ОГЛАВЛЕНИЕ

ОТ СОСТАВИТЕЛЕЙ

В настоящем номере журнала «Электронные библиотеки» опубликованы статьи, составляющие вторую часть тематического выпуска и подготовленные их авторами на основе материалов, представленных на научной конференции «Научный сервис в сети Интернет» 2019 года. Тематика конференции и, соответственно, тематического выпуска достаточно широка: от цифровых библиотек, библиографических баз и наукометрии до различных специальных областей использования возможностей интернета для научных исследований.

Первая часть тематического выпуска размещена в №3 журнала «Электронные библиотеки» за 2020 год.

М. М. Горбунов-Посадов, А. М. Елизаров

УДК 338.28

О МОДЕЛИРОВАНИИ ФУНКЦИОНИРОВАНИЯ ОБОРОННЫХ ОТРАСЛЕЙ ЭКОНОМИКИ: ИНФОРМАЦИОННАЯ ПОДДЕРЖКА

А. А. Акиншин [1], Ю. Е. Поляк [2]

[1,2] *Федеральное государственное бюджетное учреждение науки Центральный экономико-математический институт Российской академии наук, г. Москва*

[1] aaa@cemi.rssi.ru, [2]polak@cemi.rssi.ru

Аннотация

При обработке огромных объемов информации, возникающих в процессе деятельности отдельных организаций и целых отраслей экономики нужны новые подходы и решения. Для эффективной конверсии национальной оборонной промышленности разрабатывается информационно-аналитическая система. В работе обсуждается создаваемый инструментарий систематизации открытых данных о продукции военного назначения и оборонно-промышленном потенциале. В частности, разрабатывается ресурс, который будет играть роль своего рода агрегатора указанной информации.

***Ключевые слова**: оборонно-промышленный комплекс, научно-промышленная политика, информационная поддержка, сетевые информационные ресурсы, базы данных*

ВВЕДЕНИЕ

В настоящее время оборонно-промышленный комплекс (ОПК) России включает около 800 промышленных предприятий и около 570 проектных, конструкторских и технологических организаций. Около 1200 предприятий и организаций (включая ремонтные заводы Минобороны России) расположены в 70 регионах России. Число занятых на предприятиях и в организациях оборонной промышленности превышает 2 млн. чел.

На долю ОПК, включая продукцию военного назначения, приходится более 70% всей научной продукции России, в нем занято более 50% всех научных сотрудников. Оборонно-промышленный комплекс является наиболее высокотехнологичным сектором экономики страны [1].

Современный ОПК как совокупность специфических хозяйствующих субъек-

тов представляет собой сложную систему с большой степенью разнообразия и неопределенности, со сложным управлением, определяющим множество вариантов при выборе решений о функционировании. Соответственно сложной является и проблема повышения эффективности управления развитием отраслей ОПК. Чтобы анализировать, моделировать и прогнозировать данный процесс в современных условиях с учетом роста объемов производства продукции и инновационной модернизации, необходима разработка комплексного, научно обоснованного и практически применимого методического аппарата и инструментария.

ОСОБЕННОСТИ ПРОЕКТА

Отделению экономической информатики Центрального экономико-математического института (ЦЭМИ) РАН поручено выполнение проекта «Информационно-аналитический и программный инструментарий систематизации имеющихся данных о продукции военного назначения, определения и анализа оборонно-промышленного потенциала с целью инновационного роста национальной экономики». В этом исследовании должна быть проведена оценка современного состояния российского оборонно-промышленного комплекса и его роли в экономическом развитии страны, должны быть определены структурные проблемы и угрозы его эффективному развитию, оценены возможные тенденции дальнейшего развития отечественных предприятий, сотрудничающих с данным сектором экономики.

Особенность информационно-аналитического и программного инструментария, разрабатываемого в проекте, а также технологий моделирования состоит в использовании многоуровневой декомпозиции национальной инновационной системы и процесса ее моделирования. Количество уровней определяется сложностью моделируемого объекта и степенью детализации построенной модели. Используемый подход позволит представить национальную инновационную систему в виде произвольного декомпозиционного множества подсистем и элементов, а процесс ее функционирования расчленить на составляющие его процедуры. Создаваемый в рамках проекта инструментарий концептуального моделирования будет способствовать повышению качества проектируемых и функционирующих оборонных производств, сокращению общих затрат и сроков их реформирования, росту производительности труда проектировщиков и менеджеров.

Как говорилось выше, оборонно-промышленный комплекс является одним из наиболее современных, наукоемких и высокотехнологичных секторов отечественной промышленности, инновационным драйвером экономики. Для реализации эффективной научной и промышленной политики в нашей стране имеются необходимые предпосылки. Это наличие передовой науки, развитой системы образования, производственной базы, человеческого потенциала, финансовых и материальных ресурсов и т. д. Однако в условиях переходного периода и затяжного кризиса реального сектора экономики возникают новые вызовы. В частности, на научно-технический и производственный потенциал оборонной промышленности негативно влияют малое энергоснабжение, износ основных производственных фондов, низкая загрузка производственных мощностей, распад существующих кооперационных связей, сокращение численности и качества персонала, рост цен на энергоносители и так далее. Перед исследователями стоят задачи дать всестороннюю оценку современного состояния и роли оборонной промышленности в экономическом развитии страны; выявить структурные проблемы и внешние угрозы его эффективному развитию. Для этого необходимо создать информационные, аналитические и программные средства для систематизации имеющихся открытых данных о военной продукции. Важными с методологической точки зрения направлениями исследований являются выделение и описание изучаемой предметной области; подготовка четких методик и рекомендаций, обеспечивающих внедрение теоретических положений в практику; создание информационного и программного инструментария, позволяющего в автоматизированном режиме проектировать и анализировать концептуальные модели разнообразных инновационных объектов и их комплексов.

ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ПРОЕКТА

Для построения классификаторов и таксономий, последующего наполнения и актуализации информационно-аналитической системы рассматриваются следующие наиболее существенные источники информации:

- электронные коллекции и библиотеки;
- монографии, диссертации;
- материалы конференций и семинаров;
- научно-техническая периодика;

• отдельные публикации, статьи;

• отчеты;

• патенты;

• веб-сайты различных уровней: персональные страницы, официальные сайты предприятий, промышленных групп, кластеров;

• новостные ленты;

• списки ресурсов, классификаторы.

Открытые источники структурируются в плане их значимости, полноты, актуальности и регулярности обновления. Таким образом, появляется необходимость ранжирования их по всем этим параметрам.

Коллектив исполнителей имеет необходимый опыт разработки значимых информационных проектов. Лаборатория сетевых информационных ресурсов ЦЭМИ РАН с 1995 года занималась исследованием информационных ресурсов отечественного интернета. За короткий срок она вошла в число ведущих коллективов страны в этой области. Её главная разработка – база данных «Интернет в России, Россия в интернете» (1996), ставшая основой знаменитого в своё время онлайн-каталога «Ау!» [2]. В 1997 г. база зарегистрирована в РосАПО, а в 2000 г. – в НТЦ «Информрегистр». В лаборатории велись подготовка обзорных, аналитических, учебно-методических материалов о сетевых информационных ресурсах для профессионалов, разработка онлайновых аннотированных указателей веб-адресов, а также печатных справочников. С этой целью проводился анализ систем рубрикации электронных документов и структур каталогов электронных библиотек; исследовались возможности фасетной классификации, систем таксономий и метаданных, вопросы разработки навигационных систем по научно-техническим информационным ресурсам, роль и значение информационного каталога в архитектуре хранилища данных, различные архитектурные решения построения систем управления метаданными.

Этот опыт находит применение при разработке информационно-аналитического и программного инструментария систематизации открытых данных о продукции военного назначения и оборонно-промышленном потенциале. Подчеркнём: ориентация исключительно на открытые источники является нашей принципиальной позицией. Коллектив разработчиков имеет в составе первоклассных профессионалов, занимавших в своё время руководящие должности в советской

и российской армии. Их квалификация, опыт выполнения прежних проектов дают уверенность, что учёные РАН вполне способны делать обоснованные стратегические прогнозы и планы развития оборонных производств на основании имеющихся открытых данных.

Информационно-аналитическая система должна включать

- базу знаний;
- нормативные документы;
- статистические сведения;
- примеры и технические характеристики успешных разработок;
- справочный аппарат по интернет-ресурсам.

Выполняя исследовательскую, аналитическую работу, специалисты заинтересованы в наличии универсального информационного ресурса, содержащего всё перечисленное. Очень удобно, если все эти материалы будут собраны в одном месте, иметь интуитивно понятный и единообразный интерфейс. Чтобы не перегружать портал второстепенной информацией, его следует снабдить каталогом ссылок – навигатором по открытым источникам данных. Отметим, что официальные сайты оборонных организаций подобной комплексной информации, как правило, не содержат. Занимаясь вебометрикой, один из авторов убедился [3, 4], что сайты вузов силовых ведомств весьма лаконичны по сравнению с гражданскими университетами (впрочем, причины этого вполне понятны). Здесь примером может служить портал Министерства обороны США Defense.gov.

Система должна выполнять такие функции (операции), как

- поиск объектов по элементам метаданных и их комбинациям с относительно развитым языком запросов,
- навигация (переход) между взаимосвязанными объектами,
- формирование наборов объектов с заданными признаками.

Анализ информационных источников позволит определить главные причины недостаточной конкурентоспособности российских наукоемких и высокотехнологичных предприятий и выделить факторы ее повышения.

ПОРЯДОК РЕАЛИЗАЦИИ ПРОЕКТА

К числу основных функциональных задач информационно-аналитического инструментария следует отнести содержательный анализ проблемной ситуации,

определение параметров стратегии управления через процедуры уточняемых программ развития, а также наработки опыта автоматизации и применения технологии на практике. Необходимость в постоянной приспособляемости технологии планирования к ходу экономических преобразований и информационным потокам выдвигает жесткие требования к адаптивным свойствам инструментария и соответствующей интеллектуально насыщенной базе знаний. Воздействуя управляющими параметрами на процедуру подготовки решения, лицо, принимающее решение, получает различные варианты программно-плановых решений, отличающихся технико-экономическими показателями, степенью использования локальных и общесистемных ресурсов, степенью выполнения заказов на продукцию (в том числе научную) и т. п. Каждое из этих проектных решений является рациональным с точки зрения определенного критерия. Набор моделей информационно-аналитической технологии, а также методов их практической реализации должны обеспечивать оперативность выполнения расчетов, как на агрегированных, так и детализированных исходных данных.

Принятие решений, направленных на оптимизацию научно-промышленной политики в целом и ВПК, в частности, должно опираться на адекватную информационно-технологическую базу.

Рассмотрим основные этапы выполнения проекта. Декомпозиция описания предметной области. Декомпозиция процессов моделирования. Построение таксономии. Например, двухэтапный полуавтоматический метод. На первом этапе – построение основы таксономии, два или три верхних уровня в соответствии с нормативными документами, формализующими рассматриваемую предметную область. На втором этапе – пошаговое достраивание тем таксономии на основе анализа данных открытых источников, используя меру сходства (релевантность) тем тексту.

Для построения информационно-аналитической системы необходимо:

1. Разработать модель базы знаний, позволяющей собирать, хранить, обрабатывать данные о продукции оборонно-промышленного комплекса, получаемых из открытых источников.

В основе базы знаний – база данных. Одна из главных целей – база технологий. При ее создании возникает несколько классификаций (таксономий) при разработке модели БД.

«Кластерно-отраслевой» подход. Предлагается использовать классификацию Ростеха, и тогда первые уровни выглядят так: «ОПК» -> «Кластеры» –> «Пром. объединения» -> «Предприятия» -> «Продукция (тип/группа)» -> «Изделие» -> «Компоненты». Кластеры Ростеха (авиация, радиоэлектроника, вооружение) имеет смысл дополнить так, чтобы они охватывали максимально большой спектр продукции ОПК.

«Технологический» подход. Классификация на основе технологий, используемых при выпуске продукции ОПК. За основу таксономии берется международная патентная классификация.

Область применения. База должна позволять как начальное создание соответствующей таксономии, так и дальнейшее ее расширение в процессе наполнения и эксплуатации. Тогда при построении ER-модели базы возникают три набора атрибутов, соответствующих приведенным выше.

Декомпозиция предметной области («Продукция ОПК») должна проводиться по разным типам критериев на различных уровнях. От уровня «ОПК» до уровня «Изделие» – критерии «отраслевой» классификации. От уровня «Изделие» – технологическая классификация и область применения.

2. Подготовить методику сбора, систематизации и анализа данных о продукции оборонно-промышленного комплекса, а также различные варианты ее представления, в зависимости от решаемых задач.

3. Разработать программный инструментарий, позволяющий провести сбор и анализ данных, их последующее представление.

ЗАКЛЮЧЕНИЕ

Реализация изложенного позволит подойти к решению фундаментальной проблемы формирования эффективной научно-промышленной политики и реализующего ее инструментария в условиях структурных преобразований в современном инновационном и высокотехнологичном секторе отечественной промышленности и на примере оборонно-промышленного комплекса научно обосновать конкретную систему мер, обеспечивающих прогрессивные структурно-технологические сдвиги в комплексе и в российской экономике в целом.

В настоящее время разрабатывается ER-модель базы данных; отрабатывается и тестируется методика сбора и систематизации информации.

В статье использованы материалы доклада на XXI Всероссийской научной конференции «Научный сервис в сети интернет» (сентябрь 2019 года).

Благодарности

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований в рамках научного проекта № 18-00-00177 (18-00-00172).

СПИСОК ЛИТЕРАТУРЫ

1. *Хрусталёв Е.Ю.* Оборонно-промышленный комплекс России: предназначение, состояние и перспективы развития // Национальные интересы: приоритеты и безопасность. 2011. Т. 7. №35. С. 61–71.

2. *Куликов В.В., Поляк Ю.Е.* Каталог русскоязычных ресурсов интернета «Ау!» // Телематика '98. Сборник научных трудов. СПб, 1998. С. 334–335.

3. *Поляк Ю.Е.* Российский и международный опыт вебометрических исследований // Информационные ресурсы России. 2014. № 6 (142). С. 2–9.

4. *Поляк Ю.Е.* Оценивание и ранжирование веб-сайтов. Вебометрические рейтинги. // Научный редактор и издатель. 2017. №2(1). С. 19–29.

ON MODELING THE OPERATION OF THE DEFENSE INDUSTRIES OF THE ECONOMY: INFORMATION SUPPORT

A.A. Akinshin[1], Yu.E. Polyak[2]

[1,2] *Central Economics and Mathematics Institute of the Russian Academy of Sciences*

[1]aaa@cemi.rssi.ru, [2]polak@cemi.rssi.ru

Abstract

When processing huge amounts of information arising in the process of activities of individual organizations and entire sectors of the economy, new approaches and solutions are needed. For effective conversion of national defense industry, an information and analytical system is being developed. The paper discusses the created tool for systematization of open data on military products and defense industry potential. In particular, a resource is being developed that will be a kind of aggregator of this information.

Keywords*: military-industrial complex, scientific and industrial policy, information support, network information resources, databases*

REFERENCES

1. *Hrustalyov E.Yu.* Oboronno-promyshlennyj kompleks Rossii: prednaznachenie, sostoyanie i perspektivy razvitiya // Nacional'nye interesy: prioritety i bezopasnost'. 2011. T. 7. No. S. 61–71.

2. *Kulikov V.V., Polyak Yu.E.* Katalog russkoyazychnykh resursov interneta «Au!» // Telematika '98. Sbornik nauchnykh trudov. SPb. 1998. S. 334–335.

3. *Polyak Yu.E.* Rossiyskiy i mezhdunarodnyy opyt vebometricheskikh issledovaniy // Informatsionnyye resursy Rossii. 2014. No 6 (142). S. 2–9.

4. *Polyak Yu.E.* Otsenivaniye i ranzhirovaniye veb-saytov. Vebometricheskiye reytingi. // Nauchnyy redaktor i izdatel. 2017. No 2(1). S. 19–29.

СВЕДЕНИЯ ОБ АВТОРАХ

АКИНШИН Анатолий Анатольевич – зав. лабораторией локальных вычислительных сетей и компьютерных информационно-издательских технологий Центрального экономико-математического института РАН (Москва).

Anatoly Anatolievich AKINSHIN – Head of Laboratory of Local Area Networks and Computer Information and Publishing Technologies, Central Economics and Mathematics Institute. Moscow, Russia

email: aaa@cemi.rssi.ru

ПОЛЯК Юрий Евгеньевич – ведущий научный сотрудник Центрального экономико-математического института РАН (Москва). Подробнее: http://computer-museum.ru/articles/sovet-muzeya/561/

Yuri Evgenievich POLYAK – Candidate of Economic Sciences, Leading Researcher, Central Economics and Mathematics Institute. Moscow, Russia. More detailed: http://computer-museum.ru/articles/sovet-muzeya/561/

email: polak@cemi.rssi.ru

Материал поступил в редакцию 27 ноября 2019 года

УДК 004.42 + 004.415

ВЕБ-СРЕДА АНАЛИЗА И ПРЕОБРАЗОВАНИЙ ПРОГРАММ В ОПТИМИЗИРУЮЩЕЙ РАСПАРАЛЛЕЛИВАЮЩЕЙ СИСТЕМЕ

А. П. Баглий

Институт математики, механики и компьютерных наук Южного федерального университета, *г. Ростов-на-Дону*

taccessviolation@gmail.com

Аннотация

Описан опыт проектирования различных вариантов веб-среды разработки (IDE) для Оптимизирующей распараллеливающей системы и компилятора на реконфигурируемую архитектуру на основе существующих инструментов, таких, как Jupyter Notebook и Eclise Che. Сформированы требования к инструментам в составе Открытой распараллеливающей системы для поддержки их интеграции в веб-среду разработки, доступную в интернете. Описан процесс создания переносимого окружения для разработки модулей компилятора, демонстрации его работы и обучения навыкам разработки параллельных программ. Приведены примеры использования разработанных преобразований программ, используемых при оптимизации программ для ПЛИС в разработанной веб-среде, и описаны способы визуализации результатов выполнения преобразований и анализа при использовании Jupyter Notebook. Проведенная работа демонстрирует возможность организации удаленного доступа к библиотеке разрабатываемых инструментов оптимизации программ в виде, удобном прикладным разработчикам.

Ключевые слова*: интегрированная среда, распараллеливающий компилятор, преобразования программ, ПЛИС, контейнеризация, интерактивная тетрадь, облачные вычисления.*

ВВЕДЕНИЕ

При разработке сложной системы, например, оптимизирующего компилятора, возникает множество проблем, связанных с организацией как самого процесса разработки, обучением и вовлечением новых программистов, так и

предоставления доступа к результатам извне, например, для демонстрации определенных функций и результатов численных экспериментов. Проблемы, возникающие в процессе разработки компилятора, в чем-то аналогичных тем, которые возникают перед программистом, не знакомым с разработкой программ для программируемых логических схем. На сегодняшний день системы, позволяющие напрямую преобразовывать программы на высокоуровневом языке программирования в схему для ПЛИС, или недостаточно развиты, или недостаточно распространены. Для того чтобы сгладить эту проблему, целесообразно организовать доступ к подобным инструментам через интернет в виде интегрированной среды разработки в браузере, дополненной некоторыми расширениями, позволяющими программисту в диалоговом режиме итеративно модифицировать свою программу и получаемую схему на ПЛИС и оценивать их характеристики. Такой подход может решить некоторые из самых острых проблем, с которыми сталкивается все больше программистов при первых попытках разработки программ для ПЛИС.

Выделим основные проблемы, которые рассматриваются в данной работе:

— быстрое развертывание окружения со всеми необходимыми инструментами для нового члена коллектива разработчиков или нового клиента системы;

— предоставление интуитивно понятного и простого диалогового интерфейса, позволяющего программисту получить программу, оптимизированную для ПЛИС или другой архитектуры по исходной;

— демонстрация некоторых новых возможностей разрабатываемого компилятора, включая оптимизирующие преобразования и алгоритмы анализа программы;

— использование создаваемой системы для обучения разработке параллельных программ и использования преобразований программ;

— использование частей создаваемой системы в научных исследованиях, где требуются предоставление доступа к результатам и удобство их воспроизведения.

Для демонстрации возможностей Оптимизирующей распараллеливающей системы [1] ранее были разработаны веб-интерфейсы для демонстрации ее от-

дельных функций [2, 3]. Общий вид первой версии веб-интерфейса показан на рис. 1.

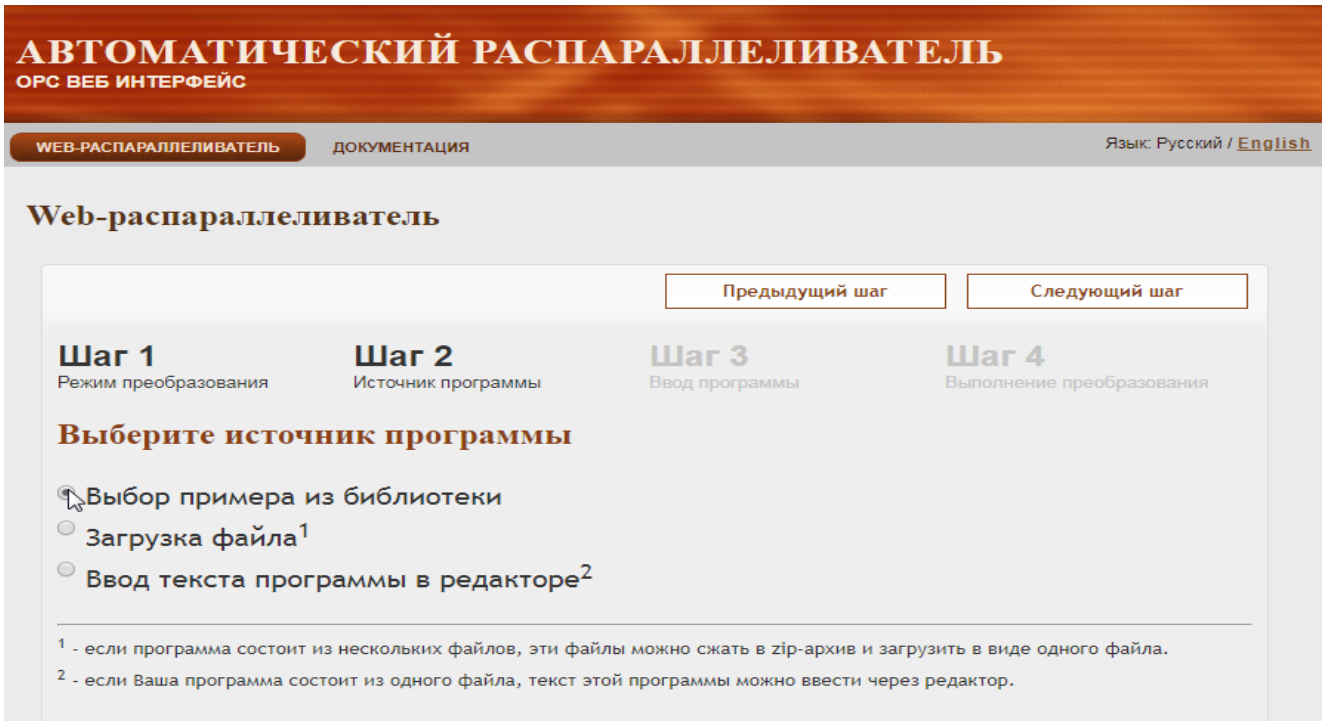


Рис. 1. Окно одной из предыдущих версий веб-интерфейса ОРС

Этот интерфейс позволяет пользователю загрузить исходный текст программы, выбрать действие из заданного набора (преобразование программы, автоматическое распараллеливание), выполнить указанной действие на сервере и скачать результирующий текст программы.

Такой веб-интерфейс решает только проблему №2 из перечисленных выше, впрочем, далеко не полностью. Пользователь не имеет возможности напрямую управлять выполнением преобразований, а от разработчиков требуется создание специальных сценариев использования системы для всех необходимых случаев.

Настоящая работа ставит задачу создания более универсальной среды разработки, доступной в интернете, которая:

— была бы основана на существующих наработках, широко применяемых в среде облачных вычислений;

— решала бы остальные указанные проблемы;

— не требовала бы в дальнейшей перспективе дополнительных усилий от разработчиков для поддержания своей функциональности при модификации самого разрабатываемого компилятора.

На основе ОРС разрабатывается компилятор на реконфигурируемые вычислительные архитектуры [4], для которого указанные проблемы стоят особенно остро, поэтому подразумевается в первую очередь применение описанных подходов к данному компилятору.

На основе перечисленных проблем ставится задача о предоставлении доступа пользователей через веб-браузер к библиотеке интерактивных документов (демонстрационных примеров, программ), которые будут сопровождаться индивидуальным для каждого пользователя окружением разработки на удаленном сервере. Эти окружения должны создаваться по требованию и позволять запускать программы из библиотеки, визуализировать результаты их работы и легко модифицировать программы.

Основными результатами данной работы являются, во-первых, новые архитектурные требования к Оптимизирующей распараллеливающей системе, которые постепенно внедряются в нее, чтобы улучшить ее модульность и дать возможность собирать на ее основе инструменты для компиляции программ, обучения параллельному программированию и программированию для ПЛИС. Во-вторых, результатом работы является прототип веб-среды разработки для синтеза программ для ПЛИС, основанный на ОРС и существующих облачных IDE с открытым исходным кодом.

Далее в разделе «Постановка задачи» описаны более подробно требования к составным частям компилятора, которые выработаны для создания веб-IDE на его основе. В разделе «Jupyterlab и OPS» описан подход к созданию среды разработки на основе OPS на языке C++ на основе интерактивных тетрадей в браузере. В разделе «Визуализация результатов» описаны выбранные способы визуализации результатов запуска преобразований программ. В разделе «Примеры преобразований программ» приведены примеры использования веб-среды для запуска преобразований, предназначенных для оптимизации программ на ПЛИС. В разделе «Результаты» представлены основные изменения, которые позволяют создать минимально функциональное окружение разработки программ на основе разрабатываемого компилятора.

ПОСТАНОВКА ЗАДАЧИ

В среде облачных вычислений существует множество широко используемых веб-ориентированных сред разработки программ с открытым исходным кодом, которые подразумевают создание расширений для той или иной предметной области, например, поддержки новых языков программирования, управления вычислительными кластерами.

Стоит особо выделить среду Eclipse Che [5], архитектура которой позволяет решить почти все проблемы, упомянутые во введении:

— создание и использование контейнеризованных рабочих сред для разработчиков системы с помощью docker[1] решает проблему быстрого развертывания окружения с повторяемыми характеристиками;

— управление рабочей средой с помощью расширяемого API решает проблему удобства демонстрации выбранных функций системы;

— возможность создания расширений для всех архитектурных частей системы позволяет добавить в нее новые представления для демонстрации важных функций и визуализации результатов.

Однако использование данной среды разработки в качестве основы для собственной облачной среды требует модификации как самой Eclipse Che, так и разрабатываемого компилятора.

В общем виде требуются:

— разбиение процесса работы компилятора на этапы, между которыми возможен экспорт или визуализация промежуточных результатов;

— добавление новых представлений в интерфейс IDE и соответствующих им методов API рабочей среды;

— создание набора контейнеров для обеспечения работы компилятора;

— обеспечение более слабых связей между модулями компилятора для удобства их использования со стороны облачной IDE.

[1] https://www.docker.com/

JUPYTERLAB И OPS

Jupyterlab [9] является на данный момент самым простым в использовании средством интерактивного предоставления результатов вычислений, это веб-приложение и связанный с ним набор инструментов поддерживают множество языков программирования, включая C++, за счет использования Xeus-Cling [10]. Это позволяет организовать доступ ко всем инструментам в составе OPC в виде интерактивных тетрадей в браузере, как показано, например, на рис. 2.

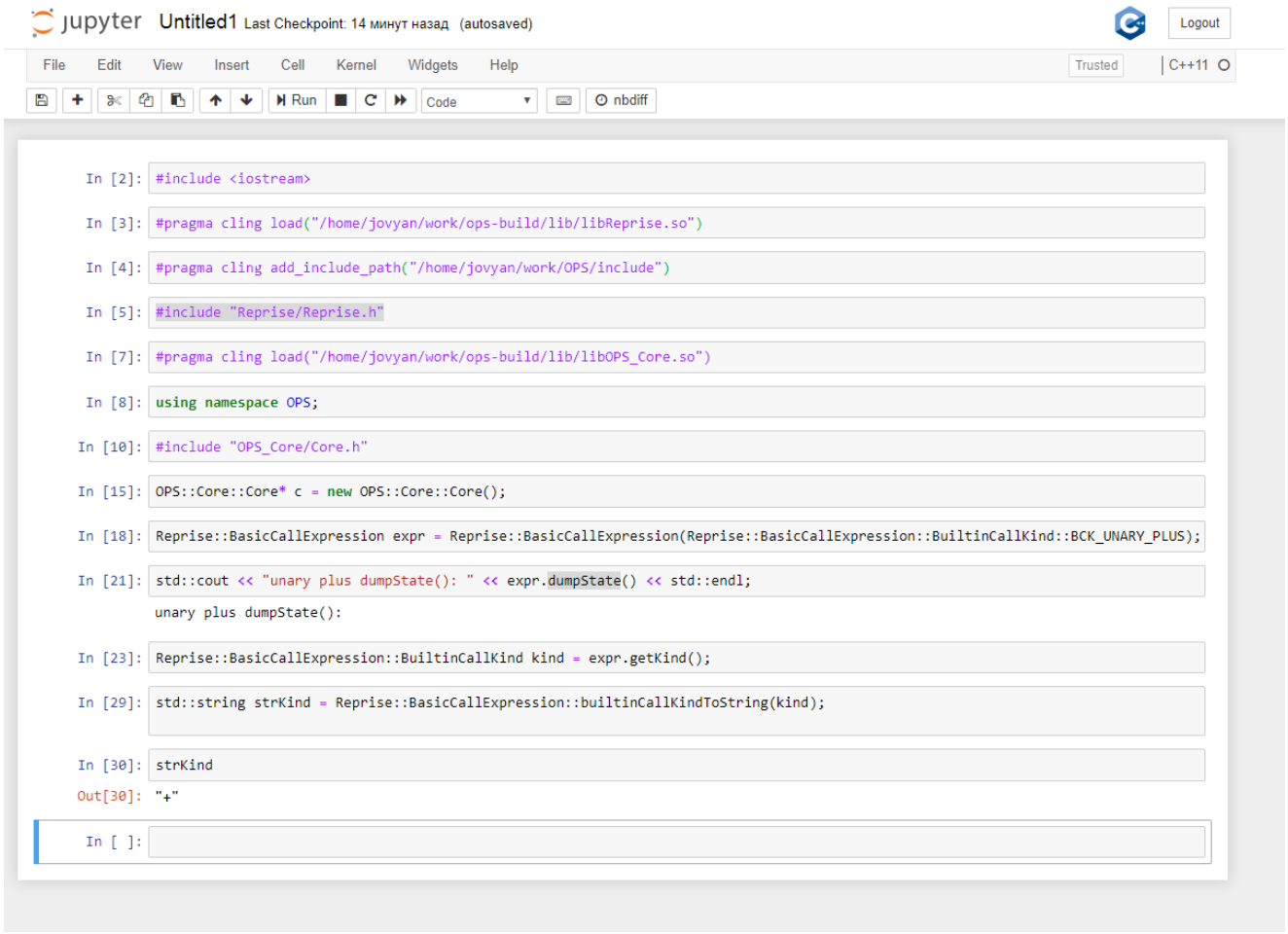


Рис. 2. Пример выполнения функций OPC в Jupyter notebook

За счет того, что в интерактивных тетрадях Jupyterlab получается использовать тот же язык программирования, на котором написана сама система, достигается простота написания сценариев ее использования разными типами пользователей. Этой возможностью уже пользуются различные проекты, например,

система анализа данных ROOT[2], так же использующая C++ и Cling в среде JupyterLab.

При использовании JupyterLab каждый пользователь получает в свое распоряжение интерактивную среду в браузере, внутри которой он может создавать тетради, использующие различные языки программирования, выполнять программы в них и использовать предоставленную библиотеку примеров. Общий вид интерфейса пользователя показан на рис. 3.

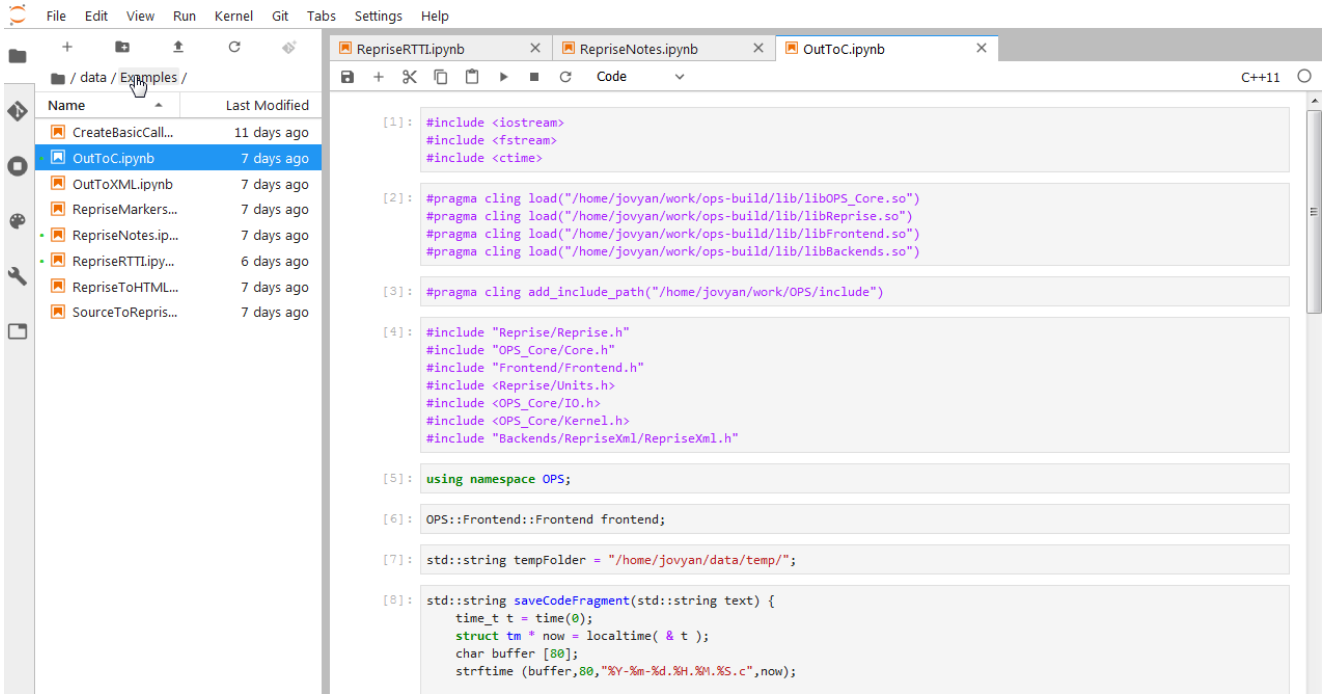


Рис. 3. Пример интерфейса пользователя системы при работе с набором готовых примеров использования функций OPC

Теперь для предоставления различных сценариев использования системы достаточно составить набор интерактивных тетрадей, которые:

— демонстрировали бы ключевые функции модулей системы, например, работу с внутренним представлением, выполнение преобразований программ, что особенно полезно в процессе обучения;

— предоставляли бы пользователям возможность проведения экспериментов по исследованию производительности и эффектов различных преобразований над тестируемой программой с помощью вспомогательных сервисов;

[2] https://root.cern.ch/

— предоставляли бы разработчикам системы простой и удобный интерфейс для обмена результатами работы, проведения тестов.

Кроме этого, использование тетрадей JupyterLab позволяет организовать работу с библиотекой примеров программ, например, используя проект Binder[3]

ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ

JupyterLab позволяет использовать различные форматы[4] содержимого ячеек, что дает возможность визуализировать данные любых типов. При выполнении преобразований программ требуется визуализировать следующие данные:

— тексты программ с пометками операторов, вхождений переменных и т. п., для которых удобно использовать формат HTML;

— графовые представления программ, в которых узлами служат вхождения переменных, выражения или операторы, для которых удобно использовать SVG-графику;

— табличные данные – результаты экспериментов, метрики исходного кода, для которых можно использовать формат Markdown;

— графовые представления программ, сложные для визуализации, например, решетчатый граф программы, для которых возможно использовать растровую графику.

Для визуализации текстов программ, например, исходной и полученной после преобразования, удобно использовать HTML содержимое, как проиллюстрировано на рис. 4., на котором показаны два результата поиска похожих участков в исходной программе для создания расширения набора инструкций процессора на ПЛИС. Кроме пометок операторов цветом возможно использовать интерактивные элементы управления. Использование текстовых форматов преставления, например, HTML или SVG, облегчает разработку модулей визуализации и передачу готовых данных.

[3] https://mybinder.org/

[4] https://jupyterlab.readthedocs.io/en/stable/user/file_formats.html

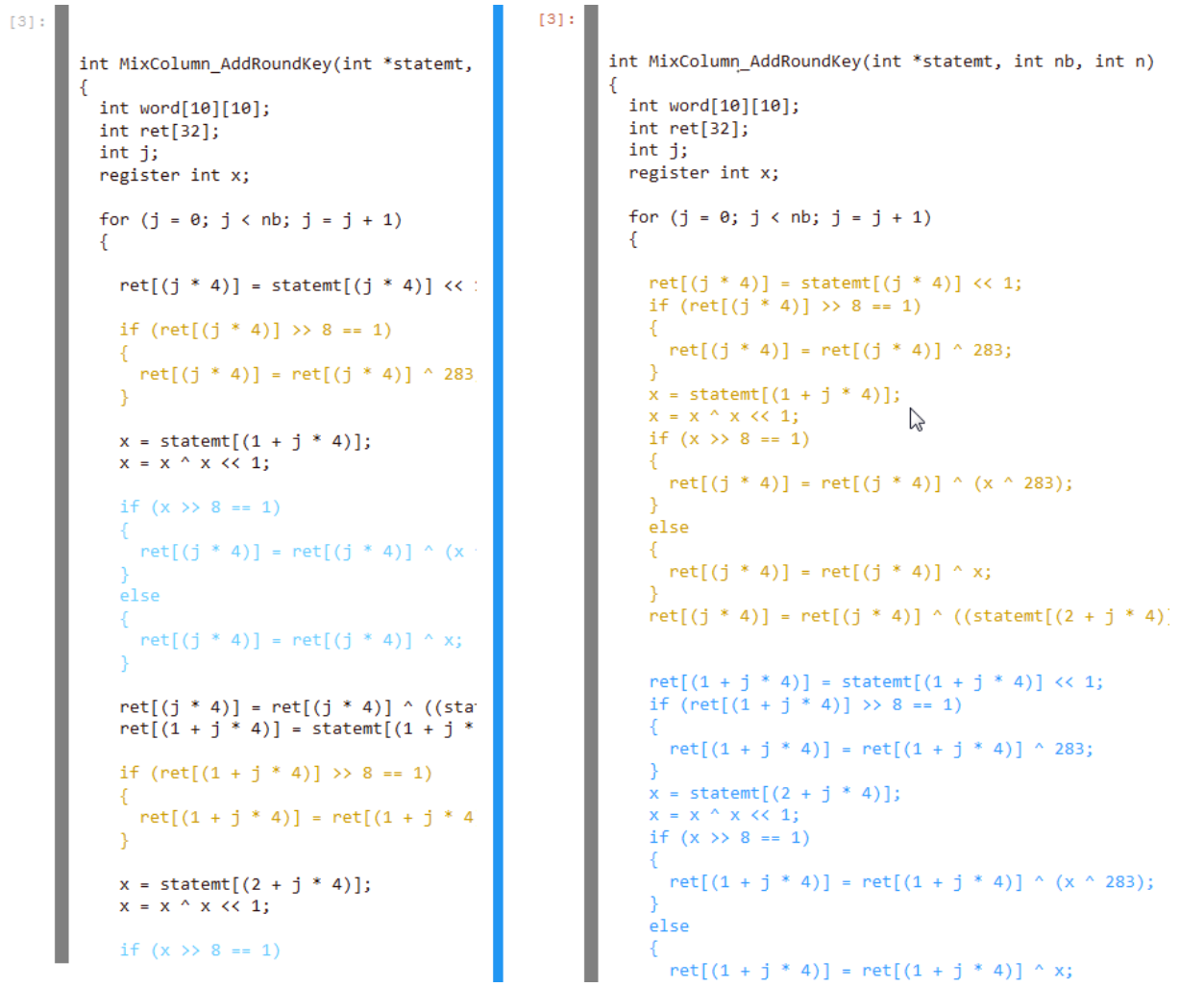


Рис. 4. Пример использования HTML для визуализации размеченного текста программ внутри JupyterLab

ПРИМЕРЫ ПРЕОБРАЗОВАНИЙ ПРОГРАММ

При разработке преобразований программ в Оптимизирующей распараллеливающей системе и оценке их применимости для компиляции программ на ПЛИС требуется удобное окружения для тестирования преобразований в отдельности и в цепочке, которое позволит легко демонстрировать полученные результаты, включать примеры в документацию. Использование JupyterLab с разработанной системой решает эти задачи.

На рис. 5 и 6 показаны примеры запуска преобразований на простых программах. На рис. 5 показано преобразование, конвертирующее функцию в макро-описание для эмулятора операций в системе ТСЕ[5]. На рис. 6 показан пример

[5] http://openasip.org/

запуска преобразования, заменяющего участок кода на вызов функции с соответствующим телом.

```
[26]: "
int main()
{
  int a;
  int b;
  int z;
  a = 0;
  b = 2;
  z = 0;
  z = add(a, b);
  return 0;
}"
```

```
[15]: #include <iostream>
std::ostringstream str;
using namespace OPS;
using namespace OPS::Frontend;
using namespace OPS::Reprise;
OPS::Backends::OutToC printer(str)
OPS::Reprise::Declarations& decls = frontend.getProgramUnit().getUnit(0).getGlobals();
OPS::Reprise::DeclIterator<OPS::Reprise::SubroutineDeclaration> it = decls.getFirstSubr();
OPS::Reprise::SubroutineDeclaration& addSub = *it;
OPS::Backends::OutToTCEBehaviour tceBehaviourGenerator(str);
tceBehaviourGenerator.printOperation(addSub, "OPADDINT");
```

```
[23]: str.str()
```

```
[23]: "#include "OSAL.hh"
OPERATION(OPADDINT)
TRIGGER
int a = INT(1);
int b = INT(2);

  int z;
  z = a + b;
IO(3) = static_cast<int>(z)

return true;
END_TRIGGER;
END_OPERATION(OPADDINT)
"
```

Рис. 5. Пример запуска генерации кода для эмулятора TCE

Подобные примеры позволяют расширить документацию разрабатываемой системы, сделать исходный код более самодокументируемым, дополнить модульные и интеграционные тесты системы более удобными сценариями использования её функций. Это также облегчает процесс экспериментирования с

модулями системы, построения цепочек преобразований и их тестирования.

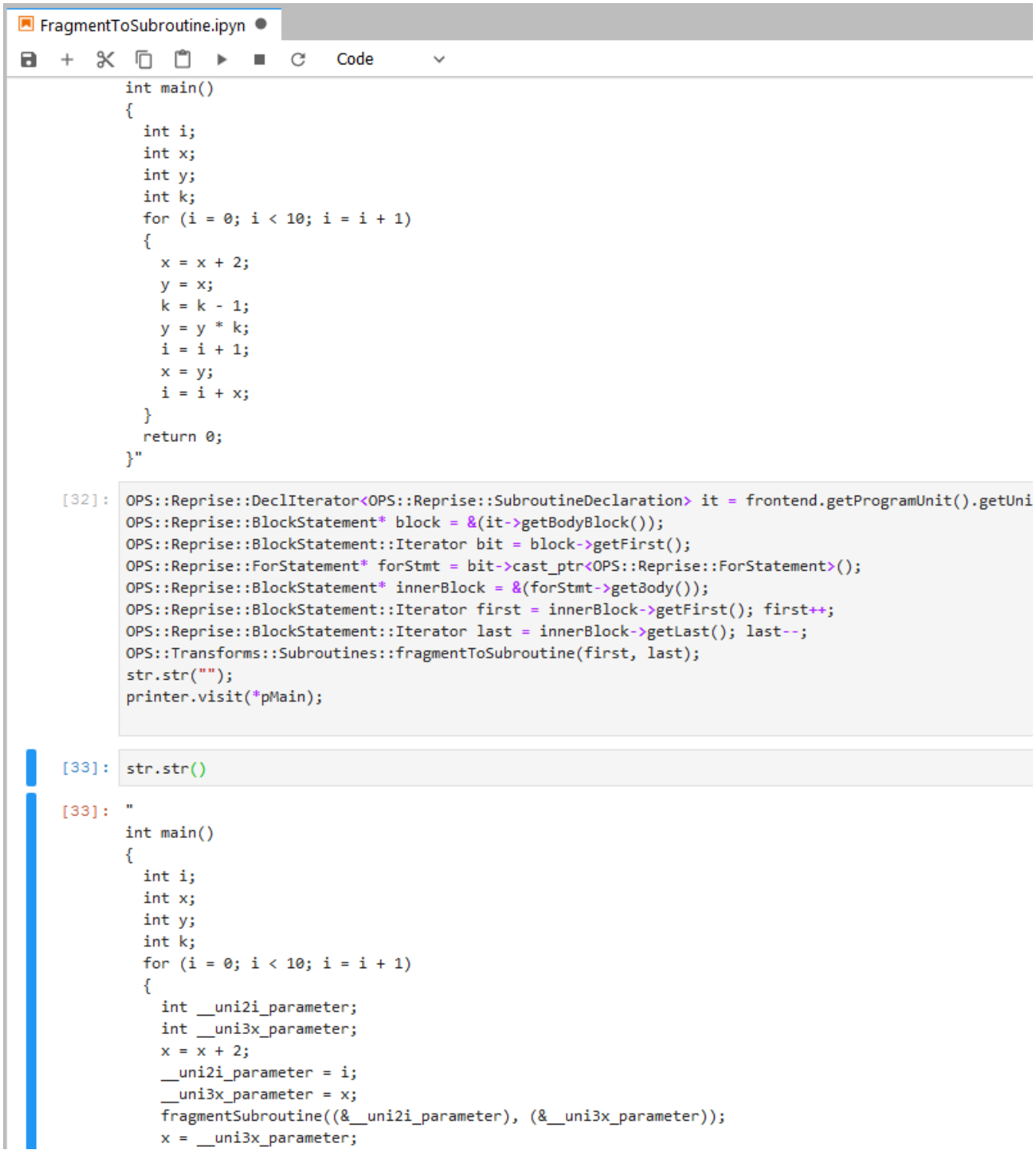


Рис. 6. Пример запуска преобразования «фрагмент в подпрограмму»

РЕЗУЛЬТАТЫ

Для решения поставленных задач в ОРС были реализованы следующие нововведения:

— набор контейнеров для надежной и повторяемой сборки самой ОРС, ее внешних зависимостей, включая компиляторы Clang/LLVM различных версий, компилятора на ее основе, запуска этого компилятора на тестовых программах;

— дополнительные сервисы для симуляции выполнения сгенерированных программ на ПЛИС, тестирования производительности преобразованных программ на языке С по принципу «черного ящика», аналогичные ранее разработанной системе тестирования из [6];

— процесс работы компилятора разбит на этапы, которыми можно управлять:

- выбор участков программы для выполнения на ПЛИС может быть автоматическим, или пользователь может выделить нужные фрагменты псевдокомментариями;
- компилятор принимает на вход набор параметров, описывающих необходимые действия;
- возможные действия включают экспорт промежуточных результатов для анализа и визуализации, генерацию выходной программы и т. п.;

— для нескольких выбранных функций компилятора и ОРС созданы соответствующие представления для визуализации данных в среде разработки, например, для статического профилировщика [7] и функции построения обобщенных конвейеров [8].

На рис. 7 показан набор основных модулей ОРС, которые реализуют плагины системы, независимые друг от друга. Это позволяет предоставлять программисту доступ только к выбранным функциям, упростить и ускорить сборку системы и ее развертывание.

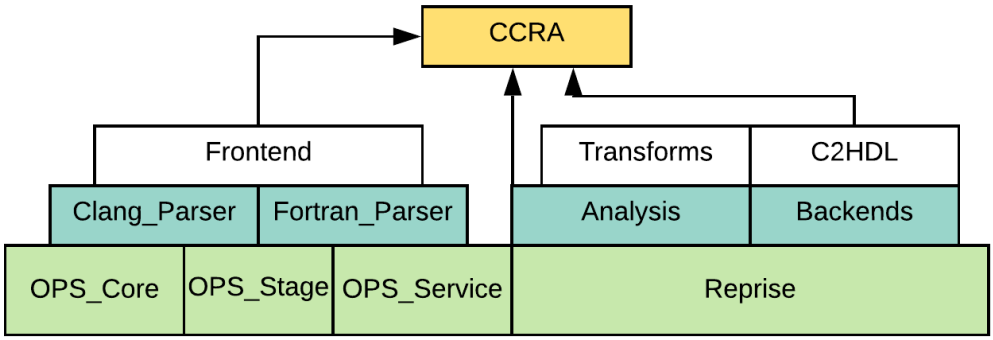


Рис. 7. Набор модульных компонент OPS, включая компилятор для ПЛИС (CCRA)

Дальнейшая работа ведется в следующих направлениях:

— поддержка работы с кодом компилятора в облачной IDE, то есть обеспечение удобства разработки самого компилятора;

— расширение функциональности специализированной облачной IDE, позволяющей использовать разрабатываемый компилятор;

— создание библиотеки примеров разного уровня сложности, документирующих использование разрабатываемых преобразований программ и позволяющих составлять из них более сложные модули.

Для начала знакомства с исходным кодом OPC и компилятора теперь достаточно открыть URL в браузере, после чего будет создано новое рабочее место с полностью сконфигурированным окружением для разработки. Это особенно облегчает непосредственное знакомство с исходным кодом системы и уменьшает время на первоначальную настройку при обучении.

ЗАКЛЮЧЕНИЕ

На основе современных облачных интегрированных сред разработки программ возможно создание специализированных инструментов, позволяющих облегчить разработку, использование и внедрение в процесс обучения сложных программных систем, включая оптимизирующие компиляторы для реконфигурируемых архитектур (ПЛИС). При этом интерактивные среды выполнения программ в браузере, ранее поддерживавшие исключительно интерпретируемые

языки, теперь возможно использовать для работы с системой значительного размера, написанной на языке C/C++.

Для того чтобы разработать функциональную веб-среду разработки на основе переносимого компилятора, требуется решить ряд задач по улучшению модульности кода, управлению внешними зависимостями системы, улучшению ее переносимости на другие платформы и предоставлению более универсальных программных интерфейсов для использования функций системы. Выполнение этих требований позволяет упростить как процесс разработки всей системы, так и использование системы, в том числе, для обучения программированию, демонстрации ключевых возможностей и визуализации результатов для пользователя web-IDE в диалоговом режиме.

На основе разработанной многопользовательской среды разработки и интерактивного выполнения программ возможно построение библиотеки примеров программ, применение которой облегчает все этапы разработки и использования сложной системы.

Благодарности

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект № 18-37-00179.

СПИСОК ЛИТЕРАТУРЫ

1. Оптимизирующая распараллеливающая система URL: www.ops.rsu.ru (дата обращения: 25.07.19).

2. *Штейнберг Б.Я., Аллазов А.Н., Алымова Е.В., Баглий А.П., Гуда С.А., Дубров Д.В., Кравченко Е.Н., Морылев Р.И., Рошаль А.С., Юрушкин М.В., Штейнберг Р.Б.* Web-ориентированный автоматический распараллеливатель программ // Параллельные вычислительные технологии (ПАВТ'2014). Труды международной научной конференции. Ростов-на-Дону: 1–3 апреля 2014.

3. *Алымова Е.В., Кравченко Е.Н., Морылев Р.И., Юрушкин М.В., Штейнберг Б.Я.* Распараллеливание и оптимизация программ с помощью Web-ускорителя OPC // Научный сервис в сети Интернет: поиск новых решений. Труды XIV Международной суперкомпьютерной конференции (17–22 сентября 2012 г., г. Новороссийск). М.: Изд-во МГУ, 2012.

4. *Steinberg B.Y., Bugliy A.P., Dubrov D.V., Mikhailuts Y V., Steinberg O.B., Steinberg R.B.* A Project of Compiler for a Processor with Programmable Accelerator // Procedia Computer Science. 2016. No 101. P. 435–438.

5. Localhost is Killing Software Delivery // Codenvy blog URL: https://blog.codenvy.com/localhost-is-killing-software-delivery-8c93cd49328 (дата обращения: 20.11.19).

6. *Штейнберг Б.Я., Алымова Е.В., Баглий А.П., Морылев Р.И., Нис З.Я., Петренко В.В., Штейнберг Р.Б.* Автоматизация тестирования элементов высокопроизводительного программного комплекса // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность. Труды Всероссийской суперкомпьютерной конференции (21–26 сентября 2009 г., г. Новороссийск). М.: МГУ им. М.В. Ломоносова, 2009. С. 287–292.

7. *Полуян С.В.* Профилирование и его применение в диалоговом оптимизирующем распараллеливателе // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи: Труды Международной суперкомпьютерной конференции (20-25 сентября 2010г., г. Новороссийск). М.: Изд-во МГУ, С. 652–653.

8. *Баглий А.П., Дубров Д.В., Штейнберг Б.Я., Штейнберг Р.Б.* Повторное использование ресурсов при конвейерных вычислениях // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18–23 сентября 2017 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2017. С. 43–46.

9. *Kluyver T et al*. Jupyter Notebooks – a publishing format for reproducible computational workflows // Positioning and Power in Academic Publishing: Players, Agents and Agendas, IOS Press Ebooks. P. 87–90.

10. Xeus-cling на Github. URL: https://github.com/QuantStack/xeus-cling (дата обращения: 20.11.19)

WEB BASED SYSTEM FOR PROGRAM ANALYSIS AND TRANSFORMATION IN OPTIMIZING PARALLELIZING SYSTEM

A. P. Bagly

Institute of mathematics, mechanics and computer science, Southern federal university, Rostov-on-Don

taccessviolation@gmail.com

Abstract

Experience of designing different variants for web-based development environment (IDE) for Optimizing parallelizing system and compiler for reconfigurable architecture is described. Designed system is based on existing tools and frameworks such as Jupyter Notebook and Eclipse Che. Set of requirements for Optimizing parallelizing system components is developed to make it possible to integrate them into web-based development environment accessible through the Internet. Designing portable environment for compiler development, compiler technology demonstration and teaching parallel program development is also described. Examples of performing newly developed program transformations are shown to be used during program optimizations for FPGA inside the designed web environment. Means of program transformation visualization are described for use with Jupyter Notebook. The work shown demonstrates possibility to organize remote access to library of instruments and tools for program optimizations currently under development that would be convenient for application developers.

Keywords*: integrated environment, parallelizing compiler, program transformations, FPGA, containerization, interactive notebook, cloud computing*

REFERENCES

1. Optimizing parallelizing system. URL: http://www.ops.rsu.ru/.

2. *Shteinberg B.Ya., Allazov A.N., Alymova E.V., Bagly A.P., Guda S.A., Dubrov D.V., Kravchenko E.N., Morylev R.I., Roshal A.S., Iurushkin M.V., Shteinberg R.B.* Web-orientirovannyi avtomaticheskii rasparallelivatel programm // Parallelnye vychislitelnye tekhnologii (PAVT'2014), Trudy mezhdunarodnoi nauchnoi konferentsii. Rostov-na-Donu: 1–3 aprelia 2014.

3. *Alymova E.V., Kravchenko E.N., Morylev R.I., Iurushkin M.V., Shteinberg B.Ya.* Rasparallelivanie i optimizatsiia programm s pomoshchiu Web-uskoritelia ORS // Nauchnyi servis v seti Internet: poisk novykh reshenii, Trudy XIV Mezhdunarodnoi superkompiuternoi konferentsii (17–22 sentiabria 2012 g/, g. Novorossiisk). M.: Izd-vo MGU, 2012.

4. *Steinberg B.Ya., Bugliy A.P., Dubrov D.V., Mikhailuts Y.V., Steinberg O.B., Steinberg R.B.* A Project of Compiler for a Processor with Programmable Accelerator // Procedia Computer Science. 2016. No 101. P. 435–438.

5. *Localhost is Killing Software Delivery* // Codenvy blog URL: https://blog.codenvy.com/localhost-is-killing-software-delivery-8c93cd49328

6. *Shteinberg B.Ya., Alymova E.V., Baglii A.P., Morylev R.I., Nis Z.Ia., Petrenko V.V., Shteinberg R.B.* Avtomatizatsiia testirovaniia elementov vysokoproizvoditelnogo programmnogo kompleksa // Nauchnyi servis v seti Internet: masshtabiruemost, parallelnost, effektivnost. Trudy Vserossiiskoi superkompiuternoi konferentsii (21–26 sentiabria 2009 g., g. Novorossiisk). M.: MGU im. M.V. Lomonosova, 2009. S. 287–292.

7. *Poluian S.V.* Profilirovanie i ego primenenie v dialogovom optimiziruiushchem rasparallelivatele // Nauchnyi servis v seti Internet: superkompiuternye tsentry i zadachi: Trudy Mezhdunarodnoi superkompiuternoi konferentsii (20–25 sentiabria 2010 g., g. Novorossiisk). M.: Izd-vo MGU, 2010. S. 652–653.

8. *Baglii A.P., Dubrov D.V., Shteinberg B.Ya., Shteinberg R.B.* Povtornoe ispolzovanie resursov pri konveiernykh vychisleniiakh // Nauchnyi servis v seti Internet: trudy XIX Vserossiiskoi nauchnoi konferentsii (18–23 sentiabria 2017 g., g. Novorossiisk). M.: IPM im. M.V. Keldysha, 2017. S. 43–46.

9. *Kluyver T. et al.* Jupyter Notebooks – a publishing format for reproducible computational workflows // Positioning and Power in Academic Publishing: Players, Agents and Agendas. IOS Press Ebooks, P. 87–90.

10. Xeus-cling na Github. URL: https://github.com/QuantStack/xeus-cling

СВЕДЕНИЯ ОБ АВТОРЕ

БАГЛИЙ Антон Павлович – ассистент Института математики, механики и компьютерных наук Южного федерального университета, специалист в области системного программирования и разработки компиляторов.

Anton Pavlovich BAGLY – teaching assistant at Institute of mathematics, mechanics and computer science, Southern federal university, a specialist in system programming and compiler development.

email: taccessviolation@gmail.com

Материал поступил в редакцию 16 ноября 2019 года

УДК 004.432

ИСПОЛЬЗОВАНИЕ DVM-СИСТЕМЫ ПРИ РАЗРАБОТКЕ ПРОГРАММЫ ДЛЯ РАСЧЕТОВ ЗАДАЧИ РАДИАЦИОННОЙ МАГНИТНОЙ ГАЗОДИНАМИКИ И ИССЛЕДОВАНИЯ ДИНАМИКИ ПЛАЗМЫ В КАНАЛЕ КСПУ

В. А. Бахтин [1,2], Д. А. Захаров [1], А. Н. Козлов [1,2], В. С. Коновалов [1]

[1] *Институт прикладной математики им. М.В. Келдыша Российской академии наук, г. Москва;*
[2] *Московский государственный университет им. М.В. Ломоносова, г. Москва*

bakhtin@keldysh.ru, s123-93@mail.ru, andrey-n-kozlov@mail.ru, v.s.konovalov@yandex.ru

Аннотация

DVM-система предназначена для разработки параллельных программ научно-технических расчетов на языках C-DVMH и Fortran-DVMH. Эти языки используют единую DVMH-модель параллельного программирования и являются расширением стандартных языков Си и Фортран спецификациями параллелизма, оформленными в виде директив для компилятора. DVMH-модель позволяет создавать эффективные параллельные программы для гетерогенных вычислительных кластеров, в узлах которых в качестве вычислительных устройств наряду с универсальными многоядерными процессорами могут использоваться ускорители, графические процессоры или сопроцессоры Intel Xeon Phi. В статье описан опыт успешного применения DVM-системы для разработки параллельного программного кода для расчетов задачи радиационной магнитной газодинамики и исследования динамики плазмы в канале КСПУ.

Ключевые слова*: автоматизация разработки параллельных программ, DVM-система, плазменный ускоритель, радиационная магнитная газодинамика.*

ВВЕДЕНИЕ

Современный уровень экспериментальных и численных исследований позволяет одновременно определять локальные значения макроскопических параметров плазмы и характеристики излучения. Это открывает новые возможности для проведения комплексных исследований, валидации моделей и сближения результатов расчетов с возможностями экспериментальных исследований. Этим обусловлена актуальность разработки численных моделей переноса излучения в квазистационарных плазменных ускорителях (КСПУ), рассматриваемых в качестве инжекторов для термоядерных установок [1]. Исследование течений ионизующегося газа и плазмы соответственно для первой и второй ступеней КСПУ проводится на основе разработанных эволюционных моделей радиационной магнитной газодинамики (РМГД) [2, 3], эффективное решение которых потребовало разработки параллельных программных кодов для расчетов на высокопроизводительных многопроцессорных вычислительных комплексах. Исследования потоков плазмы во второй ступени КСПУ на основе РМГД-модели позволили выявить условия, обеспечивающие генерацию потоков дейтерий-тритиевой или D-T плазмы с термоядерными параметрами при разрядных токах до 1 МА во второй ступени КСПУ [1].

МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ

Постановка задачи включает систему уравнений РМГД с учетом конечной проводимости среды, теплопроводности и излучения. При условии квазинейтральности $n_i = n_e = n$ в одножидкостном приближении $\mathbf{V}_i = \mathbf{V}_e = \mathbf{V}$ для полностью ионизованной плазмы имеем:

$$\frac{\partial \rho}{\partial t} + div(\rho \mathbf{V}) = 0, \qquad \rho \frac{d \mathbf{V}}{d t} + \nabla P = \frac{1}{c} \mathbf{j} \times \mathbf{H}, \qquad \frac{d}{d t} = \frac{\partial}{\partial t} + (\mathbf{V}, \nabla), \tag{1}$$

$$\frac{\partial}{\partial t}(\rho \varepsilon) + div(\rho \varepsilon \mathbf{V}) + P\, div \mathbf{V} = \frac{\mathbf{j}^2}{\sigma} - div \mathbf{q} - div \mathbf{W}\ , \qquad \varepsilon = 2 c_V T\ , \qquad \mathbf{q} = -\kappa \nabla T\ ,$$

$$\frac{\partial \mathbf{H}}{\partial t} = rot(\mathbf{V} \times \mathbf{H}) - c\, rot \frac{\mathbf{j}}{\sigma}, \qquad \mathbf{j} = \frac{c}{4\pi} rot \mathbf{H}\ ,$$

где $\rho = m_i\, n$ – плотность, $P = P_i + P_e = 2\, k_B\, n\, T$ – суммарное давление, $\mathbf{q}$ – тепловой поток, κ – теплопроводность, $\sigma = e^2 n_e / m_e \nu_e$ – электропроводность среды, $\mathbf{W}$ – поток энергии излучения, определяемый далее с помощью интегральных соотношений.

