Privacy and Security, Data Visualization, and Data Scalability. Within these challenges, some promoted by the characteristics of big data, others are through the current analytical models and methods, and others are due to boundaries of existing data processing systems [31]. In this section, a brief discussion regarding all the challenges listed in the above presented as follows.

2.3.1 Understanding of Big Data

Big data is basically about applying innovative and cost-effective techniques and technologies for solving existing and future business problems. These solutions handle difficulties that require resources and capabilities beyond the boundaries of traditional computing environments. Organizations require best understand the big data basics like what big data is, what its benefits are, what infrastructure is needed, etc.

Lack of a clear and sufficient understanding of the value that big data can offer an organization leads to the failure of the big data adoption project. The clear understanding and considering the market conditions help to avoid waste lots of time and resources on things they don't even know how to use.

Among the challenges listed before, poor understanding of what big data means, and how to use it could be considered the root of these challenges. It is very significant to differentiate between big data and any other digital trend. The power of big data helps to transform business and boost its efficiency. Organizations must invest in careful planning, come up with a decent strategy, and well-organized architecture. To achieve positive results, and to realize significant effective big data management and reduce the cost of upgrading the future.

Many evaluation criteria need to pass successfully before proceeding with the process of making decisions regarding the integration of big data solutions as a part of an enterprise information management architecture. There exist many variables that are relevant to the evaluation process. These variables include the characterization of the term big data, the examination of the reasons of inadequacy the traditional data management framework with the growing data variability with the evaluation of owned technologies, etc.

2.3.2 Data Storage

The storage and management of the massive data generated through various devices are vital challenges in Big Data. Most enterprise focus settles on analytics issues, but they will never get there without having an efficient, long-term data storage solution to provide a stable

foundation. Data requires a place to stay in, if an organization plan on keeping enormous amounts of data, there is a necessity for invest in storage infrastructure [32].

Several approaches proposed to deal with this problem [33], but the trade-off between cost and effectiveness still a significant challenge. One of the current solutions is to take advantage of another company's infrastructure to save data by using cloud hosting and cloud storage.

On the other hand, concerning a data management perspective, the limitations of existing techniques regarding the high rate of data generation is another challenge. Various techniques proposed to solve this problem that involves the activities of data clustering, data replication, and data indexing [34, 35]. However, these activities impose a development challenge to improve its effectivity and performance [36-38].

2.3.3 Data Quality

Regardless of the size of the data, the need for data quality still urgent. The influence of data quality on achieved business value from big data settled data professionals under stress, which is a fundamental property of data that determines its reliability for making decisions. Preservation of the quality, integrity, and relevance of data represents a trickier challenging task. With the lack of satisfactory quality and relevance, the data processed will be useless [39].

Depending on the type of analysis designed, certain data needs to be collected and managed in a particular way, to handle the new challenges. The collected data must be valuable for the analysis to achieve the correct results. The next step involves applying the appropriate techniques to the gathered data for assuring its quality and relevance.

Organizations look for realizing the maximum values of their data assets. The task of data quality assurance requires to deliver results like trusted analytics, operational reporting, self-service functionality, business monitoring, and governance for taking decisions.

Danette McGilvray proposed many dimensions to describe the quality of big data that represent characteristics or aspects of data quality [40], which depicted in Fig. 2 as follows: Data specifications, Data integrity fundamentals, Duplication, Accuracy, Consistency & Synchronization, Timeliness & Availability, Ease of use & Maintainability, Data coverage, Presentation quality, Perception, Relevance & Trust, Data decay, and Transactability. These dimensions present a method for measuring and managing the quality of data and information. Each one needs various tools, techniques, and processes to measure it. That results in different levels of time, money, and human resources to complete the evaluations.

Fig. 2. Quality Dimensions of Big Data

There is a possibility to adapt traditional data quality techniques like virtualization to the new paradigms of modern data management. Adjustments and optimizations make data quality tasks (e. g. Standardization, deduplication, matching, profiling & monitoring, and customer data) are relevant to big data [41].

2.3.4 Data Integration

The problem of data integration appears when organizations realize there is a need to analyze data that comes from diverse sources in a variety of different formats. The "variety" characteristic of big data makes data integration a great challenge since differences between several data structures became much more significant and matching them is problematic. Additionally, the exponential growth of volumes and velocities, driving both systems and processes to their limits [42].

Frequently, data warehouses use data integration techniques and combine multiple data sources, to consolidate operational system data, and to enhance reporting or analytical needs. An example of integrating data of various types may be that eCommerce companies need to analyze data from website logs, call-centers, e-mails, scans of competitor's websites, and social media. All listed information can be integrated and made available to support decision-makers [28].

The combining of data from diverse external sources is an additional complication imposed by big data on the data integration process, that due to the limited control of organizations over data standards at the source. The integration process of big data involves additional challenges. Such challenges need to be considered as a confusing diversity of big data management technologies that mean risk in the selection, synchronizing data across various sources, the lack of expertise, moving data into a big data platform, extracting useful information from big data, etc. [43].

2.3.5 Data Processing

Data processing is the method to be applied after the data storage to extract useful information. The primary goals of data processing are to get a significant relationship between the various

fields of data collection, and the extraction of valuable analysis. The complexity and scalability of big data make data processing tasks more complicate and imposes another challenge [44].

The traditional paradigm regarding performing operations on the consolidated datasets becomes inappropriate. Also, the combination of all related data in a whole database may cost a lot of time and require extra investments in infrastructure. The shortages of traditional techniques appear during the time it takes to analyze through a single set of datasets, while the speed of processing a query in big data is significant demand [45].

Generally, the data processing tasks employ two main techniques, classification, and prediction. Classification is a data mining technique used to divide datasets into different categories and groups. On the other hand, the prediction is responsible for making decisions based on the analysis results of past transactions. Regarding big data, in the case of manipulating massive volumes of data collected from various sources with different formats, the data processing task becomes tricky.

Therefore, a shift in paradigm includes application parallelization to simultaneously process against multiple chunks of data and divide-and-conquer are natural computational techniques for handling big data problems. This approach moves the processing to the place where the data stored by distributing the query and update requests across distributed servers rather than attempting to process against one combination of a dataset.

Several techniques introduced in the context of big data processing for storing the unstructured data in distributed databases, like HBase, Apache Cassandra, or SimpleDB , and for classifying data, techniques like MapReduce in Hadoop introduced [46]. Also, to enhance query processing, optimization techniques introduced like HiveQL, SCOPE, etc. [47].

2.3.6 Data Privacy and Security

The enormous volumes of big data may threaten to overwhelm the organization's ability to preserve its privacy, as analysis results provide complete information about activities and processes. Consequently, giving a low priority to the security of big data, and turning it off until later stages of big data adoption projects, is not always a smart move, "Security first" is an indispensable requirement [48].

Recently, the number of cyberattacks has a significant increase, and their sophistication is expanding, seeking to traverse corporate firewalls, extract critical business information, or gradually deplete individual financial accounts while working entirely under the radar. Monitoring for cybersecurity events and ensure the privacy and security of data lies in the organization that is responsible for keeping the data (e. g. Cloud providers) from various sources and a wide variety of massive streaming datasets [49].

Therefore, the necessity to quickly capture and integrate threats into a model is inevitable, which enables for identification of known attack patterns as well as the discovery of the new emerged patterns as the attacks become more complicated [50]. Hence, the new security intelligence solutions are required to combine big data with advanced analytics, to link security events across multiple data sources, and provide early detection of suspicious activities. As follows, a list of the most ferocious security challenges that big data involve [51-54]:

1. the access and usage of data by unauthorized persons;
2. absence of security audits;
3. encryption protection problems;
4. possibility to extract sensitive information;
5. high speed in the development of NoSQL databases and lack of security focus;
6. data source difficulties;
7. the generation of fake data.

2.3.7 Data Visualization

The presentation of data in a readable manner and making it understandable for users represents a difficult task. Data visualization task includes the representation of essential information and knowledge efficiently and intensively by employing various types of infographics and other visual formats. Regarding the characteristics of big data (e. g. Large volume, variety, and velocity of the information), visualization becomes a big challenge [55].

Frequently, the use of data visualization gives the ability to summarize and review large volumes of data into a format that is helpful to human consumption, which may provide the ability to move to an additional level of detail upon request. Visualization of big data aims to provide decision-makers and other business users with insights.

Several solutions for data visualization are available like Tableau, Microsoft Power BI, Sisense, QlikView, etc., and the selection of the right and the appropriate tool indicates a bit tricky. Also, choosing the most proper data visualization technique for application from a wide variety of popular methods and techniques (e. g. Symbol maps, tree map, line charts, pie charts, heat maps, bar charts, scatter plot, map chart, parallel coordinate plot, etc.) appears to be a more complicated task than it seems [56].

2.3.8 Data Scalability

The organizations may be hindered by the limitations of their owned infrastructure for data acquisition for analytics while dealing with enormous volumes of big data. Although scalability is not a unique challenge in big data solutions, the ability to grow represents a crucial feature of any big data solution. Regarding the high complexity of algorithms, software scalability has always been a problem [57].

The scalability meaning in big data refers to the ability to grow with the rapid increase of data volumes, which require a change in the storage process, management, and analytical techniques. Therefore, if the technical infrastructure of an organization designed around a traditional data warehouse information flow, it may hinder its ability to handle big data problems and capability to manage the hesitated volume of data in real-time.

Traditionally, the research efforts for solving scalability problems concentrate on parallelizing the computation, with less resolution on storage distribution. Commonly, scaling approaches classify into two main classes: vertical and horizontal scaling [4, 58].

Vertical scaling, also known as scaling-up, aims to improve the performance of the system, it could achieve by enhancing the capability of processing platforms by adding additional computing power (e. g. RAM, CPU, etc.) to accommodate further data volumes. An example regarding this type of scaling platform is High-Performance Computing (HPC) clustering. This approach hinders by its high cost and complexity in terms of maintenance, also by the restrictions imposes by the platform upper limits (e. g. Maximum capacity of RAM, number of CPUs, etc.).

On the other side, horizontal scaling, also knowns as scaling-out, could achieve by adding more machines interconnected over a network. It employs a divide-and-conquer approach (e. g. Apache Hadoop). That helps in distribute the workload and generate parallel processing over multiple independent computing machines. Accelerate data processing by adding more computing machines as much as needed to improve the overall system performance. The ability to run and maintaining different computing machines with various operating systems impose additional challenges toward managing and maintenance these instances [51, 59].

In big data, the volume and variety of data can differ dynamically in response to potentially variable user demand. Hence, the process of up-and-down scaling according to the request for computational resources represents an important characteristic of big data solutions. That is due to the difficulties concerning allocation and de-allocation of resources in real-time, as they have an impact on the overall system performance.

3. Big Data Analytics

Today, enterprises seek to discover facts they didn't know before by searching for massive amounts of highly detailed data. Analysis of large datasets unfolds and improves business values. However, the growing volumes of data, increase the difficulties with management and manipulation. This section demonstrates the concepts and basics of big data analytics and the benefits of exploiting available assets of big data regarding Business Intelligence (BI) [60].

3.1 Topic Conceptualization

Big data analytics refers to the advanced analytic techniques to apply on big data sets [61]. An example of analytical tasks may include searching for specific data, and patterns, data retrieval, and organization, etc. Therefore, the application of superior analytical techniques comprises the entire processes and tools required for knowledge extraction [62]. These processes incorporate multiple tasks that start with data acquisition and extraction, followed by the transformation of data. Next, preparing and loading data for analysis that includes the employment of appropriate tools and techniques for getting desirable results. Finally, the delivery of realized results to support decision-makers [58].

Generally, data analysis involves the examination of sets of data to uncover hidden patterns, correlations, and other insights as well as rendering conclusions. These tasks could classify into three main areas, Statistical analysis, modeling, and predictive analysis [63]:

1. *Statistical Analysis*: Regarding business intelligence, it involves the collection and examination of each sample of data in a set of elements from which it can draw. Statistical analysis strictly related to hypothesis testing. The main goal of statistical analysis is to recognize and predict future trends.
2. *Modeling*: Refers to the processes used to identify, describe, and analyze the requirements of data, also to describe the overall behavior of a system, it plays a crucial role in the growth of any business. These processes include the use of mathematical equations or some logical language. The main objective of data modeling is to support business processes and give answers to business questions more easily and quickly.
3. *Predictive Analysis*: Represent a form of advanced analytics that concerned with guessing how an individual, group, or data object will behave and forecasting trends based on historical data or the recognized behaviors of similar individuals or groups. It includes several types of algorithms like recommenders, classifiers, and clustering.

Big data analytics are different from small and traditional data analytics. From this perspective, Joshua Eckroth [6] defines big data in adaption with a definition from Philip Russom [61] as follows: "A data analysis task may be described as big data if the data to be processed have such high volume or velocity that more than one commodity machine is required to store and/or process the data".

Otherwise, from the perspective of Business Intelligence and the advantages of data analytics, Rick F. van der Lans [64] says, “Big data is revolutionizing the world of business intelligence and analytics”. Thus, big data analytics revolves around two items, big data, and analytics, as well as how the collaboration and the combination between the two items could perform to create one of the most intellectual trends in business intelligence today [61].

3.2 Advantages of Big Data Analytics

Despite the difficulties and challenges involves in the process of building and developing big data analytics platforms due to the complex nature of big data, enterprises working quickly to build analytical solutions for big data. This situation is because of the great opportunity it offers to upgrade from the traditional methods of information extraction into new dimensions.

Decisions made without data-driven answers will likely fail, so organizations must build their systems that support data-driven decision making [59].

Traditionally, organizations recognized that the insights of owned data could extensively benefit their business performance. The importance of big data does not rely only on the volume of data a company has but how a company utilizes the collected data. Hence, big data analytics represents a competitive advantage for businesses. Achieved benefits and values utilizing big data analytics solutions could be determined depending on the way enterprises use the data [65].