В данном случае исследования динамики потоков плазмы во второй ступени КСПУ проведены при наличии основной азимутальной компоненты магнитного поля в канале. Ускорение плазмы обеспечивает сила Ампера $\frac{1}{c}\mathbf{j}\times\mathbf{H}$, где $\mathbf{j}$ –ток в плазме, имеющий преимущественно радиальное направление.

В качестве единиц измерения взяты длина канала L, характерная концентрация или плотность газа на входе в канал ускорителя n_o ($\rho_o = m\, n_o$) и температура T_o. Характерная величина азимутального магнитного поля на входе в канал H_o определяется разрядным током в системе J_p так, что $H_o = 2\, J_p / c R_o$, где R_o – характерный радиус канала. С помощью этих величин формируются единицы: давления $P_o = {H_o}^2 / 4\pi$, скорости $V_o = H_o / \sqrt{4\pi\rho_o}$, времени $t_o = L / V_o$, тока в плазме $j_o = c\, H_o / 4\pi L$. В модели участвуют безразмерные параметры $\beta = 8\,\pi\, P_o / {H_o}^2$ и $\mathrm{Re}_m = \nu^{-1} = \sigma_o\, T^{3/2}$, а также безразмерные значения теплопроводности и потока $\mathbf{W}$.

Постановка МГД задачи включает традиционные граничные условия. На входе в канале при $z = 0$ плазма подается с известными значениями плотности $\rho(r) = f_1(r)$ и температуры $T(r) = f_2(r)$. Считаем, что разрядный ток не изменяется, и ток поступает в систему только через электроды, т. е. $j_z = 0$ при $z = 0$ или $r\, H_\varphi = r_o = const$, где $r_o = R_o / L$. Граничные условия на электродах $r = r_a(z)$ и $r = r_\kappa(z)$, образующих стенки канала, предполагают их эквипотенциальность ($E_\tau = 0$) и непроницаемость поверхности для плазмы ($V_n = 0$). На оси системы при $r = 0$ имеем $V_r = 0$, $V_\varphi = 0$, $H_\varphi = 0$.

Алгоритм численного решения уравнений (1) предполагает отображение расчетной области на единичный квадрат в плоскости (y, z) с помощью соотношения

$$r = (1 - y)\, r_\kappa(z) + y\, r_a(z). \qquad (2)$$

Для расчета гиперболической части МГД-уравнений используется разностная схема с коррекцией потоков. Магнитная вязкость и теплопроводность учитываются с помощью метода потоковой прогонки. Квазистационарные течения рассчитываются методом установления. Задача о переносе излучения решена в рамках разработанной 3D-модели.

Поскольку скорость распространения излучения существенно выше скоростей плазмодинамических процессов, интенсивность излучения вычисляется с помощью стационарного уравнения переноса излучения

$$\mathbf{\Omega}\cdot\nabla I_\nu(\mathbf{r},\mathbf{\Omega})=\eta_\nu(\mathbf{r})-\kappa_\nu(\mathbf{r})\cdot I_\nu(\mathbf{r},\mathbf{\Omega}), \qquad (3)$$

где $I_\nu(\mathbf{r},\mathbf{\Omega})$ – интенсивность излучения с частотой ν, распространяющегося в направлении телесного угла $\mathbf{\Omega}$, отвечает точке с координатой $\mathbf{r}$. В свою очередь плотность энергии излучения U и плотность потока энергии излучения $\mathbf{W}$ определяются через интенсивность излучения с помощью следующих соотношений:

$$U(\mathbf{r})=\frac{1}{c}\int_0^\infty\int_0^{4\pi} I_\nu(\mathbf{r},\mathbf{\Omega})\,d\,\mathbf{\Omega}\,d\nu, \qquad \mathbf{W}(\mathbf{r})=\int_0^\infty\int_0^{4\pi} I_\nu(\mathbf{r},\mathbf{\Omega})\,\mathbf{\Omega}\,d\,\mathbf{\Omega}\,d\nu. \qquad (4)$$

Коэффициент поглощения $\kappa_\nu(\mathbf{r})$ и излучательная способность $\eta_\nu(\mathbf{r})$ являются известными функциями температуры, плотности вещества и представляют собой суммы из трех частей, отвечающих а) поглощению и излучению в линиях; б) фотоионизации и фоторекомбинации и в) тормозному излучению и обратному тормозному поглощению.

В силу соотношений (4) задачу о переносе излучения необходимо решать в трехмерной постановке задачи.

В данном случае сетка для 3D-задачи о переносе излучения генерируется поворотом исходной сетки в плоскости переменных (z, r) на 360 градусов вокруг оси канала с заданным шагом. Для каждого узла или ячейки трехмерной сетки также строится дополнительная угловая сетка по азимутальному и полярному углам. Разбиение телесного угла на элементы угловой сетки производится методом, обеспечивающим равномерное распределение лучей по направлениям. Использовано 440 лучей в полном телесном угле $\mathbf{\Omega}=4\pi$. В соответствии с методом длинных характеристик для решения уравнения переноса излучения (3) осуществляется трассировка лучей с целью определить точки их пересечения с

гранями ячеек трехмерной координатной сетки и место падения лучей на стенки канала ускорителя и границы трехмерной расчетной области. Невидимые теневые области исключаются из расчета потока энергии излучения. Коэффициент поглощения $\kappa_\nu(\mathbf{r})$ и излучательная способность $\eta_\nu(\mathbf{r})$ вычисляются по среднему значению плотности и температуры в центре ячейки. Интенсивность вдоль лучей или характеристик, проходящих через любое количество однородных областей с известными коэффициентами κ_ν и η_ν, определяется в результате сшивки решений на границе однородных областей.

Детальная постановка задачи и используемые методы численного решения уравнения переноса излучения изложены в работах [2, 3].

РАЗРАБОТКА ПАРАЛЛЕЛЬНОЙ ВЕРСИИ ПРОГРАММЫ

Параллельный программный код для исследования высокоскоростных потоков плазмы в каналах квазистационарных плазменных ускорителей реализован с помощью DVM-системы [4, 5] на языке Fortran-DVMH.

Язык Fortran-DVMH представляет собой язык Фортран 95, расширенный спецификациями параллелизма, которые оформлены в виде специальных комментариев. Эти спецификации «невидимы» для стандартных компиляторов, что позволяет иметь один вариант программы для последовательного и параллельного выполнений.

Модель параллелизма DVM базируется на специальной форме параллелизма по данным: одна программа – множество потоков данных. В этой модели одна и та же программа выполняется на всех процессорах, но каждый процессор выполняет свое подмножество операторов в соответствии с распределением данных.

При разработке параллельной программы с помощью DVM-модели программист определяет массивы и витки циклов, которые должны быть распределены между процессорами. Распределенные массивы специфицируются директивами отображения данных (спецификации DISTRIBUTE и ALIGN), а циклы – директивами распределения вычислений (спецификация PARALLEL). Остальные переменные отображаются по одному экземпляру на каждый процессор (размноженные данные).

Распределение данных определяет множество локальных или собственных переменных для каждого процессора. Множество собственных переменных определяет правило собственных вычислений: процессор присваивает значения только собственным переменным.

При вычислении значения собственной переменной процессору могут потребоваться как значения собственных переменных, так и значения несобственных (удаленных) переменных. Для организации доступа к удаленным данным служат специальные директивы (SHADOW_RENEW, REMOTE_ACCESS, ACROSS и др.).

Основная сложность разработки параллельной программы для кластера – необходимость принятия глобальных решений по распределению данных и вычислений с учетом свойств всей программы, а затем выполнения кропотливой работы по модификации программы и ее отладке. Большой объем программного кода, многомодульность, многофункциональность затрудняют принятие решений по согласованному распределению данных и вычислений.

Для облегчения процесса разработки параллельной версии программы можно использовать инкрементальное распараллеливание. Идея этого метода заключается в том, что распараллеливанию подвергается не вся программа целиком, а ее части (области распараллеливания) — в них заводятся дополнительные экземпляры требуемых данных, производится распределение этих данных и соответствующих вычислений. Области могут выбираться на основе времен, полученных с помощью профилирования последовательной программы.

Для взаимодействия с теми частями программы, которые не подвергались распараллеливанию, используются операции копирования исходных (нераспределенных) данных в дополнительные (распределенные) данные и обратно. Таким образом, можно изолировать интересующие нас участки кода и распараллеливать каждую область отдельно от всей остальной программы. Это позволяет тестировать различные походы к распределению данных и вычислений внутри области, которые могут существенно изменять структуру хранения данных, но при этом любые изменения внутри области не будут требовать никаких модификаций кода вне области. Вместе с этим найти оптимальное распределение

данных и вычислений в небольшой локальной области значительно проще, чем для всей программы.

После распараллеливания отдельных областей выполняется оптимизация параллельной программы, направленная на минимизацию объема копируемых данных. Для этого несколько областей могут объединяться в одну, если для общей области можно найти эффективное общее распределение. Также возможно добавление в область менее времяемких фрагментов, если это уменьшит количество операций копирования данных.

Остановимся на особенностях распараллеливания данного программного комплекса с помощью инкрементального подхода. Наиболее времяемкие части программы были определены при помощи анализатора производительности, который является частью DVM-системы. По времени выполнения самой существенной частью программного комплекса является функция run_radiat, отвечающая расчету переноса излучения в 3D постановке задачи. Кроме того, значительное время требуется на выполнение итерационного цикла в той части программного комплекса, который отвечает расчетам осесимметричных течений на основе МГД-модели. Эти фрагменты программы и были выбраны в качестве областей распараллеливания.

Опишем процесс распараллеливания основного итерационного цикла и функции run_radiat.

Итерационный цикл представлял собой обычный цикл с фиксированным количеством итераций. Внутренние циклы были описаны при помощи меток (рис. 1):

```
          do 43 L = 1,NZ
             do 43 M = 1,NR
                HFI(M,L) = HRFI(M,L) / RAD(M,L)
43              continue
```

Рис. 1. Фрагмент исходной программы

Все внутренние циклы для удобства были переписаны без использования меток, а также была произведена замена всех использующихся в них массивов (рис. 2). Отказ от использования меток был чисто техническим преобразованием

с целью повышения уровня читаемости программы и облегчения распараллеливания.

```
do L = 1,NZ
  do M = 1,NR
    new_HFI(M,L)=new_HRFI(M,L)/new_RAD(M,L)
  end do
end do
```

Рис. 2. Фрагмент программы после преобразования

Замена массивов производилась как подготовка к дальнейшему распределению данных массивов в изоляции от остальной части программы. Так как итерационный цикл — лишь некоторая часть всего комплекса, было необходимо удостовериться, что распределение массивов не потребует вносить никаких изменений в остальную часть программы. Всего было заменено около 45 массивов. Все эти массивы, как оригинальные, так и копии, были заменены на динамические массивы, и их выделение происходило перед самым началом итерационного цикла. Оригинальные массивы в этот момент наоборот уничтожались, чтобы не использовать лишнюю память (рис. 3):

```
allocate(new_HRFI)
new_HRFI = HRFI
deallocate(HRFI)
<итерационный цикл>
allocate(HRFI)
HRFI = new_HRFI
deallocate(new_HRFI)
```

Рис. 3. Работа с копиями массивов

Указанные выше переходы от одного массива к другому ставились перед итерационным циклом и после него в случае, если данные, имевшиеся в этих массивах, использовались соответственно внутри итерационного цикла или после него.

Большая часть циклов имела достаточно простой характер обращений – либо к соответствующим для витка индексам элементов, либо к ближайшим соседям (рис. 4):

```
do L = 2,NZM1
  do M = 2,NRM1
          FGB = (GAM - 1.) / BB * new_RDY(L) / new_TEM(M,L)
          DRHDY = (new_HRFI(M + 1,L) - new_HRFI(M - 1,L)) / DY2
          DRHDZ = (new_HRFI(M,L + 1) - new_HRFI(M,L - 1)) / DZ2
          ! more code
       end do
    end do
```

Рис. 4. Типовой цикл

Для таких массивов, как new_HRFI из примера выше, при распределении были указаны соответствующие теневые грани. Механизм теневых граней, реализованный в DVM-системе, позволяет расширить локальную часть массива, которая будет отображена на процессор, слева и справа на ширину граней импортируемых данных. Если перед выполнением цикла скопировать грани импортируемых данных в соответствующие теневые грани, то в процессе выполнения цикла доступ к удаленным данным не потребуется. Максимальная ширина теневых граней может быть задана при помощи спецификации SHADOW.

Подавляющее большинство циклов программы не было тесновложенными, что не позволило провести их распределение сразу по двум измерениям. Провести распределение по какому-либо одному распределению также было нельзя, так как присутствовали циклы с прямыми зависимостями как по одному, так и по другому измерению (рис. 5):

```
do L = 2,NZM1
  ! more code
  do N = 2,NRM1
    M = NR - N + 1
    WL(M)=((1.-C(M)/A(M)*BET(M))*(B(M)*WL(M+1)-F(M))+C(M)*GAMM(M))/ A(M)
    PROR(M) = (A(M) * GAMM(M) + BET(M) * (F(M) - B(M) * WL(M + 1)) / A(M)
  end do
  !more code
end do
```

Рис. 5. Цикл с зависимостями

Аналогичные циклы были и по другому измерению. Значительное число циклов также имело существенный объем обращений к потенциально удаленным данным по одному из измерений (рис. 6):

```
      do L = 2,NZM1
         ! more code
         PPL = VR1 * new_RAD(1,L) * new_RDY(L) * new_PLT(1,L)
         !more code
      end do
```

Рис. 6. Потенциально удаленные данные

Для решения этих проблем был использован механизм шаблонов (спецификация TEMPLATE). Шаблон определяет «фиктивный» массив, который может быть распределен между процессорами и применяется для отображения данных и/или витков циклов. Элементы шаблона не требуют дополнительной памяти, они лишь указывают на процессор, на который должны быть распределены элементы выровненных массивов и/или вычисления. Для данного фрагмента программы было решено использовать сразу два шаблона для распределения данных – по одному на каждое измерение. В одном случае получалось, что каждый процесс получал часть «строк» двумерного массива и работал с ними. В другом – каждый процесс получал часть «столбцов» двумерного массива и обрабатывал их. Для каждого цикла выбиралось измерение без зависимостей и без обращений к удаленным данным, и распараллеливание производилось по

нему. Созданные ранее копии массивов также распределялись по тем измерениям, по которым распараллеливался цикл, в котором они использовались. Если массив использовался в циклах, среди которых были параллельные циклы как по одному, так и по другому измерению, для него создавалось две копии сразу. Между циклами, распараллеленными по разным измерениям, при необходимости производилось переключение между шаблонами (рис. 7):

```
!DVM$ PARALLEL(L) ON new_PLT(*,L), PRIVATE(M)
      do L = 2,NZ
        do M = 1,NR
          ! more code
        end do
      end do
      new2_PLT = new_PLT
!DVM$ PARALLEL (M) ON new2_PLT(M,*), PRIVATE(L)
      do M = 2,NR
        do L = 1,NZ
          ! more code
        end do
      end do
```

Рис. 7. Переключение между шаблонами

Серьезных переключений в программе оказалось лишь два – перевод 8 массивов с распределения по второму измерению на распределение по первому, и перевод этих же 8 массивов назад. Еще один раз в программе потребовался аналогичный перевод туда и обратно, но уже лишь для одного массива. По созданным шаблонам было распределено около 30 массивов, из которых около 20 имело копии для обоих шаблонов распределения. Также по ним было распараллелено более 50 разных циклов.

В функции run_radiat было проведено распараллеливание 5 циклов. Эти циклы использовали разнообразные структуры данных и массивы уникальных форматов (рис. 8):

```
do inode = 1,mesh3d%NNodes
         den3d%fval(inode) = 0.e0_r8p
         tem3d%fval(inode) = 0.e0_r8p
         do i = 1,valsInterp%ielem(inode)%nSrc
            n1  = valsInterp%ielem(inode)%iSrc(i)
            eps = valsInterp%ielem(inode)%wSrc(i)
            den3d%fval(inode) = den3d%fval(inode) + den2d%fval(n1)*eps
            tem3d%fval(inode) = tem3d%fval(inode) + tem2d%fval(n1)*eps
         end do
      end do
```

Рис. 8. Использование сложных структур данных

Так как основные массивы в этих циклах имели уникальную структуру, которые нельзя было хорошо соотнести друг с другом, для всех 5 циклов было создано 4 разных шаблона распределения (2 из 5 циклов удалось распараллелить, использовав один и тот же шаблон). Все распределяемые массивы были заменены на обычные массивы стандартных типов, остальные массивы и структуры данных были оставлены в изначальном виде. Итоговый вариант цикла из примера выше после преобразований и распараллеливания выглядел так (рис. 9):

```
!DVM$ PARALLEL(inode) ON new_den3d(inode),PRIVATE(n1,eps,i)
do inode = 1,m3d
     new_den3d(inode) = 0.e0_r8pч
     new_tem3d(inode) = 0.e0_r8p
     do i = 1,valsInterp%ielem(inode)%nSrc
        n1  = valsInterp%ielem(inode)%iSrc(i)
        eps = valsInterp%ielem(inode)%wSrc(i)
        new_den3d(inode) = new_den3d(inode) + new2_den2d(n1)*eps
        new_tem3d(inode) = new_tem3d(inode) + new2_tem2d(n1)*eps
     end do
end do
```

Рис. 9. Цикл после преобразований

Главной сложностью распараллеливания этой части программы была косвенная адресация, выделенная в примерах выше. Ее наличие уже означает, что любой виток цикла может обращаться к любому элементу адресуемого косвенно массива. Ее форма в данной задаче такова, что каждый виток может обращаться к произвольному числу разных элементов адресуемого массива, возможно даже ко всем. Более того, ситуация, когда каждый процесс будет запрашивать практически все элементы косвенно адресуемого массива (то есть значения n1 для каждого набора витков, выделяемых процессу, проходят через практически все возможные элементы массивов den2d%fval и tem2d%fval) для данной задачи является нормой. Поэтому было решено проводить запись данных в распределенные массивы, а если в последующих циклах этот массив адресуется косвенно – производилось его размножение по всем процессам. Учитывая то, что каждый процесс использует почти каждый элемент массива, лишние накладные расходы на совершение полного копирования являются незначительными. После цикла, указанного выше, например, происходило размножение в виде (рис. 10):

```
new2_den3d = new_den3d

new2_tem3d = new_tem3d
```

Рис. 10. Копирование данных после выполнения цикла

Префикс new_ указывал на распределенные массивы, а new2_ – на размноженные. Здесь следует отметить наличие в языке Fortran-DVMH удобных средств для копирования распределенных массивов (секций распределенных массивов), которые позволяют обеспечить совмещение обменов данными с вычислениями. В общем случае операторы присваивания вида *new2_den3d = new_den3d* преобразуются компилятором с языка Fortran-DVMH в соответствующие операторы асинхронного копирования распределенных массивов, которые реализованы в системе поддержки параллельного выполнения DVMH-программ. При использовании низкоуровневых библиотек типа MPI, SHMEM сложность реализации операций копирования данных до/после цикла, перехода от одного массива к массива-копии может стать дополнительной проблемой при распараллеливании программы.

Отдельной сложностью для распараллеливания было наличие редукционной операции суммирования в цикле с косвенной адресацией (рис. 11):

```
do idir = 1,NDirs
    do iEn = 1,NEnGroups
        do isegm = 1,NPoints
            inode = pTrace%rayTree(idir)%segm(isegm)%inode
            !more code
            cUden%fval(inode) = cUden%fval(inode) + cU_ptX
        end do
    end do
end do
```

Рис. 11. Цикл с редукцией

В данном случае получалось, что любой виток цикла мог производить запись в любой элемент массива, которая еще и суммировалась. Для решения этого вопроса был создан специальный распределенный массив, позволявший каждому витку проводить суммирование в своем собственном «столбце». После этого уже производилось суммирование по последнему измерению (рис. 12):

```
do idir = 1,NDirs
   !DVM$ PARALLEL(iEn) ON new_cuden3d(*,iEn)
   do iEn = 1,NEnGroups
      do isegm = 1,NPoints
         inode = pTrace%rayTree(idir)%segm(isegm)%inode
         !more code
         new_cuden3d(inode,iEn) = new_cuden3d(inode,iEn) + cU_ptX
      end do
   end do
end do
!DVM$ PARALLEL(iEn) ON uni_cuden3d(*,iEn), PRIVATE(inode),
!DVM$* REDUCTION(SUM(new2_cuden3d))
do iEn = 1,NEnGroups
   do inode = 1,m3d
      new2_cuden3d(inode)=new2_cuden3d(inode)+ new_cuden3d(inode,iEn)
```

```
    end do
  end do
```

Рис. 12. Реализация редукции

В итоге каждый процесс получал корректную копию редукционного массива.

По итогам распараллеливания была получена версия программы, которую возможно выполнять на кластерах, в узлах которых установлены многоядерные процессоры.

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНОЙ ВЕРСИИ ПРОГРАММЫ

Расчеты с помощью параллельной версии программы проводились на двух многопроцессорных высокопроизводительных вычислительных комплексах К-100 (ИПМ РАН) и MVS1P5 (МСЦ РАН). На рис. 13 и 14 представлены графики изменения времени выполнения программы в секундах в зависимости от числа используемых ядер данных вычислительных комплексов. Речь идет о расчетах на один временной шаг, определяемый условием Куранта в решении эволюционной задачи на основе РМГД-модели. Кривые 1 на рисунках отвечают расчетам с помощью метода длинных характеристик, который требует наибольших затрат вычислительных ресурсов в случае решения полностью трехмерной задачи и расчету интегральных характеристик во всех узлах трехмерной координатной сетки. Кривые 2 и 3 соответствуют расчетам интегральных характеристик излучения в одной плоскости переменных (r,z) с учетом аксиальной симметрии течения. Наряду с методом длинных характеристик был реализован метод коротких характеристик, которому отвечают кривые 2. Метод коротких характеристик приводит к численной диффузии при расчете поля излучения, и он требует больше времени для расчета по сравнению с методом длинных характеристик, который представлен кривыми 3 на рисунках при условии вычисления интегральных характеристик излучения в одной плоскости переменных (r,z) с учетом аксиальной симметрии течения.

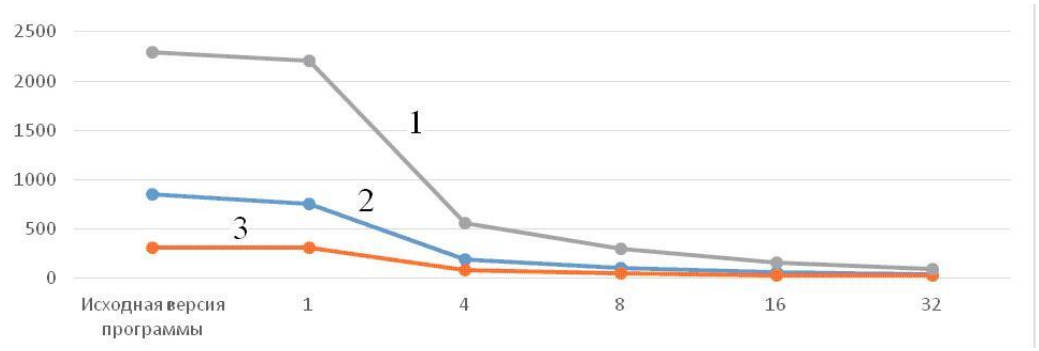


Рис. 13. Время выполнения программы в секундах на различном числе ядер К-100

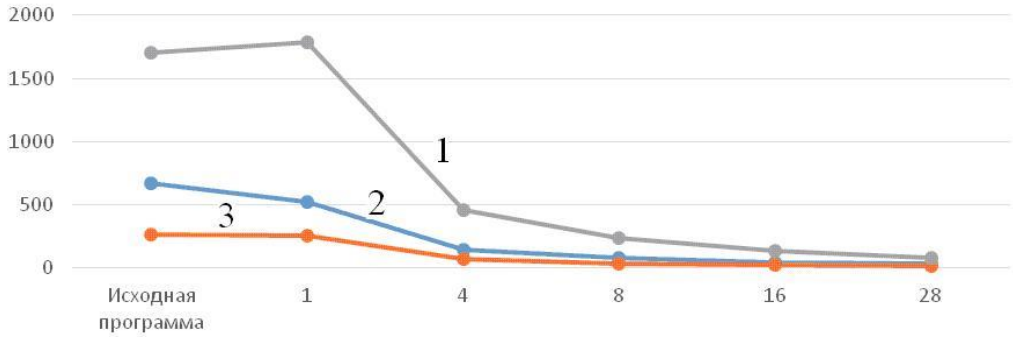


Рис. 14. Время выполнения программы в секундах на различном числе ядер MVS1P5

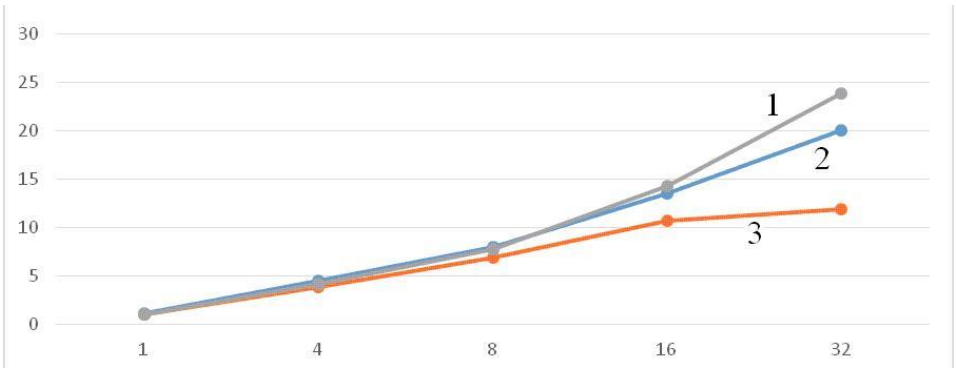


Рис. 15. Ускорение процесса выполнения программы относительно исходной версии на различном числе ядер К-100

Рис. 15 и 16 показывают, как изменяется ускорение процесса выполнения параллельной версии программы относительно исходной последовательной версии на различном числе ядер вычислительных комплексов соответственно в

ИПМ РАН и МСЦ РАН. Кривые 1, 2 и 3 на данных рисунках отвечают тем же условиям расчетов, указанным выше.

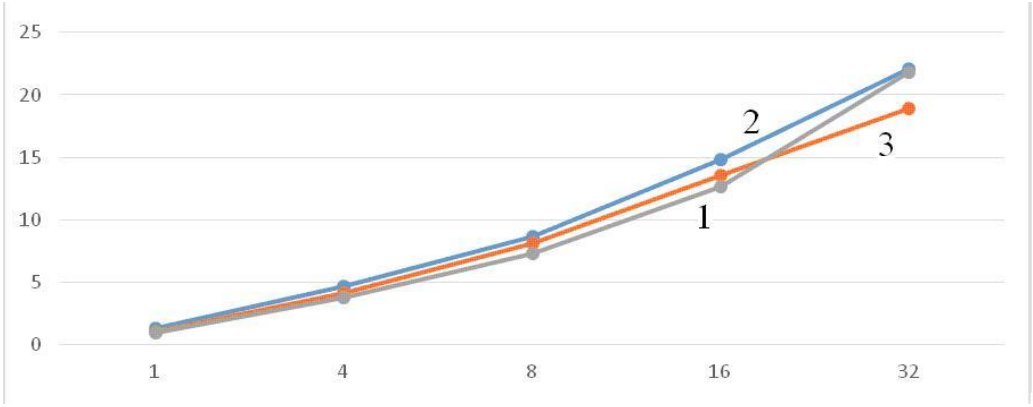


Рис. 16. Ускорение процесса выполнения программы относительно исходной версии на различном числе ядер MVS1P5

Более слабое ускорение на К-100 для кривой 3 на рис. 15 для 32 процессоров объясняется тем, что время работы программы уже очень мало, менее 30 секунд, и накладные расходы начинают занимать существенный процент времени работы программы.

Ускорение параллельной версии программы на одном процессоре можно объяснить за счет более оптимального характера обращений к памяти. Замена части структур, использующих указатели, на набор обычных массивов привела к уменьшению общего количества обращений к памяти. Это также позволило в некоторых случаях избавиться от косвенной адресации, что увеличило эффективность использования кэш-памяти. Получаемое ускорение существенно зависит от использующейся сетки, режима работы программы и характеристик памяти конкретной машины. Его может быть достаточно для того, чтобы покрыть все возникающие при распараллеливании накладные расходы и даже немного ускорить программу на одном процессоре.

ЗАКЛЮЧЕНИЕ

С помощью DVM-системы разработан параллельный программный код для расчетов задачи радиационной магнитной газодинамики и исследования динамики плазмы в канале КСПУ. При этом достигнуты следующие результаты по ускорению процесса выполнения программы относительно исходной версии:

до 24 раз на 32 процессорах вычислительного комплекса К-100 (ИПМ РАН) и до 22 раз на 28 процессорах вычислительной системы MVS1P5 (МСЦ РАН).

Многочисленные расчеты, выполненные с помощью параллельной версии программы, показали, что значение разрядного тока, необходимое для достижения термоядерных параметров потока плазмы из КСПУ и энергии ионов $\varepsilon_i = 30\ KeV$, увеличивается пропорционально размерам установки. Установлено, что уменьшение характерной концентрации плазмы n_o на входе в канал ускорителя позволяет существенно уменьшить значения разрядных токов. Определены значения разрядного тока в установке, обеспечивающие на выходе энергию ионов на уровне $30\ KeV$, который необходим для последующей реакции синтеза D-T плазмы в магнитных ловушках для удержания плазмы. Соответствующие разрядные токи являются вполне приемлемыми для существующих установок КСПУ и не приводят к оплавлению элементов конструкций.

СПИСОК ЛИТЕРАТУРЫ

1. *Kozlov A. N.* The study of plasma flows in accelerators with thermonuclear parameters // Plasma Physics and Controlled Fusion. 2017. V. 51. No. 11, Ar. 115004. P. 1–7.

2. *Kozlov A.N., Konovalov V.S.* Numerical study of the ionization process and radiation transport in the channel of plasma accelerator // Communications in Nonlinear Science and Numerical Simulation. 2017. V. 51. P. 169–179.

3. *Kozlov A.N., Konovalov V.S.* Radiation transport in the ionizing gas flow in the quasi-steady plasma accelerator // Journal of Physics: Conference Series. 2018. V. 946.

4. Язык C-DVMH. C-DVMH компилятор. Компиляция, выполнение и отладка CDVMH-программ. URL: http://dvm-system.org/static_data/docs/CDVMH-reference-ru.pdf

5. Язык Fortran-DVMH. Fortran-DVMH компилятор. Компиляция, выполнение и отладка DVMH-программ. URL: http://dvm-system.org/static_data/docs/FDVMH-user-guide-ru.pdf

THE USING OF DVM-SYSTEM FOR DEVELOPING OF A PROGRAM FOR CALCULATIONS OF THE PROBLEM OF RADIATION MAGNETIC GAS DYNAMICS AND RESEARCH OF PLASMA DYNAMICS IN THE QSPA CHANNEL

V. A. Bakhtin[1,2,*], D. A. Zakharov[1], A. N. Kozlov[1,2], V. S. Konovalov[1]

[1] *Keldysh Institute of Applied Mathematics*
[2] *Lomonosov Moscow State University*

bakhtin@keldysh.ru, s123-93@mail.ru, andrey-n-kozlov@mail.ru, v.s.konovalov@yandex.ru

Abstract

DVM-system is designed for the development of parallel programs of scientific and technical calculations in the C-DVMH and Fortran-DVMH languages. These languages use a single DVMH-model of parallel programming model and are an extension of the standard C and Fortran languages with parallelism specifications in the form of compiler directives. The DVMH model makes it possible to create efficient parallel programs for heterogeneous computing clusters, in the nodes of which accelerators, graphic processors or Intel Xeon Phi coprocessors can be used as computing devices along with universal multi-core processors. The article describes the experience of the successful using of DVM-system to develop a parallel software code for calculating the problem of radiation magnetic gas dynamics and for research of plasma dynamics in the QSPA channel.

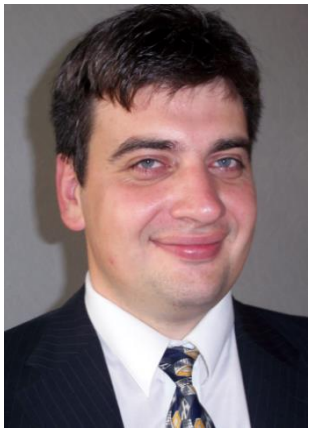
Keywords*: automation of development of parallel programs, DVM-system, plasma accelerator, radiation magnetic gas dynamics*

REFERENCES

1. *Kozlov A. N.* The study of plasma flows in accelerators with thermonuclear parameters // Plasma Physics and Controlled Fusion. 2017. V. 51. No. 11, Ar. 115004. P. 1–7.
2. *Kozlov A.N., Konovalov V.S.* Numerical study of the ionization process and radiation transport in the channel of plasma accelerator // Communications in Nonlinear Science and Numerical Simulation. 2017. V. 51. P. 169–179.

3. *Kozlov A.N., Konovalov V.S.* Radiation transport in the ionizing gas flow in the quasi-steady plasma accelerator // Journal of Physics: Conference Series. 2018. V. 946.
4. C-DVMH language, C-DVMH compiler, compilation, execution and debugging of DVMH programs. URL: http://dvm-system.org/static_data/docs/CDVMH-reference-en.pdf
5. Fortran DVMH language, Fortran DVMH compiler, compilation, execution and debugging of DVMH programs. URL: http://dvm-system.org/static_data/docs/FDVMH-user-guide-en.pdf

СВЕДЕНИЯ ОБ АВТОРАХ



БАХТИН Владимир Александрович – ведущий научный сотрудник ИПМ им. М.В. Келдыша РАН, доцент кафедры системного программирования факультета ВМК МГУ им. М.В. Ломоносова. Сфера научных интересов – математическое обеспечение, программные средства и системы для распределенных вычислений; параллельные алгоритмы; методы, средства и системы обработки данных большого объема.

Vladimir Aleksandrovich BAKHTIN – leading researcher of Keldysh Institute of Applied Mathematics, docent of the faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University. Research interests include mathematical software, software and systems for distributed computing; parallel algorithms; methods, tools and systems of large data processing.

email: bakhtin@keldysh.ru

ЗАХАРОВ Дмитрий Александрович – программист ИПМ им. М.В. Келдыша РАН. Сфера научных интересов – программные средства и системы для распределенных вычислений; параллельные алгоритмы; автоматизация параллельного программирования; распараллеливание программ, использующих неструктурные сетки.

Dmitry Aleksandrovich ZAKHAROV – programmer of Keldysh Institute of Applied Mathematics. Research interests include mathematical software, software and systems for distributed computing; parallel algorithms; automatization of parallel programming; parallelization of unstructured grid applications.

email: s123-93@mail.ru

КОЗЛОВ Андрей Николаевич – главный научный сотрудник ИПМ им. М.В. Келдыша РАН, профессор кафедры вычислительной механики Механико-математического факультета МГУ им. М.В. Ломоносова. Сфера научных интересов – механика жидкости, газа и плазмы; математическое моделирование; физика плазмы.

Andrey Nikolaevich KOZLOV – chief researcher of Keldysh Institute of Applied Mathematics, professor of the faculty of Mechanics and Mathematics of Lomonosov Moscow State University. Research interests include fluid, gas and plasma mechanics; mathematical modeling; plasma physics.

email: andrey-n-kozlov@mail.ru

КОНОВАЛОВ Вениамин Сергеевич – младший научный сотрудник ИПМ им. М.В. Келдыша РАН. Сфера научных интересов – математическое моделирование физических процессов в плазме, решение МГД уравнений, перенос изучения в плазме в многогрупповом приближении, спектры излучения плазмы, нестационарная и неравновесная поуровневая кинетика, квазистационарные плазменные ускорители, магнитные ловушки.

Veniamin Sergeevich KONOVALOV – junior researcher of Keldysh Institute of Applied Mathematics. Research interests include mathematical modeling of physical processes in plasma, solution of MHD equations, radiation transport, quasi-steady plasma accelerator, magnetic traps.

email: v.s.konovalov@yandex.ru

Материал поступил в редакцию 13 ноября 2019 года

УДК 519.6 + 519.2

ИССЛЕДОВАНИЕ УСТОЙЧИВОСТИ СОВМЕСТНОЙ МОДЕЛИ К ВОЗМУЩЕНИЮ НАЧАЛЬНЫХ ДАННЫХ

К. П. Беляев [1], Г. М. Михайлов [2], А. Н. Сальников [3], Н. П. Тучкова [4]

[1-4]Вычислительный центр им. А.А. Дородницына Федерального исследовательского центра «Информатика и управление» Российской академии наук (РАН), г. Москва

[1]Институт океанологии им. П.П. Ширшова РАН, г. Москва

[3]Московский государственный университет имени М.В. Ломоносова, факультет вычислительной математики и кибернетики, г. Москва

[1]kosbel55@gmail.com, [2]gmickail@ccas.ru, [3]salnikov@angel.cs.msu.ru, [4]natalia_tuchkova@mail.ru

Аннотация

Задача устойчивости рассматривается в терминах классического определения Ляпунова. Для этого задается множество начальных условий, состоящих их данных предварительных расчетов, и анализируется разброс траекторий, полученных в результате численного моделирования. Эта процедура реализована как серия ансамблевых экспериментов с совместной моделью MPI-ESM института метеорологии М. Планка (Германия). Для численного моделирования задавалась серия различных начальных значений полей характеристик, и модель интегрировалась, начиная с каждого из этих полей, на различные временные периоды. Изучались экстремальные характеристики уровня океана за период 30 лет. Строилось их статистическое распределение, оценивались параметры этого распределения, изучался статистический прогноз на 5 лет вперед. Показано, что статистический прогноз уровня соответствует расчетному прогнозу, полученному по модели. Изучалась локализация экстремальных значений уровня и проводился анализ этих результатов. Численные расчеты выполнялись на суперкомпьютере Ломоносов-2 Московского государственного университета имени М.В. Ломоносова.

***Ключевые слова**: нелинейные модели циркуляции, численные ансамблевые эксперименты, анализ устойчивости модельных траекторий*

ВВЕДЕНИЕ

Ансамблевые эксперименты со сложными нелинейными моделями – один из самых распространенных и относительно легко реализуемый метод исследования модельных траекторий, их поведения во времени и пространстве, а также изучения условий их устойчивости при достаточно длительном промежутке интегрирования. С другой стороны, эти численные эксперименты требуют использования большого объема вычислительных мощностей, компьютерного времени и памяти, решения задач визуализации результатов и многих сопутствующих проблем. В последние годы, благодаря значительному прогрессу в области вычислительных мощностей и численного моделирования, систем накопления и обработки больших данных, ансамблевые эксперименты становятся доступными многим исследовательским группам и отдельным пользователям, принадлежащим определенному научному сообществу. Это, в свою очередь, способствует дальнейшему развитию численного моделирования, возможностям анализа модельных данных, полученных результатов с последующим сравнением.

Сегодня имеется достаточно много исследовательских работ с современными моделями. Работы по численному моделированию климата ведутся различными международными группами, например, это модели GFDL [1], NEMO [2], MPI-ESM Макса Планка [3] и другие. Среди российских разработок наиболее известны в этой области модели ИВМ им. Г.И. Марчука РАН, например, [5, 6]. Многие результаты, признанные научной общественностью, представлены в бюллетене МГЭИК [7], где дан анализ физических характеристик и окружающей среды на основе модельных результатов.

Научный интерес к моделированию климата и численным моделям вызван не только их практической значимостью, но и тем, что эти модели значительно обогащают исследования в области нелинейных систем дифференциальных уравнений и способствуют пониманию природы моделируемых процессов. В частности, проблема устойчивости и чувствительности модели к начальным возмущениям очень интересна и требует серьезного анализа. Применительно к нашей задаче необходимо заметить, что система уравнений достаточно сложна, и аналитические методы не работают, за исключением нескольких простых случаев. Чтобы получить достаточно обоснованные результаты, необходимо обес-

печить ряд вычислительных экспериментов и провести углубленный анализ полученных результатов. Некоторые работы в этом направлении представлены в [8]. Однако эта область исследований настолько обширна, что, хотя моделирование климата началось с середины XX в., когда появились первые возможности и ресурсы ЭВМ, нельзя утверждать, что к настоящему времени не осталось направлений для фундаментального анализа.

В данной работе мы изучаем поведение основных физических характеристик за определенный временной период интегрирования по модели MPI-ESM, упомянутой выше [3]. В экспериментах был выбран набор начальных данных, состоящий из 50 различных полей, и наблюдались модельные характеристики в течение 10 и более лет интегрирования с этими данными. Известно, что модельные уравнения приводят к разбросу результатов вычисления из-за начального возмущения. Анализируя этот разброс, мы можем оценить стабильность модели, чтобы оценить ее статистические и аналитические свойства и сделать вывод о ее физических и математических особенностях. Некоторые результаты представлены ниже.

1. МОДЕЛЬ И ИСХОДНЫЕ ДАННЫЕ

В работе использована Модель земной системы института Метеорологии М. Планка, MPI-ESM [3], которая была включена экспертами в проект по взаимному сопоставлению климатических моделей (CMIP5) [4]. Основная ее конфигурация многократно была представлена в публикациях и показана на рис. 1.

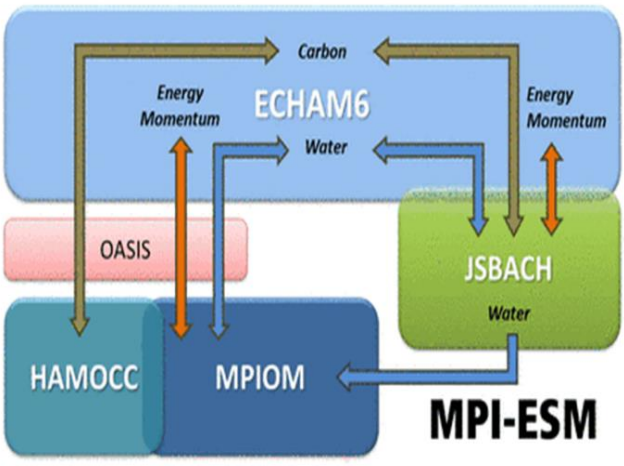


Рис. 1. Общая схема модели MPI-ESM

Модель включает различные блоки, а именно, океан (MPIOM), атмосферу (ECHAM6), углерод (HAMOCC), земной цикл водообмена (JSBACH) и другие. Блок OASIS управляет самой моделью. В нашей работе изменялись начальные данные только для блока MPIOM, хотя модель интегрировалась полностью.

Основные уравнения, определяющие этот блок, следующие:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}+fv=-\frac{\partial P}{g\rho_0\partial x}+\kappa\Delta u,$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}-fv=-\frac{\partial P}{g\rho_0\partial y}+\kappa\Delta v,$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0, \qquad (1)$$

$$\frac{\partial \theta}{\partial t}+u\frac{\partial \theta}{\partial x}+v\frac{\partial \theta}{\partial y}+w\frac{\partial \theta}{\partial z}=\mu\Delta\theta,$$

$$\frac{\partial S}{\partial t}+u\frac{\partial S}{\partial x}+v\frac{\partial S}{\partial y}+w\frac{\partial S}{\partial z}=\mu\Delta S,$$

$$\rho=\rho(\theta,S,P).$$

Система уравнений (1) определяет состояние океана в любой момент времени *t* в точке (*x*, *y*, *z*) пространства (сетки). Здесь используются стандартные обозначения, а именно: *u, v, w* – компоненты вектора скорости соответственно в северном, восточном и вертикальном (вниз) направлениях; ϑ, *S*, ρ – соответственно обозначают потенциальную температуру, соленость и плотность; *P* обозначает давление, ρ_0 – соответствующее среднее значение плотности по выбранному уровню *z*; *g* – ускорение силы тяжести; *f* – параметр Кориолиса, равный $2\Omega\sin\varphi$, где Ω – угловая скорость вращения Земли (7,2921×$10^{-5}$$с^{-1}$) вокруг оси; φ – географическая широта места; μ, *κ* – коэффициенты вязкости для скорости и температуры (солености) в уравнениях (1); Δ – стандартный 2D-оператор Лапласа. Эта система решается численно в 2*D*-сетке для выбранных уровней *z*. Граничные условия задаются параметрами атмосферы (поверхность моря) и берутся из модели атмосферы, которая моделирует силу ветра, тепловые потоки и

осадки (потоки пресной воды). Детали здесь опущены. Полностью описание модели приведено в [3].

В модели используется криволинейная 2*D*-сетка, и расстояния между точками различны от 40 км в восточном и северном направлении в Южном полушарии до 15 км в Полярной и Арктической зонах. От поверхности моря до дна задаются 40 уровней, и первые 15 уровней обеспечивают дискретизацию верхних 500 м. Другие детали конфигурации модели могут быть найдены в [9].

Были проведены серии ансамблевых экспериментов с различными сценариями. В частности, расчеты были проведены следующим образом: стартуя с 50 различных начальных условий из базы данных МГЭИК [7] (Межправительственная группа экспертов по изменению климата, МГЭИК, англ. Intergovernmental Panel on Climate Change, IPCC), модель интегрировалась последовательно в течение 10 лет. После интегрирования 50 различных полученных модельных полей записывались и анализировались. Вычислялись средние значения до и после интегрирования и аномалии относительно этих средних. Ниже представлены результаты расчетов и их анализ.

2. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ И АНАЛИЗ ПОЛУЧЕННЫХ ХАРАКТЕРИСТИК

По результатам моделирования были получены поля характеристик модели, которые изучены и представлены с помощью специальных графических средств, предназначенных для обработки netCDF, HDF и GRIB форматов (PanoplyWin https://www.giss.nasa.gov/tools/panoply/, GrADS http://cola.gmu.edu/grads/ и др.).

На рис. 2 приведена разность среднеширотной поверхностной температуры за 10 лет интегрирования. Подобное исследование выполнялось ранее в [10], наше несколько отличается по локализации представленных результатов. Поле поверхностной температуры воды (ТПО) усреднялось по долготе и представлено по широтной изменчивости от Южного (–90º) до Северного (90º) полюсов. Сама по себе изменчивость этих температур невелика, в максимуме составляет 0.2ºС и в минимуме –1.6ºС. Интересно заметить, что максимум изменчивости за 10 лет приходится на среднеширотные зоны, где температура заметно (более чем на 1.5ºС) уменьшается. В высоких широтах, как в Южном, так и Северном полушариях, она возрастает на 0.2 градуса.

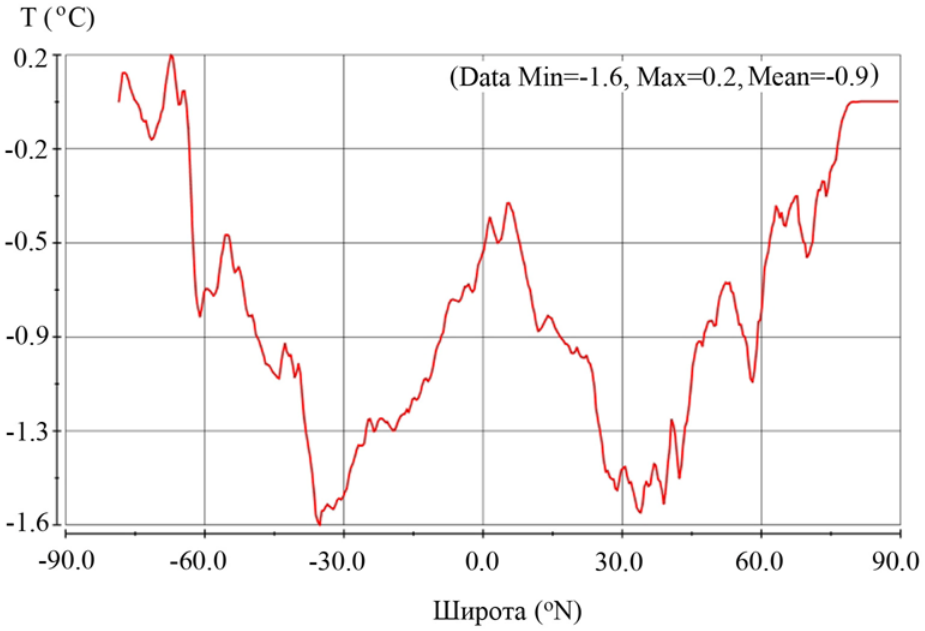


Рис. 2. Разница максимальных отклонений температуры от среднего на поверхности

Можно отметить, что в поведении уровня океана (рис. 3) заметна тенденция на увеличение, то есть уровень возрастает, причем максимум приходится на северные широты, где изменчивость уровня составляет около 0.3 м. Это соответствует наблюдаемым характеристикам поля уровня. Рост модельного уровня обусловлен внешними силами, связанными с атмосферой, которая в свою очередь зависит от значения карбона и других газов.

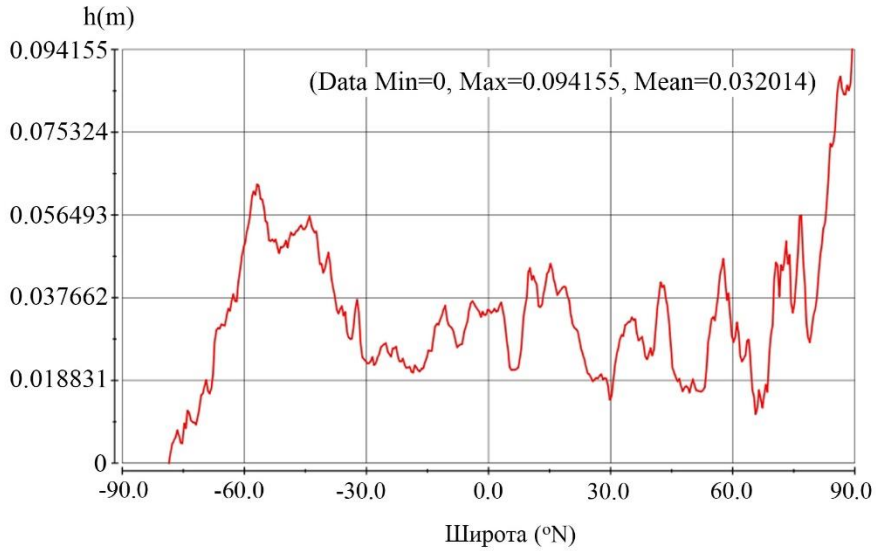


Рис. 3. Кривая разницы максимальных отклонений относительно среднего уровня океана до и после интегрирования

Отдельный интерес представляет рис. 4, где показано поле отклонений минимальных значений уровня океана от их средних за 10 лет. Величина (то есть разброс) отклонений за 10 лет заметно выросла и составляет в максимуме более метра. Отдельно выделяются полярные зоны и среднеширотные области в Тихом океане, где эти величины весьма значимы.

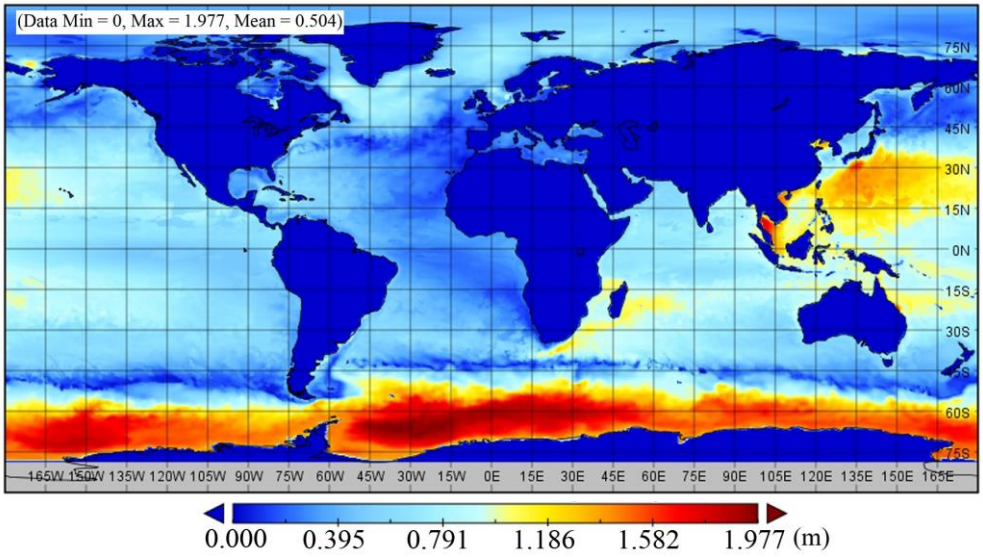


Рис. 4. Поле разницы минимальных отклонений относительно средних полей уровня океана до и после интегрирования

Рисунки 5 и 6 показывают характеристики прогностического уровня океана и его изменчивости на всем периоде интегрирования. На рис. 5 показан результат модельного расчета уровня океана на период 2000–2007 гг , а на рис. 6 – соответствующая разница значений уровня на этот же период для зоны Северного Моря и Российской зоны Арктики. Интересно отметить, что по прогнозу в Российской Арктике уровень будет в основном возрастать, причем довольно значимо, до 7 см, к востоку от Новой Земли и в районе Теребенской губы. Это хорошо согласуется с прогнозом роста поверхностной температуры воды (ТПО) и таяния льдов (см. далее рис. 8–10). В Северном море, наоборот, видно некоторое понижение уровня, около 5–6 см. Общее изменение разнонаправленное, от – 0.104 м до 0.076 м.

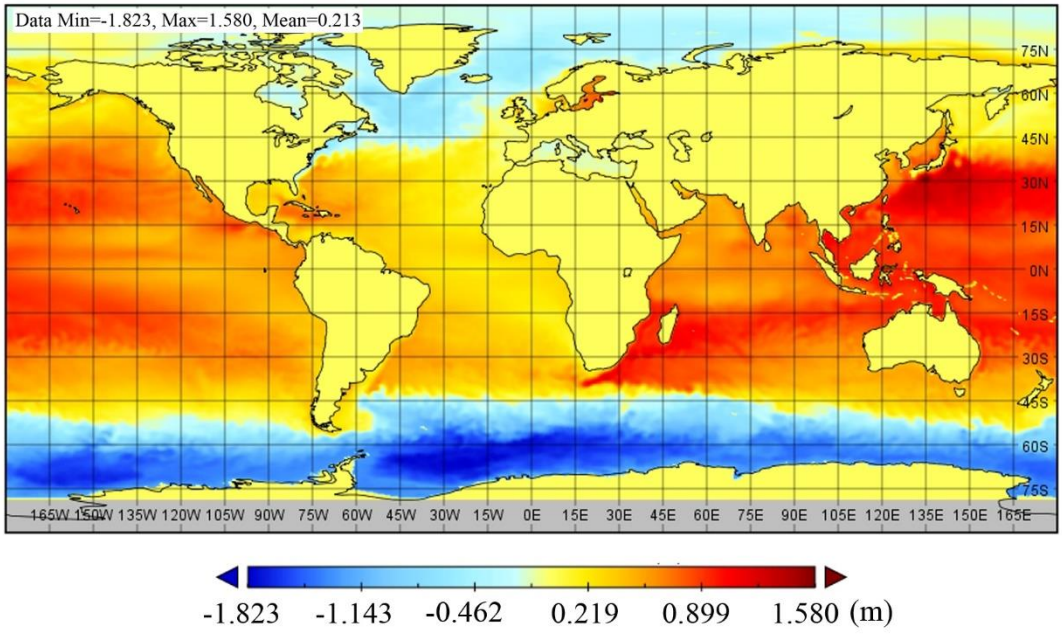


Рис. 5. Модельный прогноз уровня уровня океана (метры) 2000–2027 гг.

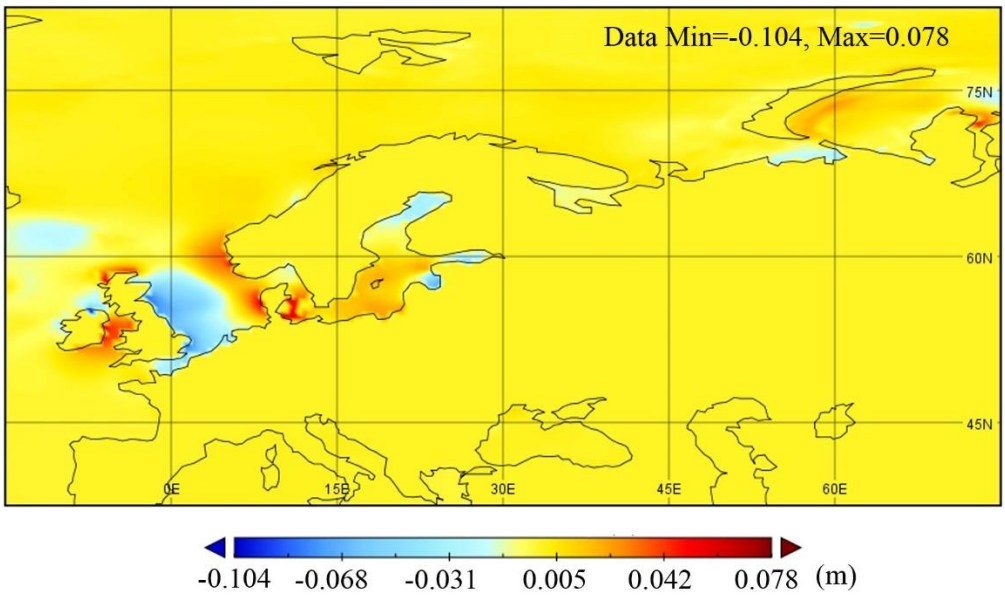


Рис. 6. Модельный прогноз изменения уровня уровня океана (метры) в зоне Российской Арктики 2000–2027 гг.

На рис. 7 приведено поведение разности дисперсии полей до и после интегрирования, осредненное по широтным кругам (то есть по долготе). Видно, что за 10 лет интегрирования заметно возрастает хаотичность, то есть величина отклонений от средних полей, которая в основном сосредоточена в северных широтах. Величины этой хаотичности небольшие, но не пренебрежимо малые, их

учет для понимания глобальных процессов существенен. Связаны эти величины также с внешним воздействием, которое обуславливается целым рядом причин, прежде всего, парниковыми газами и другими факторами, по-видимому, антропогенного воздействия. Результаты прогнозирования изменения ТПО отражены на рис. 8–10.

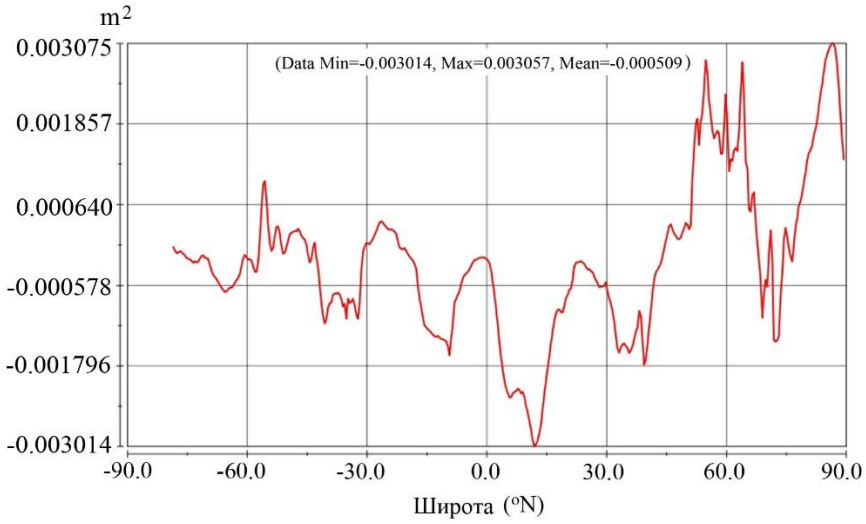


Рис. 7. Поле отклонений дисперсий уровня океана до и после интегрирования на период 10 лет

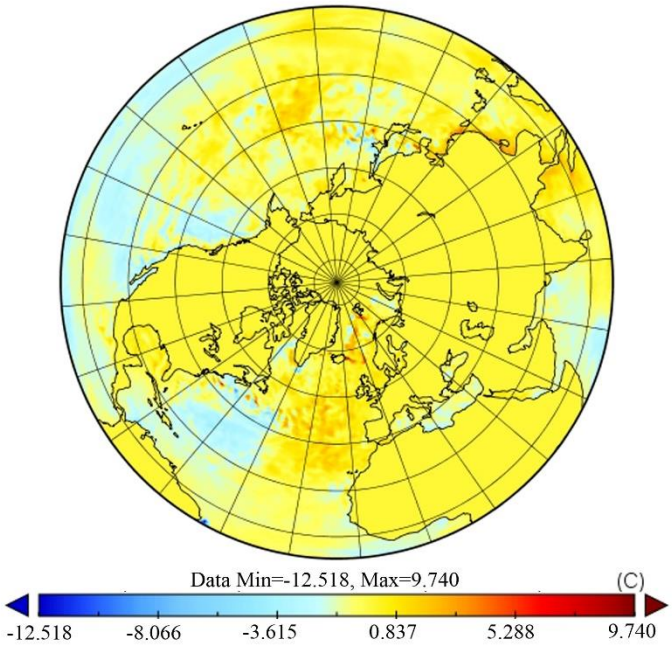


Рис. 8. Разница потенциальной поверхностной температуры в Арктике 2000–2027 (прогноз MPI-ESM)

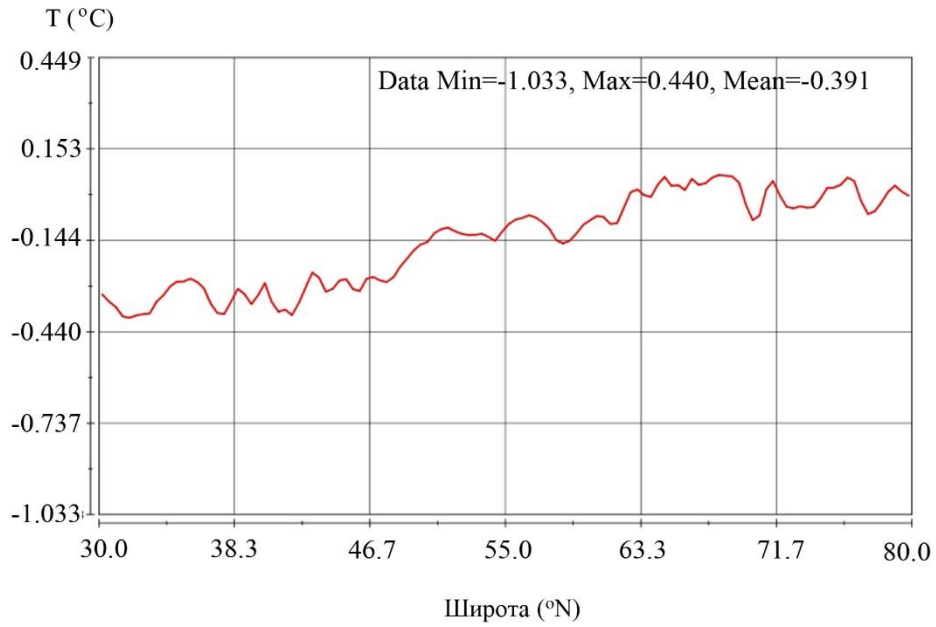


Рис. 9. Кривая роста потенциальной температуры в зоне Арктики 2000–2027 гг. (модельный прогноз MPI-ESM)

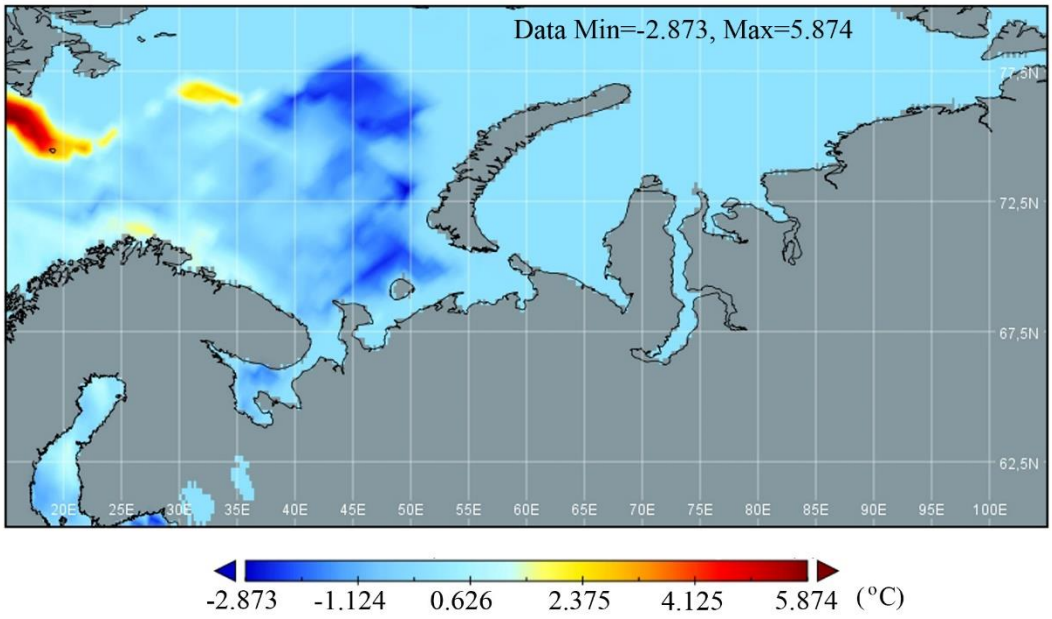


Рис. 10. Разница потенциальной поверхностной температуры в Российской зоне Арктики 2000–2027 гг (прогноз MPI-ESM)

На рис. 11, 12 показан модельный прогноз для изменения фракции льда в Арктике и Российской зоне Арктики, ранее показано в [11] по другой версии модели MPI ESM).