Enterprises are now facing challenges to create new business actions based on the benefits brought by available types of analysis. Efficient analysis of owned and collected data helps the business to find answers which will improve the organizational value [66]. Moreover, companies that use comprehensive big data analytics solutions realize the benefits, obtaining even more insights that drive intelligent decision-making. These insights represent new means of making business by leveraging new types of analytics across different types of data. Some benefits of using big data analytics solutions include [67-69]:

- the identification of the root causes of failure and issues in real-time;
- cost-saving;
- reduce processing time;
- enable the development of new products and services;
- a better understanding of market conditions;
- reassess risk portfolios quickly;
- full understanding of the potential of data-driven marketing;
- improve customer engagement and increase customer loyalty;
- personalize the customer experience;
- generate customer offers based on their buying habits;
- add value to online and offline customer interactions;
- controlling the reputation online.

Therefore, we can say that the significant contribution of big data to the organization revolves around four main elements, the essence of these items reflects on adding value to the organization by Increasing revenues, lowering costs, increasing productivity, and reducing risk.

3.3 Preliminary Assessments

Before starting to build a new big data solution, enterprises are required to perform primary assessments. The Migration from traditional analytic solutions to big data may cost a lot of time and require extra investments in infrastructure. So, it is significant to know the actual state of the business and evaluate the needs of this step. The trade-off between costs and potential benefits is essential to select the best path and decide whether to continue with traditional schemes or to transform into the new one. Without proper organizational evaluation and preparedness, neither strategy is likely to succeed [70].

The big data approach is more appropriate to address or resolve business problems that meet one or more of the following criteria as depicted in the Fig. 3 [71]: Data throttling, computation-restricted throttling, large data volumes, significant data variety, and benefits from data parallelization. These evaluation criteria can be used to assess the degree of relevance of business problems toward big data technology if there exists a correlation with business problems, whose solutions fit big data analysis applications.

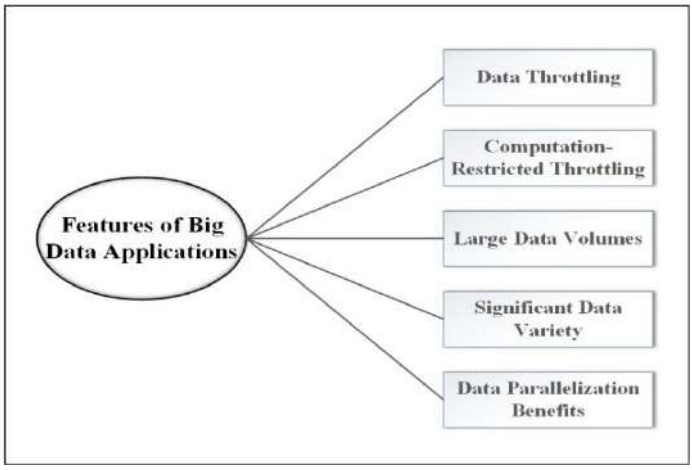


Fig. 3. Evaluation Criteria to Build a Big Data Solution

4. Big Data Applications

Concerning the continuous growth in the field of big data analytics and its substantial impact on human life, big data applications introduce cutting-edge opportunities in every aspect of our daily life. The primary goal of big data applications is to help companies make more informative-business decisions by analyzing large volumes of data, given the tremendous competition where we are living overall the world [72].

The applications of big data positively change our life for the better and smoother as well. We are utilizing big data solutions to improve our efficiency and productivity. This opportunity encourages researchers and technology providers to develop complicated platforms, frameworks, and algorithms to struggle with the challenging of big data. In this section, the most significant fields where use big data applications will discuss as follows.

4.1 Education

Education is the backbone of any country; it represents the engine of growth and prosperity in the lives of nations. Successive civilizations reflect the level of scientific and civilized development throughout the ages. Based on the great importance of education, it was necessary to continuously develop educational systems around the world to raise the efficiency of the educational process and the quality of the service provided [73].

The main characteristics of education development appear clearly in the expansion of using e-learning systems by offering the courses online [74]. These systems offer interactive online courses that involve carrying out assignments by learners as well as submitting results [75]. Data generated by using these educational systems are personal information, academic progress, attendance, student status, student activities, and interests, etc. Additionally, other types of data offered by systems to administrators like financial status, course plans, staff details, organization details, etc.

The responsibility to ensure the quality of education depends entirely on the government and educational institutions. Many of them are planning to implement smart education systems to improve e-learning systems in their countries. Since the availability of large quantities of data generated from several types of educational frameworks, the necessity to analyze such data to obtain insights is growing. Analysis of these types of data requires special techniques for efficient processing that are more suitable for big data techniques [76].

Big data solutions can produce fantastic results and offer innovative data-driven approaches for student learning. In many countries, using big data applications are common in schools and

colleges. But developing countries are also gaining new technologies. Using these technologies enables storing, managing, and analyzing large datasets with maintaining security [77]. Also, it provides relevant data on class activities that are recorded using high-definition cameras and video clips that help evaluate student facial expressions and can track their movements as well as make decisions for organizations [78].

4.2 Agriculture

Starting with the significant role of agriculture that plays in the development of the social economy of any country, this has been a growing interest in developing big data solutions in this industry. Generally, agriculture is based mainly on geographical and climatic conditions. The main factors on which crop production depends are climate, temperature, precipitation, agriculture, fertilizers, pesticides, etc.

Nowadays, due to the advent of advanced technology in the area of agriculture, the procedures of crop production have changed, and it is possible to control greenhouses and cultivate crops by managing the temperature, humidity, sunlight, etc. The most recent greenhouses are provided with the latest sensors devices to determine the quality of the soil condition that represents an essential factor for crop yield [79].

In agriculture, big data is playing an influential role in improving the performance of the firms. The goal is to optimize crop efficiency by minimizing the firm's loss and increasing the generation of necessary food grains. The amount of collected data from sensors during the process of planting crops and running simulations to measure how plants react to various changes in conditions are pretty huge [80]. So, the processing of these data using big data analytics techniques allows it to discover the optimal environmental conditions for specific gene types [81].

The use of big data analytics helps the producers in overall the processes to decide on the crops, fertilizers to use, pesticides, etc., as well as from harvesting to distribution process of agricultural products like paddy, wheat, vegetables to increase the benefits. Other advantages may include the automation of the watering system of the firm and enabling the firm's owners to use the same land for several purposes throughout the year without any interval [82, 83].

4.3 Healthcare

Big data analytics have already affected patient care and pharmaceutical manufacturers [84]. Traditionally, the use of big data solutions in healthcare manufacturing has been much delayed compared to other industries. This problem caused by several factors, one of them relies on the resistance to change by service providers of using an independent-decisions approach that employing their clinical judgment to make treatment; rather than relying on protocols of big data [85].

Additionally, critical information inside a single hospital, payor, or pharmaceutical company, is often kept silent and isolated within a single group or department. That caused by the lack of procedures in organizations to integrate data and report results. Other factors are more structural. Recently, healthcare stakeholders like pharmaceutical industries and hospitals, now have access to promising new threads of knowledge and adopting with the facilities provided by big data analytics [86].

Although big data technology is in the initial stage in healthcare, these technologies help the industry make critical decisions, which in turn play an important role in enhancing the ability of healthcare organizations to make a good profit by providing services to patients. These reasons have made the healthcare industry a significant commercial system. Typically, healthcare stakeholders are using online frameworks to publish diagnostical reports, analysis reports, schedule appointments, monitor patient status, and preserve records; these systems represent a principal resource for big data.

The growing generating of tremendous varieties of data of healthcare systems like clinical trials, medications, exercise directions, allergies, insurance data, visiting schedule, treatment follow-ups, etc., represents a rich source of data that make pharmaceutical industry experts, payers, and providers, for turning to analyze such data to obtain insights.

The volume, complexity, and diversity of these types of data make traditional relational database systems are unable to manage and process those large datasets. However, the capability of big data techniques to manipulate whatever the type of data allows the manipulation of those datasets efficiently. So, recent technological advances in the industry have improved their ability to handle this data, as well as developing secure frameworks [87-89].

The introduced solutions aim not only for treatment identification but also for improving the process of rendering healthcare. Additionally, big data has a high impact on reducing consumption of money and time; it enables the development of new infrastructure and emergency medical services. The use of big data has many advantages that are difficult to list fully in this context, for example, the evaluation of symptoms and identification of many diseases at the early stages based on the availability of medical databases, which plays a significant role in disease prediction.

4.4 Smart Cities

According to the United Nations estimations, 1.3 million people are moving into cities each week, and by 2050, 68% of the world's population is expected to be living in cities, with close to 90% of this urban population growth set to occur in Africa and Asia [90]. Therefore, experts are spending numerous efforts attempting to improve the quality of life in our cities.

Generally, the term Smart City mainly supplies by the advent of IoT (Internet of Things) and Big Data [91]. A simple definition of a smart city is "A city equipped with the basic infrastructure to provide a high-quality lifestyle to its citizens". There are many areas of development in the city that must identify to reconfigure the current situation like water management, waste management, transportation, and safety. Smart cities of the future must comprise the necessities and higher technologies for effortless and elementary living.

The massive expansion in the use of IoT technologies has made it easy to communicate with devices without human intervention to collect data in real-time [92, 93]. Sensors installed all over the city producing data regarding critical infrastructures like rails, airports, seaports, roads, power, water, and communication. These devices, in addition to other types like cameras, GPRS, etc., generate enormous amounts of data, effective use enables achieving of many improvements. Collected data from these resources serve as a tremendous source for big data [94].

Big data plays a significant role in processing the gathered data and represents an emerging trend in the field of information systems; so that further analysis can be made to recognize the patterns and needs in the city. Analytical solutions based on big data aim to get insights and extract correlations relationships to improve services provided to residents by optimizing the usage of resources, and managing maintenance activities, as well as to support the decision-making processes. Many challenges are facing the development process of city-level smart information services like the integration of generated datasets from various city domains, and analysis process as well [95, 96].

The analysis of these resources offers many benefits of smart cities, particularly three main areas attest to why data-driven innovation is crucial to the future of urban life as follows [26, 97, 98]:

1. intelligent traffic management through the proper utilization of historical data;
2. promoting public safety by using predictive analytics that helps to examine historical and geographical data to recognize when and where crimes occur;
3. managing smart cities infrastructure, the evaluation of the current situation helps to enhance city planning, effective spending, and maintain sustainability.

4.5 Criminal Analysis & Fraud Detection

With the continuous development of criminal methods, which makes it more professional and sophisticated, it becomes necessary to resort to advanced techniques that rely on data-driven analytics to meet growing risks and challenges. Criminal operations differ in their forms and domains, and among the most complex are those carried out through using electronic devices and the disappearance from behind them to cover up the eyes. The determination of fraud against enterprises that involves any type of operations like claims or transaction processing is one of the most compelling examples of big data applications.

Recently, governments, security agencies, and enterprises have started using big data analytics in the fields of security and law enforcement. Various domains are affected by well-planned crimes such as drug trafficking, kidnaps, terror attacks, fraud, and robberies in an increasing manner. Big data offers a large variety of solutions to handle these types of crimes effectively, which already proved to be very beneficial in preventing criminal activities [99-101].

Historically, the discovery of fraud has proven to be an elusive purpose. Usually, the detection of fraud done long after it occurs. That means actual injury has already happened, and all that remains is to reduce the damage and set policies to decrease the opportunity of repeating occurrence. The advantages of using a well-designed big data framework play a significant role and could change the game of fraud detection. It can investigate claims and transactions in real-time, discover general patterns across multiple transactions, or recognize abnormal behavior from an individual user [102-104].

Many areas are using big data technologies to detect fraud and electronic theft. The collected information for criminal analysis comes from various sources like bank transaction records, mobile call records, web, and social media (e. g. Social meeting sites like Facebook, Twitter, and LinkedIn), etc. [105, 106]. These areas include what is related to stealing money through fake electronic cards and using them in purchases and obtaining services illegally.

Credit card fraud considers one of the most common aspects of electronic fraud. Therefore, many approaches introduced to handle this problem and related fields, including illegal purchased for goods and services [107-109]. Another example compromises to healthcare services, that individuals and criminal networks whose commit fraud for nefarious reasons, personal gain, and obtaining medicines and medical supplies to gain private benefits illegally. That works on eliminating the ability of enterprises to effectively provide the healthcare needs of the elderly and other qualifying people [110-112].

4.6 Government

Government work provides an opportunity to help the public or provide services that add real benefit to the lives of citizens. They need to deal with various complex local, national, and global issues daily. One of the greatest strengths of big data is flexibility and the overall application of many different industries. Besides many other areas, big data in government can have a tremendous impact. Much of this work is being done in the public sector using big data and analysis, due to the use of analyzes can improve the results of general programs.

The implementation of a big data platform can leave an enormous impact on the governmental sector [113]. Governments can access vast amounts of relevant information about millions of people necessary to accomplish their daily functions, as well as help to make any decision regarding locals. Governments have to try to make sense and analyze the impact and opinions about vital decisions that affect millions of people, and to decide if any change is needed or not [114, 115].

The positive impact of using big data solutions is almost endless. It is very significant because it not only allows the government to identify areas that need attention but also gives it that information in real-time [116, 117]. Big data analytics has proven to be very useful in the government sector regarding the significant role it plays in election campaigns. Additionally,

governments utilize numerous techniques to ascertain how the electorate is responding to government action, as well as ideas for policy augmentation [118].

Big data analytics provide tremendous benefits to the public sector by improving the outcomes that have a direct impact on citizens' lives [119]. Examples of these sectors that can be applied to achieve tangible results may include Emergency response, anti-money laundering, insider threats, workforce effectiveness, etc. Moreover, it can help in the success of government campaigns to eliminate problems that affect national security, such as the drug problem and poverty. Besides, it helps in predict any terrorist attack and take necessary action to prevent unwanted conditions. Finally, the analytical ideas obtained from owned big data stores could make a difference and make the government more effective [120, 121].