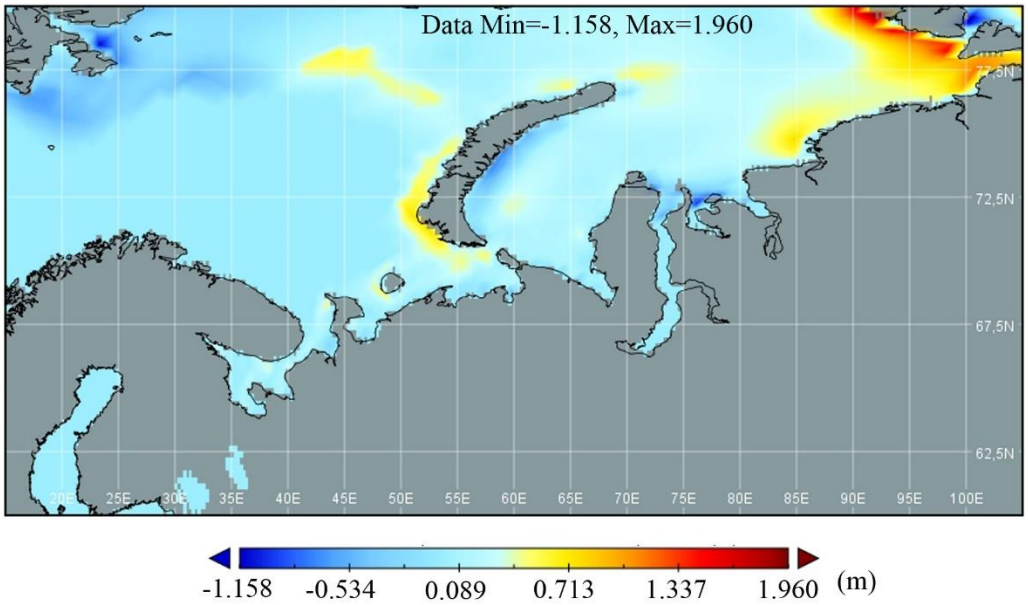


Рис. 11. Разница фракции льда в воде в Российской зоне Арктики 2000–2027 гг. (прогноз MPI-ESM)

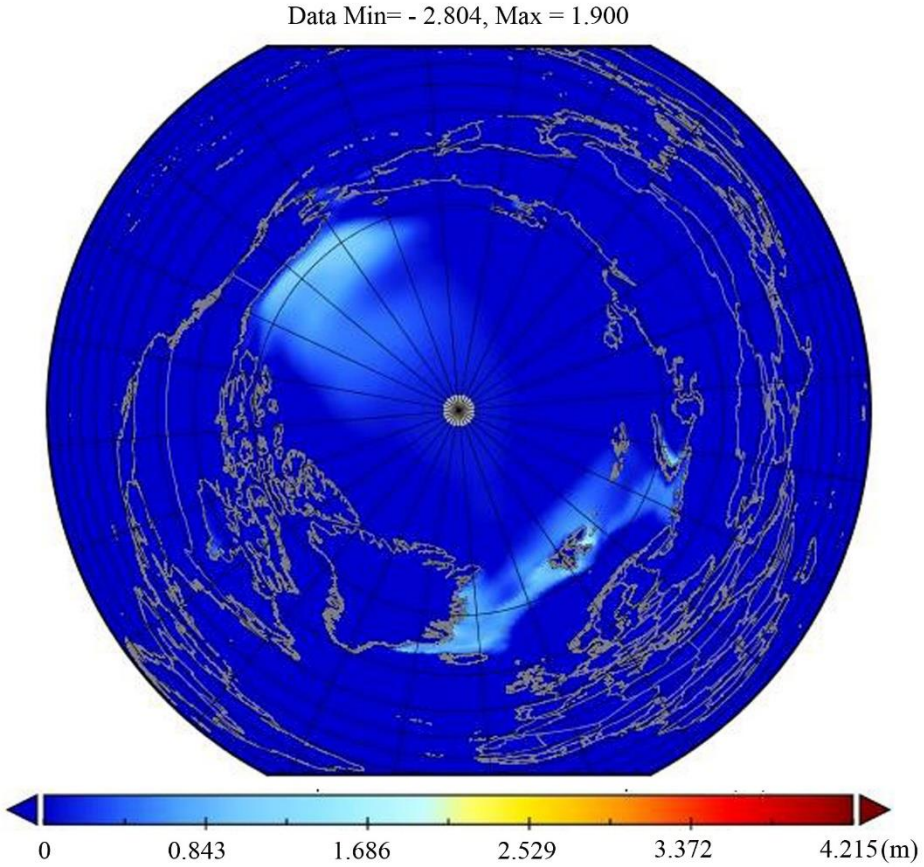


Рис. 12. Модельный прогноз (MPI-ESM) изменения льда на морской поверхности Арктики (метры) 1990–2027 гг.

Полученные результаты с точки зрения физического смысла можно трактовать следующим образом.

Проведенные эксперименты подтверждают тенденции глобального увеличения температуры, что соответствует реальным наблюдениям и выводам МГЭИК. Этот процесс происходит неравномерно. В высоких широтах нагрев происходит заметно интенсивнее, чем в среднем по глобусу. Вследствие этого повышения температуры происходит таяние льдов и другие климатические процессы, которые активно обсуждаются как в научном сообществе, так и на страницах прессы. В наших исследованиях такие процессы не показаны, однако они известны из целого ряда источников и отчетов МГЭИК [12] и ВМО [13] (Всемирная метеорологическая организация, ВМО, англ. World meteorological organization, WMO) и других исследований, например, [14].

Отметим также, что факт увеличения дисперсии характеризует увеличение хаотичности в поведении природных процессов. В частности, это видно на увеличении аномально холодной и аномально жаркой сезонных температур в различных частях планеты. Наблюдаются увеличение количества осадков, появление аномально сильных ветров, песчаных бурь в нехарактерных районах и в нехарактерное время года.

Отражение перечисленных явлений в результатах численного моделирования является следствием нелинейности рассматриваемой нами совместной модели океан–земля–атмосфера. Из этой нелинейности, например, следует, что средние значения характеристик при ансамблевом моделировании не совпадают со средними значениями самой модели, что представляет собой особый объект исследований.

Исследования такого рода нелинейных моделей нужны не только с точки зрения изучения систем нелинейных уравнений, но и для понимания природных процессов.

ЗАКЛЮЧЕНИЕ

В работе проведены численные эксперименты с глобальной совместной моделью MPI-ESM, которые направлены на выяснения характеристик устойчивости этой модели к возмущению начальных полей и влиянию различных факторов на поведение траекторий модели при длительном (декадном) интегрирова-

нии. Показано, что модель MPI-ESM достаточно хорошо «чувствует» возмущение начальных условий, производит адекватные результаты при этих возмущениях, но при этом устойчива к малым поправкам. Полученные при декадном интегрировании модельные поля вполне реалистичны, соответствуют наблюдаемым тенденциям и могут быть проанализированы, что и показано в работе. По результатам исследований сделаны оценки по поведению сложных нелинейных моделей, чувствительности к начальным возмущениям и анализу поведения этих возмущений.

БЛАГОДАРНОСТИ

Работа выполнена при частичной поддержке Российского фонда фундаментальных исследований, проект № 18-29-10085мк и в рамках бюджетных тем Министерства науки и высшего образования РФ научных организаций авторов. Третий автор выполнял исследования также по теме Министерства науки и высшего образования РФ №0149-2019-0004.

СПИСОК ЛИТЕРАТУРЫ

1. *Bronselaer B., Winton M., Griffies S.M., Stouffer R.J., Hurlin W.J., Rodgers K., Russell J.L.* Change in future climate due to Antarctic meltwater // Nature. 2018. V. 564. Issue 7734. P. 53–58. https://www.nature.com/articles/s41586-018-0712-z (доступно 07.11.2019)

2. *Holt J., Polton J., Huthnance J., Wakelin S., Enda O'Dea E., Harle J., Yool A., Artioli Y., Blackford Y., Siddorn J., Inall M.* Climate-Driven Change in the North Atlantic and Arctic Oceans Can Greatly Reduce the Circulation of the North Sea // Geophysical Research Letters. 2018. https://doi.org/10.1029/2018GL078878 (доступно 07.11.2019)

3. *Jungclaus J.H., Fischer N., Haak H., Lohmann K., Marotzke J., Matei D., Mikolajewicz U., Notz D., Storch J.S.* Characteristics of the ocean simulations in the Max Planck Institute Ocean Model (MPIOM) the ocean component of the MPI-Earth system model // J. of Advances in Modeling Earth Systems. 2013. Issue 2. P. 422–446. https://doi.org/10.1002/jame.20023 (доступно 07.11.2019)

4. *Taylor K.E., Stouffer R.J., Meehl G.A.* An overview of CMIP5 and the experiment design // Bulletin American Meteorological Society. 2012. V. 93. No. 4. https://journals.ametsoc.org/doi/abs/10.1175/BAMS-D-11-00094.1 (доступно 07.11.2019)

5. *Марчук Г.И., Дымников В.П., Залесный В.Б.* Математические модели в геофизической гидродинамике и численные методы их реализации. Л.: Гидрометиздат, 1987. 296 с.

6. *Наац В.И., Наац И.Э.* Математические модели и численные методы в задачах экологического мониторинга атмосферы. М.: ФИЗМАТЛИТ, 2010. 328 с.

7. The Intergovernmental Panel on Climate Change. https://www.ipcc.ch/ (доступно 01.07.2019)

8. *Breckling S.M., Pahlevani N.F.* A sensitivity study of the Navier-Stokes-α model // Computers and Mathematics with Applications. 2018. V. 75. P. 666-689.

9. *Belyaev K.P., Kirchner I., Kuleshov A.A., Tuchkova N.P.* Numerical Realization of Hybrid Data Assimilation Algorithm in Ensemble Experiments with the MPIESM Coupled Model. In: Velarde M., Tarakanov R., Marchenko A. (eds). The

Ocean in Motion. Springer Oceanography. 2018. P. 447–459. https://doi.org/10.1007/978-3-319-71934-4_27

10. *Baehr J., Fröhlich K., Botzet M. et al.* The prediction of surface temperature in the new seasonal prediction system based on the MPI-ESM coupled climate model // Climate Dynamic. 2015. V. 44. Issue 9-10. P. 2723–2735. https://doi.org/10.1007/s00382-014-2399-7

11. *Notz D., Haumann F.A., Haak H., Jungclaus J.H., Marotzke J.* Arctic sea-ice evolution as modeled by Max Planck Institute for meteorology's Earth system model // J. of Advances in Modeling Earth Systems. 2013. V. 5. P. 173–194. https://doi.org/10.1002/jame.20016 (доступно 07.11.2019)

12. Global warming of 1.5 °C https://www.ipcc.ch/sr15/ (access 01.07.2019)

13. WMO Statement on the State of the Global Climate in 2019. https://library.wmo.int/doc_num.php?explnum_id=5789 (доступно 07.11.2019)

14. *Koul V., Schrum C., Düsterhus A., Baehr J.* Atlantic Inflow to the North Sea Modulated by the Subpolar Gyre in a Historical Simulation with MPI-ESM // J. of Geophysical Research: Oceans, 2019. V. 124. Issue 3. P. 1807–1826. https://doi.org/10.1029/2018JC014738 (доступно 07.11.2019)

STABILITY STUDIES OF A COUPLED MODEL TO PERTURBATION OF INITIAL DATA

K. P. Belyaev[1] , G. M. Mikhaylov[2], A. N. Salnikov [3], N. P. Tuchkova[4]

[1-4]Dorodnicyn Computing Center FRC CSC of RAS, Vavilov str., 40, 11933, Moscow
[1]Shirshov Institute of Oceanology of RAS, Nahimovskiy pr., 36, 117218, Moscow
[3]Lomonosov Moscow State University, GSP-1, Leninskie Gory, 11999, Moscow,

[1]kosbel55@gmail.com, [2]gmickail@ccas.ru, [3]salnikov@angel.cs.msu.ru,
[4]natalia_tuchkova@mail.ru

Abstract

The stability problem is considered in terms of the classical Lyapunov definition. For this, a set of initial conditions is set, consisting of their preliminary calculations, and the spread of the trajectories obtained as a result of numerical simulation is analyzed. This procedure is implemented as a series of ensemble experiments with a joint MPI-ESM model of the Institute of Meteorology M. Planck (Germany). For numerical modeling, a series of different initial values of the characteristic fields was specified and the model was integrated, starting from each of these fields for different time periods. Изучались экстремальные характеристики уровня океана за период 30 лет. The statistical distribution was built, the parameters of this distribution were estimated, and the statistical forecast for 5 years in advance was studied. It is shown that the statistical forecast of the level corresponds to the calculated forecast obtained by the model. The localization of extreme level values was studied and an analysis of these results was carried out. Numerical calculations were performed on the Lomonosov-2 supercomputer of Lomonosov Moscow State University.

Keywords*: non-linear circulation models, Ensemble numerical experiments, analysis of stability of the model trajectories*

REFERENCES

1. *Bronselaer B., Winton M., Griffies S.M., Stouffer R.J., Hurlin W.J., Rodgers K., Russell J.L.* Change in future climate due to Antarctic meltwater // Nature. 2018. V. 564. Issue 7734. P. 53–58. https://www.nature.com/articles/s41586-018-0712-z (access 07.11.2019)

2. *Holt J., Polton J., Huthnance J., Wakelin S., Enda O'Dea E., Harle J., Yool A., Artioli Y., Blackford Y., Siddorn J., Inall M.* Climate-Driven Change in the North Atlantic and Arctic Oceans Can Greatly Reduce the Circulation of the North Sea // Geophysical Research Letters. 2018. https://doi.org/10.1029/2018GL078878 (access 07.11.2019)

3. *Jungclaus J.H., Fischer N., Haak H., Lohmann K., Marotzke J., Matei D., Mikolajewicz U., Notz D., Storch J.S.* Characteristics of the ocean simulations in the Max Planck Institute Ocean Model (MPIOM) the ocean component of the MPI-Earth system model // J. of Advances in Modeling Earth Systems. 2013. Issue 2. P. 422–446. https://doi.org/10.1002/jame.20023 (access 07.11.2019)

4. *Taylor K.E., Stouffer R.J., Meehl G.A.* An overview of CMIP5 and the experiment design // Bulletin American Meteorological Society. 2012. V. 93. No. 4. https://journals.ametsoc.org/doi/abs/10.1175/BAMS-D-11-00094.1 (access 07.11.2019)

5. *Marchuk G.I., Dymnikov V.P., Zalesnyj V.B.* Matematicheskie modeli v geofizicheskoj gidrodinamike i chislennye metody ih realizacii. L.: Gidrometizdat, 1987. 296 s.

6. *Naac V.I., Naac I.E.* Matematicheskie modeli i chislennye metody v zadachah ekologicheskogo monitoringa atmosfery. M.: FIZMATLIT, 2010. 328 s.

7. The Intergovernmental Panel on Climate Change. https://www.ipcc.ch/ (access 07.11.2019).

8. *Breckling S.M., Pahlevani N.F.* A sensitivity study of the Navier-Stokes-α model // Computers and Mathematics with Applications. 2018. V. 75. P. 666–689.

9. *Belyaev K.P., Kirchner I., Kuleshov A.A., Tuchkova N.P.* Numerical Realization of Hybrid Data Assimilation Algorithm in Ensemble Experiments with the MPIESM Coupled Model. In: Velarde M., Tarakanov R., Marchenko A. (eds). The Ocean in Motion. Springer Oceanography. 2018. P. 447–459. https://doi.org/10.1007/978-3-319-71934-4_27

10. *Baehr J., Fröhlich K., Botzet M. et al.* The prediction of surface temperature in the new seasonal prediction system based on the MPI-ESM coupled climate model // Climate Dynamic. 2015. V. 44. Issue 9-10. P. 2723–2735. https://doi.org/10.1007/s00382-014-2399-7

11. *Notz D., Haumann F.A., Haak H., Jungclaus J.H., Marotzke J.* Arctic sea-ice evolution as modeled by Max Planck Institute for meteorology's Earth system model

// J. of Advances in Modeling Earth Systems. 2013.V. 5. P. 173–194, https://doi.org/10.1002/jame.20016 (access 07.11.2019)

12. Global warming of 1.5 °C https://www.ipcc.ch/sr15/ (access 01.07.2019)

13. WMO Statement on the State of the Global Climate in 2019 https://library.wmo.int/doc_num.php?explnum_id=5789 (access 07.11.2019)

14. *Koul V., Schrum C., Düsterhus A., Baehr J.* Atlantic Inflow to the North Sea Modulated by the Subpolar Gyre in a Historical Simulation With MPI-ESM // J. of Geophysical Research: Oceans. 2019. V. 124. Issue 3. P. 1807–1826. https://doi.org/10.1029/2018JC014738 (access 07.11.2019)

СВЕДЕНИЯ ОБ АВТОРАХ

БЕЛЯЕВ Константин Павлович – ведущий научный сотрудник Института океанологии им. П.П. Ширшова РАН, доктор физ.-матем. наук, профессор кафедры теории вероятностей и статистики МГУ им. Ломоносова. Сфера научных интересов - математическое моделирование и усвоение данных наблюдений, статистический анализ натурных данных.

Konstantin Pavlovich BELYAEV – leading scientist of Shirshov Institute of Oceanology, Russian Academy of Science. Doctor of science, professor of Dept of Appiied Math and Cybernetics, Lomonosov Moscow State University. Research interests – math modelling and data assimilation, statistical analysis of natural data.

email: kosbel55@gmail.com

МИХАЙЛОВ Гурий Михайлович – ведущий научный сотрудник Вычислительного центра им. А.А. Дородницына ФИЦ ИУ РАН, кандидат физ.-мат. наук. Сфера научных интересов – архитектура вычислительных систем и сетей, вычислительные и информационные технологии.

Gury Mikhaylovich MIKHAYLOV – leading scientist of Dorodnicyn computing center FRC SCS RAS, PhD in physics with a math degree. Research interests include architecture of computing systems and networks, computing and information technology.

email: gmickail@ccas.ru

САЛЬНИКОВ Алексей Николаевич – ведущий научный сотрудник кафедры математической физики факультета ВМиК МГУ М.В. им. Ломоносова, кандидат физ.-матем. наук. Сфера научных интересов – параллельное программирование, биоинформатика, суперкомпьютеры.

Alexey Nikolaevich SALNIKOV – leading researcher Dept of Appiied Math and Cybernetics, Lomonosov Moscow State University, PhD in physics with a math degree. Research interests include bioinformatics, parallel and supercomputing programming

email: salnikov@angel.cs.msu.ru

ТУЧКОВА Наталия Павловна – старший научный сотрудник Вычислительного центра им. А.А. Дородницына ФИЦ ИУ РАН, кандидат физ.-мат. наук, окончила ВМиК МГУ им. М.В. Ломоносова. Специалист в области алгоритмических языков и информационных технологий.

Natalia Pavlovna TUCHKOVA – senior researcher of Dorodnicyn computing center FRC SCS RAS, PhD in physics with a math degree, graduated from CS Faculty of Lomonosov MSU. The expert in the field of algorithmic languages and information technologies.

email: natalia_tuchkova@mail.ru

Материал поступил в редакцию 15 ноября 2019 года

УДК 519.713

О РАЗДЕЛИМОСТИ ВХОДО-ВЫХОДНЫХ ПОЛУАВТОМАТОВ С НЕДЕТЕРМИНИРОВАННЫМ ПОВЕДЕНИЕМ

И. Б. Бурдонов, Н. В. Евтушенко, А. С. Косачев

Институт системного программирования им. В.П. Иванникова Российской академии наук, г. Москва

igor@ispras.ru, evtushenko@ispras.ru, kos@ispras.ru

Аннотация

При синтезе тестов для проверки функциональных и нефункциональных требований для компонентов различных управляющих систем особое значение имеет понятие различимости, поскольку должна быть возможность отличить правильно функционирующий компонент от неправильно функционирующего, и при активном тестировании для этого используются специальные различающие последовательности. Такие последовательности хорошо исследованы для детерминированных и полностью определенных автоматов, однако компоненты управляющих систем часто могут быть описаны только частично и имеют недетерминированное поведение. В настоящей работе мы рассматриваем модель входо-выходного полуавтомата, вводим понятие разделяющей последовательности для двух таких полуавтоматов, при однократной подаче которой можно однозначно распознать, какой из двух полуавтоматов представлен для эксперимента, и предлагаем алгоритм построения таких последовательностей для специального класса полуавтоматов.

Ключевые слова*: входо-выходной полуавтомат, тестирование, разделяющая последовательность*

ВВЕДЕНИЕ

Задача синтеза тестов с гарантированной полнотой для управляющих систем по-прежнему является актуальной, и при построении таких тестов широко используются трассовые модели. В частности, активно используется модель конечного входо-выходного автомата/полуавтомата, описывающая поведение системы как

отображение последовательностей действий в одном (входном) алфавите в последовательности в другом (выходном) алфавите [1–5]. При синтезе тестов для проверки различных свойств управляющей системы на основе модели «белого ящика» особое значение имеет понятие различимости, поскольку должна быть возможность отличить правильно функционирующий компонент от неправильно функционирующего, и при активном тестировании для этого используются специальные различающие / разделяющие входные последовательности. Такие последовательности хорошо исследованы для детерминированных и полностью определенных автоматов [5], однако спецификация управляющей системы достаточно часто является частичной и недетерминированной. Для таких конечных автоматов также существуют методы построения различающих / разделяющих входных последовательностей [6–8]. В настоящей работе мы рассматриваем модель входо-выходного полуавтомата [2, 4, 9], поскольку во многих системах наличие выходного символа после каждого входного символа, как это имеет место в классических конечных автоматах, является серьезным ограничением. В модели входо-выходного полуавтомата выходная реакция может появиться только после поступления последовательности входных воздействий, причем в качестве такой реакции может быть не один выходной символ, а последовательность выходных символов. Мы вводим понятие (адаптивной) разделяющей последовательности для двух полуавтоматов специального класса и предлагаем алгоритм проверки ее существования. Если разделяющая последовательность существует, то при однократной подаче такой последовательности можно однозначно распознать, какой из двух полуавтоматов (спецификация или ошибочная реализация) предъявлен для проверки, что при тестировании систем с недетерминированным поведением существенно отличает этот вид различимости от других, для которых различающая последовательность должна быть подана на проверяемую реализацию достаточно большое количество раз (выполнение требования о «всех погодных условиях» [2, 8]). В работе также оценена длина разделяющих последовательностей для полуавтоматов специального класса.

Статья структурирована следующим образом. Раздел 1 содержит определения и обозначения, используемые в работе. В разделе 2 обсуждены отношение разделимости и класс входо-выходных полуавтоматов, для которых можно по-

строить (адаптивную) разделяющую последовательность с использованием известных методов из теории автоматов; в разделе 3 приведены оценки длины неадаптивных последовательностей, если последовательности существуют. В заключении, как обычно, кратко обсуждены направления дальнейшей работы.

1. ОПРЕДЕЛЕНИЯ И ОБОЗНАЧЕНИЯ

В настоящем разделе мы напоминаем необходимые определения из теории трассовых моделей. Кроме того, в разделе определены класс входо-выходных полуавтоматов, рассматриваемый в работе, и понятие разделяющей последовательности для входо-выходных полуавтоматов рассматриваемого класса. Конечный входо-выходной *полуавтомат* (или далее просто *полуавтомат*) есть четверка $\boldsymbol{S}=(S,s_0,I,O,h_S)$, где S – конечное непустое множество состояний с выделенным начальным состояние s_0, I – конечное непустое множество входных действий, O – конечное непустое множество выходных действий, $I\cap O=\varnothing$, и $h_S\subseteq S\times(I\cup O)\times S$ есть отношение переходов. Существует переход из состояния s в состояние s' под действием символа a, если и только если тройка (s,a,s') принадлежит отношению переходов h_S. Полуавтомат *наблюдаемый*[1], если в любом состоянии по любому символу существует не более одного перехода. Полуавтомат обычно рассматривается как трассовая модель, где под трассой в заданном состоянии понимается последовательность действий из алфавита $I\cup O$, допустимая в этом состоянии. Поскольку в процессе тестирования наблюдаются конечные трассы, то мы предполагаем, что в полуавтомате отсутствуют циклы, помеченные только выходными символами. Для того чтобы исключить состязания в состояниях полуавтомата, мы рассматриваем полуавтоматы, в которых в каждом состоянии определены только входные или только выходные символы. Иными словами, в данной работе под входо-выходным полуавтоматом понимается наблюдаемый полуавтомат $\boldsymbol{S}=(S,s_0,I,O,h_S)$, в котором множество S есть объединение трех непересекающихся подмножеств S_1, S_2 и S_3: в состояниях множества S_1 определены переходы только по входным символам (и есть хотя бы один такой переход),

[1]Достаточно часто такой полуавтомат называется детерминированным [1]. Однако мы используем термин «детерминированный» для наблюдаемых полуавтоматов, у которых в состоянии определено не более одного выходного символа.

а в состояниях множества S_2 есть только переходы по выходным символам (и есть хотя бы один такой переход). В состояниях множества S_3 не определен ни один переход, т. е. эти состояния являются тупиковыми состояниями. Вообще говоря, каждое из подмножеств может быть пустым. Трасса в начальном состоянии называется *полной*, если она заканчивается в состоянии, в котором не определены переходы по выходным символам. Для возможного наблюдения таких трасс введем специальный выходной символ $\delta \notin I \cup O$, так называемый «молчащий символ (quiescence)» [2, 4], т. е. в каждом состоянии из множеств S_1 и S_3 добавляется петля по символу δ, который считается выходным символом, и в результате получается полуавтомат $\boldsymbol{S}^{\delta}$. Таким образом, трасса σ полуавтомата $\boldsymbol{S}$ является полной тогда и только тогда, когда в полуавтомате $\boldsymbol{S}^{\delta}$ есть трасса $\sigma\delta$, что соответствует тому, что после этой трассы не может появиться ни один выходной символ из алфавита O (так называемые δ-трассы). Напомним, что, по определению, в полуавтомате отсутствуют циклы, помеченные только символами из выходного алфавита O, и поэтому каждая трасса в полуавтомате является начальным отрезком некоторой полной трассы.

Если начальное состояние полуавтомата принадлежит множеству S_1, то входной символ $i \in I$ называется *строго определенным* в начальном состоянии, если в начальном состоянии определен переход по данному входному символу. Если начальное состояние полуавтомата принадлежит множеству S_2, то входной символ $i \in I$ называется *строго определенным* в начальном состоянии, если по данному входному символу определен переход в каждом состоянии, которое достижимо из начального состояния по полной трассе, помеченной выходными символами из алфавита O. Последовательность αi входных символов из множества I называется *строго определенной* в начальном состоянии полуавтомата $\boldsymbol{S}$, если последовательность α является строго определенной в начальном состоянии полуавтомата, и в каждом состоянии, достижимом из начального состояния по полной трассе с входной проекцией α, определен переход с входным символом i.

2. РАЗДЕЛИМОСТЬ ВХОДО-ВЫХОДНЫХ ПОЛУАВТОМАТОВ

Пусть $\boldsymbol{S}=(S,s_0,I,O,h_S)$ и $\boldsymbol{P}=(P,p_0,I,O,h_P)$ суть полуавтоматы из определенного выше класса. Полуавтоматы $\boldsymbol{S}$ и $\boldsymbol{P}$ назовем *неразделимыми*, если для любой входной последовательности α, строго определенной в начальных состояниях полуавтоматов $\boldsymbol{S}$ и $\boldsymbol{P}$, множества выходных проекций полных трасс полуавтоматов $\boldsymbol{S}$ и $\boldsymbol{P}$ с входной проекцией α пересекаются.

В противном случае полуавтоматы называются *разделимыми*, и входная последовательность α, такая, что множества выходных проекций полных трасс полуавтоматов $\boldsymbol{S}$ и $\boldsymbol{P}$ с входной проекцией α не пересекаются, называется *разделяющей* последовательностью для полуавтоматов $\boldsymbol{S}$ и $\boldsymbol{P}$.

При синтезе проверяющих тестов на основе модели «белого ящика» для распознавания предъявляются два полуавтомата $\boldsymbol{S}$ и $\boldsymbol{P}$, где $\boldsymbol{S}$ – обычно полуавтомат-спецификация, а $\boldsymbol{P}$ – полуавтомат, полученный из спецификации посредством некоторой «интересной» мутации. При распознавании этих полуавтоматов внешним экспериментом отношение разделимости означает, что если полуавтомат, предъявленный для распознавания, является одним из полуавтоматов $\boldsymbol{S}$ и $\boldsymbol{P}$, то, подав на предъявленный полуавтомат входную последовательность α и проанаблюдав соответствующую полную трассу, мы однозначно можем определить, какой из полуавтоматов был предъявлен, т. е. обнаружить полуавтомат-мутант. Используется следующая *гипотеза о подаче входных последовательностей* [9]. Перед подачей очередного входного символа тестер ожидает некоторый разрешенный промежуток времени – максимальный выходной таймаут t. Таким образом, эксперимент с полуавтоматом проводится следующим образом: тестер ожидает время t, если система выдает выходной сигнал, то таймер сбрасывается, и тестер вновь ждет отведенное время t. Если же за время t система не произвела выходного символа, то предполагается, что система выдала выходной символ δ. После этого тестер подает следующий входной символ, и далее вновь ожидает отведенное время t. В этом случае по каждому из полуавтоматов можно построить конечный автомат, который может оказаться частичным и недетерминированным, и использовать методы из [6–8] для проверки разделимости построенных конечных автоматов.

Конечный автомат, часто называемый просто *автоматом*, есть пятерка $S=\langle S,X,Y,h_S,s_0\rangle$, где S есть непустое конечное множество состояний с выделенным начальным состоянием s_0, X и Y суть непустые конечные *входной* и *выходной алфавиты*, и $h_S \subseteq S\times I\times O\times S$ есть *отношение переходов*. Мы говорим, что существует переход из состояния $s\in S$ в состояние $s'\in S$ по входо-выходной паре x/y (xy), если и только если четверка (s,x,y,s') принадлежит отношению переходов h_S. Автомат S – *наблюдаемый*, если для любых двух переходов (s,x,y,s'), $(s,x,y,s'')\in h_S$ справедливо $s''=s'$. Если автомат S наблюдаемый, $x\in X$ и $y\in Y$, то состояние s' называется *xy-преемником* состояния s, если $\exists\, s'\ (s,x,y,s')\in h_S$; если такого состояния s' нет, будем говорить, что xy-преемник пустой. Множество непустых xy-преемников состояния s для всех выходных символов y называется *x-преемником* состояния s. Понятие xy- и x-преемников можно определить для пары различных состояний s_1 и s_2, если входной символ x определен в каждом из состояний. В этом случае xy-преемник определяется как пара xy-преемников этих состояний. Если xy-преемники этих состояний совпадают, или xy-преемник существует только для одного из состояний s_1 и s_2, то xy-преемником пары $\{s_1, s_2\}$ является соответствующий синглетон. Множество непустых xy-преемников пары $\{s_1,s_2\}$ называется *x-преемником* этой пары состояний.

Отношение переходов автомата обычным образом расширяется на входные и выходные последовательности; по определению, для каждого $s\in S$ автомата S четверка $(s,\varepsilon,\varepsilon,s)$ находится в множестве переходов автомата S, где ε – пустая последовательность. Расширенное на последовательности отношение переходов автомата обозначается тем же символом h_S. *Входо-выходной последовательностью* или *трассой* автомата в начальном состоянии называется последовательность входо-выходных символов, которые можно последовательно пройти из начального состояния. Входной символ x определен в состоянии s, если отношение переходов содержит четверку (s,x,y,s') для некоторых y и s'. Входная последовательность αx называется *определенной* для автомата, если последовательность α определена в начальном состоянии, и входной символ x определен в каждом состоянии, достижимом из начального состояния по трассе с входной проекцией α. Если автомат S – наблюдаемый, $\alpha\beta$ – входо-выходная последовательность в состоянии и $(s,\alpha,\beta,s')\in h_S$, то состояние s' называется $\alpha\beta$-*преемником* состояния

s; если такого состояния *s*' нет, будем говорить, что $\alpha\beta$-преемник пустой. Множество непустых $\alpha\beta$-преемников состояния *s* для всех возможных входо-выходных последовательностей $\alpha\beta$ называется α-*преемником* состояния *s*.

Два автомата, определенные над одним входным алфавитом, называются *разделимыми*, если существует входная последовательность α, определенная в каждом из автоматов, такая, что множества трасс с входной проекцией α в автоматах не пересекаются. В противном случае автоматы называются *неразделимыми*. Известны алгоритмы построения разделяющей последовательности как для полностью определенных и наблюдаемых автоматов, так и для частичных ненаблюдаемых автоматов [6, 7], и эти алгоритмы можно использовать при построении разделяющей последовательности для входо-выходных полуавтоматов, рассматриваемых в данной работе. Построим по полуавтомату ***S*** конечный, возможно, недетерминированный автомат с использованием алгоритма из работы [9].

Алгоритм 1 построения конечного автомата по входо-выходному полуавтомату

Вход: Наблюдаемый входо-выходной полуавтомат $\boldsymbol{S}=(S,s_0,I,O,h_S)$, в котором множество *S* есть объединение трех непересекающихся подмножеств S_1, S_2 и S_3.

Выход: Конечный автомат $\boldsymbol{M_S}$, представляющий множество трасс полуавтомата $\boldsymbol{S}^{\delta}$.

Строим автомат $\boldsymbol{M_S}=(S_1 \cup S_3, I\cup\{null\_in\}, O\cup O^2\cup \ldots \cup O^{ns}\cup\{\delta\}, T_{MS})$, $null\_in\notin I$, с пустым множеством переходов, т. е. $T_{MS}=\varnothing$, где *ns* – наибольшая длина трассы из выходных символов в полуавтомате ***S***:

- для каждого состояния $s\in S_1$, такого, что $(s,i,s')\in T_S$, $s'\in S_1\cup S_3$, добавляем к T_{MS} переход (s,i,δ,s');

- для каждого состояния $s\in S_1$, такого, что $(s,i,s')\in T_S$, $s'\in S_2$, добавляем к T_{MS} переход $(s,i,o_1o_2\ldots o_k,s'')$, $k\le ns$, где $s''\in S_1\cup S_3$ есть $o_1\,o_2\ldots o_k$-преемник состояния s'.

Если начальное состояние полуавтомата $\boldsymbol{S}$ принадлежит S_2, то добавляем к T_{MS} переход $(s_0,null\_in,o_1\ o_2\ldots o_k,s)$, где $s\in S_1\cup S_3$, и s есть $o_1\ o_2\ldots o_k$-преемник состояния s_0. Если начальное состояние полуавтомата $\boldsymbol{S}$ принадлежит S_3, то множество переходов T_{MS} автомата является пустым, т. е. есть тривиальный автомат, множество трасс которого содержит единственную пустую последовательность ε.

□

Непосредственной проверкой можно убедиться, что по правилам построения автомата $\boldsymbol{M_S}$ имеют место следующие достаточно простые утверждения.

Предложение 1. Если $\boldsymbol{S}$ – наблюдаемый входо-выходной полуавтомат, то автомат $\boldsymbol{M_S}$, построенный по алгоритму 1, есть наблюдаемый автомат.

Предложение 2. Пусть $\boldsymbol{S}$ – наблюдаемый входо-выходной полуавтомат из рассматриваемого класса. Если начальное состояние полуавтомата есть состояние из множества S_1, то входная последовательность α является строго определенной в полуавтомате $\boldsymbol{S}$, если и только если α является определенной входной последовательностью в автомате $\boldsymbol{M_S}$. Если начальное состояние полуавтомата есть состояние из множества S_2, то входная последовательность α является строго определенной в полуавтомате $\boldsymbol{S}$, если и только если последовательность $null\_in$ α является определенной входной последовательностью в автомате $\boldsymbol{M_S}$.

Предложение 3. Пусть $\boldsymbol{S}$ – наблюдаемый входо-выходной полуавтомат. Если в начальном состоянии полуавтомата определены переходы только по входным символам, то множества трасс полуавтомата $\boldsymbol{S}^{\delta}$ и автомата $\boldsymbol{M_S}$ совпадают. Если в начальном состоянии полуавтомата определены переходы только по выходным символам, то каждой трассе α в полуавтомате $\boldsymbol{S}$ соответствует трасса $null\_in$ α в автомате $\boldsymbol{M_S}$, и обратно.

Пусть $\boldsymbol{S}=(S,s_0,I,O,h_S)$ и $\boldsymbol{P}=(P,p_0,I,O,h_P)$ суть полуавтоматы из рассматриваемого класса. По каждому из полуавтоматов $\boldsymbol{S}^{\delta}$ и $\boldsymbol{P}^{\delta}$ можно построить соответствующий конечный, возможно, недетерминированный и частичный автомат. Следствием предложений 2 и 3 является следующая теорема.

Теорема 4. Полуавтоматы $\boldsymbol{S}$ и $\boldsymbol{P}$ являются разделимыми, если и только если разделимыми являются автоматы $\boldsymbol{M_S}$ и $\boldsymbol{M_P}$. Более того, если начальные состояния полуавтоматов суть состояния из S_1 и P_1, то последовательность α является разде-

ляющей для полуавтоматов ***S*** и ***P***, если и только если α является таковой для автоматов M_S и M_P. Если начальные состояния полуавтоматов суть состояния из S_2 и P_2, то последовательность α является разделяющей для полуавтоматов ***S*** и ***P***, если и только если последовательность *null_in* α является таковой для автоматов M_S и M_P. Если начальные состояния полуавтоматов суть состояния из S_1 и P_2 или S_2 и P_1, то пустая последовательность является разделяющей для полуавтоматов ***S*** и ***P***.

Для построения разделяющей последовательности для автоматов M_S и M_P используется алгоритм построения разделяющей последовательности из [6].

Алгоритм 2 построения разделяющей последовательности для двух наблюдаемых, возможно частичных автоматов.

Вход: Два наблюдаемых, возможно частичных автомата M_S и M_P над входным алфавитом *I*.

Выход: Разделяющая последовательность для автоматов M_S и M_P, если автоматы разделимы, или сообщение «Автоматы не являются разделимыми» в противном случае.

Шаг 1. Строим пересечение автоматов M_S и M_P. Если пересечение является полностью определенным автоматом, то **Return** сообщение «Автоматы не являются разделимыми».

Шаг 2. Если пересечение M_S и M_P является частичным автоматом, то строим усеченное дерево преемников для начального состояния пересечения. Корень дерева помечается парой начальных состояний автоматов; вершины дерева помечаются подмножествами состояний автоматов из пересечения. Пусть уже построены *j* уровней дерева, $j \geq 0$, и внутренняя (не финальная) вершина $j^{\text{го}}$ уровня помечена подмножеством *P* состояний пересечения. Из вершины существует исходящая дуга, помеченная входным символом *i*, если *i* является определенным входным символом в каждом из состояний множества *P*. В этом случае конец дуги помечен множеством *i*-преемников состояний из *P*. Текущая вершина *Current* на $p^{\text{м}}$ уровне, $p \geq 0$, помеченная множеством *P* состояний, является ***листом***, если выполняется одно из следующих условий:

Правило 1: Существует входной символ *i*, для которого каждое состояние (*s*, *p*) множества *P* не имеет *i*-преемников, в то время как в каждом из состояний *s* и *p* входной символ *i* определен.

Правило 2: Существует вершина на $j^{м}$ уровне $j<p$, помеченная множеством R состояний, такая, что $P \supseteq R$.

Шаг 3.

Если ни один из путей дерева не заканчивается листом, полученным по Правилу 1, то автоматы не являются разделимыми. **Return** сообщение «Автоматы не являются разделимыми».

Если существует конечная вершина, объявленная листом по Правилу 1, т. е. помеченная множеством P состояний пересечения, такая, что существует входной символ i, для которого каждое состояние (s, p) множества P не имеет i-преемников, в то время как в каждом из состояний s и p входной символ i определен, и путь в эту вершину помечен входной последовательностью α, входная последовательность $\alpha\, i$ является разделяющей последовательностью для автоматов $\boldsymbol{M_S}$ и $\boldsymbol{M_P}$. **Return** последовательность $\alpha\, i$.

□

Заметим, что при использовании алгоритма 2, если усеченное дерево построено полностью или если обход дерева ведется в ширину, то для разделимых автоматов $\boldsymbol{M_S}$ и $\boldsymbol{M_P}$ будет построена кратчайшая разделяющая последовательность.

Таким образом, можно предложить следующий алгоритм для проверки разделимости входо-выходных полуавтоматов.

Алгоритм 3 проверки разделимости входо-выходных полуавтоматов и построения разделяющей последовательности, если полуавтоматы разделимы:

Вход: Входо-выходные наблюдаемые полуавтоматы $\boldsymbol{S}=(S,s_0,I,O,h_S)$ и $\boldsymbol{P}=(P,p_0,I,O,h_P)$.

Выход: Разделяющая последовательность α или сообщение “Полуавтоматы не являются разделимыми”.

Шаг 1. Если начальное состояние полуавтомата $\boldsymbol{S}$ ($\boldsymbol{P}$) принадлежит множеству $S_1 \cup S_3$ ($P_1 \cup P_3$), а начальное состояние полуавтомата $\boldsymbol{P}$ ($\boldsymbol{S}$) принадлежит множеству S_2 (P_2), то пустая входная последовательность является разделяющей для полуавтоматов $\boldsymbol{S}$ и $\boldsymbol{P}$. Если начальные состояние полуавтомата $\boldsymbol{S}$ ($\boldsymbol{P}$) принадлежит множеству S_3 (P_3), а начальное состояние полуавтомата $\boldsymbol{P}$ ($\boldsymbol{S}$) принадлежит множеству S_1 (P_1), то выдать сообщение «Полуавтоматы не являются разделимыми».

Шаг 2. Пусть начальные состояния полуавтоматов ***S*** и ***P*** принадлежат множествам $S_1 \cup S_3$ и $P_1 \cup P_3$ или множествам S_2 и P_2. Строим автоматы M_S и M_P по алгоритму 1.

Шаг 3. Проверяем наличие разделяющей последовательности, используя алгоритм 2. Если разделяющей последовательности для автоматов не существует, то выдаем сообщение «Полуавтоматы не являются разделимыми».

Если разделяющая последовательность α для автоматов M_S и M_P построена, то **Return** α, если α начинается с входного символа из алфавита *I*. Если разделяющая последовательность α имеет вид *null_in* β, то **Return** β.

□

Пример. Рассмотрим полуавтоматы ***S*** и ***P*** на рис. 1а и 2а с начальными состояниями *s*1 и *p*1 и соответствующие автоматы M_S и M_P на рис. 1б и 2б.

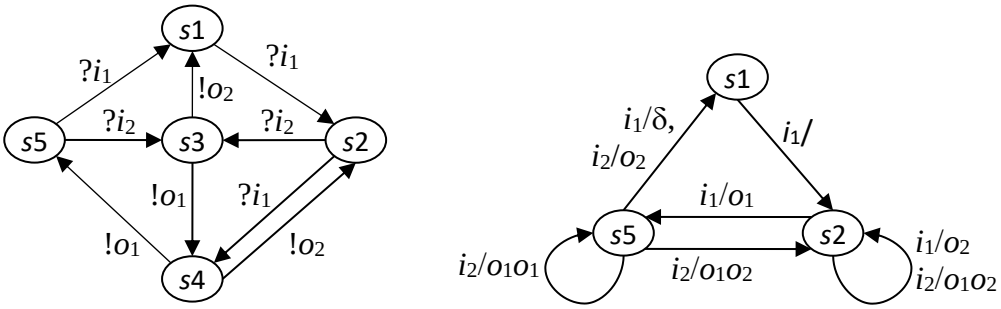


Рис. 1. Полуавтомат ***S*** (а) и автомат M_S (б)

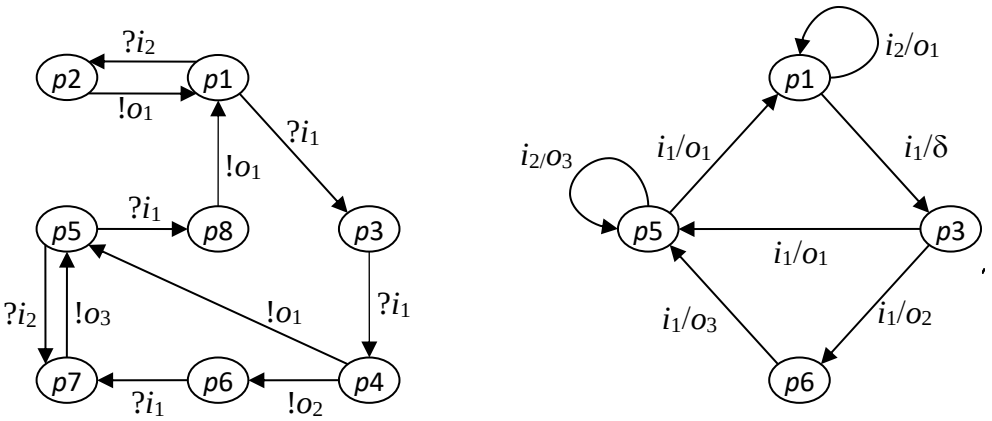


Рис. 1. Полуавтомат ***P*** (а) и автомат M_P (б)

С использованием алгоритма 2 можно построить разделяющую последовательность, фрагмент соответствующего усеченного дерева преемников представлен на рис. 3. В частности, в одном из состояний пары состояний ($s2$, $p3$) входной символ i_2 не определен, поэтому в этом состоянии в усеченном дереве есть только дуга, помеченная входным символом i_1. Состояния $s5$ и $p5$ и состояния $s2$ и $p6$ разделяются входным символом i_1, который определен в каждом из этих состояний, таким образом, входная последовательность i_1 i_1 i_1 является разделяющей для полуавтоматов $\boldsymbol{S}$ и $\boldsymbol{P}$.

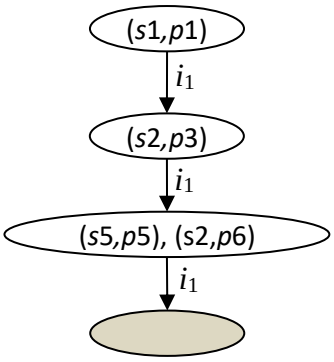


Рис. 3. Фрагмент усеченного дерева преемников для автоматов на рис. 1б и 2б

3. ОЦЕНКА ДЛИНЫ РАЗДЕЛЯЮЩЕЙ ПОСЛЕДОВАТЕЛЬНОСТИ

Для конечных полностью определенных наблюдаемых автоматов известна точная оценка длины кратчайшей разделяющей последовательности относительно числа состояний автоматов [7]. Эта оценка равна 2^{mn-1}, если автоматы $\boldsymbol{M_S}$ и $\boldsymbol{M_P}$ имеют m и n состояний, однако результат доказан только для автоматов с числом входных символов 2^{mn-1}. Соответственно, длина разделяющей последовательности для полуавтоматов $\boldsymbol{S}$ и $\boldsymbol{P}$, которые имеют m и n состояний в множествах $S_1 \cup S_3$ и $P_1 \cup P_3$, не превосходит этой величины. Для проверки достижимости этой оценки необходимы дополнительные исследования. Тем не менее, нами показано, что экспоненциальная оценка длины разделяющей последовательности достижима для наблюдаемых полуавтоматов, в которых в каждом состоянии из множеств S_1 и P_1 определен каждый входной символ, а множества S_3 и P_3 являются пустыми. Иными словами, существуют полуавтоматы, для которых кратчайшая разделяющая последовательность имеет экспоненциальную длину относительно числа состояний, по крайней мере, одного из полуавтоматов.

Теорема 5. Для любых $k \geq 3$ и $n>1$ существуют наблюдаемые полностью определенные по входным символам полуавтоматы $\boldsymbol{S}_k$ с $k(2k-3)$ состояниями и $\boldsymbol{A}_n$ с $n(2k-3)$ состояниями, $(2k-4)$ входными и k выходными символами, такие, что длина кратчайшей разделяющей последовательности равна $(n-1)2^{k-2}+1=\mathbf{O}(n2^k)$.

Для доказательства теоремы докажем сначала следующее утверждение для конечных автоматов.

Утверждение 6. Для любых $k \geq 3$ и $n>1$ существуют наблюдаемые полностью определенные автоматы $\boldsymbol{S}_k$ с k состояниями и $\boldsymbol{A}_n$ с n состояниями, $(2k-4)$ входными и k выходными символами, такие, что длина кратчайшей разделяющей последовательности равна $(n-1)2^{k-2}+1=\mathbf{O}(n2^k)$.

Доказательство. Построим конечный автомат $\boldsymbol{S}_k$, $k>2$, с множеством $\{0, \ldots, k-1\}$ состояний и начальным состоянием 1, $2k-4$ входными символами и k выходными символами. Пусть множество состояний есть $\{0, \ldots, k-1\}$, множеством $\{a,e,b_3,c_3, \ldots, b_{k-1},c_{k-1}\}$ входных символов и множество $\{y,y',y_2, \ldots, y_{k-1}\}$ выходных символов. Автомат $\boldsymbol{S}_k$ имеет следующие переходы.

Состояние 0: Под действием всех входных символов осуществляется переход в состояние 0 с выходными символами y и y'.

Состояние 1: Под действием входного символа a осуществляется переход в каждое состояние j с выходным символом y_j для каждого $j=2, \ldots, k-1$; для каждого входного символа $e, b_3, \ldots, b_{k-1}, c_3, \ldots, c_{k-1}$ осуществляется переход в состояние 0 с выходным символом y.

Состояние 2: Под действием входного символа e осуществляется переход в состояние 1 с выходным символом y; для каждого входного символа b_j, $j=3,\ldots,k-1$, осуществляется переход в состояние j с выходным символом y; для каждого входного символа a, c_j, $j=3,\ldots,k-1$, осуществляется переход в состояние 0 с выходным символом y.

При $k \geq 4$:

Состояние 3: Под действием входных символов $a, e, b_4, \ldots, b_{k-1}, c_4,\ldots, c_{k-1}$ осуществляется переход в состояние 0 с выходным символом y; для входного символа b_3 осуществляется переход в состояние 3 с выходным символом y; для входного символа c_3 осуществляется переход в состояние 2 с выходным символом y_2.

При $k \geq 5$:

Состояние *j*, *j*=4,…,*k*–1: Для каждого входного символа $b_3,\ldots, b_j, c_3,\ldots, c_{j-1}$ осуществляется переход в состояние *j* с выходным символом *y*; под действием входного символа c_j осуществляется переход в состояние *p* с выходным символом y_p для каждого $p=2,\ldots,j-1$; для каждого входного символа $a, e, b_{j+1}, \ldots, b_{k-1}, c_{j+1},\ldots, c_{k-1}$ осуществляется переход в состояние 0 с выходным символом *y*.

Автоматы $\boldsymbol{S}_5$ и $\boldsymbol{A}_n$ с $n>1$ состояниями и начальным состоянием 1 показаны на рис. 4 и 5.

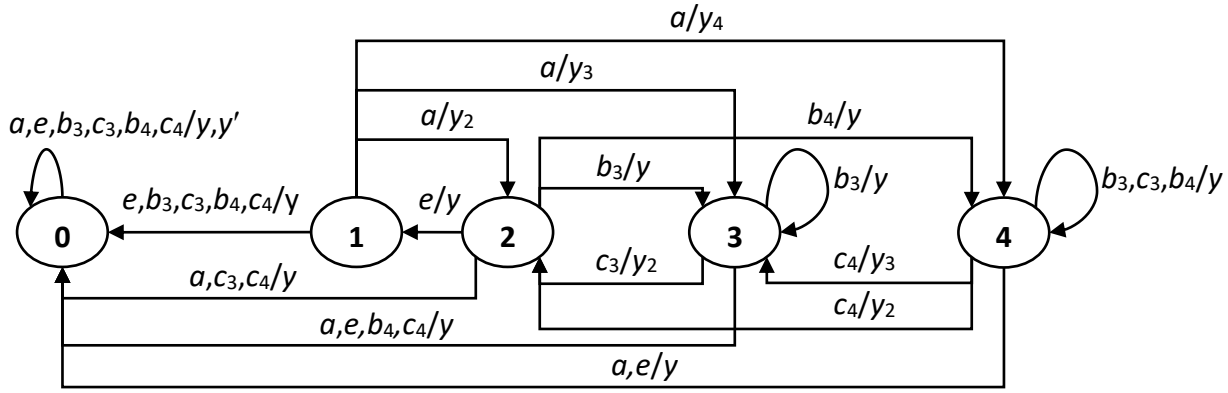


Рис. 4. Автомат $\boldsymbol{S}_5$

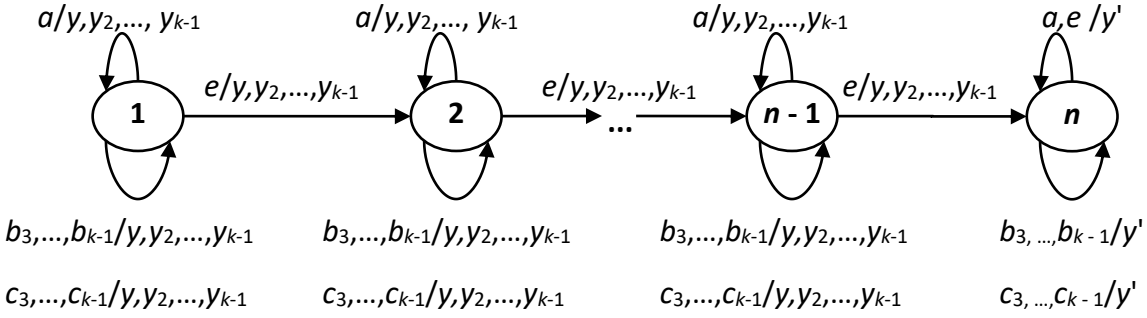


Рис. 5. Автомат $\boldsymbol{A}_n$, $n>1$

По определению автоматов $\boldsymbol{S}_k$ и $\boldsymbol{A}_n$, в автомате $\boldsymbol{A}_n$ в каждом состоянии, кроме состояния *n*, для каждого входного символа есть переход с каждым выходным символом, а в состоянии *n* – только с символом *y*′. В автомате $\boldsymbol{S}_k$ в каждом состоянии, кроме состояния 0, для каждого входного символа есть переходы с выходными символами, отличными от *y*′, а в состоянии 0 для каждой пары из входного и выходного символов есть переход. Поэтому кратчайшая разделяющая последовательность для автоматов $\boldsymbol{S}_k$ и $\boldsymbol{A}_n$ должна иметь вид $\alpha_{k,n}\,x$, где последовательность $\alpha_{k,n}$ переводит автомат $\boldsymbol{A}_n$ в состояние *n*, а автомат $\boldsymbol{S}_k$ гарантированно не переводит в состояние 0, после чего подаётся один (любой) входной символ *x*:

в автомате $\boldsymbol{A}_n$ будет получен выходной символ y', а в автомате $\boldsymbol{S}_k$ гарантированно другой выходной символ. Для определенности будем считать $x=a$.

Для того чтобы последовательность $\alpha_{k,n}$ переводила автомат $\boldsymbol{A}_n$ в состояние n, она должна содержать $n-1$ символов e. В автомате $\boldsymbol{S}_k$ переход по e ведет из состояния 2 в начальное состояние 1, а из любого другого состояния – в состояние 0. Таким образом, последовательность $\alpha_{k,n}$ должна иметь вид $(\beta_k)^{n-1}$, где последовательность β_k – это последовательность, гарантированно переводящая автомат $\boldsymbol{S}_k$ из состояния 1 в состояние 1 с однократным проходом по переходу $(2,e,y,1)$. Покажем, что $\beta_k=a\gamma_k e$, где γ_3 – пустая последовательность, а для $k>3$ $\gamma_k=\gamma_{k-1}b_{k-1}c_{k-1}\gamma_{k-1}$.

В автомате $\boldsymbol{S}_3$ в состоянии 1 есть только переходы $(1,e,y,0)$ и $(1,a,y,2)$, а в состоянии 2 – только переходы $(2,e,y,1)$ и $(2,a,y,0)$. Отсюда непосредственно следует, что $\beta_3=ae$.

При переходе от автомата $\boldsymbol{S}_3$ к автомату $\boldsymbol{S}_4$ добавляются состояние 3 и следующие переходы:

из состояния 0: $(0,b_3,y,0)$, $(0,b_3,y',0)$, $(0,c_3,y,0)$, $(0,c_3,y',0)$;

из состояния 1: $(1,b_3,y,0)$, $(1,c_3,y,0)$;

из состояния 2: $(2,b_3,y,3)$, $(2,c_3,y,0)$;

из состояния 3: $(3,b_3,y,3)$, $(3,c_3,y_2,2)$, $(3,a,y,0)$, $(3,e,y,0)$.

Таким образом, a-преемником начального состояния 1 в автомате $\boldsymbol{S}_4$ является множество {2,3}, из которого только по символу b_3 нет перехода в состояние 0; b_3-преемником множества {2,3} является синглетон {3}, и из состояния 3 только по символам b_3 и c_3 нет перехода в состояние 0. Переход из состояния 3 по символу b_3 является петлей, а по символу c_3 переход ведет в состояние 2. В состоянии 2 имеется переход $(2,e,y,1)$. Поэтому $\beta_4=ab_3c_3e$.

Пусть $k>4$ и $\beta_{k-1}=a\gamma_{k-1}e$ – кратчайшая последовательность, гарантированно переводящая автомат $\boldsymbol{S}_{k-1}$ из состояния 1 в состояние 1 с однократным проходом по переходу $(2,e,y,1)$. Рассмотрим автомат $\boldsymbol{S}_k$ и покажем, что $\beta_k=a\gamma_{k-1}b_{k-1}c_{k-1}\gamma_{k-1}e$. По сравнению с автоматом $\boldsymbol{S}_{k-1}$ в автомат $\boldsymbol{S}_k$ добавлены состояние $k-1$ и следующие переходы:

из состояния 0: $(0,b_{k-1},y,0)$, $(0,b_{k-1},y',0)$, $(0,c_{k-1},y,0)$, $(0,c_{k-1},y',0)$;

из состояния 1: $(1,b_{k-1},y,0)$, $(1,c_{k-1},y,0)$;

из состояния 2: $(2,b_{k-1},y,k\text{-}1)$, $(2,c_{k-1},y,0)$;

из состояния j=3, ..., k–2: $(j,b_{k-1},y,0)$, $(j,c_{k-1},0)$;

из состояния k–1: $(k–1,b_i,y,k–1)$, i=3, …, k–1; $(k–1,c_i,y,k–1)$, i=3, ..., k–2; $(k–1,c_{k-1},y_p,p)$, p=2, …, k–2; $(k–1,a,y,0)$; $(k–1,e,y,0)$.

Таким образом, a-преемником начального состояния 1 в автомате $\boldsymbol{S}_k$ является множество {2,3,…,k–1}; γ_{k-1}-преемником подмножества {2,3,…,k–2} является состояние 2, а γ_{k-1}-преемником состояния (k –1) является состояние (k – 1). Поэтому γ_{k-1}-преемником подмножества {2,3,…,k–1} – множество {2,k–1}; b_{k-1}-преемником множества {2,k–1} – синглетон {k–1}; c_{k-1}-преемником состояния (k–1) – множество {2,3,…,k–2}, которое гарантированно переводится в состояние 1 последовательностью $\gamma_{k-1}e$. Таким образом, последовательность $a\gamma_{k-1}b_{k-1}c_{k-1}\gamma_{k-1}e$ гарантированно переводит автомат $\boldsymbol{S}_k$ из состояния 1 в состояние 1 с однократным проходом по переходу (2,e,y,1).