4.7 Marketing & E-Commerce

The tremendous expansion in the use of modern technology and various devices in commercial transactions has made marketing and electronic commerce, one of the most informative areas. This development has brought about drastic changes in the map of the global economy due to the successive changes in trade and marketing strategies to keep pace with this tremendous development. Marketing trends for companies have completely changed [122].

Digital marketing is the key to success in any company. Now, any business can manage marketing promotional activities and run successful advertising campaigns and promote their products and services regardless of their size. Big data has made digital marketing powerful and has become an essential part of any business. Various forms of marketing campaigns appear in different places, such as social media platforms (e. g. Facebook, LinkedIn, Twitter, etc.), ads placed on YouTube and TV, companies' websites, E-Markets, Text messages, E-mails, etc.

Big data analytics offers several solutions for helping enterprises in analyzing available tremendous datasets [123]. One example investigates at what kinds of advertisements compel viewers to continue watching and what turns viewers off. It uses facial-recognition software to learn how well their advertising succeeds or fails at stimulating interest in their products. That helps marketers to create widely accepted ads to increase sales.

Another example of using big data solutions for analyzing customer calls. Analyzing the content of customer contact records with call centers helps determine their sentiment that represents a powerful barometer and influencer of the market sentiment. Big data solutions can help recognize repeating issues or patterns of customer and employee behavior not only by understanding time/quality accuracy metrics but also by recording and analyzing the content of the call.

The advent of social media is one of the most substantial contributors to big data. Based on its significant importance, various solutions introduced to analyze user's activities [124-126]. Big data analytics can provide valuable insights in real-time about how the market is responding to products and campaigns [127]. According to obtained results, companies can adjust their prices, forecasting demand and its distribution, adjusting production, update distribution strategies, promotions, and campaign placements accordingly [128-131]. Therefore, to know the consumer mindset, it is necessary to apply smart decisions derived from big data [132].

The comprehensive development of e-commerce and the expansion of establishing electronic markets for online shopping like Amazon, eBay, Walmart, Best Buy, Wish, etc., have created a competitive environment between companies to attract the highest number of customers. Enterprises seek to measure several factors like customer satisfaction, loyalty, and the success of marketing strategies by analyzing customer reviews through electronic platforms such as websites and social media [133-136]. But the analysis process must involve excluding fake reviews and negative comments released by competitors [137].

5. Conclusions

Digital data generated every second throughout the world is producing in a structured, semi-structured, and unstructured format. Unfortunately, traditional data analytics techniques are not able to handle these volumes of data considering their complex structures. Therefore, big data analytics has emerged as a substantial research area, and intensively researched to handle these problems.

In this paper, we present theoretical conceptualizations of big data and its characteristics to reveal the challenges of tackling big data analytics in various fields of applications. Uncovering these challenges helps to determine, at a high level, essential functional and non-functional requirements that should put into consideration while the process of designing and developing big data analytics frameworks. Also, we gave a demonstration of the concepts and basics of big data analytics and the benefits of exploiting available assets of big data regarding Business Intelligence.

The primary goal of big data applications is to help companies make more informative-business decisions by analyzing large volumes of data, given the tremendous competition where we are living overall the world. In this paper, we review the most significant fields that employ big data applications to demonstrate how it positively changes our life for the better and smoother as well as to improve our efficiency and productivity. Also, we state opportunities for researchers to develop complicated platforms, frameworks, and algorithms to struggle with the challenging of big data. In the future, the research will move towards the landscape of big data tools and specialized techniques offered for big data analytics, investigating its functionality and limitations.

References

[1] Ghani N.A., Hamid S., Hashem I.A.T., Ahmed E. Social Media Big Data Analytics: A Survey. Computers in Human Behavior, vol. 101, 2019, pp. 417-428.

[2] Emani C.K., Cullot N., Nicolle C. Understandable Big Data: A Survey. Computer Science Review, vol. 17, 2015, pp. 70-81.

[3] Stieglitz S., Mirbabaie M., Ross B., Neuberger C. Social Media Analytics – Challenges in Topic Discovery, Data Collection, and Data Preparation. International Journal of Information Management, vol. 39, 2018, pp. 156-168.

[4] Tsai C.W., Lai C.F., Chao H.C., Vasilakos A.V. Big Data Analytics: A Survey. Journal of Big Data, vol. 2, no. 21, 2015, pp. 1-32.

[5] Yadav K., Rautaray S.S, Pandey M. A Prototype for Sentiment Analysis Using Big Data Tools. In Proc. of the First International Conference on Computational Intelligence, Communications, and Business Analytics, 2017, vol. 775, pp. 103–117.

[6] Eckroth J. A Course on Big Data Analytics. Journal of Parallel and Distributed Computing, vol. 118, no. 1, 2018, pp. 166-176.

[7] Smirnova E., Ivanescu A., Bai J., Crainiceanu C.M. A Practical Guide to Big Data. Statistics & Probability Letters, vol. 136, 2018, pp. 25-29.

[8] Siddiqa A., Hashem I.A.T., Yaqoob I., Marjani M., Shamshirband S., Gani A., Nasaruddin F.A Survey of Big Data Management: Taxonomy and State-of-the-Art. Journal of Network and Computer Applications, vol. 71, 2016, pp. 151-166.

[9] Soufi A.M., El-Aziz A.A.A., Hefny H.A. A Survey on Big Data and Knowledge Acquisition Techniques. IPASJ International Journal of Computer Science (IIJCS), vol. 06, no. 07, 2018, pp. 15-29.

[10] Halevi G., Moed H.F. The Evolution of Big Data as a Research and Scientific Topic: Overview of the Literature. Research Trends, no. 30, 2012, pp. 3-6.

[11] Manyika J., Chui M., Brown B., Bughin J., Dobbs R., Roxburgh C., Byers A. H. Big Data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Global Institute, 2011, 143 p.

[12] Gartner Glossary: Big Data. Available at: https://www.gartner.com/en/information-technology/glossary/big-data, accessed 14.10.2019.

[13] Chapter 1. Market and Business Drivers for Big Data Analytics. In Loshin D. Big Data Analytics, Morgan Kaufmann, 2013, pp. 1-9.
[14] Chapter 1. Introduction to Big Data. In Krishnan K. Data Warehousing in the Age of Big Data, Morgan Kaufmann, 2013, pp. 3-14.
[15] Davis K. Ethics of Big Data: Balancing Risk and Innovation. O'Reilly Media, 2012, 82 p.
[16] Zikopoulos P.C., Eaton C., deRoos D., Deutsch T., Lapis G. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill Osborne Media, 2012, 176 p.
[17] Owais S. S., Hussein N. S. Extract Five Categories CPIVW from the 9V's Characteristics of the Big Data. International Journal of Advanced Computer Science and Applications (IJACSA), vol. 7, no. 3, 2016, pp. 254-258.
[18] Laney D. 3D Data Management: Controlling Data Volume, Velocity and Variety. Application Delivery Strategies, META Group Research Note, 2001.
[19] Kemp S. Digital 2019: Internet Trends in Q3 2019. Available at: https://datareportal.com/reports/digital-2019-internet-trends-in-q3, accessed 06.11.2019.
[20] Kemp S. Digital Trends 2019: Every Single Stat You Need to know About the Internet. Available at: https://thenextweb.com/contributors/2019/01/30/digital-trends-2019-every-single-stat-you-need-to-know-about-the-internet/, accessed 06.11.2019.
[21] Reinsel D., Gantz J., Rydning J. The Digitization of the World: From Edge to Core, 2018, Available at: https://www.seagate.com/our-story/data-age-2025/, accessed 06.11.2019.
[22] Hale J.L. More Than 500 Hours of Content Are Now Being Uploaded to YouTube Every Minute. Available at: https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/, accessed 07.11.2019.
[23] Mention.com. 2018 Twitter Report. Available at: https://mention.com/en/reports/twitter/, accessed 07.11.2019.
[24] Wiener J., Bronson N. Facebook's Top Open Data Problems. Available at: https://research.fb.com/blog/2014/10/facebook-s-top-open-data-problems/, accessed 07.11.2014.
[25] Torrecilla J.L., Romob J. Data Learning from Big Data. Statistics and Probability Letters, vol. 136, 2018, pp. 15-19.
[26] Osman A.M.S. A Novel Big Data Analytics Framework for Smart Cities. Future Generation Computer Systems, vol. 91, 2019. pp. 620-633.
[27] Jin X., Wah B.W., Cheng X., Wang Y. Significance and Challenges of Big Data Research. Big Data Research, vol. 2, no. 2, 2015, pp. 59-64.
[28] Reeve A. Chapter 21. Big Data Integration. In Reeve A. Managing Data in Motion, Morgan Kaufmann, 2013, pp. 141-156.
[29] Dhupia B., Rani M.U. Research Challenges in Big Data Solutions in Different Applications. In Social Network Forensics, Cyber Security, and Machine Learning, Springer, 2019, pp. 105-116.
[30] Baig M.I., Shuib L., Yadegaridehkordi E. Big Data Adoption: State of the Art and Research Challenges. Information Processing & Management, vol. 56, no. 6, 2019, article 102095.
[31] Malik S.U.R., Khan S.U., Ewen S.J., Tziritas N., Kolodziej J., Zomaya A.Y., Madani S.A., Min-Allah N., Wang L., Xu C.-Z., Malluhi Q.M., Pecero J.E., Balaji P., Vishnu A., Ranjan R., Zeadally S., Li H. Performance Analysis of Data Intensive Cloud Systems Based on Data Management and Replication: A Survey. Distributed and Parallel Databases, vol. 34, no. 2, 2016, pp. 179-215.
[32] Bellatreche L., Furtado P., Mohania M.K. Guest Editorial: A Special Issue in Physical Design for Big Data Warehousing and Mining. Distributed and Parallel Databases, vol. 34, no. 3, 2016, pp. 289–292.
[33] Lakshman A., Malik P. Cassandra: A Decentralized Structured Storage System. ACM SIGOPS Operating Systems Review, vol. 44, no. 2, 2010, pp. 35-40.
[34] Dean J., Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, vol. 51, no. 1, 2008, pp. 107-113.
[35] Rotondi Azevedo D.N., Parente de Oliveira J.M. Application of Data Mining Techniques to Storage Management and Online Distribution of Satellite Images. Studies in Computational Intelligence, vol. 169, 2009, pp. 1-15.
[36] Agrawal D., Abbadi A.E., Antony S., Das S. Data Management Challenges in Cloud Computing Infrastructures. Lecture Notes in Computer Science, vol. 5999, 2010, pp. 1-10.
[37] Buza K., Nagy G.I., Nanopoulos A. Storage-Optimizing Clustering Algorithms for High-Dimensional Tick Data. Expert Systems with Applications, vol. 41, no. 9, 2014, pp. 4148-4157.

[38] Mateus R.C., Siqueira T.L.L., Times V.C., Ciferri R.R., de Aguiar Ciferri C.D. Spatial Data Warehouses and Spatial OLAP Come Towards the Cloud: Design and Performance. Distributed and Parallel Databases, vol. 34, no. 3, 2016, pp. 425–461.
[39] Merino J., Caballero I., Rivas B., Serrano M., Piattini M. A Data Quality in Use Model for Big Data. Future Generation Computer Systems, vol. 63, 2016, pp. 123-130.
[40] McGilvray D.. Executing Data Quality Projects: Ten Steps to Quality Data and Trusted Information. Morgan Kaufmann, 2008, 352 p.
[41] Russom P. Data Quality in the Age of Big Data. Available at: https://tdwi.org/Articles/2019/04/19/DIQ-ALL-Data-Quality-in-the-Age-of-Big-Data.aspx?Page=1, accessed 18.11.2019.
[42] SAS. Data Integration Déjà Vu: Big Data Reinvigorates DI - White Paper. Available at: https://www.sas.com/ru_ua/whitepapers/data-integration-deja-vu-107865.html, accessed 18.11.2019.
[43] FlyData I. The 6 Challenges of Big Data Integration. Available at: https://www.flydata.com/the-6-challenges-of-big-data-integration/, accessed 18.11.2019.
[44] Akusok A., Björk K.-M., Miche Y., Lendasse A. High-Performance Extreme Learning Machines: A Complete Toolbox for Big Data Applications. IEEE Access, vol. 3, 2015, pp. 1011–1025.
[45] Ji C., Li Y., Qiu W., Jin Y., Xu Y., Awada U., Li K.,Qu W. Big Data Processing: Big Challenges and Opportunities. Journal of Interconnection Networks, vol. 13, no. 03 & 04, 2012, article 1250009.
[46] White T. Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale, 4 ed. O'Reilly Media, Inc., 2015, 756 p.
[47] Candela L., Castelli D., Pagano P. Managing Big Data through Hybrid Data Infrastructures. ERCIM News, no. 89, 2012, pp. 37-38.
[48] Tao H., Bhuiyan M.Z.A., Rahman M.A., Wang G., Wang T., Ahmed M.M., Li J. Economic Perspective Analysis of Protecting Big Data Security and Privacy. Future Generation Computer Systems, vol. 98, 2019, pp. 660-671.
[49] Tawalbeh L.A., Saldamli G. Reconsidering Big Data Security and Privacy in Cloud and Mobile Cloud Systems. Journal of King Saud University - Computer and Information Sciences, May 2019, 10 p. DOI: 10.1016/j.jksuci.2019.05.007
[50] Kantarcioglu M., Xi B. Adversarial Data Mining: Big Data Meets Cyber Security. In Proc. of the ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 1866-1867.
[51] Cavoukian A., Chibba M., Williamson G., Ferguson A. The Importance of ABAC to Big Data: Privacy and Context. The Privacy and Big Data Institute, Ryerson University, Toronto, Canada, 2015. Available at: https://www.ryerson.ca/content/dam/pbdce/papers/The-Importance-of-ABAC-to-Big-Data-05-2015.pdf, accessed 18.11.2019.
[52] Talha M., Kalam A.A.E., Elmarzouqi N. Big Data: Trade-off between Data Quality and Data Security. Procedia Computer Science, vol. 151, 2019, pp. 916-922.
[53] Xu L., Jiang C., Wang J., Yuan J., Ren Y. Information Security in Big Data: Privacy and Data Mining. IEEE Access, vol. 2, 2014, pp. 1149–1176.
[54] Chardin B., Lacombe J.-M., Petit J.-M. Chronos A. NoSQL System on Flash Memory for Industrial Process Data. Distributed and Parallel Databases, vol. 34, no. 3, 2016, pp. 293-319.
[55] Sivarajah U., Kamal M.M., Irani Z., Weerakkody V. Critical Analysis of Big Data Challenges and Analytical Methods. Journal of Business Research, vol. 70, 2017, pp. 263-286.
[56] Ali S. M., Gupta N., Nayak G.K., Lenka R.K. Big Data Visualization: Tools and challenges. In Proc. of the 2nd International Conference on Contemporary Computing and Informatics, 2016, pp. 656-660.
[57] Yang A., Troup M., Ho J.W.K. Scalability and Validation of Big Data Bioinformatics Software. Computational and Structural Biotechnology Journal, vol. 15, 2017, pp. 379-386.
[58] Elgendy N., Elragal A. Big Data Analytics: A Literature Review Paper. Lecture Notes in Computer Science, vol. 8557, 2014, vol. 8557, pp. 214-227.
[59] Shim J.P., French A.M., Guo C., Jablonski J. Big Data and Analytics: Issues, Solutions, and ROI. Communications of the Association for Information Systems, vol. 37, 2015, pp. 797-810.
[60] Gandomi A., Haider M. Beyond the Hype: Big Data Concepts, Methods, and Analytics. International Journal of Information Management, vol. 35, no. 2, 2015, pp. 137-144.