Покажем, что последовательность $a\gamma_{k-1}b_{k-1}c_{k-1}\gamma_{k-1}e$ есть кратчайшая последовательность с таким свойством. Пусть α – некоторая кратчайшая последовательность, гарантированно переводящая автомат $\boldsymbol{S}_k$ из состояния 1 в состояние 1 с однократным проходом по переходу (2,e,y,1). Для входного символа x справедливо, что x-преемник начального состояния 1 не содержит состояние 0 только, если x=a. Поэтому последовательность α должна начинаться символом a. В состояние 1 в автомате $\boldsymbol{S}_k$ ведет единственный переход (2,e,y,1), а переходы по e из других состояний ведут в состояние 0. Поэтому последовательность α должна заканчиваться символом e и не содержать других вхождений этого символа. Заметим также, что переход по входному символу a из любого состояния, кроме состояния 1, ведет в состояние 0. Поэтому $\alpha=a\alpha_1e$, где α_1 не содержит символов a и e. По определению, a-преемником начального состояния 1 в автомате $\boldsymbol{S}_k$ является множество {2,3,…,k–1}. Пусть α_2 – максимальный префикс α_1, не содержащий символов b_{k-1} и c_{k-1}. Тогда α_2-преемник множества {2,3,…,k–1} содержит состояние (k–1) и хотя бы одно из состояний 2, 3, …, k–2, т. е. $\alpha_1=\alpha_2x\alpha_3$, где x=b_{k-1} или x=c_{k-1}. Поскольку в каждом из состояний 2, 3, …, k–2 переход по c_{k-1} ведет в состояние 0 и в каждом из состояний 3, …, k–2 переход по b_{k-1} ведет в состояние 0, x=b_{k-1}, и α_2-преемник множества {2,3,…,k–1} есть множество {2,k–1}. Поскольку в состояние

(k–1) из других состояний можно перейти только по переходу (2,b_{k-1},y,k–1), α_2-преемник множества {2,3,…,k–2} есть синглетон {2}. Поскольку $\beta_{k-1}=a\gamma_{k-1}e$ – кратчайшая последовательность, гарантированно переводящая автомат $\boldsymbol{S}_{k-1}$ из состояния 1 в состояние 1 с однократным проходом по переходу (2,e,y,1), последовательность α_2 не короче последовательности γ_{k-1}. Из состояния (k–1) переходы в состояния, отличные от состояний k–1 и 0, суть переходы (k–1,c_{k-1},y_p,p), p=2,…,k–2. Следовательно, последовательность α_3 должна начинаться с символа c_{k-1}, т. е. $\alpha_3=c_{k-1}\alpha_4$. Кроме того, c_{k-1}-преемником множества {2,k–1} является множество {2,3,…,k–2}, для которого кратчайшая последовательность, гарантированно переводящая автомат $\boldsymbol{S}_k$ в состояние 2, есть последовательность γ_{k-1}. Поэтому последовательность α_4 не короче последовательности γ_{k-1}, и, соответственно, последовательность $\alpha=a\alpha_2b_{k-1}c_{k-1}\alpha_4e$ не короче последовательности $a\gamma_{k-1}b_{k-1}c_{k-1}\gamma_{k-1}e$. Следовательно, $a\gamma_{k-1}b_{k-1}c_{k-1}\gamma_{k-1}e$ есть кратчайшая последовательность, гарантированно переводящая автомат $\boldsymbol{S}_k$ из состояния 1 в состояние 1 с однократным проходом по переходу (2,e,y,1).

Соответственно, последовательность $(a\gamma_k e)^{n-1}a$, длина которой равна $(n-1)2^{k-2}+1=\mathbf{O}(n2^k)$, является кратчайшей разделяющей для автоматов $\boldsymbol{A}_n$ и $\boldsymbol{S}_k$.

□

Для доказательства теоремы 5 построим полуавтоматы, по которым по алгоритму 1 строятся автоматы $\boldsymbol{S}_k$ и $\boldsymbol{A}_n$. Для этого введем в каждый полуавтомат для каждой пары <состояние, входной символ>, т. е. пары (s,i), введем дополнительное состояние (s,i), т. е. множеством состояний полуавтомата $\boldsymbol{S}_k$ является объединение множеств {0,…,k–1} и {(s, i): $s\in${0,…,k–1}, $i\in\{a,e,b_3,c_3,\ldots,b_{k-1},c_{k-1}\}$}; множеством состояний автомата $\boldsymbol{A}_n$ является объединение множеств {1,…,n} и {(s,i): $s\in${1,…,n}, $i\in\{a,e,b_3,c_3,\ldots,b_{k-1},c_{k-1}\}$}. Переходу ($s$,$i$,$o$,$s'$) в соответствующем полуавтомате соответствуют два перехода (s,i,(s,i)) и ((s,i),o,s'). Таким образом, число состояний полуавтомата, построенного по автомату $\boldsymbol{S}_k$, равно $k(2k-3)$; число состояний полуавтомата, построенного по автомату $\boldsymbol{A}_n$, равно $n(2k-3)$.

Согласно теореме 4, множество разделяющих последовательностей для построенных полуавтоматов совпадает с таковым для соответствующих автоматов, т. е. кратчайшая разделяющая последовательность имеет длину $(n-1)2^{k-2}+1$.

□

Если полуавтоматы $\boldsymbol{S}$ и $\boldsymbol{P}$ могут быть ненаблюдаемыми, то соответствующие автоматы M_S и M_P могут оказаться ненаблюдаемыми. В работе [6] предложен алгоритм построения разделяющей последовательности для таких автоматов: в этом случае соответственно изменяется определение *io*-преемника состояния, который не обязательно является синглетоном. Поскольку алгоритм также основан на построении подмножеств, то можно ожидать, что оценка на длину кратчайшей разделяющей последовательности совпадет с таковой для наблюдаемых автоматов.

Еще один вопрос касается адаптивной различимости. Адаптивная разделяющая последовательность для двух конечных автоматов представляется в виде тестового примера, т. е. подходящего ациклического автомата, и методы построения соответствующих тестовых примеров для наблюдаемых автоматов известны [8]. В этом случае высота тестового примера, т. е. длина самой длинной входной последовательности, приложенной в процессе различающего эксперимента, не превышает величины mn, таким образом, адаптивные разделяющие последовательности являются более эффективными при тестировании на основе модели «белого ящика». Однако, если автоматы могут быть ненаблюдаемыми, то необходимы дополнительные исследования по синтезу и оценке длины адаптивной разделяющей последовательности.

ЗАКЛЮЧЕНИЕ

В настоящей работе мы исследуем отношение разделимости для входо-выходных полуавтоматов специального класса, которые достаточно часто используются в качестве спецификаций для описания поведения управляющих систем. При тестировании реализаций таких систем на основе модели «белого ящика» в спецификацию вносятся подходящие мутации, и в процессе тестирования (эксперимента с реализацией системы) требуется определить наличие внесенных мутаций. При наличии для каждой пары (спецификация, мутант) (адаптивной) разделяющей последовательности не требуется выполнения «всех погодных условий», т. е. каждая такая последовательность подается на проверяемую систему только один раз. С другой стороны, однократная подача различающей последовательности приводит к увеличению длины такой последовательности, и направлениями

наших дальнейших исследований являются как анализ других отношений различимости для входо-выходных полуавтоматов, так и исследование других классов полуавтоматов на предмет построения (адаптивных) разделяющих последовательностей.

СПИСОК ЛИТЕРАТУРЫ

1. *Kam T., Villa T., Brayton K.R., Sangiovanni-Vincentelli A*. Synthesis of FSMs: Functional Optimization. Springer. 1997. 282 p.

2. *Бурдонов И.Б., Косачев А.С., Кулямин В.В.* Теория соответствия для систем с блокировками и разрушением. Наука. Глав. ред. физ.-мат. лит., 2008. 412 с.

3. *Tretmans J.* A formal approach to conformance testing // The Intern. Workshop on Protocol Test Systems. 1993. P. 257–276.

4. *Starke P.* Abstract Automata. American Elsevier, 1972. 419 p.

5. *Гилл А.* Введение в теорию конечных автоматов. Наука, 1966. 272 с.

6. *Kushik N., Yevtushenko N., Cavalli A.R.* On Testing against partial nondeterministic machines // Intern. Conf. on the Quality of information and Communications Technology. 2014. P. 230–233.

7. *Евтушенко Н., Кушик Н.* Некоторые задачи идентификации состояний для недетерминированных автоматов. СТТ, 2018. 190 с.

8. *Petrenko A., Yevtushenko N.* Conformance Tests as Checking Experiments for Partial Nondeterministic FSM // Lecture Notes in Computer Science. 2005. V. 3997. P. 118–133.

9. *Kushik N., Yevtushenko N., Burdonov I., Kossachev A.* Synchronizing and Homing Experiments for Input/output Automata // System Informatics. 2017. No 10. P. 1–10.

SEPARATING INPUT/OUTPUT AUTOMATA WITH NONDETERMINISTIC BEHAVIOR

I. Burdonov, N. Yevtushenko, A. Kossachev

Ivannikov Institute for System Programming of the Russian Academy of Sciences, Moscow

igor@ispras.ru, evtushenko@ispras.ru, kos@ispras.ru

Abstract

When deriving tests for checking functional and nonfunctional requirements for components of control systems, the notion of separablity becomes very important that is used for distinguishing the fault-free component from a faulty one. In order to do this, proper separating sequences are utilized. Such sequences are well studied for complete and deterministic Finite State Machines but components of control systems can be only partially described and their behavior can be nondeterministic. In this paper, we consider the formal model of Input/Output automata, introduce the notion of a separating sequence for two such automata and propose an approach for deriving such a separating sequence.

Keywords*: Input/Output automaton, test derivation, separating sequence*

REFERENCES

1. Kam T., Villa T., Brayton K.R., Sangiovanni-Vincentelli, A. Synthesis of FSMs: Functional Optimization. Springer, 1997. 282 p.

2. *Burdonov I.B., Kossachev A.S., Kuliamin V.V.* Conformance theory for systems with blocking and destruction. Nauka, 2008. 412 p.

3. *Tretmans J.* A formal approach to conformance testing // The Intern. Workshop on Protocol Test Systems. 1993. P. 257–276.

4. *Starke P.* Abstract Automata. American Elsevier, 1972. 419 p.

5. *Gill A.* Introduction to automata theory. Nauka, 1966. 272 p.

6. *Kushik N., Yevtushenko N., Cavalli A.R.* On Testing against partial nondeterministic machines // Intern. Conf. on the Quality of information and Communications Technology. 2014. P. 230–233.

7. *Yevtushenko N., Kushik N.* Some state identification problems for nondeterministic Finite State Machines. CTT, 2018. 190 p.

8. *Petrenko A., Yevtushenko N.* Conformance Tests as Checking Experiments for Partial Nondeterministic FSM // Lecture Notes in Computer Science. 2005. V. 3997. P. 118–133.

9. *Kushik N., Yevtushenko N., Burdonov I., Kossachev A.* Synchronizing and Homing Experiments for Input/output Automata // System Informatics. 2017. No 10. P. 1–10.

СВЕДЕНИЯ ОБ АВТОРАХ

БУРДОНОВ Игорь Борисович – ведущий научный сотрудник Института системного программирования им. В.П. Иванникова РАН. Сфера научных интересов – моделирование и верификация программных систем, теория графов, теория автоматов.

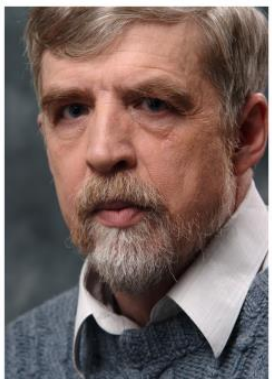
Igor Borisovich BURDONOV – Leading Researcher, Ivannikov Institute for System Programming of the RAS. Research interests - modeling and verification of software systems, graph theory, automata.

email: igor@ispras.ru

ЕВТУШЕНКО Нина Владимировна – ведущий научный сотрудник Института системного программирования им. В.П. Иванникова РАН. Сфера научных интересов – теория автоматов, моделирование, верификация и верификация программных систем, телекоммуникационные протоколы и сервисы.

Nina Vladimirovna YEVTUSHENKO – leading researcher of Ivannikov Institute for System Programming of the RAS. Research interests – automata theory, modeling, verification and testing of software systems, telecommunication protocols and services.

email: evtushenko@ispras.ru



КОСАЧЕВ Александр Сергеевич – ведущий научный сотрудник Института системного программирования им. В.П. Иванникова РАН. Сфера научных интересов – моделирование и верификация программных систем, теория графов, теория автоматов.

Alexander Sergeevich KOSSACHEV – Leading Researcher, Ivannikov Institute for System Programming of the RAS. Research interests - modeling and verification of software systems, graph theory, automata.

email: kos@ispras.ru

Материал поступил в редакцию 07 ноября 2019 года

УДК 519.6

ИССЛЕДОВАНИЕ СХОДИМОСТИ ЧИСЛЕННЫХ МЕТОДОВ РЕШЕНИЯ ЗАДАЧ С ОПЕРАТОРОМ СМЕШАННОГО ТИПА В НЕОГРАНИЧЕННОЙ ОБЛАСТИ

М. П. Галанин[1, 2], Д. Л. Сорокин[1, 2]

[1] Институт прикладной математики им. М. В. Келдыша РАН, г. Москва;

[2] Московский государственный технический университет имени Н.Э. Баумана, г. Москва

galan@keldysh.ru, sorokin.dmitr@yandex.ru

Аннотация

Проанализированы методы решения задач, базирующиеся на основной интегральной формуле Грина. Предложены новые методы решения задачи с оператором смешанного типа в неограниченной области. На основе этих методов созданы программы для решения задач с оператором смешанного типа. Приведены результаты вычислительных экспериментов, показывающие корректность применения методов.

Ключевые слова*: неограниченная область, оператор смешанного типа, электродинамические ускорители рельсового типа, система уравнений Максвелла в квазистационарном приближении*

ВВЕДЕНИЕ

При исследовании физических явлений часто возникает необходимость проводить моделирование в неограниченной области, например, при моделировании электростатического поля зарядов, решении задачи теплопроводности и т. п. В случае, когда явление можно описать с помощью простейших линейных эллиптических операторов, проблема решена и описана, например, в [1, 2]. Постановка неотражающих граничных условий для волновых уравнений подробно рассмотрена в [3]. Однако для более широкого класса задач, для которых оператор задачи может иметь смешанный тип, но вне некоторой финитной области, фундаментальное решение оператора известно и легко вычисляется, вышеупомянутые методы неприменимы.

В частности, задачи с операторами смешанного типа возникают при моделировании электромагнитного поля в электродинамических ускорителях рельсового типа. Процесс протекания тока в проводниках в этом случае может быть описан параболическим уравнением, а электромагнитное поле в диэлектрике — эллиптическим [4, 5].

1. МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ В НЕОГРАНИЧЕННОЙ ОБЛАСТИ

Для решения задачи предлагается три метода решения: метод расширения области, метод задания интегральных граничных условий, трёхэтапный метод [6].

При решении задач с помощью сеточных численных методов точность решения определяется шагом сетки и точностью задания граничных условий. Если при решении задачи в конечной области на границе поставить те же граничные условия, что и на бесконечности в исходной задаче, то ошибка, определяемая заданием граничного условия, будет, как правило, уменьшаться при увеличении размера расчётной области. В связи с этим для решения задач в неограниченной области можно использовать метод расширения области, т. е. можно найти такой размер расчётной области, при котором влияние искусственной границы будет меньше, чем применение численного метода.

Метод расширения расчётной области прост в реализации, но имеет высокую вычислительную сложность [6]. Альтернативными методами являются методы, основанные на использовании формул Грина.

Первым из них является метод задания интегрального граничного условия [6]. Он основан на том, что вне некоторой финитной области действует оператор с известным фундаментальным решением и тривиальной правой частью, тогда решение на границе расчётной области определяется с помощью основной интегральной формулы Грина. Таким образом, решение задачи с помощью метода задания интегральных граничных условий требует решения системы уравнений с заполненными строками, соответствующими границе.

В [6] построен трёхэтапный метод решения. При использовании трёхэтапного метода необходимо решать две системы, но с разреженными строками, относящимися к граничным условиям. Результаты расчётов показывают, что при решении задач, сводящихся к решению линейных уравнений с сильно заполнен-

ными матрицами, метод неэффективен, однако для задач с разреженной матрицей системы трёхэтапный метод позволяет получить результат существенно быстрее, чем метод задания интегрального граничного условия.

2. МОДИФИКАФИКАЦИЯ ТРЁХЭТАПНОГО МЕТОДА, ПОСТРОЕНИЕ ИТЕРАЦИОННОГО МЕТОДА РЕШЕНИЯ ЗАДАЧИ В НЕОГРАНИЧЕННОЙ ОБЛАСТИ

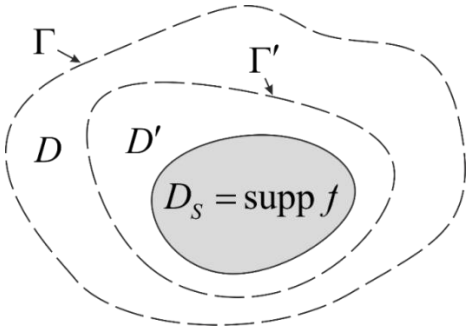


Рис. 1. Структура расчётной области

Рассмотрим подробнее трёхэтапный алгоритм [6] на примере решения задачи с оператором смешанного типа:

$$\begin{cases} u_t - a^2 \Delta u = f, \ \mathbf{r} \in D_S, \\ \Delta u = 0, \ \mathbf{r} \in \mathbb{R}^2 \backslash D_S, \end{cases} \tag{1}$$

где ∂D_S – граница области D_S (рис. 1).

После применения метода конечных разностей для аппроксимации производной по времени вместо системы (1) необходимо решить задачу:

$$\begin{cases} \Delta \hat{u} - \dfrac{1}{a^2 \tau} \hat{u} = -\dfrac{1}{a^2} \hat{f} - \dfrac{u}{a^2 \tau}, & \mathbf{r} \in D_S, \\ \Delta \hat{u} = 0, & \mathbf{r} \in \mathbb{R}^2 \backslash D_S. \end{cases} \tag{2}$$

В последующих записях решаемых задач во временных производных используется аппроксимация (2).

На первом этапе трёхэтапного метода предлагается решать систему с запаздывающими по времени граничными условиями:

$$\begin{cases} v_t - a^2 \Delta \hat{v} = f, \ \mathbf{r} \in D_S; \\ \Delta \hat{v} = 0, \ \mathbf{r} \in D \backslash D_S, \\ \hat{v}\big|_{\Gamma} = u\big|_{\Gamma}, \\ v(t_{i-1}) = u. \end{cases}$$

На втором этапе предполагается, что решение $\hat{v}$ в области D_S незначительно отличается от решения $\hat{u}$ исходной задачи (2), причём отличие связано с ошибкой в задании граничного условия. Исходя из этих соображений и основной интегральной формулы Грина, поставим следующую задачу:

$$\begin{cases} u_t - a^2\Delta\hat{u} = f,\ \mathbf{r} \in D_S; \\ \Delta\hat{u} = 0,\ \mathbf{r} \in D'\backslash D_S, \\ \hat{u}\big|_{\Gamma'} = \hat{v}_i + \int\limits_{\Gamma}\left(\Phi_P\dfrac{\partial\hat{v}_i}{\partial\mathbf{n}} - \hat{v}_i\dfrac{\partial\Phi_P}{\partial\mathbf{n}}\right)dS, \\ u(t_{i-1}) = u. \end{cases}$$

На третьем этапе согласно основной интегральной формуле Грина восстановим значение $\hat{u}$ на границе Γ по значению $\hat{u}$ в области D':

$$\hat{u}(P) = \int\limits_{\Gamma_S}\left(\Phi_P^L\frac{\partial\hat{u}}{\partial\mathbf{n}} - \hat{u}\frac{\partial\Phi_P^L}{\partial\mathbf{n}}\right)dS.$$

Главное преимущество трёхэтапного алгоритма заключается в том, что без использования итерационного процесса он позволяет получить результаты высокой точности [6, 7]. Однако наличие второго этапа заведомо создаёт ошибку, которая не устраняется итерационным повторением этапов 1–3 алгоритма и может расти с уменьшением шага интегрирования по времени τ. Для устранения данного недостатка метода предлагается исключить второй этап метода, а первый и третий этапы выполнять несколько раз на каждом временном слое.

Первый этап нового итерационного алгоритма

$$\begin{cases} u_t^k - a^2\Delta\hat{u}^k = f,\ \mathbf{r} \in D_S; \\ \Delta\hat{u}^k = 0,\ \mathbf{r} \in D\backslash D_S, \\ \hat{u}^k\big|_{\Gamma} = \hat{g}^{k-1}\big|_{\Gamma}, \\ u(t_{i-1}) = u,\ k = 1..N. \end{cases}$$

На первой итерации в качестве граничного условия предлагается выбрать запаздывающее граничное условие

$$\hat{g}^0\big|_{\Gamma} = g\big|_{\Gamma}.$$

Второй этап алгоритма

$$\hat{g}^k(P) = \int\limits_{\Gamma_S} \left(\Phi_P^L \frac{\partial \hat{u}^k}{\partial \mathbf{n}} - \hat{u}^k \frac{\partial \Phi_P^L}{\partial \mathbf{n}} \right) dS.$$

3. ПРИМЕНЕНИЕ ИТЕРАЦИОННОГО АЛГОРИТМА К РЕШЕНИЮ ЗАДАЧ В НЕОГРАНИЧЕННОЙ ОБЛАСТИ

Применим построенный алгоритм к решению следующей задачи:

$$\begin{cases} u_t - a^2 \Delta u = f, \ r \le R_0, \\ \Delta u = 0, \ r > R_0, \end{cases} \tag{3}$$

$$f(r,\varphi,t) = a^2 \left(\sum_{k=2}^{2+m} \left(1 - r^2 \lambda_n^2 - k^2\right) r^{k-2} y_k - C_1 r \lambda_n^2 \right) e^{-\lambda_n^2 a^2 t} \cos\varphi.$$

Начальное условие (при $r \le R_0$):

$$u(r,\varphi,0) = \left(\sum_{k=2}^{m+2} \left(y_k r^k \right) + C_1 r - \frac{2c}{\lambda_n R_0^2 (J_0(R_0\lambda_n) - J_2(R_0\lambda_n))} J_1(r\lambda_n) \right) \cos\varphi.$$

Здесь m и n – задаваемые параметры; J_j – функция Бесселя *j*-го порядка ($j = 0,1,2$); $\lambda_n = \frac{\mu_n}{R_0}$, где μ_n – *n*-й корень уравнения $J_1(\mu_n) = 0$; $y_k = \frac{(-1)^k C_m^{k-2}}{R_0^{k-2}\left(k^2 - 1\right)}$;

$\gamma = 2$; $C_1 = -\frac{1}{1+\gamma} \sum_{k=2}^{m+2} \left((k\gamma + 1) y_k R_0^{k-1} \right)$; $C_2 = \sum_{k=2}^{m+2} \left(y_k R_0^{k+1} \right) + C_1 R_0^2$; $c = C_2 \left(1 - \frac{1}{\gamma} \right)$.

Аналитическое решение имеет вид:

$$u(r,\varphi,t) = \begin{cases} \left(\sum_{k=2}^{m+2} \left(y_k r^k \right) + C_1 r + \frac{2c}{\lambda_n R_0^2 (J_0(R_0\lambda_n) - J_2(R_0\lambda_n))} J_1(r\lambda_n) \right) e^{-\lambda_n^2 a^2 t} \cos\varphi, \ r \le R_0; \\ \frac{C_2}{r} e^{-\lambda_n^2 a^2 t} \cos\varphi, \ r > R_0. \end{cases}$$

Зададим $m = 1$, $n = 2$, $a = 0.1$, $R_0 = 1.0$. Решим задачу с помощью нового итерационного метода и трёхэтапного алгоритма. В таблице 1 представлены результаты расчётов. Видим, что все рассмотренные методы приводят к правильному результату.

Таблица 1. Результаты решения тестовой задачи (3) при $R_0 = 1.0$, $R = 2.0$, $\tau = 0.001$, $h = 0.2$, $T = 0.008$

$\frac{h^*}{\tau}$	Число шагов	$\frac{\|u_h - u\|_C}{\|u\|_C}$	p	$\frac{\|u_h - u\|_C}{\|u\|_C}$	p	$\frac{\|u_h - u\|_C}{\|u\|_C}$	p	$\frac{\|u_h - u\|_C}{\|u\|_C}$	p

	по времени	аналитическое задание ГУ		применение трёхэтапного алгоритма		применение итерационного алгоритма (1 итерация)		применение итерационного алгоритма (2 итерации)	
h τ	8	$3.15114\cdot10^{-3}$ (за 0.2108 с.)		$3.90629\cdot10^{-3}$ (за 0.489305 с.)		$3.20485\cdot10^{-3}$ (за 0.267368 с.)		$3.25578\cdot10^{-3}$ (за 0.555425 с.)	
h / 2 τ /4	32	$4.79565\cdot10^{-4}$ (за 1.15328 с.)	2.72	$5.05743\cdot10^{-4}$ (за 2.32202 с.)	2.95	$4.50521\cdot10^{-4}$ (за 1.36682 с.)	2.83	$4.57497\cdot10^{-4}$ (за 2.66011 с.)	2.83
h / 4 τ /16	128	$6.82737\cdot10^{-5}$ (за 8.14104 с.)	2.81	$6.8335\cdot10^{-5}$ (за 15.6243 с.)	2.89	$8.60114\cdot10^{-5}$ (за 9.01307 с.)	2.39	$7.39768\cdot10^{-5}$ (за 17.7113 с.)	2.63
h / 8 τ /64	512	$1.75524\cdot10^{-5}$ (за 114.472 с.)	1.96	$1.75524\cdot10^{-5}$ (за 233.351 с.)	1.96	$2.65021\cdot10^{-5}$ (за 133.145 с.)	1.69	$2.32554\cdot10^{-5}$ (за 260.535 с.)	1.67
h/16 τ /256	2048	$4.61259\cdot10^{-6}$ (за 2959.46 с.)	1.93						

Здесь $p=\log_2\left(\left(\|u_h-u\|_C/\|u\|_C\right)\Big|_{h^*}\Big/\left(\|u_h-u\|_C/\|u\|_C\right)\Big|_{h^*/2}\right)$ – величина, характеризующая порядок метода, $(\ \cdot\ )\big|_{h^*}$ – значение, вычисленное на сетке с шагом h^*.

Заметим, что итерационный процесс можно было бы построить и на основе трёхэтапного алгоритма [6]. Однако согласно результатам таблицы 2 это не поможет за меньшее время получить ответ с требуемой точностью, т. к. одна итерация трёхэтапного метода содержит в себе почти в два раза больше вычислительных операций, чем количество операций нового итерационного метода.

Изучая таблицы 1 и 2, можно заметить, что построенные итерационные методы имеют высокую скорость сходимости, пока расстояние от приближения $\hat{u}^k$ до истинного решения велико. В окрестности же решения итерационные процессы не сходятся, что подтверждают результаты численных расчётов (таблица 3).

Таблица 2. Сходимость итерационных методов при решении задачи (3) в случае выбора на каждом шаге в качестве начального приближения нулевого вектора $\tau=1.5625\cdot10^{-5},\ h=0.025,\ T_0=0.000125$. Ошибка при аналитическом способе задания граничных условий равна $8.45492\cdot10^{-6}$

Число итераций	$\dfrac{\|u_h-u\|_C}{\|u\|_C}$ (итерационный метод)	$\dfrac{\|u_h-u\|_C}{\|u\|_C}$ (итерационный процесс на базе трёхэтапного метода)
1	$3.82473\cdot10^{-1}$	$1.39064\cdot10^{-1}$
2	$1.25412\cdot10^{-1}$	$1.65792\cdot10^{-2}$

4	$1.34993 \cdot 10^{-2}$	$2.42915 \cdot 10^{-4}$
8	$1.80879 \cdot 10^{-4}$	$1.07342 \cdot 10^{-5}$
16	$2.53095 \cdot 10^{-5}$	$1.07263 \cdot 10^{-5}$
32	$2.53009 \cdot 10^{-5}$	—
64	$2.53275 \cdot 10^{-5}$	—

Таблица 3. Сходимость итерационного метода при $R_0 = 1.0,\ R = 2.0,\ \tau = \dfrac{0.001}{64},$ $h = \dfrac{0.2}{8},\ T = 0.008$

	$\dfrac{\|u_h - u\|_C}{\|u\|_C}$ аналитическое задание ГУ	$\dfrac{\|u_h - u\|_C}{\|u\|_C}$ применение итерационного алгоритма (1 итерация)	$\dfrac{\|u_h - u\|_C}{\|u\|_C}$ применение итерационного алгоритма (2 итерации)	$\dfrac{\|u_h - u\|_C}{\|u\|_C}$ применение итерационного алгоритма (3 итерации)	$\dfrac{\|u_h - u\|_C}{\|u\|_C}$ применение итерационного алгоритма (4 итерации)
Ошибка	$1.75524 \cdot 10^{-5}$	$2.65021 \cdot 10^{-5}$	$2.32554 \cdot 10^{-5}$	$2.45878 \cdot 10^{-5}$	$4.61259 \cdot 10^{-5}$
Время (с.)	114.472	133.145	260.535	359.556	486.5

Одним из способов улучшить скорость сходимости в окрестности решения (таблица 4) является изменение второго этапа итерационного метода, например,

$$\hat{g}^k(P) = \frac{1}{2}\left(\hat{g}^{k-1}(P) + \int\limits_{\Gamma_S} \left(\Phi_P^L \frac{\partial \hat{u}^k}{\partial \mathbf{n}} - \hat{u}^k \frac{\partial \Phi_P^L}{\partial \mathbf{n}} \right) dS \right).$$

Таблица 4. Сходимость модифицированного итерационного метода при $R_0 = 1.0,$ $R = 2.0,\ \tau = \dfrac{0.001}{64},\ h = \dfrac{0.2}{8},\ T = 0.008$

	$\dfrac{\|u_h - u\|_C}{\|u\|_C}$ аналитическое задание ГУ	$\dfrac{\|u_h - u\|_C}{\|u\|_C}$ применение итерационного алгоритма (1 итерация)	$\dfrac{\|u_h - u\|_C}{\|u\|_C}$ применение итерационного алгоритма (2 итерации)	$\dfrac{\|u_h - u\|_C}{\|u\|_C}$ применение итерационного алгоритма (4 итерации)	$\dfrac{\|u_h - u\|_C}{\|u\|_C}$ применение итерационного алгоритма (8 итерации)
Ошибка	$1.75524 \cdot 10^{-5}$	$2.65021 \cdot 10^{-5}$	$2.51139 \cdot 10^{-5}$	$2.4427 \cdot 10^{-5}$	$2.42558 \cdot 10^{-5}$
Время (с.)	114.472	133.145	236.651	468.127	925.113

Стоит также отметить, что вновь построенный метод менее требователен к размеру вспомогательной подобласти $D \backslash D_S$, что позволяет уменьшить её размер, а тем самым уменьшить вычислительную сложность задачи.

ЗАКЛЮЧЕНИЕ

На основе метода задания интегральных граничных условий и трёхэтапного метода построены и программно реализованы вычислительные алгоритмы для решения ряда задач в неограниченной области. Результаты вычислительных экспериментов подтверждают корректность методов. Также они свидетельствуют о том, что если порядок точности квадратурных формул, используемых при реализации методов, согласован с порядком разностной схемы, то порядок схемы сохраняется.

СПИСОК ЛИТЕРАТУРЫ

1. *Tsynkov S.V.* Numerical solution of problems on unbounded domains. A review // Appl. Math. 1998. V. 27. P. 465–532.

2. *Галанин М.П., Низкая Т.В.* Разработка и применение численного метода решения линейных эллиптических уравнений в неограниченной области // Препринты ИПМ им. М.В. Келдыша, 2005. № 2. 29 с.

3. *Ильгамов М.А., Гильманов А.Н.* Неотражающие условия на границах расчётной области. М.: Физматлит. 2003. 240 с.

4. *Галанин М.П., Попов Ю.П.* Квазистационарные электромагнитные поля в неоднородных средах: Математическое моделирование. М.: Наука. Физматлит, 1995. 320 с.

5. *Галанин М.П., Лотоцкий А.П., Попов Ю.П., Храмцовский С.С.* Численное моделирование пространственно трехмерных явлений при электромагнитном ускорении проводящих макротел // Математическое моделирование. 1999. Т. 11. № 8. С. 3–22.

6. *Галанин М.П., Сорокин Д.Л.* Разработка и применение численных методов решения задач в неограниченной области на основе третьей формулы Грина // Препринты ИПМ им. М.В. Келдыша, 2018. № 246. 24 с.

7. *Галанин М.П., Сорокин Д.Л.* Разработка и применение численных методов решения задач с оператором смешанного типа в неограниченной области // Дифференциальные уравнения. М.: МАИК «Наука/Интерпериодика», 2019. Т. 55. № 7. С. 949–961.

CONVERGENCE RESEARCH OF NUMERICAL METHODS FOR SOLVING PROBLEMS WITH MIXED TYPE OPERATOR IN AN UNLIMITED AREA

M. P. Galanin[1, 2], D. L. Sorokin[1, 2]

[1] *Keldysh Institute of Applied Mathematics (KIAM), Moscow*

[2] *Bauman Moscow State Technical University (BMSTU), Moscow*

sorokin.dmitr@yandex.ru

Abstract

Methods for solving problems of elliptic equations, based on the third Green formula, was analyzed. New methods for solving a problem with a mixed-type operator in an unbounded domain are proposed. On the basis of the proposed methods, programs for solving problems with a mixed type operator have been created. The results of computational experiments, showing the correctness of the application of methods, are presented.

Keywords*: unlimited area, mixed operator, electrodynamic accelerator, railgun, electromagnetic field, Maxwell's equations*

REFERENCES

1. *Tsynkov S.V.* Numerical solution of problems on unbounded domains. A review // Appl. Math. 1998. V. 27. P. 465–532.

2. *Galanin M.P., Nizkaia T.V.* Razrabotka i primenenie chislennogo metoda resheniia lineinykh ellipticheskikh uravnenii v neogranichennoi oblasti // Preprinty IPM im. M.V. Keldysha, 2005. № 2. 29 s.

3. *Ilgamov M.A., Gilmanov A.N.* Neotrazhaiushchie usloviia na granitsakh raschetnoi oblasti. M.: Fizmatlit, 2003. 240 s.

4. *Galanin M.P., Popov Iu.P.* Kvazistatsionarnye elektromagnitnye polia v neodnorodnykh sredakh: Matematicheskoe modelirovanie. M.: Nauka. Fizmatlit, 1995. 320 s.

5. *Galanin M.P., Lototskii A.P., Popov Iu.P., Khramtsovskii S.S.* Chislennoe modelirovanie prostranstvenno trekhmernykh iavlenii pri elektromagnitnom uskorenii provodiashchikh makrotel // Matematicheskoe modelirovanie. 1999. T. 11. No 8. S. 3–22.

6. *Galanin M.P., Sorokin D.L.* Razrabotka i primenenie chislennykh metodov resheniia zadach v neogranichennoi oblasti na osnove tretei formuly Grina // Preprinty IPM im. M.V. Keldysha, 2018. № 246. 24 s.

7. *Galanin M.P., Sorokin D.L.* Development and Application of Numerical Methods for Equations of Mixed Type in an Unbounded Domain // Differential Equations. Pleiades Publishing. 2019. T. 55. No 7. S. 915–928.

СВЕДЕНИЯ ОБ АВТОРАХ

ГАЛАНИН Михаил Павлович – д. ф.-м. н., гл. н. с., и.о. заведующего отделом Института прикладной математики им. М.В. Келдыша РАН, профессор Московского государственного технического университета им. Н.Э. Баумана. Автор более 240 печатных работ в области вычислительного эксперимента, вычислительной математики, математического моделирования многомерных нестационарных процессов в сплошных средах.

Mikhail Pavlovich GALANIN – Doctor of science, Head of Department of Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences, Professor of Bauman Moscow State University. Author of more than 240 publications in the field of computational experiment, computational mathematics, mathematical simulation of multidimensional non-stationary processes in continuous media.

E-mail: galan@keldysh.ru

СОРОКИН Дмитрий Леонидович — младший научный сотрудник Института прикладной математики им. М. В. Келдыша РАН, ассистент Московского государственного технического университета им. Н.Э. Баумана. Область исследований: численное моделирование нестационарных электромагнитных полей.

Dmitry Leonidovich SOROKIN – Junior Researcher of Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences, Assistant of Bauman Moscow State University. Scientific interest: mathematical simulation of non-stationary electrodynamic fields.

E-mail: sorokin.dmitr@yandex.ru

Материал поступил в редакцию 15 ноября 2019 года

УДК 004.43 БК 22.18 Г70

СИСТЕМАТИЗАЦИИ ПАРАДИГМ ПРОГРАММИРОВАНИЯ ПО ПРИОРИТЕТАМ ПРИНЯТИЯ РЕШЕНИЙ

Л. В. Городняя

Институт систем информатики им. А.П. Ершова Сибирского отделения Российской академии наук, Новосибирский государственный университет, г. Новосибирск

lidvas@gmail.com

Аннотация

Цель статьи – описание методики сравнения парадигм и языков программирования, отражающей выразительную силу языков, трудоёмкость реализации систем программирования и приспособленность к обоснованию практичных, объективных критериев декомпозиции программ, что можно рассматривать как подход к решению проблемы факторизации весьма усложнённых определений языков программирования и систем их поддержки. Представлены результаты анализа наиболее известных основных парадигм программирования и намечен подход к навигации в современном расширяющемся пространстве языков программирования. Систематизация парадигм учитывает особенности постановок задач программирования и семантические характеристики языков и систем программирования с акцентом на критерии качества программ и приоритеты в принятии решений при их реализации и обучении программистов.

Ключевые слова*: определение языков программирования, парадигмы программирования, классификация сложных определений, семантические системы*

ВВЕДЕНИЕ

Понятие «парадигмы программирования» (ПП) не имеет строгого определения, поэтому возникает вопрос о принадлежности новых подходов в программировании и ИТ к множеству ПП и об упорядочении такого множества. Парадигма программирования проявляется как образ мышления, связанный с компромиссом между особенностями решаемых задач, методами их решения в

форме программ, принятыми в ПП критериями качества программ и приоритетами принятия решений в процессе программирования. Такая особенность ПП позволяет понимать выбор парадигмы как процесс принятия, представления и отладки решений при постановке разных задач, поэтому естественно систематизацию ПП выполнить по сопоставлению с приоритетами и варьированием схем постановки задач и методов их решения. Наиболее ясная систематизация ПП в настоящее время позволяет выделять основные и производные ПП, дополненные комбинированными, вспомогательными и системообразующими или перспективно-стратегическими. Следует отметить, что академик Андрей Петрович Ершов основное внимание уделял именно стратегическим ПП, включающим фундаментальные, образовательные и технологичные. Множество основных ПП можно разделить на базовые, инструментально расширяющие и неограниченные в зависимости от наполнения семантических систем организации вычислений, работы с памятью, управления вычислениями и конструирования сложных данных.

Описания современных языков программирования (ЯП) обычно содержат список из 5–10 языков-предшественников и ряд ПП, поддержанных языком [1, 2]. Такая характеристика как правило не показывает, какие свойства ПП и черты предшественника фактически унаследованы определяемым ЯП и какие особенности являются действительно новыми. Для реальной оценки уровня новизны и практичности ЯП может быть полезной коллекция постановок задач с примерами их решения на разных языках в рамках различных парадигм. Определённое количество примеров обычно присутствует в описаниях ЯП и учебных пособиях. Чаще всего это программа печати приветствия, что позволяет быстро начать эксперименты с системой программирования (СП), но, учитывая, что большинство современных ЯП поддерживает механизмы ввода-вывода на уровне библиотечных вызовов, такие примеры слабо характеризуют язык.

В данной статье рассматривается систематизация парадигмальных особенностей определения ЯП на уровне семантических систем [3], классифицированных по постановкам задач и языковым средствам, используемым при их решении. Ещё в давние времена Николас Вирт отмечал важность соответствия поста-

новки задачи и используемого для её решения инструмента, особенно если удаётся уловить подобие обрабатываемых структур данных и алгоритмов их обработки, что теперь называют гомоиконностью. На основе такого соответствия можно выстроить пространство конструкций, поддержанных в определениях языков и систем программирования (ЯСП) и сопоставленных со сложностью постановок успешно решаемых задач. Полученное пространство может быть исходной структурой при выборе критериев декомпозиции программ с учётом особенностей развития постановок задач в процессе программирования их решений [4], расширения семантических систем ЯП и их уточнения при реализации СП [5].

Методика показана на материале четырёх классических базовых парадигм программирования в рамках ЯП высокого уровня без экскурса в языки низкого и сверх высокого уровней и распространена на более широкое пространство основных парадигм, особенно новых, ещё не получивших поддержки в достаточно известных языках программирования и признания в виде примеров отлаженных программных продуктов. Выполнен поверхностный анализ языков организации параллельных вычислений (ПВ), заслуживающих не менее двух парадигм, и DSL-языков, вероятно знаменующих переход прикладного программирования на качественно новый уровень.

Автор выражает благодарность участникам неформальной дискуссии о понятии «парадигма программирования», организованной Ан.В. Климовым 26-го сентября в рамках XXI Всероссийской конференции «Научный сервис в сети Интернет**».**

РЕЗУЛЬТАТЫ ПАРАДИГМАЛЬНОГО АНАЛИЗА

Анализ и сравнение большого числа ЯП разного уровня позволили выделить наиболее существенные характеристики для выражения парадигмальной специфики широкого класса новых ЯП (см. Таблицу 1)[1].

Интересно отметить, что парадигмальная характеристика многих долгоживущих ЯП исторически претерпела заметные изменения (см. Таблицу 2). Можно

[1] Перечни в Таблицах 1 и 2 опираются на открытые источники типа Википедий и учебных пособий, имеют предварительный характер. Перечни могут быть пополнены и уточнены. специалистами при более строгом изложении.

сказать, что отдельные парадигмы проявились постепенно, по мере накопления опыта решения определённых классов задач.

Таблица 1. Языки программирования XXI века (все мультипарадигмальные)

Год	*ЯП*	*Предшественники*[2]	*Поддержанные парадигмы*
2018	Dart	Java, JavaScript, CoffeeScript, Go	объектно-ориентированный каркас веб-приложений сценарный язык императивный рефлективный функциональный
2016	Kotlin	Java, Scala, Groovy, Gosu, C#, Python, ML	объектно-ориентированный язык JVM
2014	Swift	Objective-C, C++, Java, Rust, Scala, D, Python, Ruby, LLVM, Smalltalk, Groovy,	протоколо-ориентированный объектно-ориентированный функциональный императивный
2012	Rust	Alev, C++, Camlp4, C#, Common Lisp, NIL, Erlang, Haskell, Ruby, Hermes, Scheme, Napier, Napier88, Newsqueak, Sather, OCaml, Cyclone, Standard ML, Swift	параллельный функциональный императивный структурный системный процедурный свободное программное обеспечение
2009	Go	C, Limbo, Modula, Oberon, Pascal	императивный, на уровне значений, процедурный, скалярный, структурный
2007	Clozure	Lisp, ML, Haskell, Erlang, Prolog, Scheme, Java, Ruby	функциональный гомоиконный
2005	F#	OCaml, C#, Haskell	функциональный объектно-ориентированный обобщённый, императивный
2003	Scala	Java, Haskell, Erlang, Lisp, Standard ML, OCaml, Smalltalk, Scheme, Algol68	функциональный объектно-ориентированный императивный
2001	D	Си, C++, C#, Python, Ruby, Java, Eiffel	императивный объектно-ориентированный функциональный контрактный обобщённый, процедурный
2000	C#	C++, Java, Delphi, Модула-3, Smalltalk	объектно-ориентированный обобщённый процедурный функциональный событийный рефлективный

[2] Сохранена лексика источников.

Таблица 2. ЯП – основатели базовых парадигм программирования

Год	*ЯП*	*Поддержанные парадигмы*	*Сфера влияния*
1954 1958	Fortran, Algol-60 [3]	императивный параллельный процедурный модульный структурный процедурный обобщённый объектно-ориентированный	**ИПП** – **императивно-процедурное** ALGOL 58, BASIC, C, Chapel, CMS-2, Fortress, PL/I, PACT I, MUMPS, IDL, Ratfor
1958	Lisp	экспериментальный функциональный объектно-ориентированный процедурный рефлективный метапрограммирование	**ФП** – **функциональное** Clips, Common lisp, CLOS, Clu, Dylan, Forth, Scheme, Erlang, Haskell, Logo, Lua, Perl, POP-2, Python, Ruby, Cmucl, Scala, ML, Swift, Smalltalk, Factor, Clojure, Emacs Lisp, Eulisp, ISLISP, Wolfram Language
1960	APL	векторный функциональный структурный модульный	**ПВ** – **параллельные вычисления** A, A+, FP, J, K, LyaPAS, Nial, S, MATLAB, PPL, Wolfram Language
1962	Simula 67	объектно-ориентированный	**ООП (1980)** – **объектно-ориентированное**
1968	Forth[4]	императивный стек-ориентированный	Factor, RPL, REBOL, PostScript, Factor и другие конкатенативные языки 5
1968	Algol-68[6]	параллельный императивный	C, C++, Bome shell, KornShell, Bash, Steelman, Ada, Python, Seed7, Mary, S3
1972	Prolog	декларативный логический	**ЛП** – **логическое** Visual Prolog, Mercury, Oz, Erlang, Strand, KL0, KL1, Datalog
1970	Pascal	императивный структурный	**Структурное** Ada, Component Pascal, Modula-2, Java, Go, Oberon, Object Pascal, Oxygene, Seed7, VHD, Structured text

Мультипарадигмальность большинства долгоживущих и новых ЯП показывает необходимость более точной детализации зависимостей между старыми и новыми ЯСП. Наиболее объективные понятия программирования связаны с архитектурными моделями, с методами реализации СП и с классификацией реша-

[3] Algol-60 – именно с этого языка в нашей стране началось знакомство с языками высокого уровня, доминирование его оттеснилось появлением реализаций языка Fortran.

[4] Forth – типовой механизм реализации работы с выражениями в разных ЯП.

[5] Языки, в которых программы строятся как конкатенации функций.

[6] Algol-68 представляет результат хорошо продуманной унификации и ортогонализации основных понятий программирования.

емых задач. Для показа особенностей ПП удобно выделять концептуальные монопарадигмальные ЯП или подъязыки и приводить критерии успешного применения ПП с оценкой результатов на примерах программ, подтверждающих признание практикой программирования [5].

Таблица 3. Наиболее востребованные ЯП (все мультипарадигмальные)

Год	*ЯП*	*Предшественники*	*Поддержанные парадигмы*	*Сфера влияния*
1972	C	B (BCPL, CPL), Algol-68, Assembly, PL/I, FORTRAN	императивный на уровне значений процедурный скалярный структурный	Numerous, AMPL, AWK, csh, C++, C--, C#, Objective-C, D, Go, Java, JavaScript, Julia, Limbo, LPC, Perl, PHP, Pike, Processing, Python, Ring, Rust, Seed7, Vala, Verilog (HDL), Nim, Cyclone, BitC
1974	SQL	Datalog	декларативный строгий	Agena, CQL, LINQ, Windows, PowerShell, ABAP
1983	Objective-C	Smalltalk, C	объектно-ориентированный рефлексивно-ориентированный	Java, Objective-J, Swift
1983	C++	C, Simula, Algol 68, CLU, ML, Ada	объектно-ориентированный процедурный метапрограммирование процедурный функциональный обобщённый язык свободной формы	C#, D, Falcon, Java, Lua, Perl, Pike, Python, Rust, Vala
1987	Erlang	ML, Miranda, Ada, Modula-2, CHILL, Prolog	параллельный функциональный декларативный открытое программное обеспечение свободное программное обеспечение	Rust, Sparkel
1991	Python	ABC, Modula-3, Lisp, Tcl, Smalltalk, C, Java, Icon	объектно-ориентированный рефлективный императивный функциональный аспектно-ориентированный динамический	Ruby, Boo, Groovy, ECMAScript, CoffeeScript, Swift, Nim

Продолжение Таблицы 3. Наиболее востребованные ЯП (все мультипарадигмальные)

Год	*ЯП*	*Предшественники*	*Поддержанные парадигмы*	*Сфера влияния*
1991	Html	SGML	Язык разметки и переписывания	shtml
1993	R	Common Lisp, S, Scheme, XLispStat	векторный императивный аспектно-ориентированный процедурный функциональный рефлективный	Julia
1995	Java	C++, C, C# Ada, Simula 67, Mesa, Smalltalk, Objective-C, Object Pascal, UCSD Pascal, Oberon, Eiffel, Modula-3, Simula	императивный на уровне значений объектно-ориентированный процедурный скалярный	C#, Ceylon, D, E, ECMAS, cript, Groovy, Process, ing, Scala, Vala, X10
995	JavaScript	Lua, Self, Си, Scheme, Perl, Python, Java, AWK, HyperTalk	объектно-ориентированный (прототипный) обобщённый функциональный императивный аспектно-ориентированный событийно-ориентированный	Objective-J, Dart, TypeScript
1995	Ruby	Ada, Dylan, Perl, Python, Smalltalk, C++, Клу, Eiffel, Lisp, Basic, Lua	функциональный императивный рефлективный	Falcon, Groovy, Objeck
1995	PHP	Perl, C, C++, Java, Tcl	сценарный, процедурный объектно-ориентированный императивный интерпретируемый свободное программное обеспечение	Falcon

Из огромного множества можно выделить небольшое число ЯП, привлекающих внимание интересными сочетаниями изобразительных средств и особенностями семантики, влияющими на развитие основных ПП.

Таблица 4. Интересные ЯП (все мультипарадигмальные)

Год	*ЯП*	*Предшественники*	*Поддержанные парадигмы*	*Сфера влияния*
1969	Setl	Algol-60	императивный процедурный, структурный, объектно-ориентированный	SETL2, ISETL, SETLX, ABC
1983	Sisal	VAL, Pascal, C, Fortran	функциональный, потоков данных	Haskell, SAC
1987	Perl	Си, AWK, Shell, Sed, Lisp	императивный объектно-ориентированный функциональный	Ruby, PHP, Groovy
1990	Haskell	ML, Standard ML, Lazy ML, Miranda, Lisp, Scheme, ISWIM, FP, АПЛ, Hope, Hope+, SISAL, Orwell*, Id	функциональный, ленивый, модульный	Agda, Bluespec, Clojure, C#, Cat, Cayenne, Clean, Curry, Epigram, Escher, F#, Factor, Isabelle, Java Generics, Idris. LINQ, Mercury, Ωmega, Perl 6, Python, Qi, Scala, Swift, Timber, Visual Basic 9.0
1991	Visual basic	QuickBasic, BASIC	процедурный объектно-ориентированный компонентно-ориентированный событийно-ориентированный	Visual Basic .NET, REALbasic, Gambas, Xojo, Basic4ppc
1993	LUA	C++, CLU, Modula, Scheme, SNOBOL	императивный функциональный объектно-ориентированный (прототипный) скриптовый встраиваемый	GameMonkey, Io, JavaScript, Julia, MiniD, Red, Ring, Ruby, Squirrel, MoonScript, C--
1993	Brainfuck	FALSE	императивный на уровне значений эзотерический	BrainSub, Brainfork, Brainloller, COW, Ook, Pbrain, Smallfuck, Spoon, LOLCODE, Whitespace, DoubleFuck, Feckfeck
1995	Delphi	Object Pascal, C++	императивный структурированный объектно-ориентированный компонентно-ориентированный высокоуровневый	C#, Java
1996	Ocaml	Standart ML, Caml Light	функциональный объектно-ориентированный императивный	F#, JoCaml, MetaOCaml, OcamlP3l
2002	Falcon	*C++, Lisp, Lua, PHP, Perl, Python, Ruby, Smalltalk*	обмен сообщениями объектно-риентированный прототипный, процедурный функциональный, табличный	
2003	Groovy	Java, Ruby, Python, Perl, Smalltalk	объектно-ориентированный императивный, сценарный	
2012	TypeScript	JavaScript, C#	объектно-ориентированный обобщённый функциональный императивный, прототипный аспектно-ориентированный событийно-ориентированный	

СЕМАНТИЧЕСКИЕ СИСТЕМЫ ОСНОВНЫХ ПАРАДИГМ

Рассматривая любые семантические системы, важно отметить разницу в характере выполнения функций таких систем в различных контекстах и варианты в выборе методов их реализации. Так, для любого множества данных D, представляющих значения V произвольной природы, реализационно различимы схемы функций для методов вычислений E, средств доступа к памяти M через адреса N, особенностей управления вычислениями C и коммуникации или обратимой комплексации и структурирования данных S. Это приводит к представлению об основных категориях семантических систем различно реализуемых схем функций F. Исторически на уровне аппаратуры такие категории систем обладали кумулятивным эффектом в порядке «DEMCS» (см. Таблицу 5) – представление чисел, арифмометр, калькулятор, дифференциальный вычислитель, компьютер, причём, аппаратные подсистемы могут взаимодействовать каждая с каждой. Парадигмы программирования можно отличать по приоритетам выбора решений при определении категорий семантических систем в процессе программирования, отмечая парадигмальные различия общих понятий в каждой модели, зависящие от критериев качества программ.

Таблица 5. Ряд категорий семантических систем аппаратного уровня

Подсистема	*Примечание*
D: данные	Данные из множества D представляют значения из V и шкалу прерываний
E: вычисления	Операции по двум-одному значению производят одно или два значения
M: память	Соответствие между адресами из множества N и хранимыми по этим адресам представлениями из множества D для значений из V допускает разные методы доступа к элементам памяти, включая замену хранимых значений, за исключением адреса 0
C: управление	Сравнение значений с нулём позволяет управлять ходом вычислений наряду с передачами по меткам и обработкой прерываний, не считая перехода по порядку
S: коммуникации	Конструирование сложных данных учитывает возможности команд адресации в памяти

Такая разница может быть показана для классических базовых ПП (см. Таблицы 6–9). Для ИПП данные – это адреса и хранимые коды значений, в ООП появляются хранимые методы и сигнатуры объектов, в ФП вместо адресов в памяти может использоваться связывание с любым значением, а в ЛП – с идентификатором. В ИПП и ООП операции в основном унарные или бинарные, а в ФП и ЛП

имеется и произвольная арность. Истинностные значения в ЛП включают специальное значение «ESC», позволяющее отличать нормальные значения предикатов от неуспеха в вычислениях, а ФП может в качестве истины использовать любое значение отличное от пустого списка – «NIL». Структуры данных в ИПП могут не рассматриваться как значения, обрабатываемые базовыми средствами, а в ФП такие структуры обрабатываются без особых ограничений. Можно отметить, что базовые ПП различают представления значений, формул, сигналов и контекстов.

При подготовке императивно-процедурной программы доминирует критерий эффективности, понимаемый как оптимальное соотношение пространственно-временных характеристик программы. Поэтому важнейшими считаются средства работы с памятью, в которой размещаются данные и результаты их обработки (1: M). Управление процессом обработки представляется с помощью конструкций ветвления, использующих операции сравнения и предикаты над данными (2: C). Обработка данных рассматривается как изменение состояний памяти в порядке выполнения вычислений (3: E). При необходимости можно реорганизовывать структуру данных и программы (4: S) (см. Таблицу 6).

Таблица 6. Парадигмальная шкала ИПП **(MCES)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Значения ограничены размерами их представлений в регистрах памяти по адресам из N. Шкала прерываний не представлена
E: вычисления	3	Операции различаются на унарные и бинарные с одним результатом
M: память	1	Работа с памятью без акцента на особенности нуля, разнообразие методов доступа к памяти и обработки прерываний
C: управление	2	Кроме аппаратного сравнения значений с нулём, при управлении ходом вычислений используются приоритеты операций и скобки в выражениях наряду с передачами по меткам, но без обработки прерываний
S: комплексация	4	Можно конструировать сложные данные и выбирать их элементы, используя возможности команд индексной адресации в памяти

В центре внимания ФП полнота семейства методов организации вычислений, доступных для эксперимента. Поэтому всё начинается с выбора базовых средств для обработки символьных представлений сущностей заданной пред-

метной области (1: E). Конструирование сложных объектов освобождено от обязательности соседства элементов (2: S). Работа с памятью в таком случае может не требовать привязки к физическим адресам, а ограничиться представлением ассоциирующей функции над парами данных любой природы (3: M). Управление процессом вычислений может рассматриваться как функция над фрагментами программы, рассматриваемыми как разновидность значений (4: C) (см. Таблицу 7).

Таблица 7. Парадигмальная шкала ФП **(ESMC)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Представления значений не ограничены по размеру и сложности, включая функции
E: вычисления	1	Некоторые операции могут обрабатывать любое число параметров и вырабатывать ряд значений, при необходимости объединяемых в сложное данное
M: память	3	При обработке сложных данных старые значения не изменяются, а новые значения располагаются в памяти независимо, причём соответствие между ассоциированными данными хранится в памяти, допуская изменение ассоциации
C: управление	4	Любое вычисление можно заблокировать или запустить. Программа может содержать точки ветвления из произвольного числа ветвей, выбираемых сравнением результата с «нулём» (NIL), входящим в множество значений
S: структуры	2	Можно конструировать сложные, равноправные с элементарными, данные, все элементы которых доступны с помощью функций в любом выражении

В случае ЛП важно показать существование хотя бы одного полезного решения задачи. Это приводит к доминированию логики недетерминированного поиска выполнимых решений (1: C). Формально вычисляются варианты возможных решений (2: E) в произвольном порядке. Но в качестве структур используются образцы (3: S), позволяющие управлять выбором вариантов. Именуются фрагменты с фиксированным числом параметров (4: M), от необходимости имён которых техника образцов освобождает (см. Таблицу 8).

Ведущий критерий качества программ в ООП – сопоставимость иерархии классов объектов с иерархией понятий в области приложения программ, позволяющая готовые программные решения пополнить новыми для расширения прежней области приложения. Поэтому здесь всё начинается с определения

иерархии классов объектов (1: S), размещаемых по фиксированным адресам в памяти (2: M), применяемым как указатели ради достижения некоторого уровня эффективности. Управление процессом обработки данных использует сопоставление классов объектов и допустимых методов обработки объектов, размеченных правами доступа из разных частей программы (3: C). Вычисления происходят лишь при успешном сопоставлении и соответствии прав доступа к объектам (4: E) (см. Таблицу 9).

Таблица 8. Парадигмальная шкала ЛП **(CESM)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Представления фактов или правил
E: вычисления	2	Попытки вычисления, дающего или результат, или сигнал о не выполнимости, что приводит к дальнейшему перебору вариантов
M: память	4	Некоторые правила могут иметь имена с указанием числа параметров, что позволяет их использовать как функции
C: управление	1	Недетерминированный перебор вариантов для выбора выполнимого варианта, дающего результат, отличный от «ESC»
S: шаблоны	3	Сопоставление сложных данных с образцом позволяет выделять элементы для выбора ветви вычислений

Таблица 9. Парадигмальная шкала ООП **(SMCE)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Данные представляют не только значения, но и методы их обработки
E: вычисления	4	Операции бинарные и унарные, также как любые вычисления, можно перегрузить, добавив обработку возможных прерываний
M: память	2	Дозированный доступ к элементам объектов сопровождается механизмами неявных обработчиков ситуаций, адреса могут быть значениями
C: управление	3	Выполнимость методов над объектами обусловлена проверкой их совместимости и прав доступа по иерархии классов
S: структуры	1	Классы объектов приспособлены к доопределению и наследованию по иерархии классов

Подробный анализ семантики ООП, сопровождённый сопоставлением с другими ПП и частичной формализацией основных механизмов реализации ЯП, представлен в работе [6].

Показывая отличия в схемах определения функций для разных категорий семантических систем в зависимости от ПП, связанных с различием критериев

качества программ и приоритетов используемых в их реализации средств, следует отметить, что переход от ЯП к СП обычно сопровождается расширением числа поддержанных ПП. При определении языка Haskell это привело к понятию «монада», позволяющему любому ЯП достигать практичности на уровне СП, что обычно и выполнялось с помощью библиотечных модулей.

Кроме сравнительно ясных классических базовых ПП, есть основания выделять основные инструментальные системно **расширяющие** ПП, нацеленные на подготовку и конструирование программ, операционных систем (ОС) и баз данных (БД), поддержку работы с файлами и разнообразными конфигурациями устройств, а также обеспечение обратной связи при выполнении любых программ (см. Таблицы 10–13).

Все *расширяющие* ПП, некоторые из них ещё не получили свои имена, работают со много более сложными элементами, обладающими своей жизнью, допускающими включение во многие системы и конфигурации, в которых возможно изменение их состояния. Представления данных включают в себя, кроме сложных структур данных, формальных определений и кодов, процессы, устройства, роли участников и комплексы. Методы обработки элементов и их взаимодействия подчинены более жёстким требованиям правильности, что влечёт поддержку улучшения элементов по частям, то есть целенаправленного развития по мере выявления ошибок или необходимости в повышении эффективности. Имеет место разделение труда по уровню квалификации и ответственности.

Прежде всего это ПП, поддерживающие системное программирование, использующее синтаксически и грамматико ориентированную обработку определений ЯП при конструировании СП (VDM, BNF в инструментах типа Lex/YACC и их аналогах во многих новых ЯП), метапрограммирование (Рефал), языки и системы программирования и разработки программных инструментов (Bliss, ЯРМО, С), создание промежуточных форм для поддержки особо сложных работ (внутренний язык системы БЕТА), отладки программ и их тестирования.

В области разработки СП характерно деление на разработчиков СП и применяющих СП программистов, работающих в рамках базовых ПП и при необходимости подключающих расширяющие ПП в форме побочных эффектов библиотечных модулей. Появление большого количества DSL-языков на базе технологии

Clang-LLVM показывает новый уровень сформированности парадигм системного программирования, стирающий эту границу. Предметно-ориентированные DSL-языки заслуживают отдельного рассмотрения именно как новый уровень программистского языкотворчества. Если при развитии аппаратуры успешный опыт накапливается в форме системы команд, а в обычных ЯП накопление опыта программирования выполняется в форме отлаженных процедур, то конструирование DSL – механизм накопления опыта решения задач применения ИТ в форме языков программирования, что допускает включение такого механизма в инструментально расширяющие ПП.