[61] Russom P. Big Data Analytics. TDWI Best Practices Report, Fourth Quarter, 2011. Available at: https://tdwi.org/research/2011/09/best-practices-report-q4-big-data-analytics.aspx, accessed 18.11.2019.
[62] Jha A., Dave M., Madan S. A Review on the Study and Analysis of Big Data Using Data Mining Techniques. International Journal of Latest Trends in Engineering and Technology (IJLTET), vol. 6, no. 3, 2016, pp. 94-102.
[63] Berman J.J. Principles of Big Data: Preparing, Sharing, and Analyzing Complex Information. Morgan Kaufmann, 2013, p. 288.
[64] van der Lans R. F. Analytics of Textual Big Data: Text Exploration of the Big Untapped Data Source. Independent Business Intelligence Analyst R20: Consultancy2013. Available at: http://www.data.net.ma/wp-content/uploads/2015/12/Analytics-of-Textual-Big-Data-Text-Exploration-of-the-Big-Untapped-Data-Source.pdf, accessed 18.11.2019.
[65] Malaka I., Brown I. Challenges to the Organisational Adoption of Big Data Analytics: A Case Study in the South African Telecommunications Industry. In Proc, of the Annual Research Conference on South African Institute of Computer Scientists and Information Technologists, 2015, Article No. 27.
[66] Lavalle S., Hopklns M.S., Lesser E., Shockley R., Kruschwitz N. Analytics: The New Path to Value. Research Report, Fall 2010, MIT Sloan Management Review and the IBM Institute for Business Value. Available at: https://sloanreview.mit.edu/projects/analytics-the-new-path-to-value/, accessed 18.11.2019.
[67] Fahmideh M., Beydoun G. Big Data Analytics Architecture Design - An Application in Manufacturing Systems. Computers & Industrial Engineering, vol. 128, 2019, pp. 948-963.
[68] Lopes C., Cabral B., Bernardino J. Personalization Using Big Data Analytics Platforms. In Proc. of the Ninth International C* Conference on Computer Science & Software Engineering, 2016, pp. 131-132.
[69] White C., Research B. Using Big Data for Smarter Decision Making. BI Research, IBM Big Data & Analytics Hub, 2011. Available at: https://www.ibmbigdatahub.com/whitepaper/using-big-data-smarter-decision-making, accessed 18.11.2019.
[70] Samosir R.S., Hendric H.L., Gaol F.L., Abdurachman E., Soewito B. Measurement Metric Proposed for Big Data Analytics System. In Proc. of the International Conference on Computer Science and Artificial Intelligence, 2017, pp. 265–269.
[71] Chapter 2. Business Problems Suited to Big Data Analytics. In Loshin D. Big Data Analytics, Morgan Kaufmann, 2013, pp. 11-19.
[72] Romary L. Data Management in the Humanities. ERCIM News, no. 89, 2012, p. 14.
[73] Lianzhi L. Evaluation Model of Education Service Quality Satisfaction in Colleges and Universities Dependent on Classification Attribute Big Data Feature Selection Algorithm. In Proc. of the International Conference on Intelligent Transportation, Big Data & Smart City, 2019, pp. 645-649.
[74] Li Y., Zhai X. Review and Prospect of Modern Education using Big Data. Procedia Computer Science, vol. 129, 2018, pp. 341-347.
[75] Xiong Z., Zhi L., Jiang J. Research on Art Education Digital Platform Based on Big Data. In Proc. of the IEEE 4th International Conference on Big Data Analytics, 2019, pp. 208-211.
[76] Kim Y.H., Ahn J.-H. A Study on the Application of Big Data to the Korean College Education System, Procedia Computer Science, vol. 91, 2016, pp. 855-861.
[77] Santoso L.W., Yulia. Data Warehouse with Big Data Technology for Higher Education. Procedia Computer Science, vol. 124, 2017, pp. 93-99.
[78] Ramos T.G., Machado J.C.F., Cordeiro B.P.V. Primary Education Evaluation in Brazil Using Big Data and Cluster Analysis. Procedia Computer Science, vol. 55, 2015, pp. 1031-1039.
[79] Huang Y., Chen Z., Yu T., Huang X., Gu X. Agricultural Remote Sensing Big Data: Management and Applications. Journal of Integrative Agriculture, vol. 17, no. 9, 2018, pp. 1915-1931.
[80] Sabarina K., Priya N. Lowering Data Dimensionality in Big Data for the Benefit of Precision Agriculture. Procedia Computer Science, vol. 48, 2015, pp. 548-554.
[81] Klerkx L., Jakku E., Labarthe P. A Review of Social Science on Digital Agriculture, Smart Farming and Agriculture 4.0: New Contributions and A Future Research Agenda. NJAS – Wageningen Journal of Life Sciences, vol. 90-91, 2019, article 100315.
[82] Gonzalez-Sanchez A., Frausto-Solis J., Ojeda-Bustamante W. Predictive Ability of Machine Learning Methods for Massive Crop Yield Prediction. Spanish Journal of Agricultural Research, vol. 12, no. 2, 2014, pp. 313-328.

[83] Senthilvadivu S., Kiran S.V., Devi S.P., Manivannan S. Big Data Analysis on Geographical Segmentations and Resource Constrained Scheduling of Production of Agricultural Commodities for Better Yield. Procedia Computer Science, vol. 87, 2016, pp. 80-85.
[84] Palanisamy V., Thirunavukarasu R. Implications of Big Data Analytics in Developing Healthcare Frameworks – A Review. Journal of King Saud University – Computer and Information Sciences, vol. 31, no. 4, 2019, pp. 415-425.
[85] Patel J.A., Sharma P. Big Data for Better Health Planning, In Proc. of the International Conference on Advances in Engineering & Technology Research, 2014, pp. 1-5.
[86] Pashazadeh A., Navimipour N.J. Big Data Handling Mechanisms in the Healthcare Applications: A Comprehensive and Systematic Literature Review. Journal of Biomedical Informatics, vol. 82, 2018, pp. 47-62.
[87] Abouelmehdi K., Beni-Hssane A., Khaloufi H., Saadi M. Big Data Security and Privacy in Healthcare: A Review. Procedia Computer Science, vol. 113, 2017, pp. 73-80.
[88] Kaur P., Sharma M., Mittal M. Big Data and Machine Learning Based Secure Healthcare Framework. Procedia Computer Science, vol. 132, 2018, pp. 1049-1059.
[89] Khaloufi H., Abouelmehdi K., Beni-hssane A., Saadi M. Security Model for Big Healthcare Data Lifecycle. Procedia Computer Science, vol. 141, 2018, pp. 294-301.
[90] United Nations, Department of Economic and Social Affairs, Population Division. World Urbanization Prospects: The 2018 Revision. Available at: https://population.un.org/wup/Publications/Files/WUP2018-Report.pdf, accessed 18.11.2019.
[91] DeRen L., JianJun C., Yuan Y. Big Data in Smart Cities. Science China Information Sciences, vol. 58, no. 10, 2015, pp. 1-12.
[92] Rathore M.M., Paul A., Ahmad A., Chilamkurthi N., Hong W.-H., Seo H. Real-Time Secure Communication for Smart City in High-Speed Big Data Environment. Future Generation Computer Systems, vol. 83, 2018, pp. 638-652.
[93] Rathore M.M., Paul A., Hong W.-H., Seo H., Awan I., Saeed S. Exploiting IoT and Big Data Analytics: Defining Smart Digital City Using Real-Time Urban Data. Sustainable Cities and Society, vol. 40, 2018, pp. 600-610.
[94] Hashem I.A.T., Chang V., Anuar N.B., Adewole K., Yaqoob I., Gani A., Ahmed E., Chiroma H. The Role of Big Data in Smart City. International Journal of Information Management, vol. 36, no. 5, 2016, pp. 748–758.
[95] Lima C., Kimb K.-J., Maglio P.P. Smart Cities with Big Data: Reference Models, Challenges, and Considerations. Cities, vol. 82, 2018, pp. 86-99.
[96] Pal D., Triyason T., Padungweang P. Big Data in Smart-Cities: Current Research and Challenges. Indonesian Journal of Electrical Engineering and Informatics, vol. 6, no. 4, 2018, pp. 351-360.
[97] Allama Z., Dhunny Z.A. On Big Data, Artificial Intelligence and Smart Cities. Cities, vol. 89, 2019, pp. 80-91.
[98] Doku R., Rawat DB. Chapter 8. Big Data in Cybersecurity for Smart City Applications. In Smart Cities Cybersecurity and Privacy, Rawat D.B., Ghafoor K.Z., eds. Elsevier, 2019, pp. 103-112.
[99] Hayes M.A., Capretz M.A. Contextual Anomaly Detection Framework for Big Sensor Data. Journal of Big Data, vol. 2, 2015, article no. 2.
[100] Goswami K., Park Y., Song C. Impact of Reviewer Social Interaction on Online Consumer Review Fraud Detection. Journal of Big Data, vol. 4, 2017, article no. 15,
[101] Shalaginov A., Johnsen J.W., Franke K. Cyber Crime Investigations in the Era of Big Data. In Proc. of the IEEE International Conference on Big Data, 2017, pp. 3672-3676.
[102] Pramanik M.I., Zhang W., Lau R.Y.K., Li C. A Framework for Criminal Network Analysis Using Big Data. In Proc. of the IEEE 13th International Conference on e-Business Engineering, 2016, pp. 17-23.
[103] Hu J. Big Data Analysis of Criminal Investigations. In Proc. of the 5th International Conference on Systems and Informatics, 2018, pp. 649-654.
[104] Vaughan G. Efficient Big Data Model Selection with Applications to Fraud Detection. International Journal of Forecasting, June 2018, https://doi.org/10.1016/j.ijforecast.2018.03.002.
[105] Khan E.S., Azmi H., Ansari F., Dhalvelkar S. Simple Implementation of Criminal Investigation Using Call Data Records (CDRs) Through Big Data Technology. In Proc. of the International Conference on Smart City and Emerging Technology, 2018, pp. 1-5.
[106] Zhao Q., Chen K., Li T., Yang Y., Wang X. Detecting Telecommunication Fraud by Understanding the Contents of A Call. Cybersecurity, vol. 1, no. 8, 2018, p. 12.

[107] Chen Y.-J., Wu C.-H. On Big Data-Based Fraud Detection Method for Financial Statements of Business Groups. In Proc. of the 6th IIAI International Congress on Advanced Applied Informatics, 2017, pp. 986-987.
[108] Makki S., Assaghir Z., Taher Y., Haque R., Hacid M.-S., Zeineddine H. An Experimental Study with Imbalanced Classification Approaches for Credit Card Fraud Detection. IEEE Access, vol. 7, 2019, pp. 93010-93022.
[109] Zhou H., Sun G., Fu S., Jiang W., Xue J. A Scalable Approach for Fraud Detection in Online E-Commerce Transactions with Big Data Analytics. CMC: Computers, Materials & Continua, vol. 60, no. 1, 2019, pp. 179-192.
[110] Herland M., Khoshgoftaar T.M., Bauder R.A. Big Data Fraud Detection Using Multiple Medicare Data Sources. Journal of Big Data, vol. 5, 2018, article no. 29.
[111] Castaneda G., Morris P., Khoshgoftaar T.M. Maxout Neural Network for Big Data Medical Fraud Detection. In Proc. of the IEEE Fifth International Conference on Big Data Computing Service and Applications, 2019, pp. 357-362.
[112] Castaneda G., Morris P., Khoshgoftaar T. M. Evaluation of Maxout Activations in Deep Learning Across Several Big Data Domains. Journal of Big Data, vol. 6, 2019, article no. 72.
[113] Lnenicka M., Komarkova J. Developing A Government Enterprise Architecture Framework to Support the Requirements of Big and Open Linked Data with the Use of Cloud Computing. International Journal of Information Management, vol. 46, 2019, pp. 124-141.
[114] Yang P., Xia H., Liu W., Li Z. Research on Government Integrity Evaluation Based on Big Data. In Proc. of the 2nd International Conference on Artificial Intelligence and Big Data, 2019, pp. 28-35.
[115] LaBrie R.C., Steinke G.H., Li X., Cazier J.A. Big Data Analytics Sentiment: US-China Reaction to Data Collection by Business and Government. Technological Forecasting and Social Change, vol. 130, 2018, pp. 45-55.
[116] Laude H. Chapter 6. France's Governmental Big Data Analytics: From Predictive to Prescriptive Using R. In Federal Data Science: Transforming Government and Agricultural Policy Using Artificial Intelligence, Batarseh F.A., Yang R., eds. Academic Press, 2018, pp. 81-94.
[117] Yan Z. Big Data and Government Governance. In Proc. of the International Conference on Information Management and Processing, 2018, pp. 111-114.
[118] Aron J.L., Niemann B. Sharing Best Practices for the Implementation of Big Data Applications in Government and Science Communities. In Proc. of the IEEE International Conference on Big Data, 2014, pp. 8-10.
[119] Hardy K., Maurushat A. Opening up Government Data for Big Data Analysis and Public Benefit. Computer Law & Security Review, vol. 33, no. 1, pp. 30-37.
[120] Archenaa J., Anita E.A.M. A Survey of Big Data Analytics in Healthcare and Government. Procedia Computer Science, vol. 50, 2015, pp. 408-413.
[121] Lee Y., Park S. Design of A Government Collaboration Service Map by Big Data Analytics. Procedia Computer Science, vol. 91, 2016, pp. 751-760.
[122] Amado A., Cortez P., Rita P., Moro S. Research Trends on Big Data in Marketing: A Text Mining and Topic Modeling Based Literature Analysis. European Research on Management and Business Economics, vol. 24, no. 1, 2018, pp. 1-7.
[123] [Saidali J., Rahich H., Tabaa Y., Medouri A. The Combination Between Big Data and Marketing Strategies to Gain Valuable Business Insights for Better Production Success. Procedia Manufacturing, vol. 32, 2019, pp. 1017-1023.
[124] Akter S., Wamba S.F. Big Data Analytics in E-Commerce: A Systematic Review and Agenda for Future Research, Electronic Markets, vol. 26, no. 2, 2016, pp. 173-194.
[125] Chong A.Y.L., Li B., Ngai E.W.T., Ch'ng E., Lee F. Predicting Online Product Sales Via Online Reviews, Sentiments, and Promotion Strategies: A Big Data Architecture and Neural Network Approach. International Journal of Operations & Production Management, vol. 36, no. 4, 2016, pp. 358-383.
[126] Erevelles S., Fukawa N., Swayne L. Big Data Consumer Analytics and the Transformation of Marketing. Journal of Business Research, vol. 69, no. 2, 2016, pp. 897-904.
[127] Jabbar A., Akhtar P., Dani S. Real-time Big Data Processing for Instantaneous Marketing Decisions: A Problematization Approach. Industrial Marketing Management, Sept. 2019, https://doi.org/10.1016/j.indmarman.2019.09.001.
[128] Li T. Using Big Data Analytics to Build Prosperity Index of Transportation Market. In Proc. of the 4th ACM SIGSPATIAL International Workshop on Safety and Resilience, 2018, no. 17, p. 6.

[129] See-To E.W.K., Ngai E.W.T. Customer Reviews for Demand Distribution and Sales Nowcasting: A Big Data Approach. Annals of Operations Research, vol. 270, no. 1-2, 2018, pp. 415–431.
[130] Kumar A., Shankar R., Aljohani N.R. A Big Data Driven Framework for Demand-driven Forecasting with Effects of Marketing-mix Variables. Industrial Marketing Management, June 2019, https://doi.org/10.1016/j.indmarman.2019.05.003.
[131] Zheng K., Zhang Z., Song B. E-Commerce Logistics Distribution Mode in Big-Data Context: A Case Analysis of JD.COM. Industrial Marketing Management, Oct. 2019. DOI: https://doi.org/10.1016/j.indmarman.2019.10.009.
[132] Salehan M., Kim D.J. Predicting the Performance of Online Consumer Reviews: A Sentiment Mining Approach to Big Data Analytics. Decision Support Systems, vol. 81, 2016, pp. 30–40.
[133] Malhotra D., Rishi O.P. An Intelligent Approach to Design of E-Commerce Metasearch and Ranking System Using Next-Generation Big Data Analytics. Journal of King Saud University - Computer and Information Sciences, Mar. 2018, https://doi.org/10.1016/j.jksuci.2018.02.015.
[134] Wu P.-J., Lin K.-C. Unstructured Big Data Analytics for Retrieving E-Commerce Logistics Knowledge. Telematics and Informatics, vol. 35, no. 1, 2018, pp. 237-244.
[135] Zhaoa Y., Xu X., Wang M. Predicting Overall Customer Satisfaction: Big Data Evidence From Hotel Online Textual Reviews. International Journal of Hospitality Management, vol. 76, 2019, pp. 111-121.
[136] Liu X., Shin H., Burns A.C. Examining the Impact of Luxury Brand's Social Media Marketing on Customer Engagements: Using Big Data Analytics and Natural Language Processing. Journal of Business Research, May 2019, https://doi.org/10.1016/j.jbusres.2019.04.042.
[137] Kauffmann E., Peral J., Gil D., Ferrández A., Sellers R., Mora H. A Framework for Big Data Analytics in Commercial Social Networks: A Case Study on Sentiment Analysis and Fake Review Detection for Marketing Decision-making. Industrial Marketing Management, Aug. 2019, https://doi.org/10.1016/j.indmarman.2019.08.003.

Информация об авторах / Information about authors

Ноаман Мухаммед АЛИ в 2016 году получил степень магистра на факультете компьютерных наук Каирского университета. С 2016 года Ноаман является ассистентом на кафедре информационных технологий и систем университета Порт-Саида, Египет. Ноаман является также аспирантом кафедры информатики Санкт-Петербургского государственного университета. Его научные интересы включают анализ больших данных, распознавание образов, системы рекомендаций, обработку естественного языка.

No'aman Muhammad ALI received his M.Sc. degree from the department of computer science, Cairo University, in 2016. Currently, No'aman is an assistant lecturer at the Information Technology & Systems Department, Port Said University, Egypt, since 2016. No'aman is a Ph.D. student at the Department of Computer Science, Saint Petersburg State University. His research interests involve Big data analytics, pattern recognition, recommender systems, natural language processing.

Борис Асенович НОВИКОВ – доктор физико-математических наук, профессор, кафедра информатики в НИУ ВШЭ, Санкт-Петербург, Россия. Сфера научных интересов - широкая область управления данными и их анализа, включая системы и приложения для управления базами данных, структуры данных и методы доступа, обработку запросов, контроль параллелизма, обработку и анализ дискретных потоков, а также приложения машинного обучения.

Boris Asenovitch NOVIKOV, Dr. Sci. in mathematics and physics, professor, Department of Informatics at National Research University Higher School of Economics, Saint Petersburg, Russia. Research interests are in a wide area of data management and analytics, including database management systems and applications, data structures and access methods, query processing, concurrency control, discrete stream processing and analytics, and applications of machine learning.

DOI: 10.15514/ISPRAS-2020-32(1)-11

Кэширование машинного кода в динамическом компиляторе SQL-запросов для СУБД PostgreSQL

М.В. Пантилимонов, ORCID: 0000-0003-2277-7155 <pantlimon@ispras.ru>
Р.А. Бучацкий, ORCID: 0000-0001-8522-1811 <ruben@ispras.ru>
Р.А. Жуйков, ORCID: 0000-0002-0906-8146 <zhroma@ispras.ru>

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

Аннотация. По мере увеличения производительности компьютеров и роста объёма оперативной и внешней памяти производительность СУБД для некоторых классов запросов всё чаще определяется характеристиками процессора и эффективностью его использования. Для исполнения SQL-запросов в реляционных СУБД используется различные модели выполнения, которые различаются характеристиками, но так или иначе подвержены существенным накладным расходам при интерпретации плана запроса. Накладные расходы связаны с большим количеством ветвлений, неявными вызовами функций-обработчиков и выполнением лишних проверок. Одно из решений – динамическая компиляция запросов, которая оправдана только в том случае, когда время, затрачиваемое на интерпретацию запроса, превосходит время, затрачиваемое на компиляцию и выполнение оптимизированного кода. Данное требование может быть удовлетворено только тогда, когда объем обрабатываемых запросом данных достаточно велик. Если время интерпретации запроса исчисляется миллисекундами, то затраты на динамическую компиляцию могут в сотни раз превосходить время выполнения сгенерированного машинного кода. Чтобы оправдать расходы, затрачиваемые на динамическую компиляцию таких запросов, необходимо иметь возможность повторного использования сгенерированного машинного кода в последующих выполнениях, тем самым избавившись от затратных операций по его оптимизации и компиляции. В рамках данной работы рассматривается метод кэширования машинного кода в динамическом компиляторе запросов СУБД PostgreSQL. Предлагаемый метод позволяет избавиться от накладных расходов, затрачиваемых на оптимизацию и компиляцию. Результаты проведенного тестирования показывают, что динамическая компиляция запросов с возможностью переиспользования машинного кода позволяет получить существенное ускорение на запросах типа OLTP.

Ключевые слова: динамическая компиляция; JIT-компиляция; кэширование кода; выполнение запросов; СУБД; PostgreSQL; LLVM

Для цитирования: 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 г., стр. 205-220. DOI: 10.15514/ISPRAS–2020–32(1)–11

Благодарности: Авторы выражают благодарность Е.Ю. Шарыгину, Д.М. Мельнику и А.Н. Томилину за помощь в выполнении научной работы.

Machine code caching in PostgreSQL query JIT-compiler

M.V. Pantilimonov, ORCID: 0000-0003-2277-7155 <pantlimon@ispras.ru>
R.A. Buchatskiy, ORCID: 0000-0001-8522-1811 <ruben@ispras.ru>
R.A. Zhuykov, ORCID: 0000-0002-0906-8146 <zhroma@ispras.ru>

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

Abstract. As the efficiency of main and external memory grows, alongside with decreasing hardware costs, the performance of database management systems (DBMS) on certain kinds of queries is more determined by CPU characteristics and the way it is utilized. Relational DBMS utilize diverse execution models to run SQL queries. Those models have different properties, but in either way suffer from substantial overhead during query plan interpretation. The overhead comes from indirect calls to handler functions, runtime checks and large number of branch instructions. One way to solve this problem is dynamic query compilation that is reasonable only in those cases when query interpretation time is larger than the time of compilation and optimized machine code execution. This requirement can be satisfied only when the amount of data to be processed is large enough. If query interpretation takes milliseconds to finish, then the cost of dynamic compilation can be hundreds of times more than the execution time of generated machine code. To pay off the cost of dynamic compilation, the generated machine code has to be reused in subsequent executions, thus saving the cost of code compilation and optimization. In this paper, we examine the method of machine code caching in our query JIT-compiler for DBMS PostgreSQL. The proposed method allows us to eliminate compilation overhead. The results show that dynamic compilation of queries with machine code caching feature gives a significant speedup on OLTP queries.

Keywords: dynamic compilation; JIT-compilation; machine code caching; query execution; DBMS; PostgreSQL; LLVM

For citation: 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. 205-220 (in Russian). DOI: 10.15514/ISPRAS-2020-32(1)-11

Acknowledgments: The authors are grateful to E.Y. Sharygin, D.M. Melnik and A.N, Tomilin for their help with the research.

1. Введение

Традиционно в реляционных системах управления базами данных (РСУБД) пользовательский запрос транслируется сначала в логический план запроса, представляющий собой дерево из операторов расширенной реляционной алгебры, а затем в физический, путём добавления метаинформации о выбранных методах доступа к данным и алгоритмах, реализующих реляционные операции. Готовый физический план передается на исполнение, где осуществляется его интерпретация с использованием заданной модели выполнения. Классическим примером последней является модель итераторов, также известная как Volcano-модель [1]. В рамках данной модели каждый алгебраический оператор преобразовывает входные данные в выходной поток кортежей, который управляется при помощи функции *next()*, являющейся частью общего интерфейса. Данная абстракция проста для понимания и удобна в реализации, однако неэффективным образом использует ресурсы современных центральных процессоров.

Существуют и другие модели выполнения [2, 3, 4], которые пытаются нивелировать недостатки модели итераторов различным образом. Так или иначе, вне зависимости от используемой модели выполнения, классический способ выполнения запроса сопряжен с накладными расходами по вызову виртуальных функций в ходе интерпретации его плана, состоящего из произвольной последовательности операторов, выражений и предикатов, что в свою очередь порождает значительное количество ложных

предсказаний переходов. Также в ходе интерпретации могут выполняться проверки, которые избыточны для конкретного плана запроса.

Всё чаще для решения этой проблемы привлекается метод динамической компиляции, в рамках которого выполняется кодогенерация специализированного кода под заданный план запроса. Производительность достигается за счет встраивания функций, подстановки констант, вычисления арифметических выражений, замены косвенных вызовов на явные, удаления мёртвого, с точки зрения плана запроса, кода и другие. Метод динамической компиляции предполагает замену в общем времени обработки запроса времени интерпретации $t_I(N)$ на суммарное время компиляции и выполнения скомпилированного кода $t_C + t_E(N)$, где N – размер данных, обрабатываемых запросом, t_C – время компиляции и t_E – время выполнения. Проводимые во время компиляции оптимизации делают скомпилированный код более эффективным: $t_E(N) < t_I(N)$, но чтобы динамическая компиляция имела смысл, необходимо, чтобы $t_C + t_E(N) < t_I(N)$, то есть чтобы время, затрачиваемое на интерпретацию запроса, превосходило время, затрачиваемое на компиляцию и выполнение оптимизированного кода.