Таблица 10. Парадигмальная шкала средств разработки СП **(ECMS)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	В качестве значений выступают определения реализуемого ЯП и система команд целевой машины, на которой предстоит выполнять программы, подготовленные на базе данного ЯП
E: вычисления	1	Синтаксически управляемый автомат с переводом строит переход от ЯП к его абстрактному представлению, более удобному для дальнейшей обработки программ и кодогенерации. Константные вычисления и символьные преобразования могут выполняться в этом процессе и заменяться на результат с целью повышения эффективности программ
M: память	3	Заполняется встроенная база данных, формируемая как таблицы, сопоставляющие адреса или идентификаторы отдельным конкретизациям понятий, выделяемых при анализе текста, или их атрибутам. Возможно привлечение промежуточных представлений для решения сложных технических задач. Строится схема программы, которая может быть достаточной для её выполнения или для кодогенерации
C: управление	2	Проверка текстов на принадлежность ЯП сопровождается диагностическими сообщениями и, возможно, рекомендациями по продолжению разбора текста. Результаты лексического, синтаксического и семантического анализа проверяются на соответствие шаблонам предстоящей кодогенерации. Определение ЯП работает подобно представлению типа данных. Кодогенерация может выполняться автономно. При выполнении созданной программы могут быть дополнительные проверки правильности вычислений.
S: структуры	4	Используются графовые и кодовые представления понятий, выделяемых при анализе текстов на принадлежность ЯП и выводе атрибутов, полезных для предстоящей кодогенерации и исполнения программы

Происходившее автономно развитие средств и методов работы с базами данных в настоящее время активно интегрируется в общий контекст ИТ, создаёт реализационную поддержку методам искусственного интеллекта и представле-

ния знаний с привлечением экспертов (GPS, RDF, OWL), является основой мощного пространства производственных и бизнес технологий, особенно на базе интернет-сервисов. Сформировано разделение труда на администраторов, ответственных за хранение данных, и клиентов, интересующихся их содержанием, способных его применять и уточнять.

Отдельные методы из этой сферы, изначально накапливающей решения по надёжности обработки данных, перемещаются в пространство решений при создании новых языков программирования, особенно поддерживающих параллелизм.

Таблица 11. Парадигмальная шкала СУБД **(MSEC)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Данные представляют объекты реального мира и их характеристики, а также сведения о правах доступа к данным
E: вычисления	3	Операции обеспечивают формирование запросов, направленных на доступ к накопленным данным, включая их обработку
M: память	1	Хранение содержательных представлений в форме отношений дополняется использованием имён, индексов, функций, позволяющих повышать эффективность доступа к данным. Долговременным хранилищем данных являются файлы. На время обработки данных создаётся их временная копия в оперативной памяти.
C: управление	4	Кроме прав доступа к хранимым данным используются предикаты и логические выражения. Имеются рекомендации по учёту так называемых «нормальных форм», позволяющих повысить эффективность хранения данных и доступа к ним, причём некоторые из них дают одновременный выигрыш по скорости и по объёму памяти.
S: структуры	2	Отношения между хранимыми данными могут быть представлены как записи, организованные в таблицы, графы, иерархии (деревья), сети

Работы по операционным системам весьма осторожно переходят к использованию языков высокого уровня, преимущественно скриптовых и интерпретируемых, пригодных для реагирования в динамике реального мира. Возникает смежная линия компонентного программирования (Com/Dcom, Corba, SOAP, .Net), поддерживающая ООП при решении вопросов, требующих повышенной квалификации. Обычно разделяют обычный и привилегированный доступ (Таблица 12)

Таблица 12. Парадигмальная шкала ОС **(CMSE)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Данные представляют задания, процессы, файлы, каталоги, устройства, протоколы и настроечные значения, а также права доступа.
E: вычисления	4	Работу операций выполняют доступные из командной строки команды уровня ядра, дополняемые представлениями программируемых заданий уровня оболочки
M: память	2	Основное хранение данных в файлах сопровождается использованием очередей, приоритетов, шкалы допустимых прерываний и настроечных значений, размещаемых при загрузке системы и допускающих уточнение привилегированным системным администратором
C: управление	1	Система функционирует как взаимодействие приаппаратного ядра и пользовательской оболочки. Имеется контроль прав доступа с учётом приоритетов и успеха-провала в ранее выполненных действиях, включая наличие-отсутствие необходимых устройств и файлов. Выполнение заданий инициируется с уровня командной строки.
S: структуры	3	Основная структура – иерархия файлов, дополненная очередями, строками, каналами и кодами, включая данные о регистрации пользователей и их правах доступа

Расширение спектра новой аппаратуры, пригодной для массового применения без технического сопровождения, привела к задачам формирования программно-аппаратных многопроцессорных комплексов для расширения сфер применения ИТ, как правило силами группы технической поддержки. (Таблица 13).

Не менее заметно выделяется группа **неограниченных** коммуникационно интерфейсных ПП, поддерживающих обработку большеобъёмных данных (bigdata, sematic-web, rdf), дистанционную работу в сетях, сервис-ориентированное программирование на базе языков разметки и переписывания (html, XML, PHP), параллельные, векторно-ориентированные для обработки массивов (APL) или поддерживающие теоретико-множественные инсерционное механизмы, включая динамические вставки-замены (SETL) и высокопроизводительные вычисления на суперкомпьютерах (OpenMP, mpC) и мобильных устройствах (см. Таблицы 14–17).

Языков программирования, поддерживающих неограниченные ПП, пока создано немного, но возможно их число будет возрастать, пока не получит удоб-

ной формы синхронизация процессов над общей памятью. За редким исключением потребность в неограниченных ПП реализуется в форме отдельных специализированных программных инструментов.

Таблица 13. Парадигмальная шкала многопроцессорных комплексов **(SECM)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Данные представляют процессоры и программные модули, организованные в комплекс совместно эксплуатируемого оборудования, конфигурация которого может динамически изменяться
E: вычисления	2	Потоки действий и реагирование на сигналы от оборудования, программ и операторов, выработка сигналов и проявление реальных характеристик комплекса
M: память	4	Распределение оборудования и дозированный доступ к ресурсам и процессорам, обладающим ограниченным временем жизни, зависящим от включения-выключения и времени хранения сигналов
C: управление	3	Выполнимость действий обусловлена проверкой условий готовности устройств, наличием ресурсов, временем хранения данных в буферах памяти и пропускной способностью процессоров, допускающих параллельное использование
S: структуры	1	Конфигурации устройств, включая произвольное число процессоров и программ, и потоки действий, позволяющих ими оперировать

Переход к обработке большеобъёмных данных пока развивается на уровне специализированных языков и инструментов, практику применения которых несколько сдерживает отсутствие методов декомпозиции, преобразования и проектирования сложных слабо структурированных формирований, обладающих трудно контролируемой динамикой обновления. Большинство решений сводится к выбору визуализации графов (см. Таблицу 14).

Таблица 14. Парадигмальная шкала большеобъёмных данных – bigdata **(EMSC)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Большие объёмы данных, возможно неструктурированных, представляют на многих машинах в сети, алгоритмы анализа таких данных могут иметь высокую временную сложность и содержательное разнообразие. Используются формы представления графов с большим числом вершин.
E: вычисления	1	Операции визуализации и анализа большеобъёмных данных дополнены средствами свёртки и редукции (MapReduce)для повышения наглядности и оптимизации, а также навигации с использованием поисковиков
M: память	2	Доступ к информационным ресурсам, хранимым на разных серверах, допускающих манипулирование большеобъёмными данными с некоторыми гарантиями условий хранения в стиле грид и облачных технологий
C: управление	4	Выполнимость отдельных методов манипулирования большеобъёмными данными и их визуализацией постепенно приводит к формированию практичных технологий и инструментов, технологических возможностей анализировать огромные массивы данных преимущественно средствами массово-параллельной обработки неопределённо структурированных данных и поддержки информационно-технологических решений
S: структуры	3	Графовые структуры и реальные конфигурации машин, видимых в сети с выделением серверов, регулирующих доступ к информационным ресурсам, или предоставляющим некоторые простаивающие мощности для общего употребления

Всеохватность методов дистанционного доступа резко влияет на качество жизни большинства слоёв населения, что сопряжено с вопросами социальной ответственности. Проявляются эффекты механизмов типа моды и рекламы. (см. Таблицу 15).

Постановки задач ПВ учитывают существование областей приложения, в которых недостаточна скорость получения результатов по доступным программам решения конкретных задач. Парадигмы этого направления находятся в стадии формирования из-за сложности перехода к масштабируемым решениям языков сверх высокого уровня, допускающим автоматическую настройку на реальные конфигурации оборудования, а также из-за отсутствия традиции представлять течение времени в математических моделях ЯП. Некоторые проблемы решает фрагментное программирование, предлагающее декомпозировать программу на схему управления и наполняющие её фрагменты. (см. Таблицу 16).

Таблица 15. Парадигмальная шкала дистанционного доступа **(MECS)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Данные представляют заполненные в соответствии с разными форматами из тэгов, наполнения и адресов, информационные ресурсы и сведения об их владельцах и заинтересованных лицах
E: вычисления	2	Операции встраиваются и показываются в виде меню, состав которых может регулироваться специальными настройками
M: память	1	Доступ к элементам данных подчинен некоторой дисциплине, внешне маскируемой разными изобразительными средствами, включая цвет, звуки и форматы интерфейса. Реально используются неявные механизмы дублирования и восстановления данных, обеспечивающие стабильность функционирования
C: управление	3	Выполнимость методов обработки данных отчасти зависит от динамики функционирования серверов в сети, но в некоторых пределах возможно управление маршрутизацией
S: структуры	4	Сети, включающие в себя сервера, способные поддерживать маршруты доступа к данным в сети, адресуемые ресурсы, окна, страницы, линейки, шкалы-визуализаторы процессов

Таблица 16. Парадигмальная шкала параллельных вычислений **(CSME)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Данные представляют действия, спусковые условия их выполнения и обрабатываемые значения и результаты, включая события, происходящие при выполнении действий и в окружающем контексте
E: вычисления	4	Операции, кроме обычных арифметических над значениями и доступа к памяти соответственно дисциплине, позволяют формировать сети и комплексы и извлекать из них составляющие, а также просачивать действия относительно структур
M: память	3	Память может быть неоднородной, элементы которой подчинены разным дисциплинам доступа и временным характеристикам реагирования
C: управление	1	Выполнимость действий регулируется спусковыми условиями, возможно зависящими от успеха предшествующих действий. Порядок одновременно выполнимых действий зависит от реализации, включая возможности синхросетей
S: структуры	2	Комплексы данных и сети действий, связанных со спусковыми условиями срабатывания и наполнением входных данных. Сеть может быть иерархической и структурированной, не исключая связи между уровнями и схемы синхронизации функционирования подсетей (синхросети)

Высокопроизводительные вычисления (ВПВ) расширяют спектр методов

управления вычислениями и критериев эффективности программ благодаря возможности использования процессорных резервов и наследования ранее отлаженных программ, как правило, посягая на их неприкосновенность не только методами распараллеливания, но и прямым редактированием. Это позволяет рассматривать ВПВ как бесспорный полигон для признания потенциала методов верификации, есть основания такие методы рассматривать как важную составляющую неограниченных парадигм.

В последние годы проявляются причины обуславливать конструирование средств верификации программ формализацией используемых ПП, а программистские проекты сопровождать обоснованием выбора не только инструментария, но и ПП, чтобы избегать межпарадигмальных конфликтов, чреватых трудно уловимыми ошибками, связанными с изменением обстановки функционирования программируемых компонентов. (см. Таблицу 17).

Таблица 17. Парадигмальная шкала суперкомпъютерных вычислений **(SCEM)**

Подсистема	*Приоритет*	*Примечание*
D: данные	0	Данные представляют готовые программы и измерительные модули, включая датчики энергопотребления, временные параметры срабатывания действий, события по срабатыванию разных действий
E: вычисления	3	Измерение производительности, выделение в программе критических участков, сравнение производительности, распараллеливание программ, приведение программ к распараллеливаемой форме, протоколирование событий и характеристик производительности или загрузки оборудования и программ, верификация схем и фрагментов программы
M: память	4	Неоднородная многоуровневая память, возможно с каналами связи с процессорами, дисциплиной доступа и коммуникацией по обновлению хранимых данных, включая их копирование и восстановление (транзакции и сборка мусора)
C: управление	2	Учёт результатов верификации и измерения производительности, эффективности и энергопотребления при распределении ресурсов и потоков действий, прогон программ по протоколу, реагирование на события согласно регламенту или спецификации,
S: структуры	1	Сети и схемы, приспособленные к включению в них действий и фрагментов программ

ПРОИЗВОДНЫЕ ПАРАДИГМЫ

Описание **производных** ПП можно сделать относительным и, следовательно, более **лаконичным**, выражая разницу с базовой ПП. Можно сказать, что производная ПП является проекцией базовой ПП на особенности постановок задач некоторой области приложения. Обычно в проекции видоизменяются наиболее важные элементы ПП. Вариации моделей семантических систем, поддерживающих производные парадигмы, можно использовать как объективные параметры при факторизации определений ЯСП и декомпозиции программ, начиная с учёта особенностей постановок задач.

Постановки задач базовой ИПП начинаются с определённого алгоритма решения актуальной задачи. Необходимо получить программу реализации алгоритма с практичными пространственно-временными характеристиками на конкретном доступном оборудовании. Производные ИПП выделяют разные методы представления данных в памяти (М) и организации порождаемых программой последовательных процессов, дополненных обработкой прерываний, поддержкой ПВ и сетевых процессов (С):

- автоматное – без процедур (блок-схемы);
- скалярное – без структур данных (Cobol);
- процедурное – библиотеки (Algol, PL/1);
- структурное – регулярная логика и типы данных (Pascal);
- сборочное – крупноблочность (assembler);
- модульное – повторное использование (Fortran);
- диаграммно-графовое – визуальность (Delphi);
- сценарное – одноуровневое, поверхностное (Tcl);
- операционное – управление заданиями (JCL);
- языки действий – управление акторами (Actor);
- событийно-ориентированное – обработчики событий (PL/1).

Постановка задач базовой ООП основана на контроле доступа к иерархии классов объектов с работоспособными методами решения задач некоторой предметной области. Требуется без лишних трудозатрат уточнить эту иерархию, чтобы приспособить её к решению новых задач этой области, её расширения или

перехода к подобной. Производные ООП дают разнообразные конкретизации понятия «класс объектов» (S) и механизмов их размещения в памяти (M):

- аспектно-ориентированное – разделение слоёв (AspectJ);
- обобщённое (генеративное) – с абстрактными типами данных в роли классов (Ada);
- прототипное – без иерархии классов (Self);
- табличная – с таблицами в роли классов (Falcom);
- проблемно-ориентированное – синтаксическая надстройка (AutoLisp, Perl, Verilog, VHDL, SQL);
- протоколо-ориентированное – интерфейсы (Swift);
- компонентно-ориентированное – по архитектуре (Visualbasic);
- субъектно-ориентированное – объекты с поведением (Smalltalk);
- обмен сообщениями - (Falcon);
- событийно-ориентированное – (TypeScript);
- рефлексивно-ориентированное – (Dart).

Задачи базовой ФП обычно ориентированы на известную предметную область, в рамках которой следует выбрать символьное представление данных и отладить систему универсальных функций, пригодных для не чрезмерно трудоёмкого создания прототипов решения новых задач из этой области. Производные ФП представляют вариации в методах организации вычислений (E) и структурирования данных (S):

- чистое – методика снижения стартового барьера отказом от побочных эффектов и явного управления (PureLisp и Haskell без монад);
- динамическое – изменения по ходу вычислений (Lisp, Python);
- ленивые вычисления – откладывание выполнения действий (Hope);
- мемоизация – хранение результатов (Setl);
- комбинаторное – сведение к функциям (FP);
- аппликативное – подгрузка любого базиса (Lisp);
- символьные вычисления – математические формулы (Reduce);
- гомоиконное – подобие структур данных и программы (TRAC);
- полиморфное – классы объектов (CLOS);
- правила переписывания – одноуровневая подстановка данных (РЕФАЛ);

- метапрограммирование – конструирование программ (Lisp, РЕФАЛ);
- продолжение – передачи управления (Scheme);
- алгебраическое – выделение семантических систем (CoQ);
- реактивное – обработчики ситуаций (Scala);
- экспериментальное – проявление знаний (Lisp);
- стек-ориентированное (Forth).

Решения задач базовой ЛП исходят из заданной коллекции фактов и отношений, характеризующей актуальную задачу, пока не имеющую ни эффективного решения, ни полного описания. Надо привести эту коллекцию, возможно пополнить, к форме, демонстрирующей возможность ответов на практичные запросы относительно данной задачи. Производные ЛП используют разные подходы к смягчению зависимости получения результатов от избыточного или недостаточного детерминизма (C) и разные формы представления вычисляемых выражений (E):

- откаты-бэктрекинг – перебор вариантов (Snobol);
- недетерминированное – без преждевременного упорядочения (Prolog);
- вопрос-ответное – задачи поиска (NLP, Eliza);
- индуктивное – вывод решения (MIS, Prolog);
- естественно-языковое – запросы на подмножестве языка (MicroPlanner);
- программирование в ограничениях – границы определённости (Prolog);
- контрактное – задание спецификаций (D);
- предикатное – сведение к предикатам (P).

Существует заметное число **комбинированных** ПП, объединяющих достоинства двух-трёх ПП для решения разнотипных подзадач, поддерживаемых и **мультипарадигмальными** ЯП (Lisp 1.5, Planner, Merlin, F#, C#, Scala и др.). Происходящее в настоящее время проявление ПП в области улучшения ПО, оперирования устройствами, организации ПВ и обработки больших данных вероятно повлечёт переход ряда производных ПП в новые позиции парадигмального пространства.

Можно обратить внимание, что ЛП и ФП нередко рассматривают как декларативные, учитывая их приспособленность к работе на предварительных фазах определения постановки решаемой задачи. Более естественно выделять **дополнительные формы**, такие как декларативность, абстракции, языки спецификаций и др., преимущественно решающие задачи типа «строительных лесов», т. е. в центре внимания таких форм не альтернативное, противопоставляемое представление средств и методов программирования, а задание границ поведения программ, выделение удобных для практики порождаемых процессов. Например, декларативное представление типов данных присутствует во многих ЯП. Можно сказать, что дополнительные формы представляют собой методы наложения ограничительных условий на функционирование программы. Любая парадигма программирования может быть дополнена такими ограничительными условиями[7]. Большинство ограничительных условий теряет смысл по завершении отладки программы.

Ещё один важный ряд **вспомогательных** парадигм – это парадигмы области приложения программ, прикладные парадигмы, своевременный учёт которых обеспечивает успех результатов программирования. Зависимость от таких парадигм усложняет постановки задач для ЛП и ООП, что, возможно, побуждает специалистов по обучению информатике считать ФП, ПВ и ИПП более фундаментальными, чем ЛП и ООП [8].

Встречаются **отвергнутые** ПП, не получившие признания программистским сообществом вопреки подтверждающим экспериментам, дающие материал для исследования границ общей парадигмы программирования, наряду с **эзотерическими** ПП, изобретение которых можно рассматривать как исследование возможностей представлять и распознавать информацию в стиле создания и расшифровки ребусов.

ПЕРСПЕКТИВНО-СТРАТЕГИЧЕСКИЕ ПАРАДИГМЫ (ПСП)

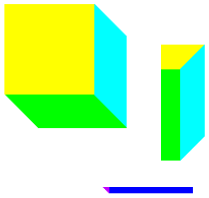
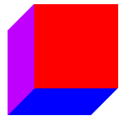
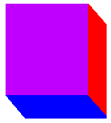
Редко упоминаются достаточно важные системообразующие **перспективно-стратегические** ПП, существование которых нацелено на общий прогресс ИТ, решение собственных задач программирования и развитие информатики как

[7] Математики говорят: «Знание границ закономерности освобождает от знания самой закономерности».

профессии. Такие ПП не сводимы к комбинации базовых и прикладных ПП, обладающих сторонней сферой приложения. Постановки задач здесь нацелены не только на улучшение качества результатов программирования, но преимущественно на стратегическую экстраполяцию перспектив повышения квалификации программистского корпуса, эстафету знания в области информатики и совершенствование технологий программирования, включая ИТ. Эксплуатационная прагматика таких ПП, в отличие от имеющих внешний полигон парадигм прикладного программирования, требует создания специальной программистской обстановки, включая элементы образовательной системы и комплекс мероприятий по обмену опытом с целью проявления и накопления имплицитных знаний, вербализация которых в программировании достаточно трудоёмка и слишком протяжённая по времени, т. к. требует «созревания головы». История ПП показывает, что появление наименований ПП обычно отстаёт от создания инструментальной поддержки примерно на 10–20 лет. Результативность специальной программистской обстановки ярко показана Ершовскими школами юных программистов, ежегодно проводимыми ИСИ СО РАН.

Следует отметить сравнительно сложившиеся ПСП, дополняющие ряд классических основных ПП. Прежде всего это фундаментальные ПП: алгоритмика и схематология, теоретическое (Standard Pascal, C-light), доказательное (CoQ) и верификационное программирования (Isabelle, PVS, LTL). Далее формируются образовательные ПП: учебное (PureLisp, Logo, Elm, Робик, Рапира, Oz), олимпиадное – спортивное (Pascal, C, Python), непрофессиональное (Basic) и экспериментальное программирования (Lisp). Производственное значение имеют технологичные ПП: оптимизационное и трансформационое программирование (работы Л. Ломбарди, Ё. Фатамуры, А.П. Ершова), любительское, отладочно-тестирующее и свободно-распространяемое программное обеспечение (GNU). Более детальная характеристика ПСП не входит в задачу данной статьи, что не мешает суммарно представить сводку ПП (Таблица 18), на которой символизировано, что общее пространство, составленное из ряда разных категорий ПП, можно рассматривать как грани куба, одна из которых выполняет роль ведущей, остальные интуитивно проявляются по мере развития постановки решаемой задачи.

Таблица 18. Парадигмы программирования (ПП)

Парадигма программирования					
	***Основные практические ПП***			***Перспективно стратегические ПП***	
Базовые императивно-процедурное функциональное логическое объектно-ориентированное	**Системно расширяющие** системное операционные системы системы управления БД многопроцессорные комплексы	**Неограниченные** большеобъёмные данные дистанционный доступ параллельное суперкомпьютерное (высокопроизводительное)	**Технологичные** оптимизационное трансформационое (проекции) отладочное и тестирование свободно-распространяемое	**Образовательные** учебное олимпиадное непрофессиональное экспериментальное (любительское)	**Фундаментальные** алгоритмическое теоретическое (схематология) доказательное верификационное
					

ЗАКЛЮЧЕНИЕ

Предложенную систематизацию ПП можно использовать при оценке сложности и трудоёмкости программирования, особенно если дополнить более ясным разделением требований к постановкам задач по сферам применения на академические и производственные, а по уровню изученности – на точные, развиваемые и усложнённые трудно удостоверяемыми требованиями. **Основные** (базовые, расширяющие и неограниченные) ПП можно различать по упорядочению категорий семантических систем и обрабатываемых данных, **производные** – по отличию от исходной парадигмы, **вспомогательные** – по областям приложения, **комбинированные** – по составляющим парадигмам. Любая парадигма программирования может быть обогащена **дополнительными** формами представления ограничительных условий на функционирование программ и вспомогательными прикладными ПП приведения к конкретной сфере приложения.

В качестве близких следует указать работы [6–8]. Е.М. Лаврищева представила достаточно полных обзор ПП, имеющих значение для технологий программирования [7], П. Ван Рой проанализировал более 30-ти ПП, преимущественно комбинированных, и представил схему их взаимосвязей, цитируемую в Википедиях [8], а П. Вегнер выполнил весьма серьёзный анализ ООП, методов поддержки этой парадигмы и её сравнение с другими классическими ПП [6].

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект № 18-07-01048-а.

СПИСОК ЛИТЕРАТУРЫ

1. https://www.levenez.com/lang/ – Диаграмма, представляющая хронологию появления и наследования многих ЯП.

2. http://progopedia.ru/. Сайт с описаниями 171 языка и 31 парадигмы.

3. *Лавров С.С.* Методы задания семантики языков программирования // Программирование, 1978. № 6. С. 3–10.

4. *Бентли Д.* Жемчужины творчества программистов. М.: Издательство «Радио и связь»: Редакция переводной литературы, 1990. 217 с.

5. *Городняя Л.В.* О представлении результатов анализа языков и систем программирования. Научный сервис в сети Интернет: труды XX Всероссийской научной конференции (17–22 сентября 2018 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2018.

6. *Peter Wegner.* Concepts and paradigms of object-oriented programming. SIGPLAN OOPS Mess. 1, 1 (August 1990). P. 7–87. https://pdfs.semanticscholar. DOI: http://dx.doi.org/10.1145/

7. *Лаврищева Е.М.* Программная инженерия и технологии программирования сложных систем. Учебник для вузов. М., 2018. 432 с.

8. *Peter Van Roy.* Диаграмма с результатами сравнения более 30-ти парадигм программирования. https://www.info.ucl.ac.be/~pvr/paradigmsDIAGRAMeng108.pdf

---

ON SYSTEMATIZATION OF PROGRAMMING PARADIGMS BY DECISION-MAKING PRIORITIES

L. V. Gorodnyaya

A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences, Novosibirsk State University, Novosibirsk

lidvas@gmail.com

Abstract

The report is devoted to the analysis of the method of comparison of programming languages, convenient for assessing the expressive power of languages and the complexity of the programming systems. The method is adapted to substantiate practical, objective criteria of program decomposition, which can be considered as an approach to solving the problem of factorization of very complicated definitions of programming languages and their support systems. The article presents the results of the

analysis of the most well-known programming paradigms and outlines an approach to navigation in the modern expanding space of programming languages, based on the classification of paradigms on the peculiarities of problem statements and semantic characteristics of programming languages and systems with an emphasis on the criteria for the quality of programs and priorities in decision-making in their implementation.

Keywords*: definition of programming languages, definition of programming languages, programming paradigms, definition decomposition criteria, semantic systems*

REFERENCES

1. https://www.levenez.com/lang/ – Chart representing the history of the emergence and inheritance of many programming languages.

2. http://progopedia.ru/. Website with descriptions 171 languages and 31 paradigms.

3. *Lavrov S.S.* Methods of definition the semantics of programming languages // Programming. 1978. No 6. P. 3–10.

4. *Bentley D.* The Pearls of creativity of programmers. Moscow: publishing House "Radio and communication": Edition of translated literature, 1990. 217 p.

5. *Gorodnyaya L.V.* On the presentation of the results of the languages and programming systems analysis. Scientific service on the Internet: proceedings of the XX All-Russian scientific conference (17–22 September 2018, Novorossiysk). Moscow: IPM im. M.V. Keldysh, 2018.

6. *Peter Wegner.* Concepts and paradigms of object-oriented programming. SIGPLAN OOPS Mess. 1, 1 (August 1990). P. 7–87. https://pdfs.semanticscholar. DOI: http://dx.doi.org/10.1145/.

7. *Lavrishcheva E.M.* Software engineering and programming technologies of complex systems. Textbook for high schools. Moscow, 2018. 432 p.

8. *Peter Van Roy.* Chart with the results of comparison of more than 30 programming paradigms. https://www.info.ucl.ac.be/~pvr/paradigmsDIAGRAMeng108. pdf.

СВЕДЕНИЯ ОБ АВТОРЕ

ГОРОДНЯЯ Лидия Васильевна – старший научный сотрудник Института систем информатики имени акад. Андрея Петровича Ершова СО РАН, доцент Новосибирского государственного университета, специалист в области системного программирования и образовательной информатики.

Lidia Vasiljevna GORODNYAYA – Senior Researcher of A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences, Associate Professor of Novosibirsk State University, a specialist in system programming and educational informatics.

email: lidvas@gmail.com

Материал поступил в редакцию 15 ноября 2019 года

УДК 004.43 ББК 22.18

РЕФАЛ-СЕРВЕР

А. А. Гусев

ЗАО «СайнАрт», г. Санкт-Петербург

a.gusev@signart.ru

Аннотация

Работа посвящена описанию проекта обновления и распространения языка программирования Рефал (далее – просто Рефал), созданного в СССР в 1960-х годах В.Ф. Турчиным. Язык изначально предназначался для различных логических преобразований, прежде всего, текстового материала и ориентирован на использование непрограммистами. На практике сфера применения оказалась шире: машинный перевод, оптимизация и компиляция программ, доказательство теорем, моделирование сложных электронных схем, решение ряда задач искусственного интеллекта. Язык сейчас имеет достаточное количество последователей, главным образом, в научных кругах.

Задачей описываемого проекта является создание продукта, позволяющего использовать Рефал в современных массовых приложениях и расширить круг его потенциальных пользователей до всего интернета. Был проведён опрос сообщества пользователей и разработчиков Рефала с целью получения представления о текущем состоянии дел, актуальных реализациях и путях развития языка. Были рассмотрены возможные средства реализации проекта. Информации о ведущихся аналогичных разработках получено не было.

Ключевые слова*: Рефал, сервер, обработка текстов, xml, json, искусственный интеллект, метавычисления*

1. ИСТОРИЯ ВОПРОСА

Язык Рефал создавался для облегчения труда по обработке различной символьной информации. Он относится к классу сентенциальных языков, что не предполагает последовательного описывания шагов, которые должен сделать компьютер для решения задачи. Задача формулируется пользователем как

набор правил, а пути решения определяет уже компьютер, точнее, исполняющая среда Рефала [1].

По мере развития средств исполнения Рефал-программ оказалось, что простота и, вместе с тем, большая выразительность языка позволяют решать весьма разнообразные и даже неожиданные на первый взгляд задачи, такие, например, как вывод химических формул.

В течение первых 10 лет после создания языка компиляторы с Рефала были созданы для основных массово используемых на то время в СССР компьютеров, и это были как раз серверные решения. Достаточное финансирование науки предоставляло такую возможность. Появились также различные интерпретации языка, развивающие его выразительные и вычислительные возможности.

В силу ряда обстоятельств, скорее всего, по причине массового перехода пользователей в научной сфере на персональные вычисления в 1990-х годах, все известные реализации языка в настоящее время также ориентированы на персональные вычисления.

С одной стороны, это позволило сильно удешевить развитие компиляторов, с другой — сузило круг задач, решаемых с помощью Рефала. Массовое обслуживание, в частности, в настоящее время немыслимо без серверных приложений. И только серверных — в части применяемой бизнес-логики.

Серверная реализация позволяет не только организовывать массовую удалённую обработку информации, но и эффективно описывать и исполнять бизнес-логику сложных человеко-машинных комплексов, сохраняя полный централизованный контроль над исполняемыми правилами.

2. ПОЧЕМУ РЕФАЛ

Выбор конкретного языка в данном случае основан на личном опыте использования различных языков и сред программирования в течение последних 40 лет. По мере развития технологий труд программиста из занятия таинственного и романтического постепенно превращается в рутинный процесс, где ценностью объявляется исключительно конечный результат. А удовольствие, предполагается, должны приносить личные доходы от трудовой деятельности либо

1 http://refal.botik.ru/

победы над конкурентами. При этом программы из произведений искусства, коими они были на заре всеобщей компьютеризации, постепенно превращаются в нагромождение заплат, стилей и технологий, несмотря ни на какие усилия руководителей процесса.

Такое развитие событий вызвало у меня желание сделать что-то для языка, давно известного, снискавшего множество похвал, но почему-то за свою 50-летнюю историю не сильно расширившего круг своих адептов.

Вот его сильные стороны:

– абсолютная простота основной вычислительной концепции, в основе которой лежат принцип *конкретизации* и логика построения программ, соответствующая способу человеческого мышления (в отличие от многих распространённых языков программирования);

– относительная лёгкость вхождения в вопрос по причине как прозрачности концепции, так и достаточно проработанной методики его компиляции и выполнения, есть достаточно исследовательских работ как непосредственно по Рефалу, так и по близким по духу системам программирования;

– сильная универсальность языка: от обработки различных текстовых данных до задач имитационного моделирования и ИИ;

– решения на Рефале красивы и способствуют развитию ума, а не его усталости (сугубо личное мнение, по мнению автора, этот пункт можно было бы поставить первым).

В случае успешной реализации полученный продукт должен предоставить:

- идентичный доступ как к разработке, так и к использованию программ на Рефале с любого устройства, в т. ч. смартфонов;
- платформу для построения приложений, в том числе, коммерческих на Рефале и с использованием Рефала;
- платформу для создания Рефал-компонентов для встраивания в программы на других языках как с помощью технологий распределённых объектов (COM и т. п.), так и путём создания блоков кода на целевых языках, эмулирующих выполнение Рефал-программ в исполняющей среде целевого языка;
- повсеместно доступную базу для обучения начинающих использовать язык, поскольку без критической массы программистов, владеющих пред-

метом, любой язык мертв.

3. РЕФАЛ И HASKELL

Первоначально, при формировании идеи разработки, вопрос в наличии «конкурирующего» языка программирования был мною упущен. Сравнение Рефала с «классическими» языками с разъяснением отличий было проведено еще В.Ф. Турчиным. Однако в 1990-х годах возник и получил развитие новый язык, имеющий с Рефалом гораздо больше пересечений — Хаскел (Haskell)[2].

Речь идёт о сопоставлении с образцом — основе Рефала и одной из сторон Хаскела. С точки зрения математической, алгоритмы реализации данного элемента языка могут быть схожими, кроме того, в Хаскел введено понятие «ленивых» или «отложенных» (Lazy) вычислений, отсутствующих в каноническом Рефале и способных поднять производительность вычислений, то есть весьма полезных.

В результате возникает вопрос: почему бы не использовать более «свежий», поддерживающий аналогичный стиль программирования и уже сильно «продвинутый» и известный в мировом сообществе язык, то есть Хаскел вместо Рефала для тех же задач?

Но, думаю, первоначальный выбор не был ошибочным. У Рефала есть неоспоримое преимущество — простота концепции языка, созданного для непрограммистов, в первую очередь. Вся суть его идеи объясняется на практике за пару лекций и не требует запоминания множества правил и шаблонов. Чего категорически нельзя сказать о Хаскел.

Пользователь Рефала сам творит свою систему понятий и занимается, прежде всего, своей задачей. Поэтому для успешного применения Рефала в любой области достаточно хорошего воображения и понимания цели, которую следует сформулировать. Это даёт шанс привлечь на сторону языка новых пользователей из различных областей науки, и не только науки.

2 https://www.haskell.org/

ТЕКУЩЕЕ ПОЛОЖЕНИЕ ДЕЛ В РЕФАЛ-СООБЩЕСТВЕ

Завершение коммунистического периода совпало с появлением доступных персональных вычислений. Одновременно с этим начался процесс сокращения затрат на науку, её децентрализации. К счастью, сохранились преданные пользователи языка, которые смогли сохранить и развить образовавшиеся к тому времени диалекты, переведя разработку и сопровождение свих локальных систем на персональную вычислительную технику. Это позволяет упростить и удешевить разработку, но затрудняет использование полученных программных средств в удалении от разработчика. Вместо решения конкретных задач удалённому пользователю нужно выбрать один из не вполне совместимых диалектов, найти соответствующие дистрибутив, документацию. В то же время изначально язык ориентировался на неискушённых в техническом смысле пользователей, т. е. таким образом теряется значительная часть потенциальной аудитории языка.

Надо отдать должное сообществу Рефала: его члены смогли наладить устойчивый обмен информацией и не опуститься до своих узкодиалектных интересов. Также ведётся постоянное ознакомление студентов с Рефалом — теми, кто связан с высшей школой.

Но при этом степень распространения Рефала вне этого круга нельзя назвать удовлетворительной, потому что язык перешел в качество «клубного интереса». Более того, многие направления эффективного применения языка заполняются средствами, явно проигрывающими ему в технологичности и скорости разработки. Как пример можно привести создание систем машинного перевода, различных ботов (класса Алисы от Яндекс), систем обработки музыкального материала и т. п.

НОВАЯ ФУНКЦИОНАЛЬНОСТЬ

Проект задуман для создания эффективного серверного окружения для многопользовательского решения задач, реализуемых на Рефале. Многопользовательский подход подразумевает возможность эффективной обработки множества одновременно выполняемых запросов от произвольного количества пользователей, безостановочную работу в режиме 7/24, одновременную поддержку различных диалектов языка.

Главная ориентация такой системы — работа в компьютерной сети, обеспечивающей доступность массового подключения к ней и должную загрузку системы. Главная проблема — обеспечение приемлемой скорости вычислений при разумной их цене.

В отличие от персонального решения, серверное требует наличия достаточных возможностей масштабирования, т. е. несложного способа увеличения производительности по мере роста загрузки системы.

Запланирована реализация ряда качеств разрабатываемой системы, которые определяют её оригинальность:

– размещение всех компонентов (функций) Рефал-программы в специальном репозитории, наделённом возможностью разграничения доступа, как следствие, наличие множества пользователей;

– поддержка версионности как на уровне кода программ, так и на уровне компилятора — это позволяет вести совершенствование системы с минимальным количеством технологических остановок;

– поддержка различных диалектов Рефала в рамках одного сервера — по мере возможности и необходимости, для начала было бы достаточно реализации Базисного Рефала;

– предоставление средств взаимодействия сервера с окружающим миром посредством обмена текстовыми данными в общепринятом открытом формате, например, XML или (и) JSON;

– предоставление единых средств оптимизации и анализа выполнения программ, а также встроенного оптимизатора с подсказками, планами выполнения, учётом времени и журналированием.

Всё вышеуказанное должно предоставить неплохую возможность для обучения и разработки с использованием сильных сторон языка Рефал.

ВЫБОР СРЕДСТВА РЕАЛИЗАЦИИ

Выбор средства является интересной и крайне важной задачей, так как по мере написания кода его замена очень сильно дорожает.

Были рассмотрены варианты скриптовых языков (Javascript), собственно Java (или Kotlin), исполняющие среды SQL-серверов Postgresql, Oracle и Microsoft как имеющие уже встроенные механизмы оптимизации исполнения.

В результате выбор был остановлен на языке Go как ориентированным на максимальную эффективность серверной работы и обладающим достаточным инструментарием для работы с интернет-запросами и базами данных и к тому же параллельным и многоплатформенным.

Кроме того, для Go можно достаточно легко адаптировать различные идеи и методы существующих компиляторов Рефала, написанных, как правило, на языке С.

4. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

Главным отличием от большинства существующих версий окружения Рефала должно стать интерпретационное исполнение программы. Однако, ввиду особенности языка, традиционная (построчная) интерпретация применительно к Рефалу предположительно годится только для достаточно простых приложений. Пока что нужно понимать, что решение задачи эффективной интерпретации может затребовать львиную долю ресурсов при разработке.

Альтернативное решение — «теневая» компиляция одним из существующих готовых компиляторов и такое же «теневое» выполнение с отправкой результата потребителю может временно закрыть вопрос, но такое решение не будет «честным» и сократит ряд преимуществ реализации в дальнейшем.

Другой важный вопрос реализации — определение входного языка интерпретатора и набора базовых библиотек. Было принято решение взять за основу синтаксис Рефала-5 в рамках возможностей Базового Рефала, т. е. для начала сделать чистое, не оптимизированное сопоставление с образцом. Что касается библиотек, то необходимы арифметика, стандартный ввод и вывод и операции с копилками и ящиками. Это даст возможность создавать самостоятельные, не интерактивные полезные программы с использованием серверной платформы.

Расширение функциональности будет необходимо вести в двух направлениях:

1. Создание специализированных встроенных функций, эффективно реализующих дополнительные возможности, например, взаимодействие с серверами баз данных; доступ к ним только от лица административных пользователей.

2. Создание библиотеки функций — оболочек к специализированным

функциям. Библиотечные функции должны быть доступны обычным пользователям.

Уже сейчас есть внутренние задачи по работе с большими массивами данных, которые желательно вынести в обработку специализированных серверных приложений.

ТЕКУЩЕЕ СОСТОЯНИЕ РАЗРАБОТКИ

Нужно признать, что к моменту формирования первой публикации в августе 2019 года проект находился только в самой начальной стадии. Реальные работы начались в сентябре 2019 года. Хотя продукт ещё рано «показывать на публике», он постепенно «оживает» и приобретает новые свойства.

Сложность интерпретатора оказалось выше, чем ожидалось. Сейчас решается вопрос о жестком соответствии сопоставления по образцу эталонному (Рефал-5) и технологии отладки и получения информации для конечного пользователя.

Далее необходимо решить вопросы о соответствии производительности разумным требованиям и разделению доступа для различных пользователей.

По завершении данных работ можно будет констатировать факт выхода первой версии продукта «в свет».

ОЖИДАЕМЫЕ ПЕРСПЕКТИВЫ

Будучи реализованной, описываемая система может быть загружена множеством приложений: сфера образования (обучения логике мышления на базе формулирования задач на Рефале), медицинская диагностика, помощь различным социальным службам, машинный перевод (я ещё верю в это), создание роботов-ботов для различных целей.

Отдельным важным преимуществом Рефала можно отметить его как средство создания интеллекта роботизированных систем. В отличие от популярных нынче нейронных сетей, здесь в основе алгоритма лежат целеполагание и правила, на которых базируется поведение робота. Нейронные сети же пусть здесь же занимаются свойственной им задачей — первичной обработкой поступающей информации. Ближайшее будущее — за гибридными системами.

Логично ожидать, что количество пользователей из разных потребительских секторов должно стать заметным, и популярность Рефала в своих приложе-

ниях может сравниться с популярностью Фортрана до сих пор у физиков. И тот, и другой языки имеют структуру, соответствующую образу мышления потенциального пользователя, каждый в своей сфере.

10. ОБЩЕСТВЕННАЯ МИССИЯ

Для продвижения любого языка программирования «в массы» нужны следующие составляющие:

1. Способность языка решать определённый круг задач. Этот круг начинается от насущных задач самого разработчика.

2. После успешного освоения начального круга задач реализация должна заинтересовать других профессиональных разработчиков. Это требует распространения информации о продукте по различным каналам, доступным разработчику. Естественно, программный продукт должен обладать всеми атрибутами продукта, начиная с качественной документации.

3. Для привлечения внимания потенциальных пользователей, не входящих в узкопрофессиональное сообщество, следует проводить информационно-просветительскую работу среди студентов и школьников, читать лекции и проводить олимпиады по освоению технологии и решению учебных задач с помощью возможностей продукта. Для этого необходима разработка пользовательской среды, дающий доступ к возможностям сервера приложений на Рефале. Возможно, это будут разные среды, ориентированные на разные аудитории. В связи с этим вспоминается замечательная среда программирования для школьников — Лого, позволяющая детям с удовольствием рисовать мультфильмы, начиная с первых занятий.

Собственно миссия здесь — распространения технологии, приносящей радость и развитие, способной привнести в повседневную работу элемент творчества, в противовес множеству современных технологий, помимо своей главной цели просто засоряющих ум.

СПИСОК ЛИТЕРАТУРЫ

1. *В.Ф. Турчин* Программирование на языке РЕФАЛ. I. Неформальное введение в программирование на языке РЕФАЛ. М.: ИПМ АН СССР, 1971. Препринт №41. 55 с.

2. *М.Ш. Исламов.* Декларативное программирование. Рефал в обучении [Электронный ресурс] / III Международная научно-практическая конференция "Современные информационные технологии и ИТ-образование", сб. докл. // Моск. гос.ун-т им. М.В. Ломоносова, Факультет вычислительной математики и кибернетики; под ред. В.А. Сухомлина. М.: МАКС Пресс, 2008. С. 252—256.

3. Сборник трудов по функциональному языку программирования Рефал под редакцией А.П. Немытых, том II. // Переславль-Залесский: Издательство «СБОРНИК», 2015, 156 с.

REFAL ON SERVER SIDE

A. A. Gusev

Company “SignArt”, Saint-Petersburg

a.gusev@signart.ru

Abstract

This work is devoted to the description of the project of updating the technology of algorithmic language application created in the USSR in the 60s of the XX century by V. F. Turchin. The language was originally intended for various logical transformations primarily of text material. In practice, the scope was wider: machine translation, optimization and compilation of programs, proof of theorems, modeling of complex electronic circuits, solving a number of problems of artificial intelligence. The language now has a sufficient number of followers, mainly in scientific circles.

The objective of the described project is to create a product that allows the use of Refal in modern mass applications and to expand the range of its potential users to the entire Internet. A survey of the community of users and developers of Refal was conducted in order to get an idea of the current state of Affairs, current implementa-

tions and ways of language development. Possible means of project implementation were considered. No information was received on similar developments under way.

***Keywords**: Refal, server-side computing, text processing, xml processing, artificial intelligence, meta-calculations*

REFERENCES

1. *V.F. Turchin.* Programmirovanie na yazyke REFAL I Neformalnoe vvedenie v programmirovanie na yazyke REFAL. M.: IPM AN SSSR, 1971. Preprint No 41. 55 s

2. *M.Sh. Islamov.* Deklarativnoe programmirovanie. Refal v obuchenii [EHlektronnyj resurs] /M. SH. Islamov III Mezhdunarodnaya nauchno prakticheskaya konferenciya “Sovremennye informacionnye tekhnologii i IT obrazovanie”: sb. Dokl. Mosk. Gos. Un-t im. M.V. Lomonosova, Fak. Vychislit. matematiki i kibernetiki pod red. V.A. Suhomlina. M.: MAKS Press, 2008, S. 252—256.

3. Cbornik trudov po funkcionalnomu yazyku programmirovaniya Refal. Tom II Pod redakciej A.P. Nemytyh. Pereslavl Zalesskij: Izdatelstvo “SBORNIK”, 2015, 156 s.

СВЕДЕНИЯ ОБ АВТОРЕ

ГУСЕВ Александр Альфредович – начальник отдела ЗАО «СайнАрт», специалист в области разработки банковских информационных систем.

Alexander GUSEV – Head of Department of CJSC SignArt, specialist in the field of development of banking information systems.

email: gusev_aleksandr@mail.ru

Материал поступил в редакцию 16 ноября 2019 года

УДК 004.021 + 004.42

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА ПОИСКА ЭКСПЕРТОВ ДЛЯ ПРОВЕДЕНИЯ НАУЧНОГО РЕЦЕНЗИРОВАНИЯ В МАТЕМАТИЧЕСКОМ ЖУРНАЛЕ

А.М. Елизаров[1][0000-0003-2546-6897], Е.К. Липачёв[2][0000-0001-7789-2332], Ш.М. Хайдаров[3][0000-0002-6835-3289]

[1–3] *Институт математики и механики им. Н.И. Лобачевского Казанского федерального университета, ул. Кремлевская, 35, г. Казань, 420008*

[1, 2] *Высшая школа информационных технологий и интеллектуальных систем Казанского федерального университета, ул. Кремлевская, 35, г. Казань, 420008*

[1]amelizarov@gmail.com, [2]elipachev@gmail.com, [3]15jkeee@gmail.com

Аннотация

Предложен подход к организации экспертной оценки научного документа, представленного для публикации в математический журнал. Ограничение предметной области связано с использованием системы математической классификации Mathematical Sciences Classification System – MSC. Представлена рекомендательная система, позволяющая сформировать список возможных экспертов для проведения процедуры научного рецензирования математической статьи. Эта рекомендательная система использует коды MSC2020, изначально представленные автором статьи. Если в статье указаны коды MSC2000 или MSC2010, производится их автоматическое преобразование в коды MSC2020.

Для каждого эксперта в системе поддерживается персональный профиль, который содержит набор кодов MSC2020, дополненный числовыми характеристиками, – весами, вычисленными для каждого кода в соответствии с системой учета компетенций, предпочтений или отказов от участия в процедуре рецензирования, сформированных в процессе предыдущей работы в качестве эксперта. Этот набор автоматически редактируется в случае включения эксперта в список возможных рецензентов – повышаются или уменьшаются веса нескольких кодов, а также добавляются новые коды. Рекомендательная система реализована в виде

встроенного инструмента (плагина) платформы Open Journal Systems (OJS). Разработанный метод апробирован в информационной системе научного журнала Lobachevskii Journal of Mathematics (https://ljm.kpfu.ru).

***Ключевые слова**: информационная система научного журнала, Open Journal Systems, рабочий процесс рецензирования, автоматический выбор рецензентов, Mathematics Subject Classification 2010, Lobachevskii Journal of Mathematics.*

ВВЕДЕНИЕ

Новые возможности распространения информации, возникшие как результат развития интернет-технологий, отразились и на научных коммуникациях, в частности, появились научные труды, оформленные в виде «живых» или «динамических» публикаций [1, 2], а также блогов [3, 4]. Функции, присущие традиционным научным журналам, также изменились: ряд процессов редакциями журналов уже не реализуется (см., например, [5]). Однако роль научных рецензируемых журналов существенно возросла, что связано, прежде всего, с внедрением академическими учреждениями систем оценки труда научных работников, существенно учитывающих показатели публикационной активности, – наличие публикаций, индексируемых в международных наукометрических базах данных, таких как Scopus, Web of Science (WoS) и ряда других, является теперь критерием, по которому оценивается работа как отдельного сотрудника, так и научного коллектива в целом.

Количество научных журналов с 2000 года увеличилось на треть, предложена новая систематическая типология академических журналов [6]. Число публикаций в научных журналах также постоянно увеличивается. Темп этого роста составляет более 5% в год, что превышает рост мировой экономики в целом (см., например, [7]). Все научные публикации, представленные к публикации, подвергаются обязательной для рейтинговых журналов процедуре экспертной оценки (двойного, тройного и в большинстве случаев «слепого» рецензирования). Наукометрические базы данных Scopus и Web of Science исключают из своего состава журналы, которые не в полной мере соблюдают установленные правила не только проведения процедуры научной экспертизы, но и привлечения специали-

стов в качестве рецензентов, а также требования к составу редколлегий журналов. Укажем в качестве примера требования к рецензентам индексируемого в Scopus открытого архива сборников трудов семинаров и конференций CEUR Workshop Proceedings (http://ceur-ws.org/). Согласно объявленным требованиям (http://ceur-ws.org/HOWTOSUBMIT.html) члены программного комитета конференции и эксперты, привлеченные к рецензированию, для подтверждения квалификации в области компьютерных наук должны иметь достаточно хорошее представительство (не менее 5 проиндексированных статей) в библиографической базе “dblp computer science bibliography” (DBLP, https://dblp.uni-trier.de/). Совокупность требований к экспертам, жесткие правила проведения процедуры рецензирования, включая временные сроки, с одной стороны, и неуклонный рост количества научных публикаций, с другой, создают очевидную проблему, связанную с подбором квалифицированных рецензентов для редакций каждого научного журнала.

Имеются и другие проблемы проведения процедуры научного рецензирования, например, обеспечение объективности экспертной оценки (см., например, [7]). В ряде работ обсуждаются преимущества открытых экспертных обзоров и публикации статей вместе с рецензиями на них (см., например, [8–10]). Мы полагаем, что разработка и внедрение программных инструментов поддержки процедуры научного рецензирования позволят уменьшить сроки опубликования научных работ, обеспечить необходимый контроль научного качества этих работ и сделать саму процедуру рецензирования более объективной.

Автоматизированные системы управления конференциями, например, EasyChar, и информационная журнальная система Open Journal Systems (OJS) имеют встроенные средства поддержки процесса рецензирования, включая предложения экспертам о возможности их назначения рецензентами, автоматическую рассылку уведомлений, контроль сроков проведения экспертной оценки, и используются многими журналами (см., например, [11]). Но подбор возможных экспертов, способных объективно, квалифицированно и в ограниченные сроки провести рецензирование рукописи, поступившей в журнал, производится сотрудниками редакции, по сути, в ручном режиме.

Мы предлагаем метод формирования рекомендаций по назначению экспертов для математического документа. Ниже представлена рекомендательная

система, позволяющая сформировать список возможных экспертов для проведения процедуры научного рецензирования математической статьи. Эта рекомендательная система использует коды системы математической классификации Mathematical Sciences Classification System MSC2020, обязательно представляемые авторами статьи при ее направлении для публикации в соответствующий журнал. Рекомендательная система реализована в виде встроенного инструмента (плагина) платформы OJS. Разработанный метод апробирован в информационной системе научного журнала Lobachevskii Journal of Mathematics (https://ljm.kpfu.ru).

1. ПРОГРАММНЫЕ ИНСТРУМЕНТЫ ПОДДЕРЖКИ НАУЧНОГО РЕЦЕНЗИРОВАНИЯ

Организация независимой экспертной оценки научных документов является обязанностью редколлегии научного журнала, поскольку научное сообщество ожидает, что качество публикуемых статей контролируется в процессе рецензирования. Как правило, научная экспертиза проводится двумя или более независимыми рецензентами. Успешное прохождение процесса рецензирования является обязательным условием публикации статьи. Помимо отзывов на статью рецензенты, как правило, приводят развернутые комментарии, направленные на улучшение представленного материала. Когда авторы получают уведомления о полученных рецензиях, они учитывают их в следующей версии своей статьи. Поиск специалистов в предметных областях для проведения рецензирования осуществляется членами редколлегии на основе личного опыта. Сам процесс рецензирования вызывает ряд вопросов, связанных, например, с корпоративными интересами [7]. Для исключения ошибочных, недоброкачественных, а также статей, скомпилированных из чужих работ, используются различные методы проведения экспертной оценки и разрабатываются вспомогательные программные инструменты [12].

Процесс научного рецензирования – как правило, наиболее трудоемкий при рассмотрении научных материалов в редакции журнала. Критическими по времени при этом являются подбор рецензентов для квалифицированной оценки работы, поступившей в журнал, а также само рецензирование. Автоматизация

этого процесса способна сократить затраты времени – это подтверждает и практический опыт (см., например, [11]).

Практически все научные журналы в настоящее время выполняют большую часть редакционных процессов, используя различные информационные платформы. Основные особенности использования информационно-коммуникационных технологий (ИКТ) в процессе издания научных журналов исследованы в [11, 13–15], проведено также сравнение существующих информационных систем с точки зрения автоматизации редакционных процессов. Из существующих открытых информационных систем как наиболее совершенная выделена платформа Open Journal Systems [16].

Выбор этой платформы обусловлен тем, что OJS поддерживает широкий спектр бизнес-моделей для периодики и настроек предоставления доступа: от полностью открытого доступа к ресурсам до предоставления кратких аннотаций и коммерческой подписки. Это позволяет использовать названную систему как единую платформу для управления комплексом электронных изданий (например, научно-исследовательской или образовательной организации) [15].

Благодаря модульной архитектуре и использованию ролевой модели пользователей, система OJS может быть настроена и адаптирована под бизнес-процессы конкретного научного издания. Функционал системы позволяет реализовать взаимодействие участников редакционного процесса в режиме онлайн [16].

В системе OJS имеется встроенная поддержка процесса рецензирования, включая назначение рецензентов путем их выбора редактором из пользователей системы, имеющих соответствующую роль. Процедура назначения предполагает автоматическую рассылку уведомлений и последующий контроль сроков проведения экспертизы. В настоящей статье предложена модификация процесса назначения рецензентов до уровня рекомендательной системы. При использовании стандартного функционала системы OJS поиск рецензентов для каждой статьи производится редактором среди всех пользователей системы, имеющих роль рецензента. Отметим, что платформу OJS используют многие российские научные журналы.

В работах [8–10] рассмотрены инструменты обеспечения открытого рецензирования – отметим, что большая часть из них может применяться и при традиционном научном рецензировании.

2. АВТОМАТИЗИРОВАННЫЙ ПОДБОР РЕЦЕНЗЕНТОВ В РЕДАКЦИИ ЖУРНАЛА LOBACHEVSKII JOURNAL OF MATHEMATICS

Процесс выбора рецензентов для научных статей был автоматизирован в журнале Lobachevskii Journal of Mathematics (LJM) (https://ljm.kpfu.ru/) ещё в 2007 году [17]. Более того, автоматически выполнялось и назначение рецензентов. Эта автоматизация рассматривалась как один из этапов внедрения веб-технологий в практику работы математических журналов [18, 19]. В 2015 году редакция журнала перешла на издательскую платформу Open Journal Systems [13, 14, 16], и ранее внедренный алгоритм был модифицирован с учетом технических требований этой платформы [15, 20, 21].

Алгоритм подбора экспертов основан на неточном сравнении, которое проводится между списком ключевых слов, указанных автором, и перечнем научных интересов рецензентов, содержащимся в базе данных журнала.

3 РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА ВЫБОРА РЕЦЕНЗЕНТОВ МАТЕМАТИЧЕСКИХ ДОКУМЕНТОВ

В этом разделе предложен алгоритм формирования списка экспертов для проведения процедуры научного рецензирования математической статьи. Выбор экспертов ограничен базой информационной системы журнала.

За основу подбора рекомендаций использована математическая предметная классификация Mathematics Subject Classification (MSC2000, MSC2010, MSC2020) [22–24], а другие классификационные признаки и ключевые слова, извлеченные из текстов статей, рассматриваются как уточняющие [25].

3.1 РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ В НАУЧНОЙ РАБОТЕ

С рекомендательными системами пользователи интернета сталкиваются ежедневно при выборе каких-либо услуг. Такие системы направлены на улучшение доступа к объектам в сложных информационных пространствах, предлагая интеллектуальный подход к навигации, в отличие от обычного поиска.

Базовой особенностью рекомендательных технологий является использование алгоритмов, способных учитывать предпочтения отдельного пользователя или категории пользователей в процессе создания персонализированных рекомендаций.

Имеется несколько определений рекомендательных систем, в каждом их которых сделан акцент на определенные особенности таких систем. Например, в [26] приведено 15 различных определений рекомендательных систем со ссылками на публикации, в которых они введены и обоснованы. В статьях, опубликованных в сборниках [27, 28], освещены наиболее важные технологии рекомендательных систем, их свойства и области применения. Существует два основных типа рекомендательных систем: контент-ориентированные и социальные (коллаборативной фильтрации). В [29] приведен онтологический подход в рекомендательных системах для физико-математического контента.

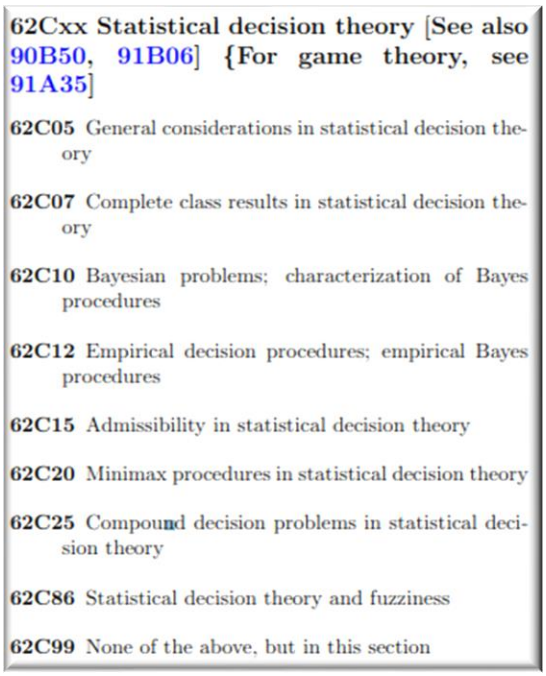
Рекомендательную систему, представленную в настоящей работе, можно отнести к типу рекомендательных систем конкретного случая (Case-Based Recommender Systems). Этот тип систем детально описан в работе [30], где представлены основные свойства таких систем и области их применения.

3.2 СИСТЕМЫ МАТЕМАТИЧЕСКОЙ КЛАССИФИКАЦИИ

Математическая предметная классификация MSC создана и поддерживается Mathematical Reviews (MR, http://www.ams.org/publications/math-reviews/math-reviews) и Zentralblatt MATH (zbMATH, https://zbmath.org/). Эта классификация используется всеми основными математическими журналами, архивом электронных публикаций научных статей и их препринтов arXiv.org (https://arxiv.org/) и цифровыми математическими библиотеками [31]. Текущей версией является MSC2010 (http://msc2010.org/; https://mathscinet.ams.org/msc/msc2010.html; https://zbmath.org/classification/), коды классификаторов MSC2000 использовались до 2010 года. С этого года рекомендована для использования следующая версия математической предметной классификации MSC2020 (https://msc2020.org/).

MSC представляет собой трехслойную схему с использованием буквенно-цифровых кодов. Код классификатора первого слоя состоит из двух цифр, второй слой определяется прописной буквой латинского алфавита “A”, “B” или “C” или же символом “-”, например, как в случае ветки “62-XX Statistics”; третий слой содержит две цифры. Например, код “62C10 Bayesian problems; characterization of Bayes procedures” относится к третьему слою иерархии, вышестоящим для него

является код “62C Statistical decision theory”, который принадлежит верхнему слою схемы (см. Рис. 1).



62Cxx Statistical decision theory [See also 90B50, 91B06] {For game theory, see 91A35]

62C05 General considerations in statistical decision theory

62C07 Complete class results in statistical decision theory

62C10 Bayesian problems; characterization of Bayes procedures

62C12 Empirical decision procedures; empirical Bayes procedures

62C15 Admissibility in statistical decision theory

62C20 Minimax procedures in statistical decision theory

62C25 Compound decision problems in statistical decision theory

62C86 Statistical decision theory and fuzziness

62C99 None of the above, but in this section

Рис. 1. Трехслойная схема классификации на примере раздела “Statistics”

MSC2020, как и MSC2010, содержит 63 узла первого слоя, начиная с “00 General mathematics” и заканчивая “97 Mathematics education”. Нумерация не сплошная, например, отсутствуют коды “02”, “04”, “07”, “21” и ряд других. Второй слой содержит 528 кодов, а третий состоит из 5606 кодов (см. Рис. 2).

В [32] классификаторы MSC2010 преобразованы в RDF Linked Data с использованием стандартизированного словаря Simple Knowledge Organization System (SKOS, https://www.w3.org/TR/skos-reference/). Это дает возможность присваивать классификаторы не только статьям, но и постам в блогах, рисункам и формулам.

В [33] представлены результаты машинного обучения MSC по полным текстам статей в математических цифровых библиотеках DML-CZ и NUMDAM. В этих библиотеках имеется сервис поиска статей по кодам MSC.

MSC2010	MSC2020

62Cxx	**Decision theory** [See also 90B50, 91B06; for game theory, see 91A35]
62C05	General considerations
62C07	Complete class results
62C10	Bayesian problems; characterization of Bayes procedures
62C12	Empirical decision procedures; empirical Bayes procedures
62C15	Admissibility
62C20	Minimax procedures
62C25	Compound decision problems
62C86	Decision theory and fuzziness
62C99	None of the above, but in this section

62Cxx Statistical decision theory [See also 90B50, 91B06] {For game theory, see 91A35]

62C05 General considerations in statistical decision theory

62C07 Complete class results in statistical decision theory

62C10 Bayesian problems; characterization of Bayes procedures

62C12 Empirical decision procedures; empirical Bayes procedures

62C15 Admissibility in statistical decision theory

62C20 Minimax procedures in statistical decision theory

62C25 Compound decision problems in statistical decision theory

62C86 Statistical decision theory and fuzziness

62C99 None of the above, but in this section

Рис. 2. Пример различий в кодах классификациях MSC2010 и MSC2020

3.3 ОСНОВНЫЕ ЭТАПЫ АЛГОРИТМА ФОРМИРОВАНИЯ РЕКОМЕНДАЦИЙ ПО ПОДБОРУ РЕЦЕНЗЕНТОВ

Алгоритм формирования рекомендаций по подбору рецензентов состоит из двух частей. Первая часть отвечает за создание и пополнение базы экспертов в информационной системе журнала. Вторая – это блок подбора экспертов для очередной рассматриваемой статьи.