Данное требование может быть удовлетворено только в том случае, когда объем обрабатываемых запросом данных достаточно велик. Таким образом, только аналитические запросы типа OLAP [5], например, из тестового набора TPC-H [6], выполняющиеся на большом объеме данных, могут нивелировать время, затрачиваемое на компиляцию, и позволяют получить прирост производительности. В случае OLTP [7] запросов, например, из набора TPC-B [8], где в основном обрабатывается небольшой объем данных, а время интерпретации может исчисляться микросекундами, метод динамической компиляции может оказаться неприемлемым по причине долгой оптимизации и компиляции динамически сгенерированного кода.

Проблема может быть решена путем сохранения и переиспользования сгенерированного машинного кода. Однако простого сохранения в общем случае недостаточно и необходимо выполнять кодогенерацию с возможностью применения патчей к сохраненному машинному коду по причине изменяемых значений и адресов структур в динамической памяти.

В данной работе рассматривается метод сохранения и переиспользования сгенерированного динамическим компилятором запросов машинного кода с целью уменьшения накладных расходов, затрачиваемых на компиляцию запросов. Работа выполняется с использованием компиляторной инфраструктуры LLVM [9] в динамическом компиляторе запросов [10, 11, 12] PostgreSQL [13], разрабатываемом в ИСП РАН [14].

2. Стандартный подход к автоматическому кэшированию запросов

В большинстве современных проприетарных РСУБД, таких как MS SQL (Microsoft), DB2 (IBM) и Oracle Database (Oracle), активным образом используется механизм кэширования планов исполняемых запросов в автоматическом режиме без использования явных синтаксических конструкций. Сохраненные планы запросов располагаются в общей памяти, которая может быть реализована по-разному в зависимости от используемой процессной модели. Таким образом, информация о планах запросов доступна всем обслуживающим клиентские подключения процессам/потокам и позволяет в общем случае уменьшить время отклика всей системы, увеличив ее производительность и пропускную способность за счет уменьшения накладных расходов на операции, связанные с построением физического плана для часто используемых запросов.

Однако у подхода автоматического кэширования в общей памяти имеются и свои минусы, в основном связанные со сложностью синхронизации доступа к разделяемым

ресурсам, а также бо́льшей чувствительностью системы к не оптимально построенным планам запросов, чем в случае локального кэширования, используемого в PostgreSQL и MySQL. Сохраненный план запроса может быть первоначально построен оптимизатором не оптимально из-за недостаточного объёма статистической метаинформации с асимметричным распределением данных, сложной конструкции самого запроса с большим количеством операций соединения, использования хранимых функций или процедур и т.д. Также может возникнуть ситуация, что оптимально построенный план запроса теряет свою актуальность после некоторого количества выполненных операций модификации базы данных, которые могли в различной степени изменить существующее распределение данных, от которого отталкивался оптимизатор при построении первоначального плана. До тех пор, пока данный план запроса не будет перестроен, все клиенты СУБД будут не оптимально расходовать ресурсы системы.

Для решения этой проблемы вендоры используют в своих СУБД различные подходы: автоматическое перестраивание плана запроса после некоторого процентного изменения данных в объекте (таблице) базы данных; сбор статистики во время выполнения, которая позволяет понять актуальность приблизительных оценок оптимизатора в момент первоначального построения; использование адаптивных планов запросов, которые могут заменять оператор-алгоритм в зависимости от количества получаемых данных, например, переходить от соединения по вложенным циклам на соединение по хэшу и т.д.

В общем случае задача динамической компиляции плана запроса слабо пересекается с проблемой его оптимального построения и может решаться независимо. По причине того, что операции по динамической генерации кода с его оптимизацией и компиляцией являются ресурсоемкими, амортизация стоимости затрачиваемых на них ресурсов становится важной задачей. В случае локального кэширования эта задача не может быть решена эффективно по причине того, что каждый процесс СУБД имеет доступ только к собственным сохраненным планам и не может переиспользовать результат работы другого процесса. В худшем случае каждый процесс может иметь абсолютную копию из некоторого набора часто выполняющихся запросов, каждый со своим динамически скомпилированным машинным кодом. При использовании такого подхода суммарные затраты на поддержание кэша планов будут линейно расти как по памяти, используемой для хранения собственной копии машинного кода плана, так и по тактам процессора, затрачиваемых на генерацию и компиляцию этой копии.

Таким образом, для реализации эффективного механизма кэширования динамически скомпилированных планов запросов в СУБД PostgreSQL его, как минимум, необходимо перенести из процесс-локальной памяти в разделяемую. Тем не менее, для преследуемых в данной работе целей указанное требование не является обязательным, и метод кэширования динамически сгенерированного машинного кода может быть исследован независимо.

3. Этапы обработки SQL запроса

Основной алгоритм выполнения SQL-запроса в реляционных СУБД состоит из следующих этапов:

1. Стадия лексического и синтаксического анализа. На этом этапе входная строка-запрос пользователя обрабатывается лексическим и синтаксическим анализаторами, и в результате получается дерево разбора. В процессе анализа выполняется только проверка синтаксиса, но не проверяется семантика. Например, если в запросе осуществляется обращение к таблице, которая не существует в базе данных, то ошибка выдана не будет.
2. Стадия семантического анализа. Дерево разбора, полученное на предыдущей фазе, проходит через семантический анализ, в результате которого получается дерево запроса, дополненное различного рода метаинформацией: системными

идентификаторами таблиц, типами и порядковыми номерами запрашиваемых полей, перечнем соединяемых таблиц и условий фильтраций в виде дерева и т. д.

3. Стадия обработки системой правил. Далее выполняется поиск в системных каталогах правил, применимых к дереву запроса, и при обнаружении подходящих правил выполняются преобразования, описанные в теле найденного правила. Примером преобразования является замена обращений к представлениям – так называемым виртуальным таблицам – на обращения к базовым таблицам из определения представления.
4. Фаза планирования и оптимизации. Планировщик получает на вход структуру с деревом запроса. Используя вспомогательные структуры данных, называемыми путями, которые представляют собой упрощенные схемы планов, планировщик осуществляет выбор наиболее эффективного пути выполнения запроса с точки зрения имеющихся оценок затрат и статистической информации на момент выполнения. Производится выбор оптимального метода доступа к данным с заданным порядком соединений и алгоритмов для их выполнения. Выбранный вариант трансформируется в полноценный план запроса и передается исполнителю.
5. Фаза выполнения итогового плана запроса. Исполнитель осуществляет рекурсивный обход по дереву плана и выполняет инкапсулированную логику соответствующего узла-оператора или выражения, получая на выходе результирующее множество строк.

В большинстве СУБД, включая PostgreSQL, используется описанный подход для трансляции запроса пользователя в физический план запроса, пригодный для выполнения. В случае динамической компиляции к последней фазе добавляются накладные расходы, связанные с процессом кодогенерации, оптимизации и компиляции. Чтобы эти расходы были оправданы, итоговое время выполнения запроса в режиме интерпретации должно существенно превосходить время, затрачиваемое на динамическую компиляцию.

4. Анализ необходимости сохранения машинного кода

На рис. 1 представлен план, построенный оптимизатором СУБД PostgreSQL, для запроса Q1 из тестового набора TPC-H[1] в базе данных, сгенерированной с параметром SCALE=2. Генерация базы данных выполнялась с заменой типов CHAR(1) на ENUM и NUMERIC на DOUBLE PRECISION.

```
                                QUERY PLAN
-----------------------------------------------------------------------
   Sort
     Sort Key: l_returnflag, l_linestatus
     ->  HashAggregate
         Group Key: l_returnflag, l_linestatus
         ->  Seq Scan on lineitem
             Filter: (l_shipdate <=
                 '1998-09-29 00:00:00'::timestamp without time zone)
```

Рис. 1. План в PostgreSQL для запроса Q1 из набора TPC-H
Fig. 1. Query plan in PostgreSQL for query Q1 from TPC-H benchmark

В запросе Q1 оператор последовательного сканирования проверяет условие предиката для каждого кортежа таблицы lineitem, состоящей примерно из 12 млн. кортежей. Среднее время интерпретации данного запроса на машине с процессором Intel Core I7-6700HQ, когда база данных полностью располагается в основной памяти, составляет 7.7 секунд. Среднее суммарное время выполнения динамически скомпилированной версии

[1] См. Q1 http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.2.pdf

этого же запроса составляет 2.1 секунды, где 350 мс занимает оптимизация сгенерированного во внутреннем представлении LLVM IR кода, 280 мс – его компиляция и 1.4 секунды – выполнение результирующего машинного кода. Таким образом, время выполнения запроса в режиме интерпретации существенно превосходит накладные расходы, затрачиваемые на динамическую компиляцию и выполнение сгенерированного машинного кода. Качество машинного кода в совокупности с размером обрабатываемых данных положительно сказываются на итоговом времени выполнения, что оправдывает ресурсы, затраченные на его генерацию.

Противоположную ситуацию можно наблюдать на примере следующего простого запроса:

```
select * from orders where o_custkey = 102022
     and o_orderdate between date '1992-11-01'
     and date '1994-01-01',
```

план которого представлен на рис. 2. Данный запрос извлекает данные о заказах клиента из соответствующей таблицы с использованием индекса и суммарно обрабатывает лишь небольшое количество кортежей. Среднее время его интерпретации на той же машине составляет примерно 0.110 мс. В случае динамической компиляции суммарное время составляет примерно 120 мс, а время выполнения сгенерированного машинного кода лишь 0.030 мс. Очевидно, что затраты на динамическую компиляцию данного запроса в сотни раз превосходят время его выполнения.

```
                                QUERY PLAN
-------------------------------------------------------------------------
   Index Scan using i_o_custkey on orders
     Index Cond: (o_custkey = 102022)
     Filter: ((o_orderdate >= '1992-11-01'::date)
                           AND (o_orderdate <= '1994-01-01'::date))
```

Рис. 2. План в PostgreSQL для OLTP запроса
Fig. 2. Query plan in PostgreSQL for OLTP query

Можно сделать вывод, что для запросов такого вида стоимость динамической компиляции чрезвычайно высока, и чтобы оправдать ее использование, необходимо использовать сгенерированный машинный код повторно в последующих выполнениях, избавившись от затратных операций по его оптимизации и компиляции. В общем случае сохранение машинного кода должно осуществляться для запросов, которые СУБД самостоятельно кэширует с целью минимизации затрат, ассоциируемых с этапами его обработки до момента выполнения. В запросе, представленном на рис. 2, для вычисления результата используются литеральные значения-константы, что существенно уменьшает вероятность последующего повторного использования данного плана запроса с идентичными аргументами.

СУБД PostgreSQL не обладает функционалом по автоматическому кэшированию запросов, но предоставляет механизм ручного сохранения плана запроса в локальную память процесса, обслуживающего клиентское подключение. Таким образом, данный механизм может быть задействован для реализации возможности переиспользования сгенерированного машинного кода, ассоциируемого с конкретным планом запроса.

5. Генерация машинного кода с возможностью переиспользования

5.1 Кэширование плана запроса в PostgreSQL

Для кэширования плана запроса, поступающего в систему из внешнего источника, СУБД PostgreSQL предоставляет механизм его ручного сохранения с помощью команды PREPARE [15]. Эта команда позволяет создать подготовленный объект-оператор для

пользовательского запроса на стороне сервера, который проходит через первые три стадии стандартного процесса обработки, описанные в разд. 3: разбор, анализ и переписывание с использованием правил, а затем сохраняется в локальной памяти процесса – текущего сеанса работы с СУБД. Последующая работа с подготовленным объектом-оператором осуществляется путем использования команды EXECUTE [16], вместе с которой также передаются значения-параметры, если они были указаны в момент подготовки объекта. Для сохраненного объекта-оператора оптимизатор генерирует наилучший план в зависимости от переданных параметров, а затем передает его на выполнение. Таким образом, повторное использование подготовленного объекта-оператора позволяет нивелировать накладные расходы, затрачиваемые на первые три стадии обработки запроса: лексический, синтаксический и семантический анализ, а также обработку системой правил.

Подготовленный оператор может также использовать некоторый обобщенный (GENERIC) план, а не перестраивать его под каждый набор полученных параметров. Для подготовленных операторов без параметров это происходит сразу; иначе общий план выбирается после пяти и более выполнений, при которых получаются планы с ожидаемой средней стоимостью, превышающей оценку стоимости общего плана. Когда общий план выбран, используется до конца жизни подготовленного оператора. Обобщенный план запроса нивелирует все накладные расходы, ассоциируемые со всеми стадиями обработки запроса, кроме его выполнения путем интерпретации.

На рис. 3 представлен пример обобщенного плана для подготовленного командой PREPARE объекта-оператора:

```
PREPARE q1(int, date, date) as
        select * from orders where o_custkey = $1
                          and o_orderdate between $2 and $3;
```

```
                              QUERY PLAN
-----------------------------------------------------------------------
 Index Scan using i_o_custkey on orders
   Index Cond: (o_custkey = $1)
   Filter: ((o_orderdate >= $2) AND (o_orderdate <= $3))
```

Рис. 3. Обобщенный план запроса в PostgreSQL
Fig. 3. Generic query plan in PostgreSQL

В данном случае оптимизатор PostgreSQL, используя накопленную статистику за первых 5 выполнений, посчитал, что стоимость обобщенного плана для данного запроса меньше, чем затраты на его планирование под каждый набор получаемых параметров. Значения $1, $2 и $3 соответствуют порядковым номерам параметров, указанных в команде PREPARE, и влияют на результат выполнения соответствующих фильтров-предикатов.

Таким образом, стандартный механизм PostgreSQL по созданию подготовленного объекта-оператора позволяет сохранить оптимизированный обобщенный план запроса для его последующего повторного переиспользования без расхода вычислительных ресурсов на стандартные фазы обработки. Данный механизм был использован в качестве базы для реализации возможности сохранения и переиспользования динамически скомпилированного машинного кода. Для этого потребовалось расширить существующие структуры PostgreSQL типа QueryDesc и CachedPlan таким образом, чтобы динамический компилятор запросов смог сохранить в них указатель на область

памяти, содержащей сгенерированный машинный код. Сохраненная в заданных структурах информация используется в динамическом компиляторе запросов для последующего переиспользования машинного кода с его предварительной модификацией, которая необходима для обновления адресов переменных PostgreSQL, используемых в процессе кодогенерации. Абсолютные адреса используемых структур и переменных PostgreSQL меняются после каждого выполнения сохраненного плана запроса и, соответственно, должны быть обновлены перед следующим запуском.

5.2 Инструменты LLVM для модификации машинного кода

Для реализации возможности переиспользования сгенерированного машинного кода некоторого обобщенного плана запроса необходимо располагать метаинформацией для установки соответствия между LLVM IR представлением, используемым в момент кодогенерации, и результирующим машинным кодом. Впоследствии данная информация может быть использована для модификации и патчинга динамически сгенерированных инструкций.

Для решения этой задачи инфраструктура LLVM предоставляет инструменты для контроля и модификации сгенерированного компонентом MCJIT [17] машинного кода – интринсики $llvm.experimental.stackmap$ и $llvm.experimental.patchpoint$. Оба этих интринсика во время компиляции представления LLVM в машинный код инициируют создание специальной секции данных, содержащей структуру $Stack\ Map$ [18]. В этой структуре сохраняется относительное смещение от начала функции в машинном коде, куда попадает вызов $stackmap/patchpoint$, а также местонахождение (слот стека, имя регистра, константа и т.п.) всех значений, переданных этим интринсикам в качестве параметров. Интринсик $llvm.experimental.patchpoint$, помимо тех же параметров, что и $llvm.experimental.stackmap$, принимает также адрес вызываемой функции. Помимо создания $Stack\ Map$, при компиляции $llvm.experimental.patchpoint$ в генерируемый код вставляется вызов этой функции в соответствии с заданным соглашением о вызовах. В дальнейшем, благодаря $Stack\ Map$, вызываемый объект можно будет подменить.

Для реализации метода кэширования сгенерированного машинного кода в динамическом компиляторе запросов используется только интринсик $llvm.experimental.patchpoint$, сигнатура которого представлена на рис. 4.

```
declare i64 @llvm.experimental.patchpoint.i64(i64<id>,
                i32<numBytes>, i8*<target>, i32<numArgs>, ...)
```

Рис. 4. Сигнатура интринсика llvm.experimental.patchpoint
Fig. 4. llvm.experimental.patchpoint intrinsic syntax

В процессе кодогенерации создаваемые инструкции в промежуточном представлении LLVM IR используют данные из структур PostgreSQL, располагающиеся в динамической памяти по некоторым абсолютным адресам, которые, в свою очередь, теряют актуальность после каждой итерации выполнения запроса. Таким образом, перед началом следующего выполнения сохраненного машинного кода необходимо актуализировать ранее используемые абсолютные адреса. В процессе динамической компиляции с целью переиспользования машинного кода для каждого обращения к полю структуры данных PostgreSQL осуществляется генерация вызова интринсика $llvm.experimental.patchpoint(ID, 13, 0x1234567890abcdef, 0)$, где ID – это уникальный идентификатор, 13 – количество резервируемых байт, $0x1234567890abcdef$ – адрес функции, которую будет вызывать сгенерированный код и 0 – количество параметров у вызываемой функции.

На рис. 5 представлен пример кодогенерации с использованием LLVM C API и его результирующее представление в LLVM IR и машинном коде.

В данном примере поле tts_nvalid принадлежит структуре $TupleTableSlot$, которая используется PostgreSQL для представления разных типов кортежей. Данная структура данных аллоцируется и освобождается при каждом выполнении плана запроса, вне зависимости от использования механизма ручного кэширования.

Вызов функций из LLVM C API внутри функции-генератора	`static LLVMValueRef`
	`top_level_consume_codegen(LLVMModuleRef mod,`
	`  LLVMBuilderRef builder, …){`
	`  ...`
	`  LLVMValueRef slot_nvalid_ptr;`
	`  __LLVMPositionBuilderAtEnd(builder, entry_bb);`
	`  slot_nvalid_ptr = GeneratePatchpoint(builder,`
	`  __LLVMPointerType(__LLVMInt32TypeInContext`
	`  (llvm_ctx), 0),`
	`  &**inputslot->tupleslot->tts_nvalid**);`
	`  __LLVMBuildStore(builder,`
	`  __LLVMConstNull(__LLVMInt32TypeInContext`
	`  (llvm_ctx)), slot_nvalid_ptr);`
	`  …`
	`}`

Сгенерированный LLVM IR	`define internal i32 @llvm_top_level_consume() {`
	`entry:`
	`  %pp_ret = call i64 (i64, i32, i8*, i32, ...)`
	`@llvm.experimental.patchpoint.i64(`
	`  i64 0, i32 13, i8* inttoptr`
	`  (i64 1311768467294899695 to i8*), i32 0)`
	`  %pp_ret_pointer = inttoptr i64 %pp_ret to i32*`
	`  store i32 0, i32* %pp_ret_pointer`
	`  …`
	`  ret i32 0`
	`}`

Сгенерированный машинный код	`<main+1966>:  49 bb ef cd ab 90 78 56 34 12`
	`  movabs $0x1234567890abcdef,%r11`
	`<main+1976>:  41 ff d3`
	`callq  *%r11`
	`<main+1979>:  c7 00 00 00 00 00`
	`movl  $0x0,(%rax)`

Пропатченный машинный код	`<main+1966>:  49 bb ef cd ab 90 78 56 34 12`
	`  movabs $**0x55b5b3195148**,%rax`
	`<main+1976>:  66 66 90`
	`data16 xchg %ax,%ax`
	`<main+1979>:  c7 00 00 00 00 00`
	`movl  $0x0,(%rax)`

Рис. 5. Пример кодогенерации с использованием интринсика $llvm.experimental.patchpoint$
Fig. 5. Example of code generation using $llvm.experimental.patchpoint$ intrinsic.

Адрес поля tts_nvalid в данной итерации кодогенерации равен $0x55b5b3195148$ и запоминается внутри функции $GeneratePatchpoint$ в глобальный массив $llvm\_pp[]$ с индексом $llvm\_pp\_n$, который затем используется в качестве ID аргумента интринсика $llvm.experimental.patchpoint$. Впоследствии переданный индекс в функцию-интринсик будет сохранен в структуру $StkMapRecord$ внутри $Stack\ Map$ как $PatchPoint\ ID$. При последующем разборе структуры $Stack\ Map$ извлекаемое значение-

индекс из поля $PatchPoint\ ID$ структуры $StkMapRecord$ позволит извлечь соответствующий адрес из массива $llvm\_pp$ и использовать его для модификации кода. Результат выполнения патчинга представлен на рис. 5, где target-адрес вызываемой функции заменяется на адрес поля tts_nvalid, регистр $\%r11$ на $\%rax$, а вызов $callq$ на nop ($xchg\ \%ax, \%ax$). Пропатченный код сохраняет семантику с точки зрения дальнейших инструкций, где работа осуществляется со значением регистра $\%rax$.

5.3 Реализация механизма модификации машинного кода в динамическом компиляторе запросов

В конструкции рассматриваемого в статьях [10, 11] динамического компилятора запросов с измененной моделью выполнения генерация кода выполняется во время обхода дерева плана в прямом порядке, во время которого для каждого оператора вызываются функции-генераторы. Для каждого оператора соответствующие функции-генераторы реализованы с использованием LLVM C API и вызываются для генерации реализующего его алгебраическую модель кода на LLVM IR.

При реализации метода кэширования машинного кода важно было избежать дублирования логики и сохранить существующий алгоритм кодогенерации в функциях-генераторах с целью упрощения дальнейшего процесса разработки и поддержки. Для реализации возможности генерации машинного кода с возможностью переиспользования были выполнены следующие изменения в существующем алгоритме кодогенерации.

- Добавлено глобальное состояние-режим кодогенерации – переменная $llvm\_patchpoint$, которая задает поведение внутри функций-оберток над LLVM C API, функции $GeneratePatchpoint$ и некоторых других.
- Все использующиеся для кодогенерации функции из LLVM C API обернуты в функцию-обертку с аналогичным названием и дополнительным префиксом, которая возвращает разный результат в зависимости от глобального режима кодогенерации. Пример функции-обертки представлен на рис. 6.
- Все вызовы функции $LLVMConstIntToPtr$ из LLVM C API заменены на специальную функцию $GeneratePatchpoint$, псевдокод которой представлен на рис. 7.

```
static LLVMValueRef inline
__LLVMBuildStore(LLVMBuilderRef B, LLVMValueRef Val,
                 LLVMValueRef Ptr)
{
    Assert(llvm_patchpoint >= 0 && llvm_patchpoint < 3);
    if (llvm_patchpoint == 1)
    {
        Assert(B == NULL && Val == NULL && Ptr == NULL);
        return NULL;
    }
    else
        return LLVMBuildStore(B, Val, Ptr);
}
```

Рис. 6. Пример функции-обертки над LLVM C API
Fig. 6. Example of LLVM C API wrapper function.

Перечислим возможные режимы кодогенерации.

- Одноразовая кодогенерация: машинный код будет использован один раз. Значение $llvm\_patchpoint$ равно 0.
- Режим патчинга: кодогенерация не выполняется, а производится сбор и сохранение новых абсолютных адресов структур данных PostgreSQL для модификации сохраненного машинного кода. Значение $llvm\_patchpoint$ равно 1.

- Режим кодогенерации с возможностью патчинга генерируемого машинного кода: в ходе кодогенерации сохраняются абсолютные адреса структур данных PostgreSQL как в режиме патчинга, а также генерируются вызовы интринсика $llvm.experimental.patchpoint$. После завершения процесса кодогенерации выполняется модификация сохраненного машинного кода. Значение $llvm\_patchpoint$ равно 2.

В режиме одноразовой кодогенерации функции обёртки над функциями LLVM C API возвращают результат вызова соответствующей LLVM функции, а $GeneratePatchpoint()$ возвращает результат вызова функции $LLVMConstIntToPtr$ без запоминания переданного адреса-аргумента.

В режиме патчинга выполняется обход плана запроса с использованием функций-генераторов, в ходе которого все вызовы функций-оберток над LLVM C API возвращают пустое значение, т. е. отсутствуют какая-либо кодогенерация. Единственная выполняемая работа – это сохранение адресов аргументов внутри функции $GeneratePatchpoint$ в глобальный массив $llvm\_pp[]$. В конце выполняется патчинг машинного кода с использованием информации из структуры Stack Map.

В режиме кодогенерации с возможностью патчинга также выполняется обход плана запроса, где функция $GeneratePatchpoint$ запоминает адрес-аргумент в глобальный массив $llvm\_pp[]$, а затем возвращает результат кодогенерации вызова интринсика $llvm.experimental.patchpoint$ с предварительным приведением к указателю нужного типа. После завершения процесса кодогенерации осуществляется патчинг адресов аналогично предыдущему режиму, а также одноразовое изменение инструкций, представленное ранее на рис. 5.

```
LLVMValueRef
GeneratePatchpoint(LLVMBuilderRef builder,
                   LLVMTypeRef type, void *address)
{
    if (llvm_patchpoint == 0)
        return LLVMConstIntToPtr(builder, address, type);
    // save address to the global array and use same index for
intrinsic in case of patchpoint gen.
    llvm_pp[llvm_pp_n] = (uintptr_t) address;
    if (llvm_patchpoint == 2)
    {
        args = { llvm_pp_n++, 13, 0x1234567890abcdef, 0 };
        ret = LLVMBuildCall(builder,
"llvm.experimental.patchpoint.i64", args);
        return LLVMBuildIntToPtr(builder, ret, type);
    }
    Assert(llvm_patchpoint == 1);
    llvm_pp_n++;
    return NULL;
}
```

Рис. 7. Псевдокод функции $GeneratePatchpoint()$
Fig. 7. $GeneratePatchpoint()$ pseudocode

Процесс патчинга, применяемый в описанных режимах, визуально представлен на рис. 8 и выполняется следующим образом:

1. Из структуры $Stack\ Map$ последовательно извлекается информация о количестве записей в массиве $StkSizeRecord$, каждая запись которого содержит адрес функции, размер стека и количество записей $StkMapRecord$.

2. Из каждой записи $StkMapRecord$ извлекается значение-индекс – $PatchPoint\ ID$, которое используется для поиска ячейки в массиве адресов, ранее собранных в процессе обхода плана запроса. Данный этап соответствует шагу 1 на рис. 8.
3. Помимо значения-индекса из $StkMapRecord$ также извлекается значение-отступ – $Instruction\ Offset$. Адрес функции и отступ позволяют получить указатель на область памяти, зарезервированную при генерации $llvm.experimental.patchpoint$. Данный этап соответствует шагу 2 на рис. 7.
4. На последнем этапе, который соответствует шагу 3 на рис. 7, выполняется патчинг с использованием нового адреса из массива $llvm\_pp[]$.