Первоначально автоматизированным поиском в реферативных и библиографических базах данных просматриваются статьи автора, включенного в состав экспертов. Коды MSC, указанные в этих статьях (см. Рис. 3), включаются в профиль эксперта и, если требуется, выполняется преобразование кодов классификаторов MSC2000, MSC2010 в коды MSC2020 [34].

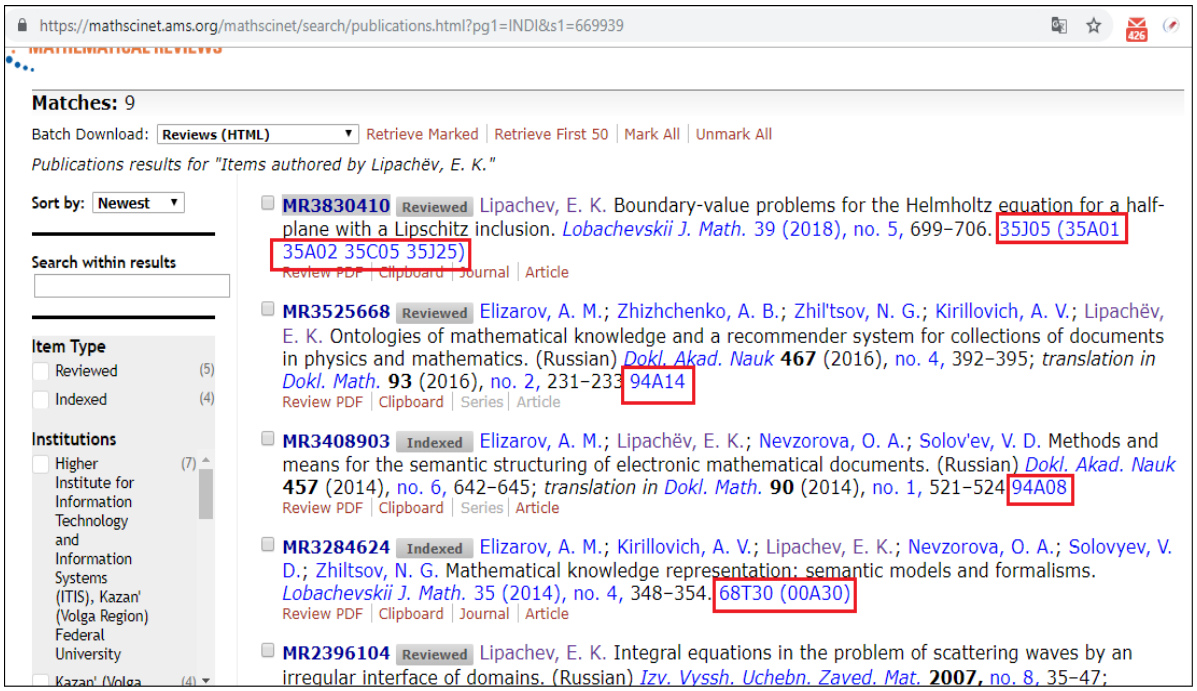


Рис. 3. Экстракция кодов классификаторов, приведенных в профиле эксперта в MathSciNet

Далее, из статей, в процессе рецензирования которых участвовал эксперт (как рецензент, составитель тематического выпуска или член редколлегии, представивший статью), извлекаются коды MSC, выполняется преобразование в коды MSC2020, после чего эти коды добавляются в профиль рассматриваемого эксперта (Рис. 4).

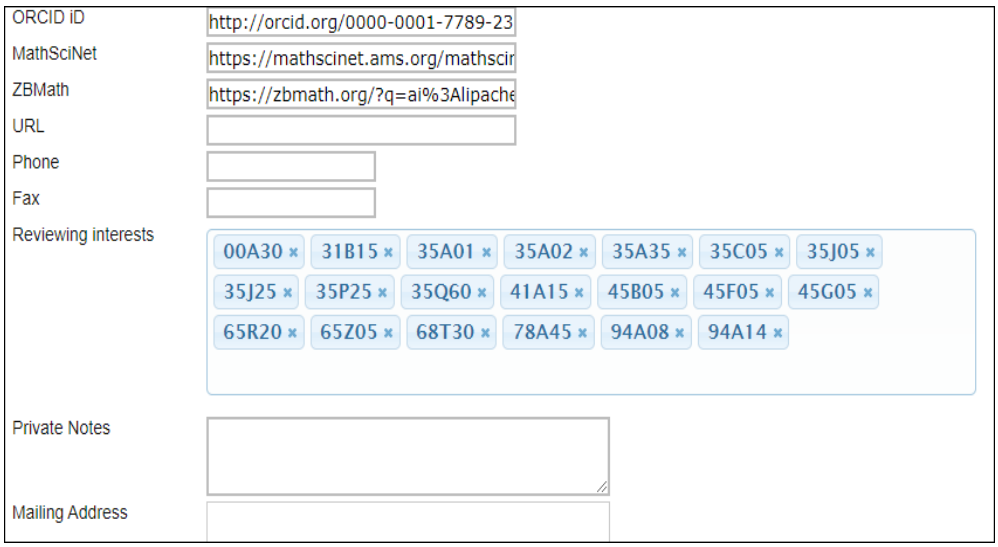


Рис. 4. Управление профилем рецензента в рекомендательной системе на платформе OJS

Если код классификатора уже присутствует в профиле, то увеличивается вес этого кода (Рис. 5).

user_id	controlled_vocab_entry_id	tf
5	440	0.33333333333333
5	441	0.083333333333333
5	442	0.083333333333333
5	443	0.083333333333333
5	444	0.16666666666667
5	445	0.083333333333333
5	446	2.75
5	447	0.33333333333333
5	448	0.083333333333333

controlled_vocab_entry_id	locale	setting_name	setting_value	setting_type
446		interest	35J05	string

Рис. 5. Учёт профессиональных интересов рецензента с помощью системы весовых коэффициентов

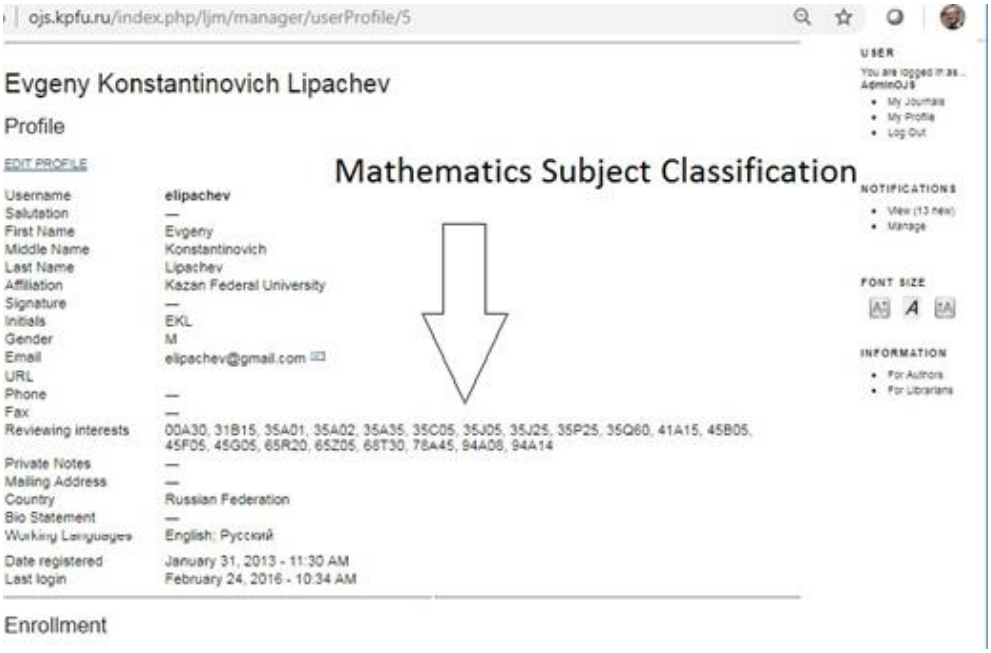


Рис. 6. Включение кодов классификаторов MSC в профиль рецензента информационной системы

В случае, если эксперт, просмотрев статью, выбрал опцию “UNABLE TO DO THE REVIEW” (отказ от рецензирования) на странице “Review” информационной системы OJS, то у кодов в профиле эксперта, совпадающих с кодами статьи, вес уменьшается. На Рис. 6 приведен фрагмент профиля эксперта с кодами MSC.

Алгоритм формирования профилей экспертов и уточнения включаемых кодов классификаторов основан на использовании методы информационного поиска [35–37]. Создается модель векторного пространства документов, в котором документами будут рецензенты, а соответствующие им термы – коды классификатора из всего списка кодов, ассоциированных с рецензентом:

$$tf(t,d) = {f_{t,d}}\Big/{\sum_{t' \in d} f_{t',\ d}},$$

где $f_{t,d}$ – число вхождений кода MSC 2010 в общий список кодов у рецензента d, а в знаменателе стоит общее число кодов в профиле рецензента.

Как правило, основным кодом, определяющим тематику статьи, является первый код. Для уточнения используются остальные коды, приведённые в статье. Поэтому в алгоритме первому в списке классификатору присваивается самый высокий вес (например, вдвое больше вторичного).

После вычисления веса каждого извлеченного кода производится запись в профиль. Для выполнения этой операции требуется внести изменения в структуру таблицы базы данных. Для этого в OJS предусмотрена возможность изменения таблицы с использованием специального XML-файла.

Подбор рецензентов производится на основе векторной модели документа [38]. В соответствии с методом *tf-idf* для рассматриваемой статьи вычисляются веса по трем слоям классификаторов MSC

$$tf(t,\ d) \times idf(t,\ D) = \left(0.5 + \left(0.5 + \frac{f_{t,d}}{\max\limits_{t' \in d} f_{t' \in d}}\right)\right) \times \log\frac{N}{df}.$$

Далее производится вычисление косинусной меры между вектором запроса, образованного с использованием предыдущей формулы, и векторами, сформированными на основе данных из профилей экспертов, представленных в системе:

$$sim(d_1,\ d_2) = \frac{(\vec{V}(d_1), \vec{V}(d_2))}{\|\vec{V}(d_1)\|\|\vec{V}(d_2)\|}.$$

Следующим шагом является вычисление косинусной меры для каждого слоя классификатора, затем производится суммирование с использованием уменьшающегося коэффициента для каждого слоя (0,5; 0,3 и 0,2 соответственно):

$$score\left(msc_{ik},\ Reviewer_j\right) = similarty\ \times\ \delta.$$

Отметим, что назначение рецензентов в системе OJS производится пользователем, имеющим роли «Редактор» или «Редактор раздела». Выбор рецензентов производится из списка пользователей системы, имеющих роль «Рецензент». Его можно выполнить только в ручном режиме – на основании профессионального опыта редактора.

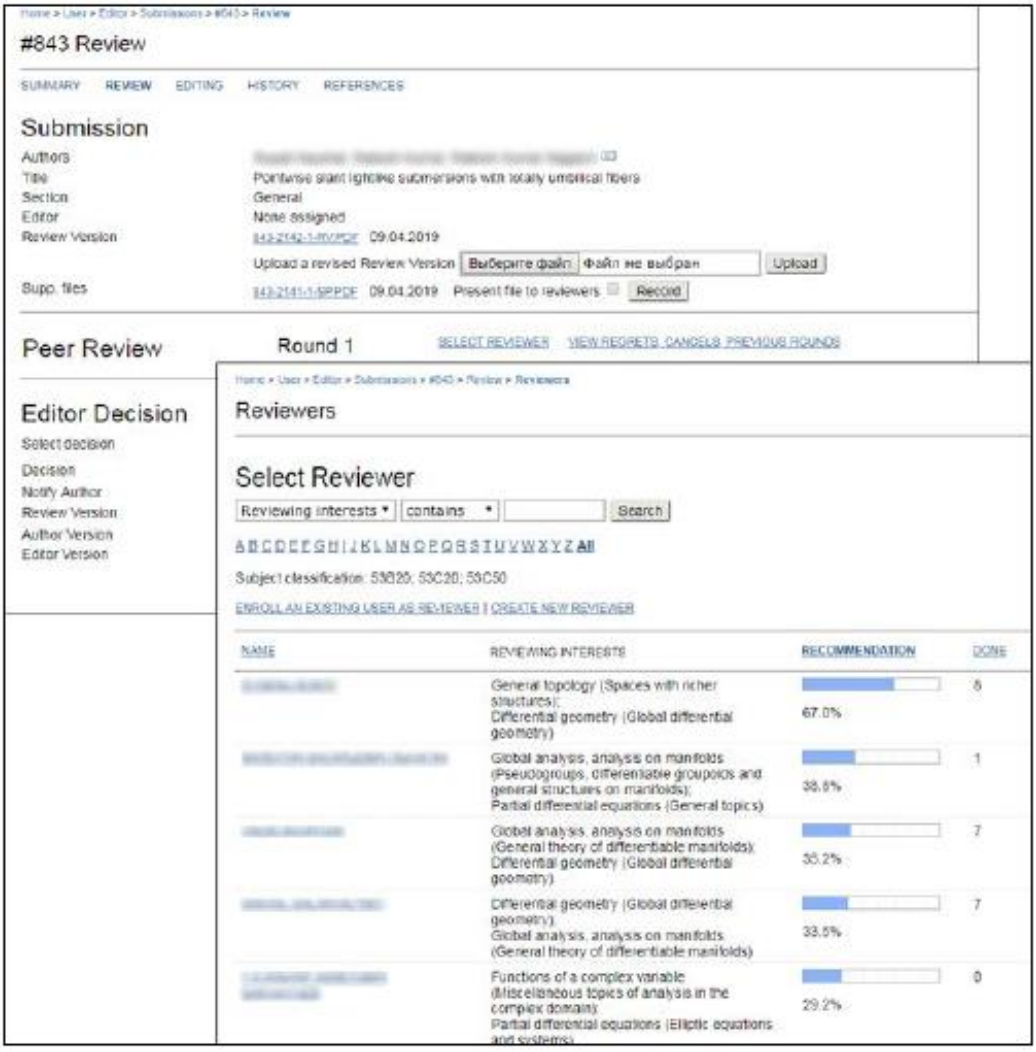


Рис. 7. Сформированная рекомендация, содержащая список наиболее предпочтительных рецензентов для выбранной статьи

Отметим, что система OJS предоставляет возможности изменения своего функционала путем добавления модулей (плагинов), составленных по определенным правилам. Поскольку система написана на языке PHP и является открытой, имеется возможность внести изменения в функционал системы без специального инструментария OJS.

ЗАКЛЮЧЕНИЕ

Таким образом, разработанный метод автоматизированного подбора возможных рецензентов для научных работ, поступающих для публикации в информационную систему научного журнала, основан на использовании Mathematics Subject Classification 2020 для определения предметной области, к которой относятся результаты исследования, представленные к публикации. Метод реализован в виде сервиса по технологии плагинов OJS и может быть использован в любом математическом журнале, функционирующем на платформе OJS. Метод апробирован в информационной системе научного журнала Lobachevskii Journal of Mathematics.

Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (РФФИ) в рамках проекта № 18-29-03086, РФФИ и Правительства Республики Татарстан в рамках проекта № 18-47-160012 и в рамках программы развития Научно-образовательного математического центра Приволжского федерального округа, номер соглашения № 075-02-2020-1478. Настоящая статья содержит также результаты, полученные в рамках проекта «Мониторинг и стандартизация развития и использования технологий хранения и анализа больших данных в цифровой экономике Российской Федерации», выполняемого в рамках реализации Программы Центра компетенций Национальной технологической инициативы «Центр хранения и анализа больших данных», поддерживаемого Министерством науки и высшего образования Российской Федерации по Договору Московского государственного университета имени М.В. Ломоносова с Фондом поддержки проектов Национальной технологической инициативы от 15.08.2019 № 7/1251/2019.

СПИСОК ЛИТЕРАТУРЫ

1. *Heller L., The R., Bartling S.* Dynamic Publication Formats and Collaborative Authoring // In: S. Bartling and S. Friesike (eds.), Opening Science, 2014. P. 191–211. https://doi.org/10.1007/978-3-319-00026-8_13.

2. *Горбунов-Посадов М.М*. Живая публикация // Открытые системы. СУБД. 2011. № 4. С. 48–49. URL: https://keldysh.ru/gorbunov/live.htm. Размещено 02.06.2011. Редакция от 04.04.2020.

3. *Henitiuk V., O'Sullivan C.* Aims and Scope: Journal Identity and Twenty-First-Century Scholarly Publishing // In: Translation and Academic Journals. The Evolving Landscape of Scholarly Publishing. Sun Y. (Ed.) Palgrave Macmillan US, 2015. P. 15–35. https://doi.org/10.1057/9781137522092_2.

4. *Елизаров А.М., Кириллович А.В., Липачёв Е.К.* Блоги в системе научных коммуникаций // Ученые записки Института социально-гуманитарных знаний. 2017. Т. 15. № 1. С. 209–214.

5. *Olver P.* Journals in Flux // Notices of the AMS. 2011. V. 58, No 8. P. 1124–1126.

6. *Brienza C.* Activism, Legitimation, or Record: Towards a New Tripartite Typology of Academic Journals // Journal of Scholarly Publishing. 2015. 46 (2). P. 141–157. https://doi.org/10.3138/jsp.46.2.02.

7. *Binswanger M.* Excellence by Nonsense: The Competition for Publications in Modern Science // In: Bartling S., Friesike S. (Eds). Opening Science. The Evolving Guide on How the Internet is Changing Research, Collaboration and Scholarly Publishing. Springer International Publishing. 2014. P. 49–72. https://doi.org/10.1007/978-3-319-00026-8_3.

8. *Sadeghi A., Capadisli S., Wilm J., Lange C., Mayr P.* Opening and Reusing Transparent Peer Reviews with Automatic Article Annotation// Publications. 2019. V. 7 (13). P. 1–12. https://doi.org/10.3390/publications7010013.

9. *Sadeghi A., Wilm J., Mayr P., Lange C.* Opening Scholarly Communication in Social Sciences by Connecting Collaborative Authoring to Peer Review // Information-Wissenschaft & Praxis. 2017. P. 163–170.

10. *Sadeghi A., Capadisli S., Wilm J., Lange C., Mayr P.* Automatically Annotating Articles Towards Opening and Reusing Transparent Peer Reviews // arXiv:1812.01027.

11. *Галявиева М.С., Елизаров А.М., Липачёв Е.К.* Цифровая инфраструктура электронного научного журнала: автоматизация редакционно-издательских процессов и система сервисов // Электронные библиотеки. 2016. 19 (5). С. 408–465. URL: https://elibrary.ru/download/elibrary_28882615_54981820.pdf.

12. *Horbach S.P.J.M., Halffman W.* The ability of different peer review procedures to flag problematic publications // Scientometrics. 2019. V. 118. P. 339–373. https://doi.org/10.1007/s11192-018-2969-2. URL: https://link.springer.com/article/10.1007/s11192-018-2969-2.

13. *Елизаров А.М., Зуев Д.С., Липачёв Е.К.* Управление жизненным циклом электронных публикаций в информационной системе научного журнала // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. 2014. № 4. С. 81–88.

14. *Елизаров А.М., Зуев Д.С., Липачёв Е.К.* Сервисы поддержки жизненного цикла электронных научных публикаций // В сборнике: Научный сервис в сети Интернет: многообразие суперкомпьютерных миров Труды Международной суперкомпьютерной конференции. Российская академия наук. Суперкомпьютерный консорциум университетов России. 2014. С. 436–438.

15. *Елизаров А.М., Зуев Д.С., Липачёв Е.К.* Информационные системы управления электронными научными журналами // Научно-техническая информация. Серия 1: Организация и методика информационной работы. 2014. № 3. С. 31–38.

16. *MacGregor J., Stranack K., Willinsky J.* The Public Knowledge Project: Open Source Tools for Open Access to Scholarly Communication // In: Bartling S., Friesike S. (Eds). Opening Science. The Evolving Guide on How the Internet is Changing Research, Collaboration and Scholarly Publishing, Springer International Publishing, 2014. P. 165–175. https://doi.org/10.1007/978-3-319-00026-8_3.

17. *Елизаров А.М., Липачёв Е.К., Малахальцев М.А.* Веб-технологии в работе электронного математического журнала Lobachevskii Journal of Mathematics // В сборнике: Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ Труды Всероссийской научной конференции. Московский государственный университет им. М.В. Ломоносова, Южный федеральный университет, Институт вычислительной математики РАН. 2007. С. 355–356.

18. *Глухов В.А., Елизаров А.М., Липачёв Е.К., Малахальцев М.А.* Электронные научные издания: переход на технологии семантического веба // Электронные библиотеки. 2007. Т. 10. № 1.

19. *Веселаго В.Г., Елизаров А.М., Липачёв Е.К., Малахальцев М.А.* Формирование и поддержка физико-математических электронных научных изданий: переход на технологии семантического веба // В сборнике: Научно-исследовательский институт математики и механики им. Н.Г. Чеботарева Казанского государственного университета, 2003–2007 гг. Монография. Казань, 2008. С. 456–476.

20. *Ахметов Д.Ю., Елизаров А.М., Липачёв Е.К.* Автоматизация редакционных процессов в информационной системе управления электронными научными журналами // Электронные библиотеки. 2015. Т. 18. № 1–2. С. 32–45.

21. *Ахметов Д.Ю., Елизаров А.М., Липачёв Е.К.* Информационные системы и сервисы комплексной поддержки периодических научных изданий // В сборнике: Научный сервис в сети Интернет труды XVII Всероссийской научной конференции. ИПМ им. М.В. Келдыша. 2015. С. 16–25. URL: https://keldysh.ru/abrau/2015/proc.pdf#page%3D16.

22. Mathematics Subject Classification MSC2010, http://msc2010.org/Default.html, last accessed 2020/04/1.

23. Mathematics Subject Classification MSC2020, https://zbmath.org/static/msc2020.pdf, last accessed 2019/11/21.

24. *Dunne E., Hulek K.* Mathematics Subject Classification 2020 // Notices Amer. Math. Soc. 2020. V. 67 (3). P. 410–411. https://dx.doi.org/10.1090/noti2052.

25. *Khaydarov S.M, Yamalutdinova G.S.* Recommender System of Physical and Mathematical Documents Classification // CEUR Workshop Proceedings. 2018. V. 2260. P. 480–486.

25. *Manouselis N., Drachsler H., Verbert K., Duval E.* Recommender Systems for Learning. Springer, 2013. https://doi.org/10.1007/978-1-4614-4361-2.

26. *Manouselis N., Drachsler H., Verbert K., Duval E.* Recommender Systems for Learning. Springer, 2013. https://doi.org/10.1007/978-1-4614-4361-2.

27. *Ricci F., Rokach L., Shapira B., Kantor P.B. (Eds.)* Recommender Systems Handbook. Springer-Verlag New York, 2011. 842 p.

28. *Ricci F., Rokach L., Shapira B. (Eds.)* Recommender Systems Handbook.

Springer-Verlag New York, 2015. 1003 p.

29. *Елизаров А.М., Жижченко А.Б., Жильцов Н.Г., Кириллович А.В., Липачёв Е.К.* Онтологии математического знания и рекомендательная система для коллекций физико-математических документов // Доклады Академии наук. 2016. Т. 467. № 4. С. 392–395. https://doi.org/10.7868/S0869565216100042.

30. *Smyth B.* Case-based recommendation // In: Brusilovsky A., Kobsa W. (eds). The Adaptive Web: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, Springer, Berlin, 2007. P. 342–376.

31. *Elizarov A.M., Lipachev E.K., Zuev D.S.* Digital Mathematical Libraries: Overview of Implementations and Content Management Services // CEUR Workshop Proceedings. 2017. V. 2022. P. 317–325.

32. *Lange C., Ion P., Dimou A., Bratsas B., Sperber W., Kohlhase M., Antoniou I.* Bringing Mathematics to the Web of Data: The Case of the Mathematics Subject Classification // In: Simperl E., Cimiano P., Polleres A., Corcho O., Presutti V. (eds). The Semantic Web: Research and Applications, ESWC 2012, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2012. V. 7295. P. 763–777: https://doi.org/10.1007/978-3-642-30284-8_58.

33. *Řehůřek R., Sojka P.* Automated Classification and Categorization of Mathematical Knowledge // In: Intelligent computer mathematics, 9th International Conference, AISC 2008, 15th Symposium, CALCULEMUS 2008, 7th International Conference, MKM 2008, Birmingham, UK, July 28 – August 1, 2008. Proceedings, P. 1–15 (2008). https://doi.org/10.1007/978-3-540-85110-3_44.

34. Table of conversions between the 2000 and 2010 versions of the Mathematics Subject Classification (MSC), http://msc2010.org/2000to2010.html, last accessed 2019/11/21.

35. *Ахметов Д.Ю., Елизаров А.М., Липачёв Е.К., Хайдаров Ш.М.* Программный комплекс формирования рекомендаций по подбору рецензентов для научных документов в информационных издательских системах // Свидетельство о регистрации программы для ЭВМ RU 2018611617, 02.02.2018. Заявка № 2017662838 от 11.12.2017. URL: https://elibrary.ru/download/elibrary_39290682_49150826.PDF.

36. *Елизаров А.М., Липачев Е.К., Хайдаров Ш.М.* Метод автоматизированного подбора рецензентов научных статей, реализованный в

информационной системе научного журнала // Научный сервис в сети Интернет: труды XXI Всероссийской научной конференции (23–28 сентября 2019 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2019. С. 318–328. URL: http://keldysh.ru/ abrau/2019/theses/94.pdf doi:10.20948/abrau-2019-94

37. *Elizarov A., Khaydarov Sh., Lipachev E.* The Formation Method of Recommendations in the Process of Scientific Peer Review of Mathematical Papers // CEUR Workshop Proceedings. 2020. V. 2543. P. 126–135. URL: http://ceur-ws.org/Vol-2543/rpaper12.pdf.

38. *Ингерсолл Г.С., Мортон Т.С., Фэррис Э.Л.* Обработка неструктурированных текстов. Поиск, организация и манипулирование. М.: ДМК Пресс, 2015. 414 с

RECOMMENDER SYSTEM IN THE PROCESS OF SCIENTIFIC PEER REVIEW IN MATHEMATICAL JOURNAL

A. M. Elizarov [1][0000-0003-2546-6897], E. K. Lipachev [2][0000-0001-7789-2332], S. M. Khaydarov [3][0000-0002-6835-3289]

[1–3] *N. I. Lobachevskii Institute of Mathematics and Mechanics, Kazan (Volga Region) Federal University, ul. Kremlyovskaya, 35, Kazan, 420008*

[1, 2] *Higher School of Information Technologies and Intelligent Systems, Kazan (Volga Region) Federal University, ul. Kremlyovskaya, 35, Kazan, 420008*

[1]amelizarov@gmail.com, [2]elipachev@gmail.com, [3]15jkeee@gmail.com

Abstract

An approach is proposed for organizing expert evaluation of a scientific document submitted to a mathematical journal. Domain restriction is associated with the use of the Mathematical Sciences Classification System – MSC. A recommendation system is presented that allows you to create a list of possible experts for conducting scientific peer-reviewing on a mathematical article. The recommender system uses the MSC codes presented by the author of the article on the MSC2020 classifiers. If the codes MSC2000 or MSC2010 are indicated in the article, they are automatically converted to codes MSC2020. For each expert, the system supports a personal profile that

contains a set of codes MSC2020, supplemented by numerical characteristics – weights calculated for each code in accordance with the system of accounting for competencies, preferences or refusals to participate in the review procedure. This set is automatically edited if the expert is included in the list of possible reviewers – the weights of several codes increase or decrease, as well as new codes are added. The recommendation system is implemented as an integrated tool (plug-in) of the Open Journal Systems (OJS) platform. The developed method has been tested in the information system of the Lobachevskii Journal of Mathematics (https://ljm.kpfu.ru).

Keywords*: scientific journal information system, Open Journal Systems, peer review workflow, automated reviewers selection, Mathematics Subject Classification 2010, Lobachevskii Journal of Mathematics.*

REFERENCES

1. *Heller L., The R., Bartling S.* Dynamic Publication Formats and Collaborative Authoring // In: S. Bartling and S. Friesike (eds.), Opening Science, 2014. P. 191–211. https://doi.org/10.1007/978-3-319-00026-8_13.

2. *Gorbunov-Posadov M.M.* Zhivaya publikaciya // Otkrytye sistemy. SUBD. 2011. № 4. S. 48–49. URL: https://keldysh.ru/gorbunov/live.htm. Razmeshcheno 02.06.2011. Redakciya ot 04.04.2020.

3. *Henitiuk V., O'Sullivan C.* Aims and Scope: Journal Identity and Twenty-First-Century Scholarly Publishing // In: Translation and Academic Journals. The Evolving Landscape of Scholarly Publishing. Sun Y. (Ed.) Palgrave Macmillan US, 2015. P. 15–35. https://doi.org/10.1057/9781137522092_2.

4. *Elizarov A.M., Kirillovich A.V., Lipachyov E.K.* Blogi v sisteme nauchnyh kommunikacij // Uchenye zapiski Instituta social'no-gumanitarnyh znanij. 2017. T. 15. № 1. S. 209–214.

5. *Olver P.* Journals in Flux // Notices of the AMS. 2011. V. 58, No 8. P. 1124–1126.

6. *Brienza C.* Activism, Legitimation, or Record: Towards a New Tripartite Typology of Academic Journals // Journal of Scholarly Publishing. 2015. 46 (2). P. 141–157. https://doi.org/10.*3138/jsp.46.2.02.*

7. Binswanger M. Excellence by Nonsense: The Competition for Publications in Modern Science // In: Bartling S., Friesike S. (Eds). Opening Science. The Evolving Guide

on How the Internet is Changing Research, Collaboration and Scholarly Publishing. Springer International Publishing. P. 49–72. https://doi.org/10.1007/978-3-319-00026-8_3.

8. *Sadeghi A., Capadisli S., Wilm J., Lange C., Mayr P.* Opening and Reusing Transparent Peer Reviews with Automatic Article Annotation// Publications. 2019. V. 7 (13). P. 1–12. https://doi.org/10.3390/publications7010013.

9. *Sadeghi A., Wilm J., Mayr P., Lange C.* Opening Scholarly Communication in Social Sciences by Connecting Collaborative Authoring to Peer Review // Information-Wissenschaft & Praxis. 2017. P. 163–170.

10. *Sadeghi A., Capadisli S., Wilm J., Lange C., Mayr P.* Automatically Annotating Articles Towards Opening and Reusing Transparent Peer Reviews // arXiv:1812.01027.

11. *Galyavieva M.S., Elizarov A.M., Lipachyov E.K.* Cifrovaya infrastruktura elektronnogo nauchnogo zhurnala: avtomatizaciya redakcionno-izdatel'skih processov i sistema servisov // Elektronnye biblioteki. 2016. 19 (5). S. 408–465. URL: https://elibrary.ru/download/elibrary_28882615_54981820.pdf.

12. *Horbach S.P.J.M., Halffman W.* The ability of different peer review procedures to flag problematic publications // Scientometrics. 2019. V. 118. P. 339–373. https://doi.org/10.1007/s11192-018-2969-2. URL: https://link.springer.com/article/10.1007/s11192-018-2969-2.

13. *Elizarov A.M., Zuev D.S., Lipachyov E.K.* Upravlenie zhiznennym ciklom elektronnyh publikacij v informacionnoj sisteme nauchnogo zhurnala // Vestnik Voronezhskogo gosudarstvennogo universiteta. Seriya: Sistemnyj analiz i informacionnye tekhnologii. 2014. № 4. S. 81–88.

14. *Elizarov A.M., Zuev D.S., Lipachyov E.K.* Servisy podderzhki zhiznennogo cikla elektronnyh nauchnyh publikacij // V sbornike: Nauchnyj servis v seti Internet: mnogoobrazie superkomp'yuternyh mirov Trudy Mezhdunarodnoj superkomp'yuternoj konferencii. Rossijskaya akademiya nauk. Superkomp'yuternyj konsorcium universitetov Rossii. 2014. S. 436–438.

15. *Elizarov A.M., Zuev D.S., Lipachyov E.K.* Informacionnye sistemy upravleniya elektronnymi nauchnymi zhurnalami // Nauchno-tekhnicheskaya informaciya. Seriya 1: Organizaciya i metodika informacionnoj raboty. 2014. № 3. S. 31–38.

16. *MacGregor J., Stranack K., Willinsky J.* The Public Knowledge Project: Open Source Tools for Open Access to Scholarly Communication // In: Bartling S., Friesike S.

(Eds). Opening Science. The Evolving Guide on How the Internet is Changing Research, Collaboration and Scholarly Publishing, Springer International Publishing, 2014. P. 165–175. https://doi.org/10.1007/978-3-319-00026-8_3.

17. *Elizarov A.M., Lipachev E.K., Malahal'cev M.A.* Veb-tekhnologii v rabote elektronnogo matematicheskogo zhurnala Lobachevskii Journal of Mathematics // V sbornike: Nauchnyj servis v seti Internet: mnogoyadernyj komp'yuternyj mir. 15 let RFFI Trudy Vserossijskoj nauchnoj konferencii. Moskovskij gosudarstvennyj universitet im. M.V. Lomonosova, Yuzhnyj federal'nyj universitet, Institut vychislitel'noj matematiki RAN. 2007. S. 355–356.

18. *Gluhov V.A., Elizarov A.M., Lipachyov E.K., Malahal'cev M.A.* Elektronnye nauchnye izdaniya: perekhod na tekhnologii semanticheskogo veba // Elektronnye biblioteki. 2007. T. 10. № 1.

19. *Veselago V.G., Elizarov A.M., Lipachev E.K., Malahal'cev M.A.* Formirovanie i podderzhka fiziko-matematicheskih elektronnyh nauchnyh izdanij: perekhod na tekhnologii semanticheskogo veba // V sbornike: Nauchno-issledovatel'skij institut matematiki i mekhaniki im. N.G. Chebotareva Kazanskogo gosudarstvennogo universiteta, 2003–2007 gg. Monografiya. Kazan', 2008. S. 456–476.

20. *Ahmetov D.Yu., Elizarov A.M., Lipachev E.K.* Avtomatizaciya redakcionnyh processov v informacionnoj sisteme upravleniya elektronnymi nauchnymi zhurnalami // Elektronnye biblioteki. 2015. T. 18. № 1–2. S. 32–45.

21. *Ahmetov D.Yu., Elizarov A.M., Lipachev E.K.* Informacionnye sistemy i servisy kompleksnoj podderzhki periodicheskih nauchnyh izdanij // V sbornike: Nauchnyj servis v seti Internet trudy XVII Vserossijskoj nauchnoj konferencii. IPM im. M.V. Keldysha. 2015. S. 16–25. URL: https://keldysh.ru/abrau/2015/proc.pdf#page%3D16.

22. Mathematics Subject Classification MSC2010, http://msc2010.org/Default.html, last accessed 2020/04/1.

23. Mathematics Subject Classification MSC2020, https://zbmath.org/static/msc2020.pdf, last accessed 2019/11/21.

24. *Dunne E., Hulek K.* Mathematics Subject Classification 2020 // Notices Amer. Math. Soc. 2020. V. 67 (3). P. 410–411. https://dx.doi.org/10.1090/noti2052.

25. *Khaydarov S.M, Yamalutdinova G.S.* Recommender System of Physical and Mathematical Documents Classification // CEUR Workshop Proceedings. 2018. V. 2260.

P. 480–486.

26. *Manouselis N., Drachsler H., Verbert K., Duval E.* Recommender Systems for Learning. Springer, 2013. https://doi.org/10.1007/978-1-4614-4361-2.

27. *Ricci F., Rokach L., Shapira B., Kantor P.B. (Eds.)* Recommender Systems Handbook. Springer-Verlag New York, 2011. 842 p.

28. *Ricci F., Rokach L., Shapira B. (Eds.)* Recommender Systems Handbook. Springer-Verlag New York, 2015. 1003 p.

29. *Elizarov A.M., Lipachev E.K., Zhizhchenko A.B., Zhil'tsov N.G., Kirillovich A.V.* Mathematical Knowledge Ontologies and Recommender Systems for Collections of Documents in Physics and Mathematics // Doklady Mathematics. 2016. T. 93. № 2. C. 231–233.https://doi.org/10.1134/S1064562416020174.

30. *Smyth B.* Case-based recommendation // In: Brusilovsky A., Kobsa W. (eds). The Adaptive Web: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, Springer, Berlin, 2007. P. 342–376.

31. *Elizarov A.M., Lipachev E.K., Zuev D.S.* Digital Mathematical Libraries: Overview of Implementations and Content Management Services // CEUR Workshop Proceedings. 2017. V. 2022. P. 317–325.

32. *Lange C., Ion P., Dimou A., Bratsas B., Sperber W., Kohlhase M., Antoniou I.* Bringing Mathematics to the Web of Data: The Case of the Mathematics Subject Classification // In: Simperl E., Cimiano P., Polleres A., Corcho O., Presutti V. (eds). The Semantic Web: Research and Applications, ESWC 2012, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2012. V. 7295. P. 763–777. https://doi.org/ 10.1007/978-3-642-30284-8_58.

33. *Řehůřek R., Sojka P.* Automated Classification and Categorization of Mathematical Knowledge // In: Intelligent computer mathematics, 9th International Conference, AISC 2008, 15th Symposium, CALCULEMUS 2008, 7th International Conference, MKM 2008, Birmingham, UK, July 28 – August 1, 2008. Proceedings, P. 1–15 (2008). https://doi.org/10.1007/978-3-540-85110-3_44.

34. Table of conversions between the 2000 and 2010 versions of the Mathematics Subject Classification (MSC), http://msc2010.org/2000to2010.html, last accessed 2019/11/21.

35. *Ahmetov D.Yu., Elizarov A.M., Lipachyov E.K., Hajdarov Sh.M.* Programmnyj kompleks formirovaniya rekomendacij po podboru recenzentov dlya nauchnyh

dokumentov v informacionnyh izdatel'skih sistemah // Svidetel'stvo o registracii programmy dlya EVM RU 2018611617, 02.02.2018. Zayavka № 2017662838 ot 11.12.2017. URL: https://elibrary.ru/download/ elibrary_39290682_49150826.PDF.

36. *Elizarov A.M., Lipachev E.K., Hajdarov Sh.M.* Metod avtomatizirovannogo podbora recenzentov nauchnyh statej, realizovannyj v informacionnoj sisteme nauchnogo zhurnala // Nauchnyj servis v seti Internet: trudy XXI Vserossijskoj nauchnoj konferencii (23–28 sentyabrya 2019 g., g. Novorossijsk). M.: IPM im. M.V. Keldysha, 2019. S. 318–328. URL: http://keldysh.ru/abrau/2019/theses/94.pdf doi:10.20948/ abrau-2019-94

37. *Elizarov A., Khaydarov Sh., Lipachev E.* The Formation Method of Recommendations in the Process of Scientific Peer Review of Mathematical Papers // CEUR Workshop Proceedings. 2020. V. 2543. P. 126–135. URL: http://ceur-ws.org/Vol-2543/rpaper12.pdf.

38. *Ingersoll G.S., Morton T.S., Farris A.L.* Taming Text. How to Find, Organize, and Manipulate It. Manning Publications Co., 2013. 299 p.

СВЕДЕНИЯ ОБ АВТОРАХ

ЕЛИЗАРОВ Александр Михайлович – доктор физико-математических наук, профессор, заслуженный деятель науки Республики Татарстан, профессор кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Alexander Mikhailovich ELIZAROV – Doctor of Physics and Mathematics, Professor, Honoured Worker of Science of the Republic of Tatarstan, Kazan Federal University. Current scientific interests: data mining, recommender systems, cloud computing, knowledge extraction technologies.

email: amelizarov@gmail.com; ORCID: 0000-0003-2546-6897

ЛИПАЧЁВ Евгений Константинович – кандидат физико-математических наук, доцент кафедры Интеллектуальных технологий поиска Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Evgeny Konstantinovich LIPACHEV – Candidate of Physics and Mathematics, Associate Professor, Kazan Federal University. Current scientific interests: data mining, recommender systems, cloud computing, knowledge extraction technologies.

email: elipachev@gmail.com; ORCID: 0000-0001-7789-2332

ХАЙДАРОВ Шамиль Махмутович – ассистент кафедры компьютерной математики и информатики Института математики и механики им. Н.И. Лобачевского Казанского (Приволжского) федерального университета.

Shamil Makhmutovich KHAYDAROV – Assistant, Department of Computer Mathematics and Informatics, Institute of Mathematics and Mechanics N.I. Lobachevsky Kazan (Volga) Federal University. Current scientific interests: data mining, recommender systems, cloud computing, knowledge extraction technologies.

email: 15jkeee@gmail.com; ORCID: 0000-0002-6835-3289

Материал поступил в редакцию 15 января 2020 года

УДК 001.89 + 001.812 + 004.051 + 004.738.5 + 004.832.28

ТАКТИЧЕСКАЯ СОРТИРОВКА УПРАВЛЕНЧЕСКИХ ЗАДАЧ ПРИ ИХ АДМИНИСТРИРОВАНИИ ПОСРЕДСТВОМ МЕТОК ПРИОРИТЕТОВ, СПЕЦИФИКАЦИЙ И АФФИЛИАЦИЙ

Ф. О. Каспаринский

ООО «МАСТЕР-МУЛЬТИМЕДИА», г. Москва

felix@kasparinsky.pro

Аннотация

Проанализирована специфика функционала программ управления стратегическими, тактическими и оперативными задачами. Предложена методика предварения названий оперативных задач тактическими метками Приоритетов, Спецификаций и Аффилиаций. Аббревиатуры меток формируются таким образом, чтобы обеспечить правильную расстановку приоритетов при сортировке задач по алфавитному порядку. Квадранты матрицы Приоритетов Д. Эйзенхауэра обозначаются двухбуквенными метками: важно срочно (IF – Important, Fast); важно бессрочно (IS – Important, Slow); не важно, но оперативно (UF – Unimportant, Fast): не важно и не срочно (US – Unimportant, Slow). Метки матрицы Спецификаций информационной среды (RA, RI, SA, SI) компонуются из взаимоисключающих свойств доступности Сети (I – Internet и A – Autonomous) и наличия редуцированного или специального функционала (R– Reduced и S – Special). Метки Транспортной спецификации (TA, TB, TC, TP) позволяют сортировать задачи, требующие перемещения (T – Translocation) на самолёте (A – Airplane), автобусе (B – Bus), автомобиле (C – Car) и пешком (P – Pedestrian), соответственно. Трёхбуквенные метки Аффилиации (принадлежности физическому или юридическому лицу) формируются из первых букв имени, отчества и фамилии или наименования лаборатории, компании, проекта. Тактические метки ускоряют принятие решений при формировании ежедневного списка оперативных задач.

***Ключевые слова**: задача, планирование, управление, приоритет, спецификация, аффилиация, метка, оперативный, тактический*

ВВЕДЕНИЕ

Планирование — это вид деятельности, связанный с постановкой целей будущих действий и обеспечением выполнения задач посредством получения и рационального распределения ресурсов. Эффективность планирования зависит от корректного определения цели и соответствующих ей задач на начальном этапе действий. Цель отвечает на вопрос «Чего нужно достигнуть?» и предусматривает положительную динамику, изменение текущего состояния некоей сущности в сторону улучшения, удовлетворения потребностей или требований. При постановке реальных целей важна иерархическая систематизация действий, выполнение которых необходимо для достижения успеха. Задачи отвечают на вопрос «Какими действиями можно достигнуть цели?». Когда количество задач увеличивается с десятков до нескольких сотен, требуется их постоянное перераспределение в соответствии с модификацией целевых приоритетов и доступностью ресурсов.

Искусство администрирования — способность выделять из комплексной задачи ансамбли параллельных операций (к примеру, по сходству используемых ресурсов) и назначать их соответствующим по возможностям исполнителям. Программы управления задачами влияют на эффективность их администрирования в соответствии с адекватностью использования функционала применительно к масштабу и структуре. В статье резюмирован 25-летний опыт административного управления задачами посредством программного инструментария.

Эффективность работы с задачей определяется результатами фазы инициации [1], в ходе которой происходят:

1. Определение новизны задачи. Если задача не нова, алгоритм ее решения находят в группе задач категории «Инструкции и схемы работ». Если задача нова, то анализ переходит к Стадии 2.
2. Установление структуры задачи, её основных стадий и их этапов (при наличии) с определением приоритетности и специфики используемых ресурсов.
3. Выявление параллельных процессов разных задач и организация их синхронного выполнения.
4. Идентификация задачи в соответствии с фазами, стадиями и этапами достижения цели (текстовая, пиктографическая, цветовая, звуковая).

Для управления комплексными задачами с последовательными ансамблями параллельных операций (фазами) используется стратегическое планирование. Тактическое планирование заключается в распределении ресурсов внутри ансамбля параллельных операций разных задач. Оперативное планирование необходимо для формирования правильной линейной последовательности этапов каждой задачи. Организация идентификации задач в соответствии с приоритетами и специфическими особенностями дополняет оперативное планирование тактическим функционалом. Опыт показывает, что использование избыточного функционала программ для управления стратегическими задачами в оперативной деятельности контрпродуктивно. С другой стороны, эффективность решения оперативных задач может быть увеличена посредством использования инструментария тактического планирования.

1. СПЕЦИФИКА ФУНКЦИОНАЛА ПРОГРАММ УПРАВЛЕНИЯ СТРАТЕГИЧЕСКИМИ, ТАКТИЧЕСКИМИ И ОПЕРАТИВНЫМИ ЗАДАЧАМИ

Информационный инструментарий для стратегического планирования должен обеспечивать рациональное перераспределение ресурсов, определять критические стадии и аргументировать материальное вознаграждение или наказание сотрудников. Для визуализации стратегического планирования с начала XX века применяются диаграммы Гантта [2], которые широко используются в современных программах профессионального управления проектами [3]. Тактическое управление приоритетами задач с разнообразной спецификацией параллельно используемых ресурсов эффективно осуществляется в программах ассоциативного картирования [1, 4–6]. Организация оперативного планирования в приложениях для работы с почтой, событиями и задачами, подобных классическому Microsoft Office Outlook, была упрощена до формирования списка задач с опциональными пометками «Важно», возможностью окраски категорий и установки напоминаний о дате выполнения. С 1995 года функционал подобных программ неоднократно модифицировался, но сохранял непригодность для управления сотнями задач и синхронизации между разнообразными устройствами пользователя. В первое десятилетие XXI века предпринимались попытки организовать синхронизацию оперативных задач между классическими приложениями стационарных компьютеров и мобильной техникой, но они были далеки от ути-

литарности вследствие постоянных сбоев в работе, приводивших к дублированию задач или блокированию работы с потерей данных [7, 8]. По этой причине в ближайшее время классические программы с функционалом оперативного управления событиями и задачами, такие, как Microsoft Office Outlook 2010, прекратят своё существование [9].

Ситуация начала изменяться к лучшему после появления «облачных» сервисов оперативного обмена данными между кроссплатформенными мобильными устройствами по стандарту BYOD [10]. Современные кроссплатформенные приложения оперативного управления задачами поддерживают учетные записи Office 365, Exchange, Outlook.com, Gmail, Yahoo! и других популярных служб. На смену программам, подобным Microsoft Office Outlook, пришли мобильные приложения, такие, как «Почта и календарь» [11] и сопряжённые с ними облачные сервисы [12]. Однако замена не привязанных ко времени задач событиями календаря на практике доказала свою неэффективность: несвоевременные напоминания о задачах приучают пользователя игнорировать их. Для оперативного управления задачами требовалось приложение со специфическим функционалом.

2. ОПТИМАЛЬНЫЙ ФУНКЦИОНАЛ ПРОГРАММ ДЛЯ УПРАВЛЕНИЯ ОПЕРАТИВНЫМИ ЗАДАЧАМИ

19 апреля 2017 года компания Microsoft представила новый сервис “To-Do” для организации персональных задач. Базовый функционал “To-Do” был унаследован от сервиса Wunderlist, приобретённого Microsoft вместе с командой разработчиков в июне 2015 года [13]. К моменту трансформации сервиса Wunderlist у него было уже 13 миллионов пользователей. Адаптированный к кроссплатформенной информационной среде сервис “To-Do“ [14] в реальном времени обеспечивает синхронизацию задач для Windows, Android, iPhone, iPad, а также «умных» часов Apple Watch и Android Wear. Традиционный функционал задачи (срок выполнения, время напоминания, повтор, комментарий) дополнен опцией создания этапов, названия которых могут содержать гиперссылки. Ещё одна важная возможность – добавление файлов к задачам. К началу 2019 года базовый вариант “To-Do“ содержал папки «Мой день», «Важно», «Запланирова-

но» и «Задачи», которые могли быть видоизменены или дополнены пользователем.

Многие задачи с различным приоритетом, принадлежностью и сроком выполнения создаются в результате работы с ежедневным набором корреспонденции. Техническая задача по копированию названий создаваемых задач из сообщений почтового менеджера и формирование ссылок на них требуют затрат времени. Для увеличения эффективности работы с производными от корреспонденции задачами в мае 2019 года в программу управления задачами Microsoft To-Do был добавлен функционал автоматического формирования задач с гиперссылками вида «Открыть в Outlook» на одноименные сообщения, помечаемые при работе в почтовом менеджере. Такие задачи помещаются в специальную папку «Помеченные сообщения» и сортируются по сроку создания. Наименования этих задач в Microsoft To-Do могут быть модифицированы посредством дополнения меток Приоритетов, Спецификаций, Действующих лиц. Между помеченными сообщениями в Outlook и соответствующими задачами в Microsoft To-Do устанавливается реципрокная связь, обеспечивающая двунаправленную синхронизацию изменения статуса выполнения или возобновления задачи со снятием или обратной постановкой метки сообщения. Для надёжной синхронизации статусов сообщений Outlook и связанных с ними задач Microsoft To-Do рекомендуется заблаговременный запуск обоих приложений до начала работы с содержимым.

3. ПРОБЛЕМА СТРУКТУРИРОВАНИЯ ПОЛИМОРФНЫХ ЗАДАЧ

На основании опыта длительного использования функционала планирования задач в Microsoft Outlook [9] и сопряженных мобильных приложениях [8] была доказана практическая неэффективность оперативной выборки ежедневного набора задач из множества папок с иерархической структурой. Многие важные задачи не имеют срока выполнения, и их обнаружение для включения в план действий сопряжено со значительным расходом ресурса времени.

Для формирования набора задач в папке «Мой день» сервис Microsoft To-Do автоматически предлагает предстоящие задачи с близким сроком выполнения или напоминания, просроченные и недавно добавленные задачи. Практический опыт показал, что эффективность работы с задачами снижается, если их

ежедневное количество превышает десяток. Ещё одна проблема – полиморфизм задач (принадлежность, приоритет, специфика). Распределение по папкам – не выход, поскольку логика распределения задач нарушается при их группировке в разделе «Мой день». Для оптимизации управления многочисленными полиморфными задачами инструментарий оперативного планирования следует дополнить элементами тактического планирования.

4. МЕТКИ ПРИОРИТЕТОВ МАТРИЦЫ ЭЙЗЕНХАУЭРА

В 2017 году при оптимизации способов тактического управления проектной деятельностью мы адаптировали матрицу распределения времени Д. Эйзенхауэра к использованию в ассоциативной карте TheBrain [1]. Классическая матрица принятия решений, называемая по имени ее создателя, Дуайта Эйзенхауэра (34-й президент США), представляет из себя 4 квадранта, образованные комбинациями проекций парных альтернативных свойств степени важности (важно/не важно) и срочности (срочно/не срочно). В целях обеспечения когнитивного комфорта при интеграции матрицы Эйзенхауэра в ассоциативную карту проекта мы предложили модернизировать оригинальные наименования квадрантов до двухбуквенных аббревиатур-меток Приоритетов: важно срочно (IF – Important, Fast); важно бессрочно (IS – Important, Slow); не важно, но оперативно (UF – Unimportant, Fast): не важно и не срочно (US – Unimportant, Slow). Эксперименты в течение последнего полугодия показали, что вставка меток приоритетов перед названием задач обеспечивает их эффективную автоматическую сортировку в папке «Мой день» сервиса управления оперативными задачами Microsoft To-Do. Рассортированные задачи стало проще перераспределять по квантам времени:

IF – 14% времени – 4 кванта – 1 час (авральные проблемы);

IS – 66% времени – 20 квантов – 5 часов (саморазвитие и передача опыта);

UF – 20% времени – 6 квантов – 1.5 часа (делегируемая рутина);

US - 0% – экстра-время – 2 кванта – 0.5 часа (непродуктивные нагрузки).

5. МЕТКИ СПЕЦИФИКАЦИЙ

Для распределения задач, зависящих от доступа в интернет и наличия специфического программно-аппаратного функционала (медиаредакторы, бухгалтерские и управленческие программы, квалифицированные электронные подписи, криптопровайдеры и т. п.), мы предлагаем использовать матрицу Спецификаций информационной среды. По сторонам этой матрицы располагаются взаимоисключающие свойства доступности Сети (I – интернет-доступ и A – автономная работа) и наличия специального функционала (R – редуцированный функционал и S – специальные функции). Пересечение проекций пар взаимоисключающих свойств в соответствующих квадрантах даёт 4 варианта комбинаций-меток: RI –редуцированный по функционалу доступ в Сеть; SI – доступ в Сеть со специальным функционалом; RA – автономная работа с редуцированным функционалом; SA – автономная работа со специальным функционалом.

Для перераспределения задач в соответствии с матрицей принятия решений Эйзенхауэра при изменении доступности интернета и программно-аппаратной базы достаточно дополнить тактические метки Приоритетов в начале названия задач метками Спецификаций.

В соответствии с особенностями ресурсов, требуемых для выполнения задач, целесообразно формировать соответствующие наборы тактических меток Спецификаций. К примеру, метка TP позволяет группировать задачи, требующие пешего перемещения (T – Translocation, P – Pedestrian), а метки TC, TB и TA – ассоциировать задачи, требующие перемещения на автомобиле (C – Car), автобусе (B – Bus) и самолёте (A – Airplane), соответственно.

6. МЕТКИ АФФИЛИАЦИИ

Если выполнение задач связано с разными физическими или юридическими лицами и проектами, целесообразно после тактических меток Приоритетов и Спецификаций позиционировать трехбуквенную аббревиатуру – метку Аффилиации, формируемую из первых букв имени, отчества и фамилии или наименования лаборатории, компании, проекта. К примеру, задачи, относящиеся лично к автору этой статьи, идентифицируются меткой “FOK”, задачи администрирования Лаборатории мультимедийных технологий и компании MASTER-MULTIMEDIA Ltd. отличаются метками “LMT” и “MML” соответственно, задачи управления

компонентами Инфоконтинуума имеют метки ”ICM” (Infocontinuum Components Management), а задачи по организации деятельности самозанятого плательщика налога на профессиональный доход обозначаются аббревиатурами “NPD”. Опыт показал, что при использовании штатного функционала для группировки задач посредством папок наивысшая эффективность достигается, если их наименования соответствуют меткам аффилиации. Это обстоятельство обусловлено относительно большей стабильностью свойства аффилиации задачи по сравнению с приоритетом и технической спецификацией. Изменение меток аффилиации может происходить в результате назначения задач другим действующим лицам (соответствующий функционал «Назначено» появился в программе Microsoft To-Do летом 2019 года).

7. ОПТИМИЗАЦИЯ НАИМЕНОВАНИЯ ЗАДАЧ

Названия задач могут формироваться автоматически из заголовков помеченных сообщений почты. В этом случае их достаточно предварять соответствующими метками Приоритетов, Спецификаций и Действующих лиц. Самостоятельное наименование задач целесообразно осуществлять унифицировано и таким образом, чтобы облегчать полнотекстовый поиск, чувствительный к различиям между заглавными и строчными буквами. Опыт показал, что для самоорганизации порядка выполнения задач по закупкам оборудования их названия рекомендуется начинать с обратной даты заказа (формат ГГГГММДД), а далее с разделением нижним подчеркиванием указывать номер заказа, поставщика и наименования приобретаемого оборудования. К примеру, название задачи «UF TP FOK 20190422_N11080222_OnlineTrade_ROCK-T14-TravelCharger_Logitech-K375s-Keyboard» означает срочную и важную (UF) закупку автором этой статьи (FOK) зарядного устройства ROCK-T14-TravelCharger и клавиатуры Logitech-K375s у поставщика OnlineTrade (номер заказа 11080222 от 22.04.2019) с личным посещением (TP) пункта выдачи заказов. Подобные названия удобно использовать для наименования папок файловой системы, содержащих финансовые и прочие документы (руководства, фотографии комплекта поставки и т. п.), а также в комментариях бухгалтерских программ. В результате все относящиеся к задаче документы и транзакции могут быть быстро найдены системой полнотекстового поиска по характерным признакам.

ЗАКЛЮЧЕНИЕ

Методика распределения оперативных задач в программе Microsoft To-Do посредством дополнения их наименований предварительными тактическими метками Приоритетов, Спецификаций и Аффилиаций формировалась в течение полугода в процессе организации перемещения ресурсов Лаборатории мультимедийных технологий из Биологического факультета МГУ в компанию МАСТЕР-МУЛЬТИМЕДИА (150 задач), модернизации аппаратной базы Инфоконтинуума (50 задач) и организации деятельности самозанятого физического лица (15 задач). Опыт показал, что дополнение названий оперативных задач тактическими метками способствует эффективному достижению поставленных целей множеством действующих лиц, координирующих действия как в стационарных, так и в мобильных условиях.

СПИСОК ЛИТЕРАТУРЫ

1. *Каспаринский Ф.О.* Интеграция матриц интернет-функционала, Эйзенхауэра и SWOT-анализа в ассоциативную карту для управления проектной деятельностью // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18–23 сентября 2017 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2017. С. 194–206. doi:10.20948/abrau-2017-26

2. *Gantt H.L.* Work, Wages and Profit. The Engineering Magazine. New York: The Engineering Magazine, 1910.

3. Microsoft Project поможет вам оптимизировать управление своими проектами, ресурсами и портфелями // Microsoft. URL: https://products.office.com/ ru-ru/project/project-and-portfolio-management-software

4. *Каспаринский Ф.О., Полянская Е.И.* Программы интеллектуального картирования как инструмент научной работы // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи: Труды Международной суперкомпьютерной конференции (20–25 сентября 2010 г., г. Новороссийск). М.: Издательство МГУ, 2010. С. 521–524.

5. *Каспаринский Ф.О.* Использование программ ассоциативного картирования для управления распределенными информационными ресурсами // Научный сервис в сети Интернет: труды XVII Всероссийской научной

конференции (21–26 сентября 2015 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2015. С. 127–134 .

6. *Каспаринский Ф.О.* Администрирование информационной среды посредством ассоциативной карты TheBrain 9 // Научный сервис в сети Интернет: труды XX Всероссийской научной конференции (17–22 сентября 2018 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2018. С. 275–283. URL: http://keldysh. ru/ abrau/2018/theses/40.pdf doi:10.20948/abrau-2018-40

7. Outlook Connector // Microsoft. URL: http://www.microsoft.com/ru-ru/download/confirmation.aspx?id=24677

8. Agenda Fusion // Developer One, 2008. URL: http://www.developerone.com/agendafusion/index.htm

9. Outlook 2010 will be retired in 2020 // Microsoft Office. URL: https://products.office.com/en-us/previous-versions/microsoft-outlook-2010

10. *Каспаринский Ф.О.* Оптимизация структур динамических ассоциативных карт TheBrain 9 для интернет-публикаций по стандарту BYOD // Научный сервис в сети Интернет: труды XX Всероссийской научной конференции (17–22 сентября 2018 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2018. С. 265–274. URL: http://keldysh.ru/abrau/2018/theses/39.pdf doi:10.20948/abrau-2018-39

11. Почта и календарь // Microsoft Store. URL: https://www.microsoft.com/store/productId/9WZDNCRFHVQM

12. Outlook.com. Удобство электронной почты и календаря. URL: https://outlook.live.com/owa/

13. Microsoft To-Do — обзор менеджера задач, который заменит Wunderlist // Android Mobile Review. URL: http://android.mobile-review.com/articles/robosoft/48860/

14. Microsoft To-Do: List, Task & Reminder. URL: https://www.microsoft.com/store/productId/9NBLGGH5R558

15. *Каспаринский Ф.О.* Интернет-сервис как зависимость // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18–23 сентября 2017 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2017. С. 170–182. doi:10.20948/abrau-2017-24

TACTICAL SORTING OF MANAGERIAL TASKS DURING THEIR ADMINISTRATION BY MEANS OF PRIORITY, SPECIFICATIONS AND AFFILIATIONS LABELS

F. O. Kasparinsky

MASTER-MULTIMEDIA LLC, Moscow

felix@kasparinsky.pro

Abstract

The article analyzes the specifics of the functional programs for managing strategic, tactical and operational tasks. A technique for prefixing operational task names with tactical labels of Priorities, Specifications and Affiliations is proposed. Label abbreviations are formed in such a way as to ensure the correct prioritization when sorting tasks in alphabetical order. The quadrants of the D. Eisenhower Priorities matrix are indicated by two-letter marks: important urgently (IF – Important, Fast); important indefinitely (IS – Important, Slow); not important, but promptly (UF – Unimportant, Fast): neither important nor urgent (US – Unimportant, Slow). The labels of the Specifications matrix for the information environment (RA, RI, SA, SI) are composed of mutually exclusive properties of the availability of the Network (I – Internet and A – Autonomous) and the presence of reduced or special functionality (R – Reduced and S – Special). Labels of the transport specification (TA, TB, TC, TP) allow you to sort tasks that require moving (T – Translocation) on an airplane (A), a bus (B), a car (C) and on foot (P – Pedestrian), respectively. Three-letter marks of Affiliations (belonging to an individual or legal entity) are formed from the first letters of the name, middle name and last name or name of the laboratory, company, project. Tactical marks accelerate decision-making when forming a daily list of operational tasks.

Keywords*: task, planning, management, priority, specification, affiliation, label, operational, tactical*

REFERENCES

1. *Kasparinskii F.O.* Integratsiia matrits internet-funktsionala, Eizenkhauera i SWOT-analiza v assotsiativnuiu kartu dlia upravleniia proektnoi deiatelnostiu // Nauchnyi servis v seti Internet: trudy XIX Vserossiiskoi nauchnoi konferentsii (18–23

sentiabria 2017 g., g. Novorossiisk). M.: IPM im. M.V. Keldysha, 2017. S. 194–206. doi:10.20948/abrau-2017-26

2. *Gantt H.L.* Work, Wages and Profit. The Engineering Magazine. New York: The Engineering Magazine, 1910.

3. Microsoft Project pomozhet vam optimizirovat upravlenie svoimi proektami, resursami i portfeliami // Microsoft. URL: https://products.office.com/ru-ru/project/project-and-portfolio-management-software

4. *Kasparinskii F.O., Polianskaia E.I.* Programmy intellektualnogo kartirovaniia kak instrument nauchnoi raboty // Nauchnyi servis v seti Internet: superkompiuternye tsentry i zadachi: Trudy Mezhdunarodnoi superkompiuternoi konferentsii (20–25 sentiabria 2010 g., g. Novorossiisk). M.: Izdatelstvo MGU, 2010. S. 521–524.