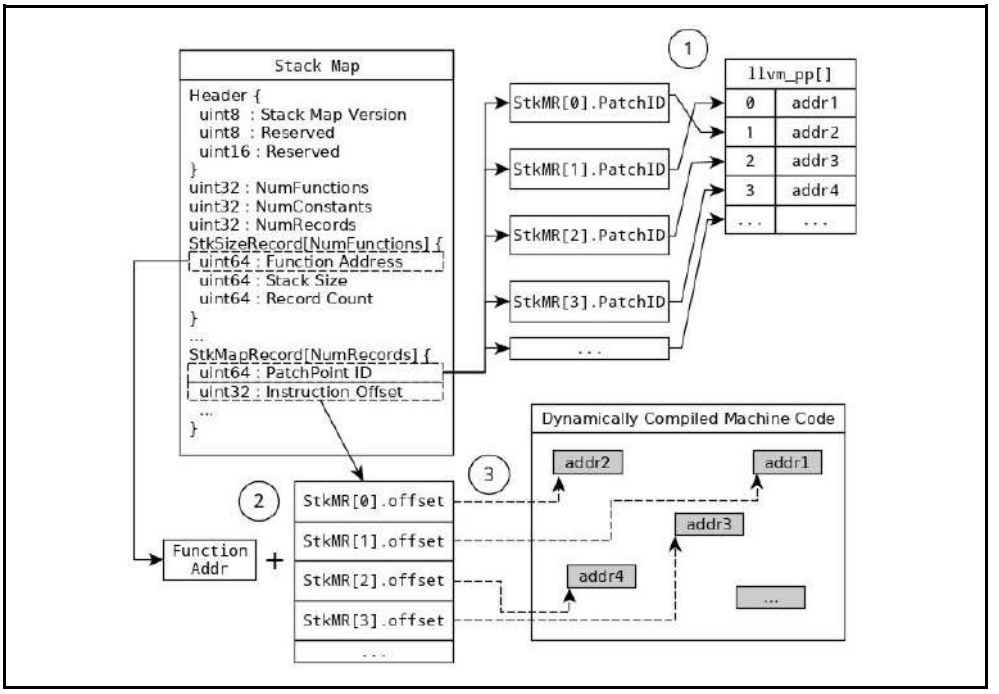


Рис. 8. Схема выполнения патчинга машинного кода
Fig. 8. Scheme of machine code patching

6. Результаты

Тестирование метода кэширования сгенерированного машинного кода с возможностью переиспользования осуществлялось на запросе Q1 из тестового набора TPC-H и запросах типа OLTP, представленных в табл. 1.

Для тестирования производительности механизма сохранения и патчинга сгенерированного машинного кода использовалась база данных из тестового набора TPC-H. Типы колонок базы данных были модифицированы следующим образом: тип CHAR(1) был изменен на тип ENUM, тип NUMERIC на DOUBLE PRECISION. Данная модификация позволяет использовать встроенные типы LLVM во время динамической компиляции. База данных генерировалась с параметром SCALE=2. Суммарный объем директории с базой данных составил 6,4 Гб.

Тестирование производительности выполнялось на компьютере с четырёхъядерным процессором Intel Core i7-6700HQ с ограничением тактовой частоты в 2.5 ГГц и с 16 гигабайтами оперативной памяти под управлением 64-битной операционной системы Ubuntu Linux версии 18.04. При тестировании база данных полностью располагалась в

оперативной памяти. Сравнение производительности интерпретатора и компилятора выполнялось с использованием СУБД PostgreSQL версии 9.6.3.

Для сбора статистических данных о результате выполнения запросов из табл. 1 использовалась программа $pgbench$ [19] – утилита, входящая в состав проекта PostgreSQL. Для каждого запроса выполнялось несколько запусков утилиты $pgbench$: $pgbench\ -n\ -M\ prepared\ -t\ 10000\ -l\ -f\ q[1,2,3].script\ -z\ 1$, где z – добавленный путём модификации исходного кода флаг, позволяющий выполнять инициализацию генератора случайных чисел одинаковым образом, а $q[1,2,3].script$ – файл с запросом.

Табл 1. SQL запросы для тестирования производительности метода кэширования
Table 1. SQL queries to test the performance of the machine code caching method

№ запроса	Текст запроса

1	`select customer.c_custkey, customer.c_name,`
	`customer.c_phone, customer.c_acctbal,`
	`orders.o_orderstatus, orders.o_totalprice,`
	`orders.o_orderdate, orders.o_clerk,`
	`lineitem.l_linenumber, lineitem.l_quantity,`
	`lineitem.l_discount, lineitem.l_tax,`
	`lineitem.l_shipdate, partsupp.ps_availqty,`
	`partsupp.ps_supplycost, part.p_name,`
	`part.p_brand, part.p_retailprice,`
	`supplier.s_name, supplier.s_address,`
	`supplier.s_phone`
	`from customer join orders on c_custkey = o_custkey`
	`join lineitem on l_orderkey = o_orderkey`
	`join partsupp on ps_partkey = l_partkey`
	`and ps_suppkey = l_suppkey`
	`join part on p_partkey = ps_partkey`
	`join supplier on s_suppkey = ps_suppkey`
	`where c_custkey between :bid1 and :bid1 + 20`
	`order by o_orderdate desc;`

2	`select l_returnflag, l_linestatus,`
	`sum(l_quantity), sum(l_extendedprice),`
	`sum(l_extendedprice * (1 - l_discount)),`
	`sum(l_extendedprice * (1 - l_discount)`
	`* (1 + l_tax)),`
	`avg(l_quantity), avg(l_extendedprice),`
	`avg(l_discount), count(*) as count_order`
	`from lineitem`
	`where l_shipdate <= date '1998-12-01'`
	`- interval '105 days'`
	`and l_partkey between :bid1 and :bid1 + 200`
	`group by l_returnflag, l_linestatus`
	`order by l_returnflag, l_linestatus;`

3	`select l_returnflag, l_linestatus,`
	`sum(l_quantity) as sum_qty,`
	`avg(l_discount) as avg_disc,`
	`count(*) as count_order`
	`from lineitem where l_shipdate <= date '1998-12-01'`
	`- interval '105 days'`
	`and l_partkey between :bid1`
	`and :bid1 + 200`

```
group by l_returnflag, l_linestatus
order by l_returnflag, l_linestatus;
```

Среднее число транзакций в секунду было получено в результате выполнения утилиты $pgbench$. Среднее время выполнения подсчитывалось на основе генерируемых pgbench лог-файлов. Результаты тестирования запросов из табл. 1 на 10000 транзакций отражены в табл. 2. Протокол $prepared$ в утилите $pgbench$ использует механизм кэширования, описанный в 4.1. Динамическая компиляция запроса с возможностью переиспользования машинного кода выполняется в момент создания оптимизатором PostgreSQL обобщенного (GENERIC) плана, т.е. при выполнении 6-ой транзакции. Дальнейшее выполнение запроса осуществляется путем повторного использования сгенерированного машинного кода. Таким образом, значение со средним количеством транзакций в секунду, подсчитанное утилитой $pgbench$, включает расходы, связанные с динамической компиляцией в 6-ой транзакции. Максимальное среднее ускорение без учета первых 6-ти транзакций в 1,78 раз было получено на запросе 2, где присутствует наибольшее число выражений.

Помимо тестирования запросов типа OLTP из табл. 1 был также протестирован запрос Q1 из набора TPC-H, результаты которого представлены в табл. 3. Сравнительное тестирование запроса Q1 в динамическом компиляторе запросов выполнялось с целью анализа влияния интринсика $llvm.experimental.patchpoint$ на качество результирующего машинного кода.

Табл. 2. Сравнение времени выполнения JIT компилятора с кэшированием машинного кода на тестовых запросах из табл. 1
Table 2. Comparison of the execution time of JIT compiler with machine code caching on test queries from table 1

Наименование единицы измерения	Запрос 1, vanilla PG	Запрос 1, JIT	Запрос 2, vanilla PG	Запрос 2, JIT	Запрос 3, vanilla PG	Запрос 3, JIT
Среднее кол-во транзакций в секунду (больше – лучше)	70,67	72,38	105.25	183,71	145,37	199,83
Компиляция обобщенного плана на 6-ой итерации, мс	-	1342,5	-	1118,9	-	997,2
Среднее время выполнения без учета первых 6 итераций, мс	14,12	13,65	9,49	5,31	6,87	4,89
Среднее ускорение выполнения без учета первых 6 итераций, X раз	1,03		1,78		1,40	

Среднее время выполнения запроса Q1 с использованием интерпретатора PostgreSQL составило 10 секунд, а время выполнения динамически скомпилированной версии этого же запроса составило 2.65 секунды, где 820 миллисекунд затрачивается на оптимизацию и компиляцию, а 1.73 секунд на выполнение машинного кода.

Табл. 3. Сравнение времени выполнения запроса Q1 из набора TPC-H в интерпретаторе PostgreSQL и динамическом компиляторе в режимах одноразовой кодогенерации и кэширования

машинного кода

Table 3. Comparison of execution time of query Q1 from TPC-H benchmark in PostgreSQL interpreter and JIT compiler in one-time code generation mode and machine code caching.

vanilla, PG	LLVM JIT			LLVM JIT + PREPARE		
	компиляция + оптимизация	выполнение	сумма	компиляция + оптимизация	выполнение	сумма
10 сек	(370 + 450) мс	1.73 сек	2.65 сек	(380 + 560) мс	2.4 сек	3.4 сек
				0.140 мс	2.4 сек	2.4 сек

Динамическая компиляция подготовленного плана запроса с возможностью переиспользования сгенерированного машинного кода суммарно составила 3.4 сек, а его среднее время выполнения на всех итерациях 2.4 сек. В случае использования подготовленного плана запроса накладные расходы на компиляцию и оптимизацию на всех итерациях после подготовки близки к 0.

Таким образом производительность машинного кода, сгенерированного с использованием $llvm.experimental.patchpoint$, в среднем на 38% меньше, чем результат одноразовой кодогенерации. Это связано с тем, что использование $llvm.experimental.patchpoint$ ограничивает возможности компилятора по выполнению оптимизаций над LLVM IR.

7. Заключение

В рамках данной работы был разработан метод сохранения и переиспользования сгенерированного динамическим компилятором запросов машинного кода, позволяющий нивелировать накладные расходы на его создание при повторном использовании. Возможность повторной утилизации сгенерированного машинного кода позволяет применять метод динамической компиляции для SQL запросов типа OLTP, время интерпретации которых измеряется миллисекундами.

Метод реализован в динамическом компиляторе запросов СУБД PostgreSQL с использованием технологии $Stack\ Map$ из инфраструктуры LLVM. Результаты проведенного тестирования показывают, что динамическая компиляция запросов с помощью JIT-компилятора LLVM с возможностью дальнейшего переиспользования результирующего машинного кода позволяет получить существенное ускорение на OLTP-запросах с достаточным количество выражений.

В будущем планируется расширить существующий механизм сохранения и переиспользования сгенерированного машинного кода, реализовав его автоматическое сохранение для часто выполняющихся однотипных запросов с использованием стоимостных оценок и эвристик.

Список литературы / References

[1]. Graefe G. Volcano – an extensible and parallel query evaluation system. IEEE Transactions on Knowledge and Data Engineering, vol. 6, issue 1, 1994, pp. 120–135.

[2]. Stefan Manegold, Martin L. Kersten, and Peter Boncz. Database architecture evolution: mammals flourished long before dinosaurs became extinct. Proceedings of the VLDB Endowment, vol. 2, 2009, pp. 1648-1653.

[3]. S. Padmanabhan, T. Malkemus, A. Jhingran and R. Agarwal. Block oriented processing of relational database operations in modern computer architectures. In Proc. of the 17th International Conference on Data Engineering, 2001, pp. 567-574.

[4]. Thomas Neumann. Efficiently compiling efficient query plans for modern hardware. Proceedings of the VLDB Endowment, vol. 4, no. 9, 2011, pp. 539-550.

[5]. А.Н.Андреев. Классификация OLAP-систем вида xOLAP / A.N. Andreev. OLAP systems of XOLAP type classification. Available at: http://citforum.ru/consulting/BI/xolap_classification/, accessed: 25.07.2019 (in Russian).

[6]. TPC-H benchmark for testing OLAP workload. Available at: http://www.tpc.org/tpch/, accessed 25.07.2019.

[7]. What is an OLTP System? Available at: https://docs.oracle.com/database/121/VLDBG/GUID-0BC75680-5BD4-43A9-826F-CD8837D30EB2.htm#VLDBG1367, accessed: 25.07.2019.

[8]. TPC-B benchark for testing OLTP workload. Available at: http://www.tpc.org/tpcb/, accessed: 25.07.2019.

[9]. The LLVM Compiler Infrastructure. Available at: http://llvm.org/, accessed: 25.07.2019.

[10]. 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 / 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). DOI: 10.15514/ISPRAS-2016-28(4)-13

[11]. 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 / 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). DOI: 10.15514/ISPRAS-2016-28(6)-3

[12]. E. Sharygin, R. Buchatskiy, R. Zhuykov, and A. Sher. Runtime Specialization of PostgreSQL Query Executor. Lecture Notes in Computer Science, vol. 10742, pp. 375–386, 2018.

[13]. PostgreSQL official site. Available at: https://www.postgresql.org/, accessed: 25.07.2019.

[14]. ISP RAS website. Available at: https://www.ispras.ru/, accessed: 25.07.2019.

[15]. PREPARE command, PostgreSQL. Available at: https://www.postgresql.org/docs/9.6/sql-prepare.html, accessed: 25.07.2019.

[16]. EXECUTE command, PostgreSQL. Available at: https://www.postgresql.org/docs/9.6/sql-execute.html, accessed: 25.07.2019.

[17]. MCJIT Design and Implementation. Available at: https://releases.llvm.org/4.0.0/docs/MCJITDesignAndImplementation.html, accessed: 25.07.2019.

[18]. Stack maps and patch points in LLVM. Available at: https://llvm.org/docs/StackMaps.html, accessed: 25.07.2019.

[19]. pgbench utility. Available at: https://www.postgresql.org/docs/9.6/pgbench.html, accessed: 25.07.2019.

Информация об авторах / Information about authors

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.

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

Ruben Arturovich BUCHATSKIY – Researcher in Compiler Technology department. Research interests: compiler technologies, optimizations.

Роман Александрович ЖУЙКОВ – научный сотрудник отдела компиляторных технологий. Научные интересы: компиляторные технологии, оптимизации.

Roman Aleksandrovich ZHUYKOV – Researcher in Compiler Technology department. Research interests: compiler technologies, optimizations.