5. *Kasparinskii F.O.* Ispolzovanie programm assotsiativnogo kartirovaniia dlia upravleniia raspredelennymi informatsionnymi resursami // Nauchnyi servis v seti Internet: trudy XVII Vserossiiskoi nauchnoi konferentsii (21–26 sentiabria 2015 g., g. Novorossiisk). M.: IPM im. M.V. Keldysha, 2015. S. 127–134.

6. *Kasparinskii F.O.* Administrirovanie informatsionnoi sredy posredstvom assotsiativnoi karty TheBrain 9 // Nauchnyi servis v seti Internet: trudy XX Vserossiiskoi nauchnoi konferentsii (17–22 sentiabria 2018 g., g. Novorossiisk). M.: IPM im. M.V. Keldysha, 2018. S. 275–283. URL: http://keldysh.ru/abrau/2018/theses/40.pdf doi:10.20948/abrau-2018-40

7. Outlook Connector // Microsoft. URL: http://www.microsoft.com/ru-ru/download/confirmation.aspx?id=24677

8. Agenda Fusion // Developer One, 2008. URL: http://www.developerone.com/agendafusion/index.htm

9. Outlook 2010 will be retired in 2020 // Microsoft Office. URL: https://products. office. com/en-us/previous-versions/microsoft-outlook-2010

10. *Kasparinskii F.O.* Optimizatsiia struktur dinamicheskikh assotsiativnykh kart TheBrain 9 dlia internet-publikatsii po standartu BYOD // Nauchnyi servis v seti Internet: trudy XX Vserossiiskoi nauchnoi konferentsii (17–22 sentiabria 2018 g., g. Novorossiisk). M.: IPM im. M.V. Keldysha, 2018. S. 265–274. URL: http://keldysh.ru/abrau/2018/theses/39.pdf doi:10.20948/abrau-2018-39

11. Pochta i kalendar // Microsoft Store. URL: https://www.microsoft.com/ store/productId/9WZDNCRFHVQM

12. Outlook.com. Udobstvo elektronnoi pochty i kalendaria. URL: https://outlook. live.com/owa/

13. Microsoft To-Do — obzor menedzhera zadach, kotoryi zamenit Wunderlist. // Android Mobile Review. URL: http://android.mobile-review.com/ articles/robosoft/48860/

14. Microsoft To-Do: List, Task & Reminder. URL: https://www.microsoft. com/ store/productId/9NBLGGH5R558

15. *Kasparinskii F.O.* Internet-servis kak zavisimost // Nauchnyi servis v seti Internet: trudy XIX Vserossiiskoi nauchnoi konferentsii (18–23 sentiabria 2017 g., g. Novorossiisk). M.: IPM im. M.V. Keldysha, 2017. S. 170–182. doi:10.20948/abrau-2017-24

СВЕДЕНИЯ ОБ АВТОРЕ

КАСПАРИНСКИЙ Феликс Освальдович – кандидат биологических наук, основатель и научный руководитель Лаборатории мультимедийных технологий Биологического факультета МГУ имени М.В. Ломоносова, учредитель и Генеральный директор ООО «МАСТЕР-МУЛЬТИМЕДИА». Сфера научных интересов – формирование информационной среды, дидактически целенаправленное использование мультимедийных технологий.

Felix Osvaldovich KASPARINSKY – Founder and Scientific Director of Multimedia Technologies Laboratory (Biological Faculty, M.V. Lomonosov Moscow State University), Founder and General Director of MASTER-MULTIMEDIA LLC. Research interests include creating an information environment and didactically targeted use of multimedia technologies.

email: felix@kasparinsky.pro

Материал поступил в редакцию 15 ноября 2019 года

УДК 001.51 + 001.812 + 004.051 + 004.057.5 + 004.738.5

СПЕЦИАЛИЗАЦИЯ ИСПОЛЬЗОВАНИЯ МИКРОКОМПЬЮТЕРОВ

Ф. О. Каспаринский

ООО «МАСТЕР-МУЛЬТИМЕДИА», г. Москва

felix@kasparinsky.pro

Аннотация

С 2015 года в информационной среде появились микрокомпьютеры (микроПК), представляющие собой компактный системный блок с минимальным функционалом без периферии. В статье представлены результаты анализа использования 6 различных микрокомпьютеров в различных сферах деятельности. Цель исследования – определить лимитирующие факторы, влияющие на эффективность целевого применения микрокомпьютеров. Установлено, что для научно-образовательных презентаций, офисной и трейдерской деятельности в настоящее время целесообразно использовать безвентиляторные микрокомпьютеры с перфорированным корпусом и внутренней WiFi-антенной, не менее 4 Гб оперативной и 64 Гб постоянной памяти, разъёмом карты памяти microSD (TF, не менее 128 Гб, файловая система NTFS), графическим ускорителем GPU Intel HD Graphics, интерфейсами USB3.0 и HDMI. На основе сравнительных экспериментов созданы методические рекомендации по оптимизации конфигурирования аппаратно-программной среды микрокомпьютеров в стационарных и мобильных условиях. Проанализированы проблемы крупных обновлений Windows 10, а также совместимости программного обеспечения Microsoft Store и сторонних производителей. Рекомендовано специализировать отдельные микрокомпьютеры для работы с 32-битными приложениями; бухгалтерскими и криптографическими программами; а также проведения презентаций с их видеозаписью. Предложены варианты оптимального конфигурирования меню «Пуск» рабочего стола Windows 10. Сделан вывод, что специализация аппаратно-программной конфигурации современных микрокомпьютеров позволяет увеличить эффективность работы с применением одиночных устройств и их сопряжённых систем в соответствии со стандартами BYOD (Bring Your Own Device).

***Ключевые слова**: микрокомпьюрер, микроПК, Windows 10, конфигурация, наука, образование, бизнес, трейдинг, BYOD*

ВВЕДЕНИЕ

Микрокомпьютеры – один из классов устройств для обработки информации, которые активно эволюционируют в течение последних 5 лет. В отличие от прочих мобильных устройств (ноутбуков, нетбуков, ультрабуков и смартфонов) микрокомпьютеры, подобно стационарным десктопам, легко специализируются посредством варьирования периферии и программного обеспечения. В настоящее время встречаются микрокомпьютеры с операционными системами Windows 10, Android и Linux. Как правило, в настройках BIOS предусмотрен выбор приоритетно используемой операционной системы. Анализ прецедентов использования показал, что наиболее широкий спектр применения обеспечивает операционная система Windows 10, и для проведения экспериментов по специализации микрокомпьютеров для разных сфер применения были выбраны устройства на вышеуказанной платформе. Операционная система Windows 10 [1] с момента своего выхода 29 июля 2015 г. находится в фазе активной адаптации к информационной среде полиморфных пользовательских устройств (серверы, десктопы, ноутбуки, ультрабуки, нетбуки, мини-компьютеры, микрокомпьютеры (микроПК), мобильные телефоны и прочие носимые персональные устройства) в соответствии с парадигмой BYOD [2]. Об этом свидетельствуют адаптация всех элементов интерфейса Windows 10 к использованию на мобильных устройствах, внедрение системы контроля и экономии энергопотребления аппаратными и программными компонентами, всестороннее использование телеметрии для контроля работы компонентов с целью исправления дефектов системы, клонирование своих программных продуктов в пространства других операционных сред (устройства Apple и Android).

В настоящее время разработка миниатюрных компьютеров для разных сфер применения находится в фазе активного эксперимента, когда обновленные модели устройств появляются небольшими партиями и включают в себя официальную лицензию Windows 10. В течение первого полугодия 2019 года стали недоступны две модели микроПК (BBen [3], MeLE [4]) и появилась одна новая

(ACEPC [5]). Уже доступны первые смартфоны с полноценной Windows 10, ориентированные на использование в профессиональной деятельности [6].

Стоимость микроПК сопоставима с ценой приобретения Windows 10 Home [1] для установки в любой компьютер без операционной системы. Современная ценовая политика распространения Windows 10 с инновационными устройствами ($5) позволяет инициативным исследователям активно экспериментировать с применением компьютерных новинок в разнообразных сферах, приближая момент официального выхода смартфонов с полноценной Windows 10, предназначенных для широкого круга пользователей.

Для проекта МАСТЕР-МУЛЬТИМЕДИА оказалось актуальным провести эксперимент по замещению традиционно используемой офисной и персональной вычислительной техники (десктопы, ноутбуки и ультрабуки) на микрокомпьютеры, представляющие собой системный блок персонального компьютера в корпусе размером с пару спичечных коробков. Этот интерес был обусловлен исторически сложившейся практикой. Последние 10 лет все ноутбуки использовались исключительно в роли вычислительных системных блоков, а интерфейсная периферия (мониторы, клавиатуры, мыши, графические планшеты, веб-камеры, микрофоны, акустические системы) находилась в общем пользовании и присоединялась к каждому из устройств по мере необходимости. Таким образом, все комплектующие ноутбуков, кроме компонентов системного блока, оказывались практически невостребованными.

Выбор микроПК для экспериментов по модернизации информационной среды проекта МАСТЕР-МУЛЬТИМЕДИА был обусловлен их мобильностью по стандарту BYOD [2], в отличие от мини-компьютеров размером с книгу формата A5 [7]. Одни модели микроПК позиционируются как развлекательные интернет-медиаплееры для домашних кинотеатров [8], другие рекомендованы для офисного, публичного (интерактивные справочные системы) и специального применения (мониторинг охранных систем) [4], а некоторые модели предполагается использовать в научно-образовательной деятельности при чтении лекций и докладов [3, 5]. Целью этого исследования было определить минимальную аппаратно-программную конфигурацию современных микроПК для применения в научно-образовательной и офисной сферах деятельности.

1. АППАРАТНАЯ КОНФИГУРАЦИЯ МИКРОКОМПЬЮТЕРОВ, ИСПОЛЬЗОВАННЫХ В ЭКСПЕРИМЕНТАХ ПО СПЕЦИАЛИЗАЦИИ ПРИМЕНЕНИЯ

Эксперименты по конфигурированию информационной среды проводились с 6 микроПК 4 моделей, у которых отличалась базовая аппаратная комплектация (Таблица 1): центральный процессор (CPU Intel), размер оперативной (RAM LPDDR3) и постоянной памяти (eMMC); тип охлаждения (активное/пассивное); наличие батареи CMOS, внешней антенны (Ext WiFi), разъемов для подключения интернет-кабеля (LAN Rj45) и аудиопериферии (Audio jack 3,5 мм). Следует обратить внимание, что в устройствах без батареи CMOS необходимо непосредственно после включения оперативно корректировать системную дату и время во избежание конфликтов синхронизации данных с облачными хранилищами, которые подключаются на финальных стадиях загрузки операционной системы компьютера.

Таблица 1. Аппаратные характеристики микрокомпьютеров

Имя	Производитель	CPU Intel	RAM, Гб	eMMC, Гб	Батарея CMOS	Ext WiFi	LAN Rj45	Audio jack
FOKWSR08	Rombica [8]	Z3735F	2	32	Нет	Есть	Нет	Нет
FOKWSB09	BBen [3]	Z8350	4	64	Есть	Нет	Нет	Нет
FOKWSM10	MeLE [4]	N3450	4	32	Есть	Есть	Есть	Есть
FOKWSA11	ACEPC [5]	Z8350	4	64	Есть	Нет	Нет	Нет
FOKWSA12	ACEPC [5]	Z8350	4	64	Есть	Нет	Нет	Нет
FOKWSA13	ACEPC [5]	Z8350	4	64	Есть	Нет	Нет	Нет

2. ОПЕРАТИВНАЯ, ПОСТОЯННАЯ И РАСШИРЯЕМАЯ ПАМЯТЬ МИКРОПК

В настоящее время минимальный размер оперативной памяти микроПК – 2 Гб. Опыты показали, что для комфортной работы недостаточно 2 Гб оперативной памяти: модель Rombica [8] отличалась задержками вызова меню «Пуск», замедлением работы при одновременном открытии более полудюжины окон браузера со страницами интернет-сайтов и выводом рекомендаций увеличить размер оперативной памяти до 3 Гб при работе с базами данных 1С-Предприятие. МикроПК с 2Гб оперативной памяти справляется с работой однозадачного воспроизведения контента (вещание интернет-ТВ, проигрывание ме-

диафайлов, демонстрация слайдов PowerPoint, мониторинг процессов). Все остальные микроПК с 4 Гб памяти функционировали с достаточной скоростью при работе в многозадачной научно-образовательной и деловой информационных средах (многооконное открытие нескольких браузеров, ассоциативных карт, электронных таблиц и текстовых процессоров, бухгалтерских баз данных с криптопровайдерами). В настоящий момент появились модели микроПК с 6 Гб оперативной памяти [9], и их можно порекомендовать для работы с ресурсоёмкими программами для многодорожечного редактирования видео. Стоимость оперативной памяти в начале 2019 года составляла порядка $25/Гб.

В 2019 году модели микроПК с накопителями данных объемом 32 Гб начали уступать место устройствам с 64 Гб постоянной памяти вследствие недостатка места для штатных процедур обновления операционной системы Windows 10, требующих не менее 15 Гб свободного места, тогда как базовый набор модулей операционной системы и минимального программного обеспечения может составлять порядка 25 Гб. Стоимость постоянной памяти для микроПК (eMMC) в начале 2019 года составляла порядка $2,5/Гб.

У всех протестированных моделей микроПК имелось по одному разъёму для карт памяти microSD (TF с максимальной ёмкостью 128 Гб), в который вставлялись отдельно приобретаемые microSD карты. На момент написания настоящей статьи лучшими microSD-картами для использования в микрокомпьютерах оказались Kingston Canvas React microSDXC 128Gb UHS-I U3 V30 A1 + ADP (100/80 Mb/s), позволяющие в реальном времени осуществлять на них видеозапись в формате 4K.

Опыты показали, что связь микроПК с картой памяти может утрачиваться вследствие сбоев в работе операционной системы и воздействия сильных электромагнитных полей. Надежность работы microSD карты зависит от конкретного образца микроПК (зарегистрированы частые сбои у FOKWSM10, редкие сбои у FOKWSR08, FOKWSB09 и FOKWSA13, отсутствие сбоев FOKWSA11 и FOKWSA12). Для восстановления работы microSD-карт достаточно выключения, обесточивания и включения микроПК. Помимо экономии места в основном накопителе, организация хранения программ и пользовательских файлов на извлекаемой microSD-карте позволяет с лёгкостью воспользоваться ими в аварийных ситуациях повреждения неизвлекаемого основного накопителя.

3. ГРАФИЧЕСКАЯ ПОДСИСТЕМА МИКРОКОМПЬЮТЕРОВ

Все протестированные модели микроПК имели графические сопроцессоры с декларированной поддержкой разрешения 4K ULTRA HD 60 fps: Intel Gen8 HD Graphics (BBen [3]) GPU Intel HD Graphics 400 (ACEPC [5]) GPU Intel HD Graphics 500 (Rombica [8] и MeLE [4]). На практике оказалось, что все модели микроПК, кроме BBen [3], без проблем воспроизводили интернет-видео и эфирное вещание с разрешением HD.

Вывод генерируемых микроПК аудиовизуальных сигналов осуществляется через интерфейс HDMI, устойчивость работы которого может зависеть от высокого качества кабелей и используемых переходников. При наличии соответствующего HDMI-разъема в мониторе, микроПК может быть подключен к нему непосредственно, при помощи комплектного удлинителя (BBen [3] и ACEPC [5]) или адаптера для HDMI-кабеля (Rombica [8], MeLE [4]). Опыт показал, что устройство Rombica [8] успешно подключалось к любым мониторам посредством разнообразных кабелей и переходников (HDMI, DP, DVI, VGA), для подключения остальных устройств требовалась высококачественная коммутация, а микроПК BBen [3] и ACEPC [5] оказались несовместимы с VGA-мониторами. При организации подключения мониторов к микрокомпьютерам посредством HDMI-DP-конвертера следует учитывать, что передача сигналов осуществляется только в одном направлении, а для обеспечения лучшей совместимости нужно выбирать устройство с активным хранением информации EDIO на конвертере, возможностью передачи через HDMI несжатого двухканального звука LDPCM и поддержкой разрешения графики до 3840x2160 P/30 Гц.

4. ИНТЕРФЕЙСЫ ОБМЕНА ДАННЫМИ МИКРОКОМПЬЮТЕРОВ

Для сетевого обмена данными микроПК снабжены двухдиапазонными модулями WiFi (802.11 a/b/g/n/ac, 2.4 ГГц & 5 ГГц) с внутренней (BBen [3], ACEPC [5]) или внешней антеннами (MeLE [4], Rombica [8], Azulle [9]), а модели MeLE [4] и Azulle [9] дополнены разъемом Rj45 Гигабитного LAN. Опыты показали, что локализация антенны не влияет на скорость WiFi-передачи данных (порядка 20 Мбит/с на вход и выход), но крепление внешней антенны может обламываться без нарушения полезного функционала после 2-месячной эксплуатации, требу-

ющей изменения положения антенны для доступа к находящемуся под ней USB-разъему (Rombica [8]).

У всех микроПК наличествовал беспроводной модуль Bluetooth 4.0, который отключался системой для экономии электропотребления, вследствие чего эксплуатация протестированных «мышей» (SONY VGP-BMS15, Logitech MX Master, Logitech MX Master 2S, Logitech MX Anywhere 2S) и клавиатур (SONY VGP-BKB1, Logitech K375s, Logitech K480, Logitech K810) с интерфейсом Bluetooth оказалась невозможной вследствие задержек реакции и потери связи. Изменения параметров настроек Bluetooth в BIOS не помогли решить эту проблему. Подключение Bluetooth-донглов посредством USB-портов оказалось также неэффективным вследствие системных блокировок. На настоящий момент проблема с подключением Bluetooth-периферии к микрокомпьютерам остается нерешенной.

Надёжное подключение периферии у всех протестированных микроПК обеспечивали 2 разъема USB (устройство Rombica [8] было снабжено разъёмами USB 2.0 и microUSB 2.0, а остальные имели по одному USB 2.0 и USB 3.0 разъёму). Очевидно, что для переноса больших массивов данных эффективно использовать модели, способные использовать протокол USB 3.0. Активировать или деактивировать протоколы USB 2.0 и USB 3.0 для каждого порта возможно в настройках BIOS.

5. ЭЛЕКТРОПИТАНИЕ И ОХЛАЖДЕНИЕ МИКРОКОМПЬЮТЕРОВ

Электропитание всех микрокомпьютеров осуществлялось через специализированный разъём microUSB (5В/3А), весьма требовательный к высокому качеству коммутируемого кабеля. Все блоки питания из комплекта поставок микроПК в ходе эксплуатации пришлось заменить на более качественные сетевые устройства (Rock T14 Travel Charger 3xUSB3A), каждое из которых обеспечивало одновременное питание одного микроПК и подключенного к нему периферийного USB-разветвителя (см. раздел 2). При необходимости избавления от акустических помех в процессе работы со звуком (запись с микрофона, воспроизведение) рекомендуется организовывать электропитание звуковой аппаратуры (приёмники, передатчики, звуковые карты, микшеры, усилители и акустические системы) от блоков питания, электрически не связанных с микрокомпьютерами.

Модель BBen [3] отличалась наличием активного охлаждения (Internal Intel Mute fan), проблемы с запуском которой препятствовали использованию этого устройства в режиме частого включения-выключения. Следует учитывать, что звук вентиляторной системы охлаждения компьютера мешает выполнению работ, связанных со звукозаписью. По этой причине для работы в студиях звукозаписи целесообразно выбирать безвентиляторные микрокомпьютеры. Было установлено, что безвентиляторное устройство Rombica [8] с наименьшим размером закрытого корпуса перегревается в режиме многозадачного использования с последующей остановкой работы операционной системы. Наиболее крупное устройство MeLE [4] с закрытым корпусом в процессе многозадачной работы ощутимо нагревается, но функционирует безотказно, а модели BBen [3] и ACEPC [5] с перфорированными корпусами в любых режимах нагреваются слабо и функционируют устойчиво.

Таким образом, для научно-образовательных, деловых и развлекательных целей мы рекомендуем использовать безвентиляторные микрокомпьютеры с перфорированным корпусом и внутренней WiFi-антенной, не менее 4Гб оперативной и 64 Гб постоянной памяти, разъёмом карты памяти Micro SD (TF, не менее 128 Гб), графическим ускорителем GPU Intel HD Graphics, интерфейсами USB3.0 и HDMI. Поскольку каждый модуль интерфейса (Rj45, Audio и пр.) является источником дополнительного нагрева, целесообразно минимизировать аппаратный функционал микроПК и специализировать свойства информационной среды подключаемой периферией. Для предотвращения возникновения акустических помех рекомендуется организовать электроснабжение микрокомпьютера и связанных с ним акустических систем через разные блоки электропитания.

В полевых условиях микрокомпьютеры и подключаемые периферийные устройства (USB-разветвители и хранилища, аудиоустройства, экраны и т. д.) могут получать электропитание от аккумуляторов (банков энергии) с USB-выходами (5В/2-3А) и ёмкостью не менее 10000 мА/ч. В стационарных условиях электропитание микроПК и периферийных устройств от USB-банков энергии, постоянно подключенных к электрической сети, может рассматриваться как альтернатива бесперебойным источникам питания. В настоящее время доступны компактные банки энергии, обеспечивающие одновременное электропитание 3–4 устройств. При организации длительной работы в стационарных условиях с использовани-

ем банков энергии в качестве бесперебойных источников питания следует отдавать предпочтение моделям, допускающим собственную зарядку во время электроснабжения подключенных устройств.

6. ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА МИКРОКОМПЬЮТЕРОВ

В соответствии со стандартами BYOD [2], микроПК можно подключать посредством HDMI-разъема не только к компьютерным мониторам, но и к телевизорам, дисплеям с сенсорными интерфейсами и т. п. Для мониторинга звука удобно использовать встроенные динамики графических мониторов, получающие сигнал через HDMI или Display Port (DP) посредством специального конвертера [12]. Вывод звука на качественные акустические системы целесообразно осуществлять посредством подключаемой к микроПК внешней USB-звуковой карты [13].

При выборе сенсорных экранов для работы с микроПК в полевых условиях целесообразно отдавать предпочтение моделям (Kenowa и т. п.), получающим питание от микроПК через USB-порт одновременно с обменом данными многоточечного позиционирования (поддержка 10-пальцевого способа ввода информации) и 1024 градациями силы нажатия. Опыт показал, что для устойчивой работы такого монитора с микрокомпьютером достаточно одного источника электроснабжения.

В отсутствие сенсорного экрана позиционирование курсора и ввод текста осуществляются посредством компьютерных «мышей» и клавиатур. В результате экспериментов с подключением беспроводных «мышей» и клавиатур к микроПК было установлено, что в условиях полиморфных информационных сред с различным электромагнитным фоном наиболее устойчиво работают устройства Logitech с приёмником Unifying, подключаемым через разъем USB. Для информационной среды из 2–3 микрокомпьютеров целесообразно использовать модели «мышей» и клавиатур с поддержкой единой среды (Logitech Flow), в которой снабженные Unifying-приёмниками компьютеры управляются одним комплектом «мыши» и клавиатуры без применения аппаратных кнопок переключения фокуса (см. раздел 9).

Для управления микроПК во время лекционно-презентационной деятельности рекомендуется использовать специальную миниатюрную клавиатуру с

собственным USB-приёмником, сенсорной панелью, а также эргономичными кнопками-гомологами левой и правой кнопок «мыши» [10].

В условиях интенсивной многозадачной деятельности работа приемников может нарушаться вследствие перегрева, после чего USB-приёмники целесообразно присоединять через USB-удлинители или разветвители (hub). Опыты показали, что наиболее эффективно использовать разветвители USB 3.0 с дополнительными разъемами для подключения гигабитного LAN (Rj45) и внешнего электропитания [11]. При необходимости, посредством такого USB-разветвителя к микроПК могут присоединяться накопители данных, графические планшеты, веб-камеры, микрофоны, аппаратные USB-ключи доступа, носители квалифицированных электронных подписей и др. Для скоростного обмена данных (до 1 Гб/с) между устройствами информационной среды могут эффективно использоваться внешние SSD-накопители с интерфейсом NVMe, подключаемые посредством USB 3.0.

7. ОПЕРАЦИОННАЯ СИСТЕМА WINDOWS 10 И ПРОБЛЕМЫ ЕЁ ОБНОВЛЕНИЯ В МИКРОКОМПЬЮТЕРАХ

При первом включении компьютера с лицензионной операционной системой Windows 10 происходят активация лицензии с её привязкой к email-адресу пользовательского аккаунта Microsoft и телефону; выбор параметров телеметрии системы; формирование в папке c:\Users\ персональной директории пользователя, автоматически называемой в соответствии с пятью первыми буквами регистрационного email-адреса пользователя, а также подключение облачного хранилища OneDrive (начиная со сборки 1809) к одноименной папке внутри пользовательской директории. Для экономии места в накопителях микроПК с 32 Гб оперативной памяти расширенную телеметрию рекомендуется отключать на стадии базовой установки системы. Следует подчеркнуть, что политика использования Windows 10 делает обновление системы неизбежным, и его можно отключить на срок не более месяца, после чего оно осуществляется принудительно.

Опыт показал, что после установки Windows 10 со сборками от 1507 (Rombica [8]) до 1803 (BBen [3]) процедура установки всех обновлений занимает длительное время (до 2 суток), требует контроля оператора и свободного места

порядка 15 Гб постоянной памяти. Для микроПК с 32 Гб постоянной памяти крупные обновления вызывают необходимость подключения внешнего USB-диска и оставляют по окончании 2–6 Гб свободного места, что критически мало для устойчивой работы системы. Переход на версию Windows 10 сборки 1903 в таких условиях оказался невозможен (устройства Rombica [8] и MeLE [4]) даже при подключении внешнего жесткого USB-диска. Освобождение 8 Гб на диске C: позволяет перейти к сборке 1903, после чего проблемы с обновлениями Windows 10 прекращаются: начиная со сборки 1909 (сентябрь 2019 г.), крупные обновления осуществляются успешно даже на устройствах с 6 Гб свободного места на системном диске.

Указанием на необходимость крупных обновлений является Лицензионное соглашение с пользователем, датированное 2015–2017 годами. Было обнаружено, что компьютеры с устаревшими сборками Windows 10 целесообразно не обновлять, а сразу после первого входа в систему подвергнуть её процедуре восстановления из раздела (Параметры/Обновление и безопасность/ Восстановление/Вернуть компьютер в исходное состояние). Опыт показал, что в результате вызываемого подобным образом процесса в устройство за минимальное время устанавливается последняя сборка Windows 10, минуя промежуточные обновления, что экономит порядка 10 Гб места в постоянной памяти. Процедура восстановления системы невозможна на системах, в которых после инсталляции программной среды и загрузки данных пользователя осталось менее 8 Гб свободного места (MeLE [4]).

Было замечено, что фоновое изображение экрана блокировки заменяется на новое при обновлениях системы, а после сбоев и восстановления компонентов возвращается к старому варианту. Эту особенность можно использовать как индикатор предшествующего состояния системы при её загрузке.

Следует избегать условий, нарушающих работу микроПК (сбой электропитания, зависание в режиме многозадачности и пр.) во время длительной процедуры очистки предыдущих установок и ненужных файлов обновления Windows, поскольку в результате её нештатного завершения очищаемое место изымается из дальнейшего оборота.

8. ПРОГРАММНЫЕ ПРИЛОЖЕНИЯ MICROSOFT STORE

В базовых вариантах поставки микрокомпьютеров операционная система Windows 10 поставляется в состоянии запроса на автоматическую установку набора из 30–40 программных приложений из интернет-магазина Microsoft Store, которая начинается после первого входа в систему. После окончания установки (порядка 1 Гб места в постоянной памяти) эти приложения обновляются в автоматическом режиме, что вызывает увеличение расхода занимаемого ими места. Многие из этих приложений являются условно-бесплатными и используются на основе регулярных подписок. Набор условно-бесплатных приложений Microsoft Store зависит от производителя микрокомпьютера.

Из базового комплекта бесплатных приложений мы рекомендуем обратить внимание на Microsoft Edge (браузер); Microsoft OneDrive (облачное хранилище, размер которого зависит от пользовательской подписки); OneNote и Sticky Notes (сервисы для создания синхронизируемых на всех пользовательских устройствах мультимедийных и текстовых заметок, соответственно); а также унаследованный из Windows 10 Mobile набор программ (Будильники и часы, Запись голоса, Калькулятор, Камера, Карты, Кино и ТВ, Люди, Музыка Groove, Погода, Почта и Календарь, Скайп, и Фотографии). Базы данных у некоторых из перечисленных приложений (OneDrive, Люди, Почта и Календарь, Фотографии) увеличиваются по мере использования, что необходимо принимать во внимание при конфигурировании информационной среды микроПК с дефицитным объемом постоянной памяти. В экстренных случаях базу данных приложения Фотографии и резервную базу данных OneDrive можно удалить с высвобождением нескольких Гб пространства (фотографии в пользовательских папках сохранятся, и базы данных впоследствии автоматически восстановятся). Набор установленных в систему приложений Microsoft Store целесообразно варьировать в соответствии со спецификой работы пользователей и использования компьютера в распределённой информационной среде (см. раздел 9). Базовый набор для формирования единой инфосреды микроПК Инфоконтинуума проекта МАСТЕР-МУЛЬТИМЕДИА включает четыре десятка приложений Microsoft Store, среди которых – менеджер задач «Microsoft ToDo: List, Task And Reminders», графические приложения «Фрагмент и набросок», «Microsoft Whiteboard» и «Microsoft Office

Lens». Новейшие видеокодеки HEVC, необходимые для воспроизведения и монтажа аудиовизуальных записей, сделанных на современных смартфонах, приобретаются дополнительно ($0.99). После установки кодека посредством бесплатного приложения Video Convertor (V3TApps) можно преобразовать новые аудиовизуальные записи к формату, совместимому с большинством программных видеоредакторов.

9. ПРОГРАММНЫЕ ПРИЛОЖЕНИЯ СТОРОННИХ ПРОИЗВОДИТЕЛЕЙ

Функционал информационной среды микрокомпьютеров может быть дополнен посредством установки программных приложений сторонних производителей, которые могут конфликтовать друг с другом, провоцировать сбои работы операционной системы или помогать их преодолевать. К примеру, запрещаемая операционной системой Windows 10 установка работоспособного векторного графического редактора CorelDraw 13 в форсированном административном режиме возможна, но результатом является необратимый отказ функционирования всех приложений Microsoft Store и меню «Пуск». В информационной среде с такими дефектами доступ в интернет возможен при помощи классического приложения Internet Explorer или посредством браузеров сторонних производителей программного обеспечения, среди которых набольшим соотношением производительность/ресурсоёмкость обладает Google Chrome.

В качестве многофункционального файлового менеджера в инфопространстве проекта МАСТЕР-МУЛЬТИМЕДИА с 1993 года используется Total Commander [14], ранее известный как Windows Commander. Пятилетний опыт использования позволяет рекомендовать в качестве надёжного и утилитарного менеджера паролей для всех устройств информационной среды программу Safe-in-Cloud [15]. Для обеспечения информационной безопасности системы рекомендуется заменить штатный «Защитник Windows» антивирусным комплексом Norton Internet Security [16], который отличается низкой ресурсоёмкостью и эффективным быстродействием, а также не вызывает сбоев в работе баз данных программ платформы 1С-Предприятие (нарушение консолидации новых данных со старыми). Следует отметить, что штатный механизм архивирования баз данных 1С работает ненадежно, и созданные им архивы следует проверять на читаемость, а восстановленные из них базы данных тестировать на целостность данных в режиме

конфигуратора. Отмечено, что криптопровайдеры и модули работы с квалифицированными средствами электронной подписи, интегрирующиеся не только в специализированные приложения, но и в браузеры, резервируют ресурсы системы и замедляют работу сторонних приложений. Модули телеметрии, собирающие сведения о сбоях программно-аппаратного функционала (Logitech Options и др.), также могут вызывать замедление работы микрокомпьютеров с ограниченными ресурсами.

10. КОНФИГУРИРОВАНИЕ МЕНЮ «ПУСК» WINDOWS 10

В Windows 10 было обновлено меню «Пуск», появившееся еще в Windows 95. Помимо кнопок обращения к Учётной записи пользователя, Проводнику, Параметрам системы и Алфавитного меню приложений, в меню Пуск были включены так называемые «плитки» – разноформатные потенциально анимированные интерактивные элементы для вызова избранных приложений интерфейса «Metro» [17]. Исходно плитки группировались в два блока: «События и общение» и «Развлечения и отдых».

Интерфейс «Metro» оказался исключительно эффективен в разработанной для смартфонов ОС Windows Phone Mobile, но мало востребован в информационной среде стационарных компьютеров и ноутбуков. Предлагаемые в сборках Windows 10 2018–2019 годов выпуска варианты конфигурирования меню «Пуск» включают блоки «Создавайте», «Играйте» и «Исследуйте». В дистрибутивных сборках Windows 10, начиная с 1903 (май 2019 года), меню «Пуск» было редуцировано из двухколоночного варианта в одноколоночный, а для обновлённых конфигураций предусмотрено автоматическое перемещение блоков второй колонки в нижнюю область первой колонки при изменении ориентации экрана с альбомной на книжную. Учитывая наметившуюся конвергенцию информационных сред мобильных устройств и прочих компьютеров, целесообразно конфигурировать меню «Пуск» в унифицированном стиле для всех компьютеров общего инфопространства и наполнять его в соответствии со спецификой каждого устройства (см. выше). К примеру, в Инфоконтинууме проекта МАСТЕР-МУЛЬТИМЕДИА все устройства имеют общие блоки плиток меню «Пуск»: «Организовать», «Воспроизвести», «Исследовать». В зависимости от целевого назначения компьютера меню «Пуск» дополняется специализированными блоками:

«Создать» «Финансировать», «Использовать». Свойства блоков и плиток, а также их взаимное расположение конфигурируются пользователем в соответствии со своими предпочтениями и спецификой частых действий, включая изменение ориентации подключаемого экрана. Практический опыт показал целесообразность включения в настройках меню «Пуск» режима отображения секции «Часто используемые приложения».

11. ОПТИМИЗАЦИЯ ИСПОЛЬЗОВАНИЯ ДОЛГОВРЕМЕННОЙ ПАМЯТИ МИКРОКОМПЬЮТЕРОВ

Контроль использования постоянной памяти в Windows 10 осуществляется в специальном разделе (Параметры/Система/Память устройства), где сосредоточены сведения об использовании локальных хранилищ и подключённых к устройству носителей (SD-карты, дисковые пространства и т. д.), а также находятся регулировки функционала контроля памяти и ее автоматического освобождения. Для экономии дефицитного места на основном носителе целесообразно изменить место сохранения нового содержимого (приложения, документы, музыка, фотографии и видео, фильмы и ТВ-передачи, автономные карты). Для этого до первоначальной загрузки системы целесообразно подключить к устройству надёжную, ёмкую и быструю microSD-карту (см. выше), предварительно отформатированную в формате NTFS, и незамедлительно после первого входа в систему указать эту карту в качестве нового места хранения. В результате все приложения Microsoft Store, загружающиеся по запросу инсталляционного сценария, не будут занимать места в основном хранилище, а пользовательская директория перемещается из папки c:\Users\ в корень microSD-карты. Теоретически в системе присутствует привычный для смартфонов механизм транслокации приложений между разными носителями, но для его использования требуется остановка переносимого приложения, которая в ряде случаев оказывается невозможной.

В Windows 10, начиная со сборки 1809, облачное хранилище OneDrive ассоциируется с аккаунтом пользователя в состоянии доступности всех файлов по запросу, что позволяет подключить к локальному накопителю превосходящие его ёмкость объёмы данных. К примеру, в многопользовательской конфигурации компьютера (см. раздел 8) к его накопителю с объемом 32 Гб можно под-

ключить до 6 облачных хранилищ общим объемом 6 Тб. До запроса доступа к файлам хранилищ OneDrive на компьютере хранятся только базы данных, контролирующие состояние синхронизации. Размер локальной базы данных – порядка 1 Гб/1 Тб данных в облачном хранилище. После скачивания файлов из облака и их модификации обновленные варианты автоматически заменяют предшественников в облаке и удаляются с локального носителя в соответствии с расписанием настроек Контроля памяти: ежедневно, еженедельно, ежемесячно и т. п. Однако при необходимости одновременной работы с крупными файлами возникает вероятность исчерпания места в системном накопителе, где по умолчанию располагаются папки OneDrive (см. раздел 3). По этой причине при базовом конфигурировании системы папку OneDrive из пользовательской директории в папке c:\Users\ целесообразно переместить на microSD-карту (допустим, только формат NTFS, тогда как исходным форматом microSD-карт является ExtFAT). Для обеспечения возможности изменения месторасположения следует согласиться с предлагаемой по умолчанию настройкой OneDrive, вслед за чем сразу удалить образованную связь с облачным хранилищем пользователя и создать новую связь с директорией на microSD-карте. При однопользовательской конфигурации компьютера папку OneDrive целесообразно размещать в корневой директории microSD-карты, а в случае многопользовательской конфигурации – внутри пользовательских директорий microSD-карты. Внутренние папки директории OneDrive создаются в соответствии с рекомендациями по созданию распределенной системы хранения данных и медиаконтента [18].

Следует иметь в виду, что в состоянии паузы синхронизации OneDrive (отсутствие связи с Сетью или подключение с лимитируемым трафиком) в локальном хранилище файлов блокируются создание и модификация файловых комментариев descript.ion. По этой причине файлы с комментариями (архивы с протоколами проверки, дистрибутивы с аннотациями и т. п.) в несинхронизируемом состоянии целесообразно аккумулировать в буферной директории OneDrive.new [18], а при возобновлении синхронизации перемещать в место постоянного хранения внутри папки OneDrive.

12. МНОГОПОЛЬЗОВАТЕЛЬСКАЯ КОНФИГУРАЦИЯ МИКРОКОМПЬЮТЕРОВ

Особенности подключения к облачным хранилищам и политика совместного использования приложений Microsoft Store позволяют создавать многопользовательские конфигурации компьютеров без существенного расхода пространства памяти накопителя на каждого нового пользователя. Для сторонних приложений может потребоваться разрешение на их использование для каждого пользователя, как это происходит в процессе восстановления функций Microsoft Office с принятием политики использования для пользователей, в конфигурациях которых не осуществлялась его первичная установка.

Следует учитывать, что конфигурирование подключения к хранилищам OneDrive для каждого пользователя происходит независимо, и пользователи не имеют доступа к просмотру содержимого директорий друг друга. Директории всех пользователей могут быть сосредоточены на microSD-карте, что уменьшает вероятность необратимой потери данных в аварийных ситуациях (см. выше). Конфигурирование меню «Пуск» каждого пользователя и модификации содержимого Рабочих столов осуществляются индивидуально. Опыт показал, что режим «Доступа по запросу» к данным пользователей позволяет организовать и устойчиво использовать многопользовательскую конфигурацию даже на микрокомпьютерах с минимальным размером постоянной памяти системного диска.

13. ОРГАНИЗАЦИЯ РАСПРЕДЕЛЕННОЙ ИНФОРМАЦИОННОЙ СРЕДЫ ИЗ НЕСКОЛЬКИХ МИКРОКОМПЬЮТЕРОВ

В ходе выполнения научно-образовательных и офисных задач часто возникают ситуации, когда требуется одновременная работа на нескольких компьютерах с разной конфигурацией, чтобы избежать программных конфликтов и дефицита ресурсов системы. К примеру, эффективность работы с базами данных платформы 1С-Предприятие выше в 64-битной среде, но многие формы отчётов с двумерными штрих-кодами поддаются распечатке только из 32-битных версий. Оказалось, что присутствие криптопровайдеров в информационной среде негативно влияет на возможность многозадачной работы с базами данных, ассоциативными картами и графическими приложениями в режиме демонстрации рабочего стола и аудиовизуальной записи его содержимого во время проведения видеоконференций Skype. По этой причине целесообразно использовать специ-

ализированные компьютеры для деятельности в сферах дистанционного обучения и бухгалтерско-управленческого учета, при котором необходимо обеспечение зависимой от криптопровайдеров работы со средствами квалифицированной электронной подписи, аппаратными USB-ключами и т. п. Особая конфигурация компьютеров требуется для обработки массивных аудиовизуальных данных и одновременного мониторинга кинетики множества процессов (инструменты технического анализа состояния финансовых рынков и т. п.). При проведении видеоконференций (консультации, научная и образовательная деятельность) удобно один компьютер использовать для демонстрации рабочего стола, а другим пользоваться в справочно-организационных целях.

Несколько компьютеров с разной аппаратно-программной конфигурацией (до 3 шт.) могут быть объединены в единую среду с использованием технологии Logitech Flow [19], когда курсор «мыши» перемещается в сопряженное инфопространство соседнего компьютера при пересечении границы экрана с соответствующим автоматическим переключением фокуса клавиатуры. В общем инфопространстве Logitech Flow доступны непосредственное копирование и перемещение файлов между рабочими столами всех подключённых компьютеров, что значительно увеличивает эффективность оперативной работы.

Информационная среда каждого компьютера может быть индивидуализирована посредством создания множества рабочих столов, контролировать изменение содержимого которых помогает временная шкала, сохраняющая историю работы со всеми файлами. Для облегчения экранной идентификации совместно используемых компьютеров рекомендуется присваивать microSD-картам имена соответствующих компьютеров, а в левом верхнем углу каждого рабочего стола размещать ярлык с названием компьютера и ссылкой на корневую директорию microSD-карты.

На уровне философских обобщений специализированные компьютеры единой информационной среды можно уподобить органам, совместное действие которых переводит систему на качественно новый уровень функционирования единого организма в соответствии с парадигмой инфоцентризма [20, 21].

ЗАКЛЮЧЕНИЕ

Аппаратно-программная конфигурация современных микрокомпьютеров позволяет использовать их в индивидуальном режиме в соответствии со стандартами BYOD [2] или специализировать дополнением программных и периферийных аппаратных устройств для использования в качестве сервисных компонентов распределенной информационной среды [21], обеспечивающей решение задач научно-образовательной, офисной, управленческой и трейдерской деятельности в стационарных и мобильных условиях.

СПИСОК ЛИТЕРАТУРЫ

1. Windows 10: лучшая Windows в истории поможет вам достичь большего. // Microsoft. URL: https://www.microsoft.com/ru-ru/store/b/windows?icid =TopStripe-windows-cat-page.

2. *Ballagas R., Rohs M., Sheridan J., Borchers J.* BYOD: Bring Your Own Device // UbiComp 2004 Workshop on Ubiquitous Display Environments. Nottingham, UK, September 2004. URL: http://www.vs.inf.ethz.ch/publ/papers/rohs-byod2004.pdf

3. BBen MN9 Mini PC Stick Windows 10 Ubuntu Intel X5 Z8350 Quad Core 2G 4GB RAM Mute Fan WiFi Smart TV Stick PC Mini Computer Micro. URL: https://www.aliexpress.com/item/32767461014.html

4. Fanless Mini Computer Mini PC Stick 4GB 32GB MeLE PCG02 Apo Quad Core Intel Celeron N3450 Windows 10 HDMI 4K LAN WiFi URL: https://www.aliexpress.com/item/32857950812.html

5. ACEPC T6 Pocket Computer stick pc mini fanless 4GB 64GB 2GB 32GB Windows 10 Home licenced Intel Atom x5-Z8350 BT4.0/WiFi/USB 3.0. URL: https://www.aliexpress.com/item/32974950520.html

6. 2018 Pocket PC Mini PC Mobile Computer Windows 10 6 inch Intel Cherry Trail Z8350 Bluetooth Single SIM GPS Shockproof Tablet PC. URL: https://www.aliexpress.com/item/32854798966.html

7. Mini Desktop Core i7 7500u Fanless PC 7th CPU Kabylake Architecture HD Graphics 620 Max 3.5GHz Windows 10 Gaming PC 300M Wifi. URL: https://www.aliexpress.com/item/32815422763.html

8. Rombica WinStick v01. URL: https://rombica.ru/product/winstick-v01/

9. Access 3 / Azulle. URL: https://azulletech.com/product/access3/

10. Original iPazzPort 19s Russian Keyboard 2.4G Mini Wireless Keyboard Air Mouse with Touchpad for Google Android TV Box, Mini PC. URL: https://www.aliexpress.com/item/32952092637.html

11. Ugreen USB Ethernet USB 3.0 2.0 to RJ45 HUB for Xiaomi Mi Box 3/S Set-top Box Ethernet Adapter Network Card USB Lan. URL: https://www.aliexpress.com/item/32401431234.html

12. HDmatters Active 4K HDMI to Displayport 1.2 converter cable 6ft 1.8M HDMI in Displayport out. URL: https://www.aliexpress.com/item/1859873621.html

13. Vention USB Sound Card External 3.5mm USB Adapter USB to Microphone Speaker Audio Interface for Computer Audio Card Sound Card. URL: https://www.aliexpress.com/item/32950973632.html

14. Total Commander, Version 9.22a, is a Shareware file manager for Windows® 95/98/ME/NT/2000/XP/Vista/7/8/8.1/10, and Windows® 3.1. URL: https://www.ghisler.com/

15. SafeInCloud Менеджер паролей для Android, iOS, Windows и Mac. URL: http://safe-in-cloud.com

16. Защита от вирусов — Norton™ Internet Security. URL: https://ru.norton.com/ internet-security

17. Дизайн в стиле Metro. URL: https://www.ixbt.com/soft/wp7-interface-donahue.shtml

18. *Каспаринский Ф.О., Полянская Е.И.* Оптимизация распределения данных, информации и медиаресурсов между локальными и облачными хранилищами // Научный сервис в сети Интернет: труды XVIII Всероссийской научной конференции (19-24 сентября 2016 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2016. С. 170–181. doi:10.20948/abrau-2016-13

19. Технология Logitech Flow: одновременное управление несколькими устройствами. URL: https://www.logitech.com/ru-ru/product/options/page/flow-multi-device-control

20. *Каспаринский Ф.О.* Интернет-сервис как зависимость // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18–23 сентября 2017 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2017. С. 170–182. doi:10.20948/abrau-2017-24

21. *Каспаринский Ф.О.* Образовательные функции сети Интернет // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18–23 сентября 2017 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2017. С. 183–193. doi:10.20948/abrau-2017-25

SPECIALIZATION OF MICROCOMPUTERS FOR TARGETED USE

F. O. Kasparinsky

MASTER-MULTIMEDIA LLC, Moscow

felix@kasparinsky.pro

Abstract

Since 2015, microcomputers have appeared in the information environment, which are a compact system unit with minimal functionality without peripherals. The article published the results of the analysis of the use of 6 different microcomputers in various fields of activity. The purpose of the study is to determine the limiting factors affecting the efficiency of the targeted use of microcomputers. It has been established that for scientific and educational presentations, office and trading activities, it is currently advisable to use fanless microcomputers with a perforated case and an internal WiFi antenna, at least 4 GB of operational and 64 GB of permanent memory, and a microSD (TF) memory card slot, at least 128 GB, NTFS file system), Intel HD Graphics, USB3.0 and HDMI interfaces. Based on comparative experiments, methodological recommendations were created on optimizing the configuration of the hardware-software environment of microcomputers in stationary and mobile conditions. The problems of major updates to Windows 10, as well as the compatibility of Microsoft Store software and third-party manufacturers, are analyzed. It is recommended to specialize individual microcomputers for working with 32-bit applications; accounting and cryptographic programs; as well as conducting presentations with their video. Options for optimal configuration of the Start menu of the Windows 10 desktop are suggested. It is concluded that specialization in the hardware-software configuration of modern microcomputers allows you to increase the efficiency of using

single devices and their paired systems in accordance with BYOD (Bring Your Own Device).

***Keywords**: microcomputer, microPC, Windows 10, configuration, science, education, business, trading, BYOD, specialization*

REFERENCES

1. Windows 10: лучшая Windows в истории поможет вам достичь большего. // Microsoft. URL: https://www.microsoft.com/ru-ru/store/b/windows?icid =TopStripe-windows-cat-page .

2. *Ballagas R., Rohs M., Sheridan J., Borchers J.* BYOD: Bring Your Own Device // UbiComp 2004 Workshop on Ubiquitous Display Environments. Nottingham, UK, September 2004. URL: http://www.vs.inf.ethz.ch/publ/papers/rohs-byod2004.pdf

3. BBen MN9 Mini PC Stick Windows 10 Ubuntu Intel X5 Z8350 Quad Core 2G 4GB RAM Mute Fan WiFi Smart TV Stick PC Mini Computer Micro. URL: https://www.aliexpress.com/item/32767461014.html

4. Fanless Mini Computer Mini PC Stick 4GB 32GB MeLE PCG02 Apo Quad Core Intel Celeron N3450 Windows 10 HDMI 4K LAN WiFi URL: https://www. aliexpress. com/item/32857950812.html

5. ACEPC T6 Pocket Computer stick pc mini fanless 4GB 64GB 2GB 32GB Windows 10 Home licenced Intel Atom x5-Z8350 BT4.0/WiFi/USB 3.0. URL: https://www. aliexpress.com/item/32974950520.html

6. 2018 Pocket PC Mini PC Mobile Computer Windows 10 6 inch Intel Cherry Trail Z8350 Bluetooth Single SIM GPS Shockproof Tablet PC. URL: https://www. aliexpress.com/item/32854798966.html

7. Mini Desktop Core i7 7500u Fanless PC 7th CPU Kabylake Architecture HD Graphics 620 Max 3.5GHz Windows 10 Gaming PC 300M Wifi. URL: https://www. aliexpress.com/item/32815422763.html

8. Rombica WinStick v01. URL: https://rombica.ru/product/winstick-v01/

9. Access 3 / Azulle. URL: https://azulletech.com/product/access3/

10. Original iPazzPort 19s Russian Keyboard 2.4G Mini Wireless Keyboard Air Mouse with Touchpad for Google Android TV Box, Mini PC. URL: https://www. aliexpress. com/item/32952092637.html

11. Ugreen USB Ethernet USB 3.0 2.0 to RJ45 HUB for Xiaomi Mi Box 3/S Set-top Box Ethernet Adapter Network Card USB Lan. URL: https://www.aliexpress.com/item/32401431234.html

12. HDmatters Active 4K HDMI to Displayport 1.2 converter cable 6ft 1.8M HDMI in Displayport out. URL: https://www.aliexpress.com/item/1859873621.html

13. Vention USB Sound Card External 3.5mm USB Adapter USB to Microphone Speaker Audio Interface for Computer Audio Card Sound Card. URL: https://www.aliexpress.com/item/32950973632.html

14. Total Commander, Version 9.22a, is a Shareware file manager for Windows® 95/98/ME/NT/2000/XP/Vista/7/8/8.1/10, and Windows® 3.1. URL: https://www.ghisler.com/

15. SafeInCloud Менеджер паролей для Android, iOS, Windows и Mac. URL: http://safe-in-cloud.com

16. Zashchita ot virusov – Norton™ Internet Security. URL: https://ru.norton.com/internet-security

17. Dizain v stile Metro. URL: https://www.ixbt.com/soft/wp7-interface-donahue.shtml

18. *Kasparinskii F.O., Polianskaia E.I.* Optimizatsiia raspredeleniia dannykh, informatsii i mediaresursov mezhdu lokalnymi i oblachnymi khranilishchami // Nauchnyi servis v seti Internet: trudy XVIII Vserossiiskoi nauchnoi konferentsii (19–24 sentiabria 2016 g., g. Novorossiisk). M.: IPM im. M.V. Keldysha, 2016. S. 170–181. doi:10.20948/abrau-2016-13 .

19. Tekhnologiia Logitech Flow: odnovremennoe upravlenie neskolkimi ustroistvami. URL: https://www.logitech.com/ru-ru/product/options/page/flow-multi-device-control

20. *Kasparinskii F.O.* Internet-servis kak zavisimost // Nauchnyi servis v seti Internet: trudy XIX Vserossiiskoi nauchnoi konferentsii (18–23 sentiabria 2017 g., g. Novorossiisk). M.: IPM im. M.V. Keldysha, 2017. S. 170–182. doi:10.20948/abrau-2017-24

21. *Kasparinskii F.O.* Obrazovatelnye funktsii seti Internet // Nauchnyi servis v seti Internet: trudy XIX Vserossiiskoi nauchnoi konferentsii (18–23 sentiabria 2017 g., g. Novorossiisk). M.: IPM im. M.V. Keldysha, 2017. S. 183–193. doi:10.20948/abrau-2017-25

СВЕДЕНИЯ ОБ АВТОРЕ

КАСПАРИНСКИЙ Феликс Освальдович – кандидат биологических наук, основатель и научный руководитель Лаборатории мультимедийных технологий Биологического факультета МГУ имени М.В. Ломоносова, учредитель и Генеральный директор ООО «МАСТЕР-МУЛЬТИМЕДИА» Сфера научных интересов – формирование информационной среды, дидактически целенаправленное использование мультимедийных технологий.

Felix Osvaldovich KASPARINSKY – Founder and Scientific Director of Multimedia Technologies Laboratory (Biological Faculty, M.V. Lomonosov Moscow State University), Founder and General Director of MASTER-MULTIMEDIA LLC. Research interests include creating an information environment and didactically targeted use of multimedia technologies.

email: felix@kasparinsky.pro

Материал поступил в редакцию 15 ноября 2019 года

УДК 004.432

ВОССТАНОВЛЕНИЕ МНОГОМЕРНОЙ ФОРМЫ ОБРАЩЕНИЙ К ЛИНЕАРИЗОВАННЫМ МАССИВАМ В СИСТЕМЕ SAPFOR

Н. А. Катаев[1], В. Н. Василькин [2]

[1]*Институт прикладной математики им. М.В. Келдыша РАН, г. Москва*
[2]*Московский государственный университет им. М.В. Ломоносова, г. Москва*

[1]kataev_nik@mail.ru, [2]volandtymim@gmail.com

Аннотация

Система автоматизированного распараллеливания SAPFOR (System FOR Automated Parallelization) включает инструменты для анализа и преобразования программ, основной ее целью является снижение сложности распараллеливания программ. Система SAPFOR ориентирована на исследования многоязыковых вычислительных комплексов, разрабатываемых на языках программирования Фортран и Си. Для анализа программ в этой системе используется низкоуровневое их представление в виде LLVM IR, которое позволяет проводить различные оптимизации с целью повышения качества анализа программ. При этом оно теряет некоторые особенности программы, отражаемые ее представлением на языке высокого уровня. Одной из таких особенностей является многомерная структура используемых массивов. Анализ зависимостей по данным является одним из ключевых при исследовании возможности параллельного выполнения программ. При этом такой анализ относится к классу NP-трудных задач. Знание многомерной структуры массивов позволяет во многих случаях учесть структуру индексных выражений в обращениях к массивам и снизить сложность проводимого анализа. Кроме того, использование многомерных массивов позволяет повысить уровень параллелизма в программе за счет использования многомерных решеток процессоров и распараллеливания гнезд циклов, а не отдельных циклов в гнезде. Данная возможность естественным образом поддерживается в DVM-системе. В настоящей работе рассмотрен подход, применяемый в системе SAPFOR для восстановления формы многомерных массивов и обращений к ним по их линеаризованному представлению в LLVM IR. Предложенный подход был

успешно протестирован на различных приложениях, включая тесты производительности из набора NAS Parallel Benchmarks.

***Ключевые слова**: анализ программ, автоматизация распараллеливания, SAPFOR, DVM, LLVM*

ВВЕДЕНИЕ

Использование многомерных массивов характерно для многих вычислительных задач. Например, они позволяют описать свойства объекта в различных точках многомерного вычислительного пространства. Система DVM [1, 2] позволяет учесть многомерную структуру задачи и отобразить измерения массивов на измерения многомерной решетки процессоров. Таким образом, при распараллеливании, в том числе, полуавтоматическом, необходимо учитывать особенности использования многомерных структур данных в программе.

Система SAPFOR [3, 4] выполняет распараллеливание программ с помощью DVMH-языков. Распараллеливание в модели DVM позволяет согласованно распределять данные по процессорам, образующим многомерную процессорную решетку. Использовав директиву *align,* можно указать правила, задающие расположения элементов массивов относительно друг друга на процессорах системы. Данные правила должны быть заданы в виде аффинных выражений, связывающих измерения массивов. Чтобы избежать дополнительных накладных расходов на коммуникации, используемые правила должны отображать элементы массива, используемые на одной и той же итерации цикла на один и тот же процессор. Таким образом, обращения к массивам в циклах и форма индексных выражений в этих обращениях диктуют правила расположения элементов массива друг относительно друга.

Сохранение многомерной структуры данных также позволяет значительно повысить точность и снизить вычислительную сложность анализа зависимостей по данным, который является неотъемлемым этапом распараллеливания программ, как ручного, так и автоматизированного. Если выражения, вычисляющие смещения относительно начала массива в тестируемых обращениях к элементам данного массива, являются линейными относительно индексных переменных объемлющих циклов, то проверка факта наличия/отсутствия зависимости по данным сводится к решению задачи целочисленного линейного программирования, которая является

NP-трудной. Чтобы понизить сложность решаемой задачи, используют эвристики, вводя ограничения на рассматриваемые индексные выражения, и выполняют попарное сравнение индексных выражений, соответствующих одному и тому же измерению массива. Знание многомерной структуры массивов позволяет выделить сравниваемые индексные выражения из линеаризованного представления обращений к массивам. В работе [5] рассмотрены тесты, применяемые в зависимости от вида индексных выражений и обладающие существенно меньшей сложностью.

Стоит отметить, что если размеры измерений массива не являются константными, то линеаризованное представление обращений к элементам массива перестает быть линейным, и проверить наличие зависимостей по данным, не учитывая многомерную структуру массива, становится практически невозможно.

Система SAPFOR при распараллеливании прикладных программ в модели DVM опирается на их представление в виде LLVM IR [6]. Это позволяет проводить анализ программ для разных языков программирования (Фортран, Си), в том числе, решая одну из проблем, характерных для больших вычислительных комплексов (многоязыковость) [7], а также обеспечивает возможность скрытого от пользователя преобразования программ для повышения качества проводимого анализа [8, 9]. Использование LLVM для анализа также устраняет необходимость учитывать синтаксические особенности разных языков и анализировать все многообразие доступных языковых конструкций. Для сохранения специфики исходного языка достаточно воспользоваться отладочной информацией, которая представлена в стандартизованном виде (DWARF [10]) и может быть расширена при необходимости. Недостатком применения LLVM является использование линеаризованного представления обращений к элементам массива, которое полностью соответствует способу хранения массива в памяти, но скрывает многомерную структуру используемых данных. Явным образом в LLVM IR могут быть представлены только массивы с фиксированными размерами каждого измерения (например, тип [100 x [200 x float]] может быть использован для объявления в LLVM IR массива, содержащего 100*200 элементов).

Целью делинеаризации может быть как восстановление формы массивов, которые были многомерными в исходном коде, но оказались линеаризованы при построении LLVM IR, так и поиск для массивов, представленных в исходном коде программы в линеаризованном виде, эквивалентного описания в виде многомер-

ных массивов. В рамках данной работы нас в первую очередь интересует восстановление формы массивов. Это связано с тем, что ограничением, накладываемым на распараллеливаемую программу DVM-системой, является использование именно многомерных массивов в исходном коде программы.

1. ДЕЛИНЕАРИЗАЦИЯ В LLVM

В работе [11] описан подход к делинеаризации, применяемый в LLVM. Будучи частично реализованным в LLVM, данный подход изначально ориентирован на использование в проекте Polly [12], нацеленном на оптимизацию циклов и повышение локальности используемых данных, а также распараллеливание для систем с общей памятью (OpenMP, GPU). В данном случае оптимизация выполняется над LLVM IR, поэтому не требуется соотнесение делинеаризованных массивов с объектами исходного кода. Кроме того, оптимизации подвергаются отдельные циклы, и не требуется согласованная делинеаризация массивов в рамках всей программы. Требование согласованной делинеаризации отличает распараллеливание для систем с распределенной памятью, которое предполагает принятие решений по распределению данных для всей программы. В [11] речь идет о делинеаризации для массивов, измерения которых имеют только переменные размеры. Отдельно в Polly реализована делинеаризация для массивов с фиксированными размерами измерений. В LLVM для представления таких массивов используется встроенный тип многомерного массива, таким образом, делинеаризация фактически не требуется, так как структуру индексных выражений можно восстановить, исходя из типа массива. Массивы, часть измерений которых имеют переменные размеры, а часть – фиксированные, будут рассматриваться как массивы с переменными размерами, поэтому количество измерений и индексные выражения, используемые для доступа по каждому измерению, могут не соответствовать описанию массивов и обращениям к ним в исходной программе. Реализованная в Polly функциональность опирается на используемые в нем структуры данных и лишь частично входит в основную сборку LLVM. В связи с вышесказанным использование предложенных алгоритмов в SAPFOR напрямую оказывается невозможным.

Рассмотрим идею алгоритма, предложенного в [11]. В линеаризованной форме обращение A[S_0]…[S_N–1] к массиву A[D_0]…[D_N–1] будет иметь вид

(1) A + S_0 * D_1 *…* D_N–1 + … + S_N–1.

В данном случае D_I – размер измерения массива с номером I, а S_I – соответствующее индексное выражение, I принимает значения от 0 до N–1. Из данной формулы следуют следующие соотношения:

(2) C_N–1 * D_N–1=GCD(S_0 * D_1 * … * D_N–1, …, S_N–2 * D_N–1),

(3) C_I * D_I=GCD(S_0 * D_1 * … * D_N–1, … S_I–1 * D_I * … D_N–1) / (D_I+1 *…* D_N–1), 0<I<N–1.

Таким образом, чтобы вычислить размер измерения массива с номером I, необходимо найти наибольший общий делитель слагаемых в (1), расположенных слева от слагаемого I, и разделить его на произведение вычисленных ранее размеров измерений от I+1 до N–1.

Множитель C_I может быть не равен 1, например, если в каждом индексном выражении присутствует один и тот же множитель, функция GCD выполняет символьное вычисление наибольшего общего делителя среди всех своих параметров.

Основная сложность алгоритма делинеаризации состоит в том, чтобы выделить из суммы (1), вычисляющей адрес элемента массива, правильное количество слагаемых (равное количеству измерений в массиве), упорядочить их в соответствии с порядком измерений и определить значение коэффициента C_I. Количество слагаемых в сумме (1) может не совпадать с количеством измерений массива, если некоторые из индексных выражений равны нулю, являются константными одновременно с размерами массивов, на которые они умножаются, или при построении формулы (1) было выполнено раскрытие скобок.

Будем называть используемые для делинеаризации слагаемые термами. В работе [11] предложено использовать в качестве термов слагаемые, которые содержат произведение индексной переменной объемлющего цикла на переменные программы. Константные множители в этом случае игнорируются, а коэффициент C_I считается равным 1. Упорядочивание термов выполняется в соответствии с количеством входящих в них множителей. В результате делинеаризованное представление массива может не соответствовать представлению массива в исходной программе. Это является допустимым, так как основная цель делинеаризации в Polly – получить аффинные индексные выражения.

Делинеаризация в SAPFOR основана на идее алгоритма, использованного в Polly, но содержит некоторые особенности, которые позволяют соотнести восста-

новленную многомерную структуру массивов с многомерными массивами в исходной программе.

2. ДЕЛИНЕАРИЗАЦИЯ В SAPFOR

В первую очередь стоит отметить, что мы рассматриваем делинеаризацию для массивов, которые были многомерными в исходной программе. Для остальных массивов делинеаризация может проводиться с целью преобразования исходной программы, чтобы добиться использования в ней многомерных массивов. В этом случае согласованное с исходным кодом восстановление формы не требуется, и может быть использован подход, реализованный в LLVM.

Как было отмечено выше, для массивов, все измерения которых являются известными на этапе компиляции, делинеаризация не требуется. Таким образом, нас будут интересовать массивы, часть или все измерения которых вычисляются во время выполнения программы.

Для вычисления адреса элемента некоторого агрегатного типа данных в LLVM IR используется специальная инструкция getelementptr (см. Рис. 1).

```
%6 = zext i32 %I.0 to i64
%7 = mul nuw nsw i64 %6, %1
%arrayidx11 = getelementptr inbounds [2 x double], [2 x double]* %vla, i64 %7
%8 = zext i32 %J.0 to i64
%arrayidx13 = getelementptr inbounds [2 x double], [2 x double]* %arrayidx11, i64 %8
%arrayidx14 = getelementptr inbounds [2 x double], [2 x double]* %arrayidx13, i64 0, i64 1
```

Рис. 1. Пример LLVM IR вычисляющего смещение для доступа к элементу A[I][J][1] массива размерности M*N*2

Параметрами данной инструкции являются адрес начала участка памяти и одно или несколько значений, используемых для вычисления смещения относительно заданного адреса. В случае многомерных массивов параметрами данной инструкции являются слагаемые из (1), используемые для вычисления смещения по каждому измерению, либо индексное выражение (без умножения на размеры измерений).

Рассмотрим пример вычисления смещения на Рис. 1. Первая инструкция getelementptr сдвигает адрес начала массива A (%vla) на величину I*N (%7), следующая инструкция сдвигает полученный адрес на величину J*2 (%8). Стоит отметить, что умножение на размер последнего измерения массива выполняется неявно инструкцией getelementptr (регистр %8 содержит только значение переменной J). Это

связано с тем, что размер последнего измерения фиксирован, а сам массив A в LLVM IR имеет тип [2 x double] *.

Зависимость количества параметров инструкции getelementptr от количества измерений массива можно считать эвристикой (можно получить эквивалентное LLVM IR, заменив все инструкции getelementptr на Рис. 1 одной с двумя параметрами: адрес массива и предварительно вычисленное смещение), аналогичной той, которая используется в [11] для выделения термов. Это позволяет нам сделать вывод о количестве измерений в массиве и определить количество и порядок термов, необходимых для делинеаризации конкретного обращения к элементу массива.

При этом для вычисления количества измерений используются только инструкции, заведомо обращающиеся к отдельным элементам массива, такие как load и store. Это связано с тем, что другие инструкции, например, инструкции вызова функции, могут принимать в качестве параметра целое измерение массива, и количество операндов в инструкции getelementptr может оказаться меньше количества измерений. Другим источником несоответствия количества операндов и количества измерений может стать использование нулевых индексных выражений. Такие выражения будут опущены в инструкции getelementptr. Чтобы точнее определить количество измерений в массиве, одновременно рассматриваются все обращения к его элементам и среди них выбираются обращения с максимальным количеством измерений.

Во многих ситуациях количество измерений также доступно из отладочной информации. Кроме того, из отладочной информации доступны размеры измерений, известные на этапе компиляции, а также (в некоторых случаях) размеры, заданные с помощью переменных.

Далее, на основе формул (2) и (3) вычисляются размеры неизвестных измерений массива. Чтобы уменьшить вероятность ошибки, наибольший общий делитель вычисляется среди термов, полученных для всех обращений к элементам заданного массива внутри анализируемой функции, которые использовались при определении количества измерений массива. Тот факт, что термы определяются на основе инструкций getelementptr, позволяет рассматривать выражения, не содержащие индексные переменные циклов, а также являющиеся константами. Это также уменьшает вероятность того, что коэффициент C_I окажется отличным от 1, так

как в противном случае все слагаемые, входящие в состав всех индексных выражений по измерениям от 0 до I–1 во всех обращениях к элементам массива, умножались на одну и ту же величину.

Для вычисления наибольшего общего делителя выполняется операция символьного деления термов друг на друга. Допустим, необходимо вычислить наибольший общий делитель (НОД) для последовательности термов T_1, …, T_N. В соответствии с приведенными выше формулами каждый терм представляет собой произведение нескольких множителей. Для вычисления НОД все термы упорядочиваются в соответствии с количеством входящих в них множителей, начиная с наиболее коротких термов. Множители первого терма в последовательности рассматриваются как потенциально возможные множители, образующие НОД всей группы термов (обозначим данную группу множителей DIVS (от англ. divider)). Для оставшихся термов выполняется символьное деление на каждый множитель из DIVS, и если остаток от деления не равен 0, то множитель удаляется из DIVS. После обработки всей последовательности термов в DIVS остаются множители, являющиеся делителями каждого терма последовательности, и НОД принимается равным произведению данных множителей. Если DIVS не содержит ни одного множителя, то НОД найти не удалось, и делинеаризация для рассматриваемого массива оказывается невозможной.

После того как были вычислены размеры измерений массива, вычисляются индексные выражения для каждого обращения к массиву. Начиная с 0 измерения, выполняется деление термов на произведение размеров следующих измерений. Если количество термов для заданного обращения меньше, чем количество измерений массива, и текущий терм не делится на произведение измерений, то индексное выражение считается равным 0. Таким образом восстанавливаются нулевые индексные выражения, опущенные в инструкции getelementptr. Важно отметить, что в процессе восстановления индексных выражений участвуют все обращения к массивам, в том числе, те, которые не могли быть использованы для вычисления количества измерений (например, фактические параметры, передающие в вызов функции целое измерение массива).

Выражения, являющиеся операндами инструкции getelementptr, могут иметь достаточно сложный вид и вложенные приведения типов. Для качественной рабо-

ты алгоритма нахождения наибольшего общего делителя, а также выполнения символьного деления выражений друг на друга, необходимо проведение максимального раскрытия скобок в выражениях и выделение наиболее простых множителей. С точки зрения строгости вычислений, операции приведения типов могут изменять результаты вычислений, по этой причине раскрытие скобок часто оказывается недопустимым в компиляторе и существенно ухудшает результаты делинеаризации в Polly. В системе SAPFOR результаты делинеаризации не используются для генерации кода, на их основе выполняется анализ зависимостей по данным и строится распределение данных, а затем выполняется вставка в исходный код соответствующих DVMH-директив. При необходимости корректность расстановки директив может быть проверена средствами функциональной отладки, входящими в состав DVM-системы. Кроме того, для полного контроля процесса делинеаризации пользователю доступна опция анализатора -fsafe-type-cast, запрещающая небезопасные приведения типов во время анализа.

3. ЭКСПЕРИМЕНТАЛЬНЫЕ РЕЗУЛЬТАТЫ

Чтобы проверить соответствие делинеаризованного представления массива и обращений к нему представлению массива в исходной программе, было проведено тестирование на автоматически сгенерированных тестовых программах. Чтобы максимально покрыть все возможные случаи использования массивов, тесты были классифицированы по следующим параметрам:

- количество измерений массива;
- размеры измерений: переменные (в стиле C99), константные, заданные через перечисления, макросы и литеральные константы разных целочисленных типов;
- динамические и статические массивы;
- локальные (объявленные в теле функции или переданные в качестве параметра) и глобальные массивы;
- различные способы задания индексных выражений: содержащие индексную переменную цикла, константные, содержащие переменные, отличные от индексных переменных;

- наличие приведения типов в объявлениях и обращениях к массивам.

С целью автоматической генерации и запуска тестов, а также анализа результатов запусков одним из авторов данной статьи была разработана библиотека тестирования Ctestgen [13]. Исходный код библиотеки написан на языке Python 3 и доступен на GitHub. Библиотека распространяется под свободной лицензией MIT.

Библиотека состоит из двух модулей: модуля генерации Си-программ через описание их абстрактных синтаксических деревьев и модуля запуска целевой программы на наборах тестов. Использование каждого модуля основано на наследовании базовых абстрактных классов и описании необходимого поведения на языке Python 3.

Каждой генерируемой программе соответствует объект класса Program, который состоит из множества директив подключения заголовочных файлов Include, множества определений макросов Define, множества перечислений Enum, множества глобальных переменных Var и множества функций Function. Функция задается названием, списком аргументов и телом функции – блоком кода. Для создания генератора нужно наследовать абстрактный класс ctestgen.generator.TestGenerator и переопределить метод _generate_programs(), возвращающий абстрактные синтаксические деревья описываемых программ. Пример определения класса для генерации программ с суммирующей функцией, принимающей на вход от 2 до 5 аргументов, приведен в Таблице 1, пример сгенерированной функции для 3 аргументов – в Таблице 2.

Модуль запуска тестов предназначен для запуска целевой программы (в данном случае – анализатора системы SAPFOR) и передаче ей на вход каждой из автоматически сгенерированных тестовых программ. Данный модуль определяет успешность прохождения тестов и собирает метрики запусков. Для запуска программ необходимо наследовать абстрактный класс ctestgen.runner.TestRunner и переопределить метод _on_test(), вызываемый для каждой программы в заданной директории.

Таблица 1. Пример определения класса для генерации программ с суммирующей функцией, принимающей на вход от 2 до 5 аргументов

```
from ctestgen.generator import TestGenerator
```

```
class ExampleTestGenerator(TestGenerator):
  def _generate_programs(self):
    generated_programs = list()
    for i in range(2, 6):
      # Создаем список аргументов функции (от 0 до i)
      sum_arguments = [Int('num_' + str(arg_idx)) for arg_idx in range(i)]
      # Объявляем функцию с заданным именем,  списком аргументов и результатом типа int.
      sum_function = Function('sum_' + str(i) + '_nums', Int, sum_arguments)
      sum_result = Int('sum')
      # Определяем тело функции.
      sum_body = CodeBlock(
        Assignment(VarDeclaration(sum_result), Add(sum_arguments)),
        Return(sum_result)
      )
      sum_function.set_body(sum_body)
      # Создаем программу, содержащую единственную функцию.
      example_program = Program('sum_' + str(i))
      example_program.add_function(sum_function)
      generated_programs.append(example_program)
    return generated_programs
# Генерируем множество программ.
example_generator = ExampleTestGenerator('example_generator_output')
example_generator.run()
```

Каждый сгенерированный тест помимо программы на языке Си 99 содержал ожидаемый результаты делинеаризации, описанные в JSON-формате. Возможность генерации результата делинеаризации в формате JSON также была добавлена в анализатор системы SAPFOR.

Таблица 2. Пример сгенерированной программы

```
int sum_3_nums(int num_0, int num_1, int num_2) {
```

```
 int sum = num_0 + num_1 + num_2;
 return sum;
}
```

В системе SAPFOR модуль делинеаризации используется при построении параллельного варианта исходной последовательной программы, а также при анализе свойств программы, например, определении зависимостей по данным. Как было отмечено в [9], чтобы повысить качество проводимого анализа программы, во многих случаях требуется ее предварительное преобразование. Подход, описанный в [8], позволяет выполнять необходимые преобразования скрыто от пользователя над представлением программы в виде LLVM IR. Общая схема выполнения анализа в системе SAPFOR выглядит следующим образом: выполнение некоторого анализа до преобразования LLVM IR, выполнение преобразования LLVM IR, повторное выполнение анализа и уточнение ранее полученных результатов. При этом результаты анализа будут связаны с объектами исходной непреобразованной программы. Модуль делинеаризации запускается на каждом шаге анализа с целью уточнения результатов анализа зависимостей по данным. Для анализа зависимостей используются наборы тестов, описанных в [5] и реализованных в виде модуля, входящего в состав LLVM. Данный модуль был соответствующим образом модифицирован с целью использования реализованного алгоритма делинеаризации.

Корректная работа модуля делинеаризации, а также модифицированного модуля анализа зависимостей по данным были вручную проверены на тестах производительности NAS Parallel Benchmarks 3.3 [14] и тестовом наборе Polybench/C the Polyhedral Benchmark suite 4.2.1 [15].

ЗАКЛЮЧЕНИЕ

Рассмотрен подход к восстановлению формы многомерных массивов языка C99, представленных в LLVM IR в линеаризованном виде. Предложенный подход опирается на отладочную информацию, доступную в LLVM, и структуру инструкций, используемых для вычисления смещений относительно начала массива. Использование стандартизованного представления отладочной информации позволяет в будущем расширить область применимости предложенного алгоритма на язык Fortran, избежав дополнительного анализа специфичных языковых конструкций.

Предложенный подход основан на идее, используемой в модуле делинеаризации, входящем в состав LLVM, но в отличие от него обеспечивает более точное соответствие результатов делинеаризации описанию многомерных массивов в исходной программе на языке высокого уровня. Данное соответствие является необходимым в силу направленности системы SAPFOR на распараллеливание программ на уровне исходного кода.

Дальнейшие работы планируется направить на совместное использование предложенного подхода и подхода, реализованного в LLVM, для построения эквивалентного многомерного представления массивов, явно заданных в исходной программе в линеаризованном виде, с целью последующей делинеаризации данных массивов в исходном коде программы.

Исходные коды системы SAPFOR доступны на GitHub [16].

СПИСОК ЛИТЕРАТУРЫ

1. *Konovalov N.A., Krukov V.A, Mikhajlov S.N., Pogrebtsov A.A.* Fortan DVM: a Language for Portable Parallel Program Development // Programming and Computer Software, 1995. V. 21. No 1. P. 35–38.

2. *Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Сазанов Ю.Л.* Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами // Вестник Южно-Уральского государственного университета, серия «Математическое моделирование и программирование». 2012. №18 (277), выпуск 12. Челябинск: Издательский центр ЮУрГУ. С. 82–92.

3. *Клинов М.С., Крюков В.А.* Автоматическое распараллеливание Фортран-программ. Отображение на кластер // Вестник Нижегородского университета им. Н.И. Лобачевского. 2009. № 2. С. 128–134.

4. *Бахтин В.А., Жукова О.Ф., Катаев Н.А., Колганов А.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Савицкая О.А., Смирнов А.А.* Автоматизация распараллеливания программных комплексов // Труды XVIII Всероссийской научной конференции «Научный сервис в сети Интернет», Новороссийск, Россия, 19–24 сентября 2016 г. М.: ИПМ им. М.В. Келдыша, 2016. С. 76–85. doi:10.20948/abrau-2016-31

5. *Goff G., Kennedy K., Tseng C.W.* Practical Dependence Testing // Proceedings of the ACM SIGPLAN 1991 conference on Programming language design and implementation (PLDI '91), 1991. ACM, New York, NY, USA, 1991. P. 15–29.

6. *Lattner C., Adve V.* LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation // Proc. of the 2004 International Symposium on Code Generation and Optimization (CGO'04), 2004. Palo Alto, California.

7. *Бахтин В.А., Жукова О.Ф., Катаев Н.А., Колганов А.С., Крюков В.А., Кузнецов М.Ю., Поддерюгина Н.В., Притула М.Н., Савицкая О.А., Смирнов А.А.* Распараллеливание программных комплексов. Проблемы и перспективы // Труды XX Всероссийской научной конференции «Научный сервис в сети Интернет», Новороссийск, Россия, 17–22 сентября 2018 г. М.: ИПМ им. М.В. Келдыша, 2018. С. 63–72. doi:10.20948/abrau-2018-33

8. *Kataev N.A.* Application of the LLVM Compiler Infrastructure to the Program Analysis in SAPFOR // Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2018. Communications in Computer and Information Science, 2018. Vol 965. Springer, Cham. P. 487-499. doi:10.1007/978-3-030-05807-4_41

9. *Катаев Н.А., Козырев В.И.* Преобразование программ на высокоуровневом языке программирования на основе результатов анализа низкоуровневого представления программ в системе SAPFOR // Труды XIII международной конференции «Параллельные вычислительные технологии, ПаВТ'2019», Калининград, Россия, 2–4 апреля 2019 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2019. С. 251–262.

10. Dwarf 3 Standard. URL: http://eagercon.com/dwarf/dwarf3std.htm

11. *Grosser T., Pop S., Ramanujam J., Sadayappan P.* On recovering multi-dimensional arrays in Polly // 5th International Workshop on Polyhedral Compilation Techniques, IMPACT 2015. P. 1–9.

12. Polly – Polyhedral optimizations for LLVM. URL: https://polly.llvm.org/

13. Ctestgen. URL: https://github.com/VolandTymim/ctestgen

14. *Seo S., Jo G., Lee J.* Performance Characterization of the NAS Parallel Benchmarks in OpenCL // 2011 IEEE International Symposium on. Workload Characterization (IISWC), 2011. P. 137–148.

15. PolyBench/C the Polyhedral Benchmark suite. URL: http://web.cse.ohio-state.edu/~pouchet.2/software/polybench/polybench.html

16. SAPFOR. URL: https://github.com/dvm-system

RECONSTRUCTION OF MULTI-DIMENSIONAL FORM OF LINEARIZED ACCESSES TO ARRAYS IN SAPFOR

N. A. Kataev[1], V. N. Vasilkin [2]

[1]*Keldysh Institute of Applied Mathematic RAS;* [2] *Lomonosov Moscow State University*

[1]kataev_nik@mail.ru, [2] volandtymim@gmail.com

Abstract

The system for automated parallelization SAPFOR (System FOR Automated Parallelization) includes tools for program analysis and transformation. The main goal of the system is to reduce the complexity of program parallelization. SAPFOR system is focused on the investigation of multilingual applications in Fortran and C programming languages. The low-level LLVM IR representation is used in SAPFOR for program analysis. This representation allows us to perform various IR-level optimizations to improve the quality of program analysis. At the same time, it loses some features of the program, which are available in its higher level representation. One of these features is the multi-dimensional structure of the arrays. Data dependence analysis is one of the main problems which should be solved to automate program parallelization. Moreover, such an analysis belongs to the class of NP-hard problems. Knowledge of the multidimensional structure of arrays allows in many cases to take into account the structure of index expressions in calls to arrays and reduce the complexity of the analysis. In addition, the use of multi-dimensional arrays allows us to use multi-dimensional processor matrix and to parallelize a whole loop nests, rather than a single loop in the nest. So, parallelism of a program is going to be increased. These opportunities are natively supported in the DVM system. This paper discusses the approach used in the SAPFOR system to recover the form of multi-dimensional arrays by their linearized representation in LLVM IR. The proposed approach has been

successfully evaluated on various applications including performance tests from the NAS Parallel Benchmarks suite.

***Keywords**: program analysis, semi-automatic parallelization, SAPFOR, DVM, LLVM*

REFERENCES

1. *Konovalov N.A., Krukov V.A, Mikhajlov S.N., Pogrebtsov A.A.* Fortan DVM: a Language for Portable Parallel Program Development // Programming and Computer Software. 1995. V. 21. No 1. P. 35–38.

2. *Bakhtin V.A., Klinov M.S., Kriukov V.A., Podderiugina N.V., Pritula M.N., Sazanov Iu.L.* Rasshirenie DVM-modeli parallelnogo programmirovaniia dlia klasterov s geterogennymi uzlami // Vestnik Iuzhno-Uralskogo gosudarstvennogo universiteta, seriia "Matematicheskoe modelirovanie i programmirovanie", No 18 (277), vypusk 12. Cheliabinsk: Izdatelskii tsentr IuUrGU, 2012. P. 82–92.

3. *Klinov M.S., Kriukov V.A.* Avtomaticheskoe rasparallelivanie Fortran-programm. Otobrazhenie na klaster // Vestnik Nizhegorodskogo universiteta im. N.I. Lobachevskogo. 2009. No 2. S. 128–134.

4. *Bakhtin V.A., Zhukova O.F., Kataev N.A., Kolganov A.S., Kriukov V.A., Podderiugina N.V., Pritula M.N., Savitskaia O.A., Smirnov A.A.* Avtomatizatsiia rasparallelivaniia programmnykh kompleksov // Trudy XVIII Vserossiiskoi nauchnoi konferentsii “Nauchnyi servis v seti Internet”, Novorossiisk, Russia, 19–24 sentiabria. M.: IPM im. M.V. Keldysha, 2016. S. 76–85. doi:10.20948/abrau-2016-316

5. *Goff G., Kennedy K., Tseng C.W.* Practical Dependence Testing // Proceedings of the ACM SIGPLAN 1991 conference on Programming language design and implementation (PLDI ’91), 1991. ACM, New York, NY, USA, 1991. P. 15–29.

6. *Lattner C., Adve V.* LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation // Proc. of the 2004 International Symposium on Code Generation and Optimization (CGO’04), 2004. Palo Alto, California.

7. *Bakhtin V.A., Zhukova O.F., Kataev N.A., Kolganov A.S., Kriukov V.A., Kuznetsov M.Iu., Podderiugina N.V., Pritula M.N., Savitskaia O.A., Smirnov A.A.* Rasparallelivanie programmnykh kompleksov. Problemy i perspektivy // Trudy XX Vserossiiskoi nauchnoi konferentsii “Nauchnyi servis v seti Internet”, Novorossiisk, Russia, 17–22

sentiabria 2018 g. M.: IPM im. M.V. Keldysha, 2018. S. 63–72. doi:10.20948/abrau-2018-33

8. *Kataev N.A.* Application of the LLVM Compiler Infrastructure to the Program Analysis in SAPFOR // Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2018. Communications in Computer and Information Science, 2018. V. 965. Springer, Cham. P. 487–499. doi:10.1007/978-3-030-05807-4_41

9. *Kataev N.A., Kozyrev V.I.* Preobrazovanie programm na vysokourovnevom iazyke programmirovaniia na osnove rezultatov analiza nizkourovnevogo predstavleniia programm v sisteme SAPFOR // Trudy XIII mezhdunarodnoi konferentsii “Parallelnye vychislitelnye tekhnologii, PaVT'2019”, Kaliningrad, Russia, 2–4 aprelia 2019 g. Korotkie stati i opisaniia plakatov. Cheliabinsk: Izdatelskii tsentr IuUrGU, 2019. S. 251–262.

10. Dwarf 3 Standard. URL: http://eagercon.com/dwarf/dwarf3std.htm

11. *Grosser T., Pop S., Ramanujam J., Sadayappan P.* On recovering multi-dimensional arrays in Polly // 5th International Workshop on Polyhedral Compilation Techniques, IMPACT 2015. P. 1–9.

12. Polly – Polyhedral optimizations for LLVM. URL: https://polly.llvm.org/

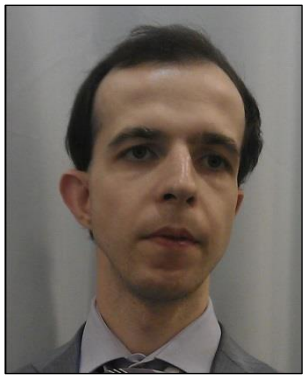
13. Ctestgen. URL: https://github.com/VolandTymim/ctestgen

14. *Seo S., Jo G., Lee J.* Performance Characterization of the NAS Parallel Benchmarks in OpenCL // 2011 IEEE International Symposium on. Workload Characterization (IISWC), 2011. P. 137–148.

15. PolyBench/C the Polyhedral Benchmark suite. URL: http://web.cse.ohio-state.edu/~pouchet.2/software/polybench/polybench.html

16. SAPFOR. URL: https://github.com/dvm-system

СВЕДЕНИЯ ОБ АВТОРАХ



КАТАЕВ Никита Андреевич – научный сотрудник ИПМ им. М.В. Келдыша, специалист в области системного программирования. Сфера научных интересов – компиляторные технологии, автоматизация распараллеливания программ.

Nikita Andreevich KATAEV – Researcher of KIAM RAS, a specialist in system programming. Research interests include compiler technologies, semi-automatic program parallelization.

email: kataev_nik@mail.ru

ВАСИЛЬКИН Владислав Николаевич – студент 1 курса магистратуры факультета ВМиК МГУ им М.В. Ломоносова.

Vladislav Nikolaevich VASILKIN – student of CMC faculty of Lomonosov Moscow State University

email: volandtymim@gmail.com

Материал поступил в редакцию 15 ноября 2019 года

УДК 519.68

ДОБАВЛЕНИЕ СТАТИЧЕСКОЙ ТИПИЗАЦИИ В ЯЗЫК ФУНКЦИОНАЛЬНО-ПОТОКОВОГО ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

А. И. Легалов, И. А. Легалов, И. В. Матковский

Сибирский федеральный университет, г. Красноярск

legalov@mail.ru, igor@legalov.ru, alpha900i@mail.ru

Аннотация

Предложено добавить статическую систему типов в функционально-потоковую модель параллельных вычислений и разработанный на ее основе язык функционально-потокового параллельного программирования. Использование статической типизации повышает возможность трансформации функционально-потоковых параллельных программ в программы, выполняемые на современных параллельных вычислительных системах. Предложены языковые конструкции. Описаны их синтаксис и семантика. Отмечена необходимость использования принципа единственного присваивания при формировании хранилищ данных конкретного типа. Рассмотрены особенности инструментальной поддержки предлагаемого подхода.

Ключевые слова*: парадигмы программирования, параллельное программирование, функционально-потоковое параллельное программирование, статическая типизация, модели параллельных вычислений*

ВВЕДЕНИЕ

Современные методы разработки параллельных программ сильно зависят от особенностей архитектур параллельных вычислительных систем (ПВС), что находит соответствующее отражение в языках программирования. Практически любые изменения архитектуры ПВС ведут к переписыванию и модификации уже разработанного и отлаженного кода. Попыткой преодолеть эту ситуацию является применение концепции архитектурно-независимого параллельного программирования (АНПП), ориентированного на разработку программ с использованием языковых и инструментальных средств, предназначенных для абстрактных

(виртуальных) параллельных систем с неограниченными вычислительными ресурсами и стратегиями управления вычислениями по готовности данных. Такие подходы развиваются в разных направлениях. Можно отметить язык программирования COLAMO, разработанный для систем на кристалле [1, 2]. Создание универсальных языков, напрямую не связанных с архитектурными ограничениями, можно проследить на примере функциональных языков параллельного программирования Sisal [3] и Пифагор [4].

Наиболее последовательно концепция АНПП нашла свое отражение в языке Пифагор. Она непосредственно учитывается в его модели функционально-потоковых параллельных вычислений. Модель программы описана как ресурсно-неограниченный ациклический безусловный граф, в котором управление осуществляется по готовности данных. Помимо этого в ней реализован принцип единственного использования вычислительных ресурсов [4]. На уровне модели предполагается, что для выполнения любых операций выделяются свои уникальные ресурсы, реальное распределение которых осуществляется после того, как разработана и отлажена логическая структура программы. Для апробации возможностей языка разработаны инструментальные средства, поддерживающие процесс создания, преобразования и выполнения функционально-потоковых параллельных программ [5].

Однако следует отметить невысокую эффективность выполнения программ, что обуславливается использованием интерпретатора. Последнее связано с тем, что в языке применяется динамическая типизация данных, а операторы, представленные в модели вычислений, обладают динамическим поведением, позволяя формировать списки произвольной размерности во время вычислений. В связи с этим практически невозможна эффективная трансформация написанных программ в современные статически типизированные языки, используемые при реальном параллельном программировании.

Вместе с тем эксперименты, проводимые с применением разработанных инструментальных средств, показали возможность эффективного применения данной парадигмы для оптимизации [6], формальной верификации [7] и отладки [8] программ еще до того момента, как начнется их преобразование к конкретной архитектуре. Это позволяет иметь программу, перенос которой на реальные ПВС мог бы осуществляться более формально за счет наложения ресурс-

ных ограничений, учитывающих особенности конкретной архитектуры, с сохранением уже отлаженной общей логики функционирования.

В связи с этим перспективной видится модификация функционально-потоковой модели параллельных вычислений (ФПМПВ), направленная на учет особенностей организации данных в современных языках программирования, что позволило бы упростить процесс трансформации функционально-потоковых параллельных (ФПП) программ. В основном эта модификация связана с применением статической типизации и фиксацией размерности списковых и контейнерных структур данных, что ведет к пересмотру ряда концепций ФПМПВ. Вполне естественно, что в соответствии с этими изменениями должен измениться и язык ФПП программирования.

В результате проведенных исследований сформирована статически типизированная модель функционально-потоковых параллельных вычислений (СТМФППВ). Как и предшествующая ей ФПМПВ, она определяет программу как информационный граф с управлением по готовности данных. Однако операторы, описывающие алгоритм программы, разработаны с учетом возможных преобразований в статически типизированные языки программирования, что ведет к изменению ряда аксиом и алгебры преобразований. На основе предложенной модели разработан статически типизированный язык функционально-потокового параллельного программирования (СТЯФППП) Smile.

1. СТАТИЧЕСКАЯ ТИПИЗАЦИЯ НА УРОВНЕ ОПЕРАТОРОВ

Как и в предшествующей ФПМПВ [4], операторы задают узлы информационного графа, в котором вычисления выполняются по готовности данных. Однако существует ряд особенностей, связанных с изменением требований. Необходимо обеспечить поддержку следующих свойств, характерных для статически типизированных языков программирования:

— эффективную трансформацию статически типизированных функционально-потоковых параллельных программ в другие модели вычислений вместо их интерпретации;

— повышение контроля за счет использования сильной типизации;

— сохранить принцип управления по готовности данных и общую концепцию функционально потоковой модели параллельных вычислений;

— каждый из программоформирующих операторов должен опираться на типизированные данные, контролируемые на этапе компиляции;

— контейнерные (списковые) данные должны иметь фиксированный размер, определяемый либо во время компиляции, либо во время выполнения;

— аксиоматика языка должна быть упрощена, чтобы уменьшить число динамических проверок и преобразований во время выполнения;

— упрощение алгебры эквивалентных преобразований.

Приведенные требования ведут к изменению практически всех операторов ФПМПВ, в результате чего сформирована модель вычислений, обладающая иными свойствами. Эти свойства определяются через особенности функционирования программоформирующих операторов СТМФППВ.

Оператор интерпретации описывает функциональные преобразования аргумента. Он имеет два входа, на которые через информационные дуги поступают функция **F** и аргумент **X**. Как аргумент, так и функция могут являться результатами предшествующих вычислений. Основными особенностями новой версии данного оператора являются:

— типы аргументов на входах оператора должны быть известны во время компиляции;

— тип результата на выходе также вычисляется во время компиляции;

— на входе и выходе оператора допускаются именованные типы данных, неименованные структуры и кортежи;

— для именованных типов допустима только именованная эквивалентность;

— для кортежей допускается структурная эквивалентность;

— предопределенность базовых операций, для которых на уровне языка прописаны все возможные типы данных аргументов и результатов.

Исходя из этого, для базовых функций языка изначально определяются сигнатуры, задающие типы аргументов и результатов. Для функций, определяемых пользователем, типы аргументов и результатов явно задаются во время определения функций. Допускается дуализм некоторых базовых данных, которые в зависимости от использования в операторе интерпретации могут выступать как в качестве аргумента, так и функции. В этом случае для них возможно определение двойного типа (тип данных и сигнатура функции).

Оператор интерпретации запускается по готовности данных, что фиксируется появлением разметки на входных дугах. Получение результата задается разметкой выходной дуги.

Вместо группировки в список данных в СТМФППВ используется **группировка в кортеж**. Можно выделить следующие основные свойства данного оператора:

— размер кортежа определяется во время компиляции (что обусловливается необходимостью знать типы группируемых данных и их размер);

— элементы кортежа являются данными именованных типов;

— обеспечивается сравнение на структурную эквивалентность с другими кортежами;

— готовность кортежа к выполнению определяется по готовности всех его данных;

— отсутствуют внутренние эквивалентные преобразования, изменяющие размер кортежа во время выполнения (сигнал, удаляемый из списка в ФПМПВ, является типом данных без значения и сохраняется в явном виде).

Изменены также аксиомы, определяющие преобразование кортежей во время вычислений, что также обуславливается введением дополнительного контроля во время компиляции.

Группировка в параллельные списки заменяется на **группировку в рой**. Используется для объединения данных, над которыми выполняется одна массовая операция. К свойствам роя относятся:

— размер роя определяется во время компиляции;

— элементами роя являются данные одного именованного типа или все элемента роя структурно эквивалентны;

— готовность роя к выполнению определяется по готовности хотя бы одного элемента (асинхронность в обработке отдельных его элементов);

— отсутствуют внутренние эквивалентные преобразования, изменяющие размер роя во время выполнения;

— внутри кортежей рой не вырождается в последовательность элементов кортежа, а является единым элементом.

— алгебра эквивалентных преобразований роев реализуется только во время компиляции.

Приведенные характеристики позволяют рассматривать рой в качестве набора независимых данных, запускаемых по мере их поступления. На выходе оператора интерпретации также формируется рой, состоящий из элементов одного типа.

Группировка в задержанный список заменяется на **оператор задержки вычислений**, который отличается от ранее предлагаемой возвратом только одного значения, тип которого определяется во время компиляции и может быть любым. В языке с динамической типизацией результатом являлся параллельный список. В новой модели выдача вместо параллельного списка роя тоже возможна, но только при явном его задании в качестве результата задержки. Раскрытие задержки осуществляется сразу же после того, как она становится аргументом оператора интерпретации. Это позволяет в ряде случаев использовать данный оператор в качестве скобочного выражения, изменяющего приоритет операций.

2. СТАТИЧЕСКАЯ ТИПИЗАЦИЯ НА УРОВНЕ ДАННЫХ

В отличие от языка ФПП программирования Пифагор, в котором представлены только базовые типы данных, язык программирования Smile имеет развитую систему типов, обуславливаемую необходимостью повышения контроля на этапе компиляции. Вводимые базовые типы данных во многом повторяют типы, используемые в современных статически типизированных языках. Однако помимо этого предлагаются типы, обеспечивающие возможность манипуляции с параллельными списками, что ведет к их определенному влиянию на СТМФППВ.

Выделяются следующие базовые типы: целый, булевский, сигнальный, функциональный, ошибки. Эти типы являются основообразующими и используются не только при обработке произвольных данных, но и в ключевых операторах языка. Дополнительные типы, такие, как действительные числа, символы и другие, рассматриваются в качестве расширений, определяемых проблемной ориентацией, и могут включаться в различные предметно-ориентированные версии языка. В целом можно отметить, что вопросы, связанные с расширением базовых типов, не являются принципиальными на уровне модели вычислений.

К составным типам относятся: массив, структура, кортеж, обобщение, рой, поток, функциональный, ссылочный. Эти типы используются для формирования производных абстракций, определяемых программистом, и состоят как из базо-

вых, так и производных типов. Они в основном подменяют ранее используемые понятия списка данных и параллельного списка, однако при этом являются описаниями, а не операторами, что позволяет на их основе формировать соответствующие хранилища данных, используемые по принципу единственного присваивания. Массив, структура и кортеж являются специализированными разновидностями списка данных ФПМПВ.

Тип «массив» предназначен для описания данных одного типа. Во многом он аналогичен использованию многомерных массивов традиционных императивных языков программирования. Массив имеет фиксированные размерность и длины по каждому измерению. Описание данного типа на уровне языка программирования задается с использованием следующего синтаксиса:

Массив ::= ИмяТипа «(» Размерность «)»

Размерность ::= Целое { «,» Целое }

Примеры массивов:

```
A << type int(100)
B << type bool(30, 40)
```

Тип «структура» обеспечивает группировку разнотипных данных по аналогии со структурными типами различных языков программирования. Структура состоит из полей, каждое из которых имеет имя и тип. Описание структуры имеет следующий синтаксис:

Структура ::= «(» ПолеСтруктуры { «,» ПолеСтруктуры } «)»

ПолеСтруктуры ::= ИмяПоля «@» ИмяТипа

| «[» ИмяПоля { «,» ИмяПоля } «]» «@» ИмяТипа

Примеры структурных типов:

```
Triangle << type (a@int, b @ int, c @int)
Rectangle << type ([x,y]@int)
```

Тип «кортеж» отличается от структуры отсутствием именованных полей. Он аналогичен вектору, но может содержать разнотипные элементы. Обращение к элементам кортежа осуществляется по номеру поля. Для задания кортежей используется следующий синтаксис:

Кортеж ::= «(» ИмяТипа { «,» ИмяТипа } «)»

Примеры задания типов кортежей:

```
C << type (int)
```

```
B << type (int, bool, signal)
```

Тип «обобщение» во многом аналогичен по организации и использованию обобщениям, используемым в императивных языках. Основной его задачей является описание вариантных данных. Существуют различные подходы к организации обобщений, включая методы, поддерживающие полиморфизм. В языке используются обобщения, поддерживающие процедурно-параметрическую парадигму программирования, которая обеспечивает более гибкую поддержку эволюционного расширения программ по сравнению с другими подходами [9]. Правила, определяющие синтаксис обобщений, имеют следующий вид:

Обобщение ::= «{» ПолеОбобщения { «,» ПолеОбобщения } «}»

ПолеОбобщения ::= ИмяТипа { «,» ИмяТипа }

| ИмяПризнака «@» ИмяТипа

| «[» ИмяПризнака { «,» ИмяПризнака } «]» «@» ИмяТипа

Примеры описания обобщений:

```
Figure1 << type {Triangle, Rectangle}
Figure2 << type {trian@Triangle,
                 rect@Rectangle,
                 rhomb@Rectangle}
WeekDay << type{[Sun,Mon,Tue,Wen,Thu,Fri,Sat]@signal}
```

Тип «рой» используется для описания независимых данных, над которыми возможно выполнение массовых параллельных операций. Все элементы роя имеют один тип, а функция, осуществляющая их обработку, может одновременно выполняться над каждым элементом. Результатом является также рой, размерность которого равна размерности роя аргументов. Синтаксические правила, определяющие данный тип, имеют следующий вид:

Рой ::= ИмяТипа «[» Целое «]»

Пример описания типа

```
R << type int[100]
```

Тип «поток» является альтернативой асинхронного списка [10]. Он используется для обработки данных поступающих последовательно и асинхронно в произвольные промежутки времени. Размерность поступающих данных при этом неизвестна, поэтому завершение обработки возможно только по признаку конца потока. Поток готов к обработке при наличии в нем хотя бы одного эле-

мента. Тип всех элементов потока одинаков. Синтаксические правила, определяющие поток:

Поток ::= ИмяТипа «{» «}»

Пример описания потокового типа:

```
A << type int{}
```

Тип «функция» (или функциональный тип) позволяет задать сигнатуру функции, определяя имя типа, тип аргумента, а также тип результата. В целом определение функционального типа отличается от общепринятых только тем, что любая функция имеет только один аргумент и возвращает только один результат. Синтаксические правила, определяющие описание функционального типа:

ФункциональныйТип ::= func Аргумент «->» Результат

Аргумент ::= ИмяТипа | Кортеж

Результат ::= ИмяТипа | Кортеж

Примеры описаний:

```
F << type func int -> int
F2 << type func (bool, int, int) -> (int, bool)
```

Тип «ссылка» (или ссылочный тип) обеспечивает поддержку указателей на различные хранилища определенного типа, что позволяет передавать значения между функциями без их копирования. Основное назначение заключается в дополнительном контроле типов в ходе передач. Синтаксические правила, определяющие описание ссылочного типа:

Ссылка ::= «&» ИмяТипа

ОткрытыйМассив ::= ИмяТипа «(» «» { «,» «*» } «)»*

3. СТАТИЧЕСКАЯ ТИПИЗАЦИЯ ПРИ ОПИСАНИИ ФУНКЦИЙ

В отличие от языка ФПП программирования Пифагор, при описании функций используется явное задание типов аргумента и результата, что обеспечивает дополнительный контроль во время компиляции. Эти изменения затрагивают заголовок функции, что определяется следующим синтаксическим описанием:

Функция ::= func Аргумент «->» Результат ТелоФункции

Аргумент ::= ИмяАргумента «@» (ИмяТипа | Кортеж) | Структура

Результат ::= ИмяТипа | Кортеж | Структура

Примеры:

```
Factorial << func n@int -> int {...}
TrianPerimeter << func ([a,b,c]@int) -> int {...}
Sum << func t@(int, int) -> int {t:+ >> return}
```

4. ДОПОЛНИТЕЛЬНЫЕ РАСШИРЕНИЯ

Наличие статической типизации ведет к появлению дополнительных возможностей по разработке функционально-потоковых параллельных программ. К ним следует отнести описание хранилищ данных предопределенного типа. С ними можно взаимодействовать как с использованием принципа единственного присваивания, так и с применением многократного доступа, аналогичного с доступом к переменным в императивных языках программирования. Последнее ведет к потере надежности программ, но может иногда использоваться после формального доказательства непротиворечивости, например, для ускорения вычислений. Хранилища описываются следующими синтаксическими правилами:

Хранилище ::= (safe | var) Тип

ИменованноеБезопасноеХранилище = ИмяХранилища «@» Тип

Первое правило явно определяет безопасное (safe) хранилище, заполняемое по принципу единственного присваивания (с контролем во время заполнения) или небезопасный вариант (var), при котором не контролируется возможность повторной записи. Второе правило является «синтаксическим сахаром» для описания безопасных хранилищ. Примеры:

```
t1 << safe Triangle ≡ t1@Triangle
t << var Triangle
```

Для взаимодействия с хранилищами необходимо ввести дополнительный набор операций, ведущих к изменению семантики модели вычислений и влияющих на синтаксис языка программирования. Эти операции обеспечивают доступ к хранилищам по чтению и записи. Наличие хранилищ разного типа предопределяет и разнообразие описаний, отражаемых в соответствующих синтаксических правилах:

ЧтениеВсегоХранилища ::= ИмяХранилища «:» Функция

ЧтениеЭлементаМассива ::=

ИмяХранилища «(» индексы «)» «:» Функция

ЧтениеЭлементаКортежаИлиОдномерногоМассива ::=

ИмяХранилища «:» индекс «:» Функция

ЧтениеЭлементаСтруктуры ::=

ИмяХранилища «.» ИмяПоля «:» Функция

ЧтениеЭлементаПотока ::= ИмяХранилища «:» get «:» Функция

Запись_в_хранилище ::= Значение «:» ИмяХранилища

Запись_в_массив ::= Значение «:» ИмяХранилища «(» индексы «)»

Запись_в_кортеж ::= Значение «:» ИмяХранилища «(» индексы «)»

Запись_в_структуру ::= Значение «:» ИмяХранилища «.» ИмяПоля

Запись_в_поток ::= Значение «:» ИмяХранилища

Применение данных операций сопровождается выполнением в надежных хранилищах принципа единственного присваивания и управления по готовности данных.

5. ОСОБЕННОСТИ ИНСТРУМЕНТАЛЬНОЙ ПОДДЕРЖКИ

Добавление в язык статической системы типов ведет к модификации инструментальных средств, обеспечивающих поддержку функционально потокового параллельного программирования [5]. Разработанный язык обеспечивает представление параллелизма на уровне элементарных операций, при котором каждая функция описывает только информационный граф алгоритма без каких-либо управляющих связей. Транслятор преобразует исходный текст функции в промежуточное представление, которое используется для оптимизации существующих зависимостей по различным критериям, а также для построения на его основе управляющего графа, задающего порядок выполнения в соответствии с выбранной стратегией управления вычислениями [11]. Трансформация управляющего графа и его оптимизация позволяют получать стратегии, отличающиеся от управления по готовности данных и учитывающие различные ограничения, свойственные, например, реальным вычислительным системам.

Общая схема, отображающая различные варианты использования предлагаемых инструментальных средств, приведена на рис. 1. В рамках создаваемой среды выделяются следующие подсистемы:

Рис. 1. Состав инструментальных средств, поддерживающих функционально-потоковое параллельное программирование

— транслятор с языка функционально-потокового параллельного программирования в промежуточное представление, называемое реверсивным информационным графом (РИГ);

— генератор управляющего графа (УГ), формирующий граф управления вычислениями;

— событийная машина, обеспечивающая выполнение функционально-потоковых параллельных программ в автоматическом и отладочном режимах, использующая в качестве программы РИГ и УГ;

— средства оптимизации реверсивного информационного графа;

— средства оптимизации управляющего графа;

— средства формальной верификации ФПП программ;

— средства преобразования ФПП программ в программы для других архитектур ПВС.

Транслятор ориентирован на обработку текстовых файлов, каждый из которых содержит один из артефактов языка. Для каждой функции в памяти компьютера порождается реверсивный информационный граф (РИГ), который сохраняется в репозитории функций в текстовой форме. Выбор текстового пред-

ставления для описания РИГ обусловлен тем, что формирование на его основе внутреннего представления в памяти компьютерной системы может быть легко выполнено с помощью простых транслирующих программ. Помимо этого разработчик может легко читать и анализировать оттранслированные функции, рассматривая данную форму как аналог языка ассемблера. В отличие от РИГ языка с динамической типизацией, данный граф содержит дополнительную информацию по типам для каждого узла.

Реверсивный информационный граф, порождённый транслятором, позволяет построить управляющий граф, определяющий выполнение функции. Для этого предназначена специальная утилита, которая формирует УГ, задающий управление вершинами РИГ по готовности данных. УГ сохраняется в репозитории в текстовом виде.

Тестирование и отладка функционально-потоковых параллельных программ на текущем этапе осуществляется специальным интерпретатором (событийной машиной), состоящим из множества событийных процессоров (СП), управляемых менеджером событийной машины. Каждый из этих процессоров (рис. 2) осуществляет обработку только одной функции, запускаемой в отдельном потоке. Выполнение операций внутри функции в настоящий момент осуществляется последовательно за счёт изменения состояния вершин УГ, которые инициируют вычисления в вершинах РИГ.

Функционирование СП осуществляется следующим образом. Исходные сигналы, фиксирующие протекание в системе различных событий и определяемые начальной разметкой УГ, загружаются в очередь, из которой передаются обработчику в соответствии с дисциплиной обслуживания. В простейшем случае это может быть дисциплина FIFO. Обработчик управляющих сигналов анализирует поступившее событие и выбирает указанный в нем узел управляющего графа. На основе анализа состояния узла УГ он может обратиться к связанной с ним вершине информационного графа за кодом выполняемой операции. В случае, когда операция обработки данных должна быть выполнена, происходит обращение к обработчику узлов РИГ, который осуществляет требуемые функциональные преобразования и сохраняет промежуточные результаты. После обработки данных управляющий узел переходит в новое состояние и при необходи-

мости формирует сигнал, передаваемый следующему узлу, который поступает в очередь управляющих сигналов.

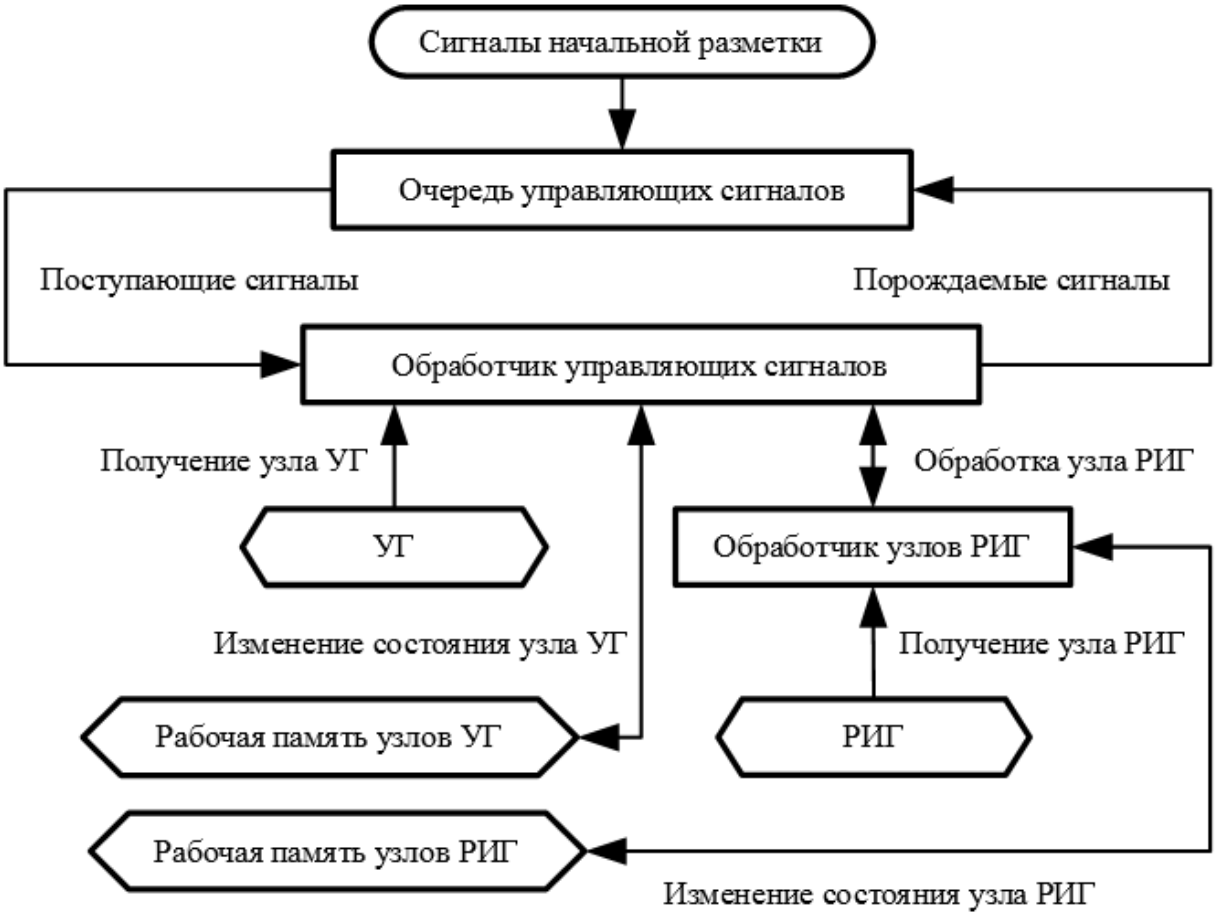


Рис. 2. Обобщённая структура событийного процессора

Основные методы оптимизации, разработанные в настоящее время, затрагивают преобразование промежуточных представлений функционально-потоковых параллельных программ. Они направлены на изменение информационного и управляющего графов. Проводимые преобразования во многом аналогичны методам, используемым при оптимизации исходных текстов программ и их промежуточных представлений в других языках программирования, и предназначены для решения схожих задач. Специфика функционально-потоковой модели параллельных вычислений накладывает свои особенности на реализацию этих методов. Она обусловлена алгеброй эквивалентных преобразований модели, реализованной в языке: информационный и управляющий графы могут изменяться независимо друг от друга. В ходе оптимизации необходимо обеспечивать согласованность РИГ и УГ, однако для многих задач доста-

точно обработки РИГ. В таких случаях оптимизация управляющего графа должна осуществляться после трансформации информационного графа и построении на его основе нового УГ. Следует отметить, что разрабатываемые в настоящее время утилиты не затрагивают распределение реальных вычислительных ресурсов.

Наличие в программе только информационных зависимостей и отсутствие ресурсных ограничений позволяют облегчить формальную верификацию. Основными задачами в данном направлении работ являются: исследование специфики применения формальных методов доказательства корректности; разработка инструментальных средств, упрощающих верификацию. Акцент сделан на доказательство корректности программы с использованием дедуктивного анализа на основе исчисления Хоара [12]. Тройка Хоара представляется как информационный граф программы, к входной и выходной дугам которого привязаны формулы на языке спецификации (предусловие и постусловие). Процесс доказательства корректности программы заключается в разметке дуг информационного графа формулами на языке спецификации, модификации графа и его свёртке. В результате получается несколько информационных графов, у которых все дуги размечены. Каждый из полностью размеченных графов может быть преобразован в формулу на языке логики. Тождественная истинность всех полученных формул свидетельствует о корректности программы. Методы, разработанные для языка Пифагор [13], применимы и к языку со статической типизацией.

Процесс доказательства достаточно трудоёмок, так как требует рассмотрения большого количества различных вариантов графов и преобразований. Поэтому разработаны основные концепции архитектуры инструментального средства для поддержки формальной верификации программ на языке ФПП программирования [14]. Система получает на вход информационный граф программы и формулы предусловия и постусловия на языке спецификации. Она находит неразмеченные дуги графа и помогает с выбором аксиом и теорем, необходимых для их разметки. Весь процесс доказательства представляется в виде дерева, каждый узел которого является частично размеченным графом. Построение дерева завершается, когда все его листья содержат полностью размеченные информационные графы программы. После этого для каждого графа из листа генерируется формула на языке логики. Если все формулы тождественно истинны, то программа корректна.

ЗАКЛЮЧЕНИЕ

Наличие статической типизации в языке функционально-потокового параллельного программирования обеспечивает более строгий контроль данных, что повышает надежность разрабатываемых программ. Также повышается возможности по проведению более полной оптимизации и формальной верификации. Помимо этого трансформация функционально-потоковых параллельных программ в традиционные языки параллельного программирования становится более простой и эффективной, так как большинство типов данных используют практически однозначное отображение.

Исследование выполнено при финансовой поддержке РФФИ в рамках проекта № 17-07-00288.

СПИСОК ЛИТЕРАТУРЫ

1. *Левин И.И., Дордопуло А.И., Гудков В.А.* Программирование реконфигурируемых вычислительных узлов на языке COLAMO. Учебное пособие. Таганрог: Изд-во ТТИ ЮФУ, 2011. 114 с.

2. *Дордопуло А.И., Левин И.И.* Ресурсонезависимое программирование гибридных реконфигурируемых вычислительных систем // Суперкомпьютерные дни в России: Труды международной конференции (25–26 сентября 2017 г., г. Москва). М.: Изд-во МГУ, 2017. С. 714–723.

3. *Kasyanov V.* Sisal 3.2: functional language for scientific parallel programming // Enterp. Inf. Syst. 2013. V. 7. No 2. P. 227–236.

4. *Легалов А.И.* Функциональный язык для создания архитектурно-независимых параллельных программ // Вычислительные технологии. 2005. № 1 (10). С. 71–89.

5. *Legalov A.I., Vasilyev V.S., Matkovskii I.V., Ushakova M.S.* A Toolkit for the Development of Data-Driven Functional Parallel Programmes // Parallel Computational Technologies. PCT 2018. Communications in Computer and Information Science, vol 910. Springer, Cham. P. 16–30.

6. *Vasilev V.S., Legalov A.I.* Loop-invariant Optimization in the Pifagor Language // Automatic Control and Computer Sciences, 2018. V. 52. No 7. P. 843-849.

7. *Ushakova M.S., Legalov A.I.* Verification of Programs with Mutual Recursion in Pifagor Language // Automatic Control and Computer Sciences, 2018. V. 52. No 7. P. 850–866.

8. *Удалова Ю.В., Легалов А.И., Сиротинина Н.Ю.* Методы отладки и верификации функционально-потоковых параллельных программ // Журнал Сибирского федерального университета. Серия «Техника и технологии». Апрель 2011 (том 4, номер 2). С. 213–224.

9. *Legalov A.I., Legalov I.A., Matkovsky I.V.* Instrumental support of the evolutionary expansion of programs using a incremental development // 20th Conf. Scientific Services and Internet, SSI 2018; Novorossiysk-Abrau; Russian Federation; 17–22 September 2018. CEUR Workshop Proc. V. 2260. 2018. P. 346–359.

10. *Легалов А.И., Редькин А.В., Матковский И.В.* Функционально-потоковое параллельное программирование при асинхронно поступающих данных // Параллельные вычислительные технологии (ПаВТ'2009): Труды международной научной конференции, Нижний Новгород, 30 марта – 3 апреля 2009 г. Челябинск: Изд. ЮУрГУ, 2009. С. 573–578.

11. *Легалов А.И.* Об управлении вычислениями в параллельных системах и языках программирования // Научный вестник НГТУ. 2004. № 3 (18). С. 63–72.

12. *Hoare C.A.R.* An axiomatic basis for computer programming // Communications of the ACM. 1969. V. 10. No 12. P. 576–585.

13. *Kropacheva M., Legalov A.* Formal Verification of Programs in the Pifagor Language // Parallel Computing Technologies, 12th International Confernce PACT September-October, 2013. St. Petersburg, Russia. Lecture Notes in Computer Science 7979, Springer, 2013. P. 80–89.

14. *Ushakova M.S., Legalov A.I.* Automation of Formal Verification of Programs in the Pifagor Language // Modeling and Analysis of Information Systems. 2015. V. 22. No 4. P. 578–589.

EVOLUTION OF VISUALIZATION METHODS FOR RESEARCH PUBLICATION COLLECTIONS

A. I. Legalov, I. A. Legalov, I. V. Matkovsky

Siberian Federal University, Krasnoyarsk

legalov@mail.ru

Abstract

It is proposed to add a static system of types to the dataflow functional model of parallel computing and the dataflow functional parallel programming language developed on its basis. The use of static typing increases the possibility of transforming dataflow functional parallel programs into programs running on modern parallel computing systems. Language constructions are proposed. Their syntax and semantics are described. It is noted that the need to use the single assignment principle in the formation of data storages of a particular type. The features of instrumental support of the proposed approach are considered.

Keywords*: visualization of document collections, text analysis, text and metadata visualization algorithms, LDA, NMF, word2vec*

REFERENCES

1. *Levin I.I., Dordopulo A.I., Gudkov V.A.* Programming reconfigurable computing nodes in the COLAMO language. Study guide. Taganrog: Publishing TTI of South Federal Univ., 2011. 114 p.

2. *Dordopulo A.I., Levin I.I.* Resource-independent programming of hybrid reconfigurable computing systems // Supercomputer days in Russia: Proceedings of the international conference (September 25–26, 2017, Moscow). M.: Publishing of Moscow State University. 2017. P. 714–723.

3. *Kasyanov V.* Sisal 3.2: functional language for scientific parallel programming // Enterp. Inf. Syst. 2013. V. 7. No 2. P. 227–236.

4. *Legalov A.I.* Functional language for architecture-independent programming // Computation technologies. 2005. № 1 (10). P. 71–89.

5. *Legalov A.I., Vasilyev V.S., Matkovskii I.V., Ushakova M.S.* A Toolkit for the Development of Data-Driven Functional Parallel Programmes // Parallel

Computational Technologies. PCT 2018. Communications in Computer and Information Science, vol 910. Springer, Cham. P. 16–30.

6. *Vasilev V.S., Legalov A.I.* Loop-invariant Optimization in the Pifagor Language // Automatic Control and Computer Sciences, 2018. V. 52. No 7. P. 843-849.

7. *Ushakova M.S., Legalov A.I.* Verification of Programs with Mutual Recursion in Pifagor Language // Automatic Control and Computer Sciences, 2018. V. 52. No. 7. P. 850–866.

8. *Udalova J., Legalov A., Sirotinina N.* Debug and verification of function-stream parallel programs // J. of SibFU. Engineering & Technologies. 2011. V. 4. No 2. P. 213–224.

9. *Legalov A.I., Legalov I.A., Matkovsky I.V.* Instrumental support of the evolutionary expansion of programs using a incremental development // 20th Conf. Scientific Services and Internet, SSI 2018; Novorossiysk-Abrau; Russian Federation; 17–22 September 2018. CEUR Workshop Proc. V. 2260. 2018. P. 346–359.

10. *Legalov A.I., Redkin A.V., Matkovsky I.V.* Dataflow Functional Parallel Programming using asynchronously incoming data // Parallel Computing Technologiws (PCT'2009): International Conf. 2009. P. 573–578.

11. *Legalov A.I.* About computation control in parallel system and programming languages // NGTU's Science Messenger. 2004. No 3 (18). P. 63–72.

12. *Hoare C.A.R.* An axiomatic basis for computer programming // Communications of the ACM. 1969. V. 10. No 12. P. 576–585.

13. *Kropacheva M., Legalov A.* Formal Verification of Programs in the Pifagor Language // Parallel Computing Technologies, 12th International Confernce PACT September-October, 2013. St. Petersburg, Russia. Lecture Notes in Computer Science 7979, Springer. 2013. P. 80–89.

14. *Ushakova M.S., Legalov A.I.* Automation of Formal Verification of Programs in the Pifagor Language // Modeling and Analysis of Information Systems. 2015. V. 22. No 4. P. 578–589.

СВЕДЕНИЯ ОБ АВТОРАХ

ЛЕГАЛОВ Александр Иванович – профессор Сибирского федерального университета. Сфера научных интересов – программная инженерия, системное программирование, параллельное программирование, языки и системы программирования, компиляторы

Alexander Ivanovich LEGALOV – Professor of Siberian Federal University. Research interests include software engineering, system programming, parallel programming, programming languages, compilers.

email: legalov@mail.ru

ЛЕГАЛОВ Игорь Александрович – доцент Сибирского федерального университета. Сфера научных интересов – языки программирования, компиляторы, web программирование.

Igor Alexandrovich LEGALOV – Associate Professor of Siberian Federal University. Research interests include programming languages, compilers, web programming.

email: igor@legalov.ru

МАТКОВСКИЙ Иван Васильевич – старший преподаватель Сибирского федерального университета. Сфера научных интересов – системное программирование, параллельное программирование, языки программирования, компиляторы

Ivan Vasilievich MATKOVSKY – Senior Lecturer of Siberian Federal University. Research interests include system programming, parallel programming, programming languages, compilers.

email: alpha900i@mail.ru

Материал поступил в редакцию 5 ноября 2019 года

УДК 004.822 + 004.021

МЕТОДЫ И АЛГОРИТМЫ ПОВЫШЕНИЯ ВЫРАЗИТЕЛЬНОСТИ СВЯЗАННЫХ ДАННЫХ (ОБЗОР)

О. А. Невзорова[0000-0001-8116-9446]

Казанский федеральный университет, ул. Кремлевская, 18, Казань, 420008
onevzoro@gmail.com

Аннотация

В обзорной статье рассмотрены методы и алгоритмы повышения выразительности связанных данных, подготовленных для публикации в Вебе. Представлены основные подходы к обогащению онтологий, описаны методы, на которых они базируются, а также приведен инструментарий, реализующий эти подходы и инструменты применения соответствующих методов.

Основным этапом в общей схеме жизненного цикла данных в облаке открытых связанных данных является этап построения набора связанных RDF-триплетов. Для улучшения классификации данных и анализа их качества применяются различные методы повышения выразительности связанных данных. Основные идеи рассматриваемых методов связаны с обогащением существующих онтологий (расширением базовой схемы знаний) путем добавления или совершенствования терминологических аксиом. Методы обогащения опираются на методы, применяемые в различных областях, таких как представление знаний, машинное обучение, статистика, обработка текстов на естественном языке, анализ формальных понятий и теория игр.

Ключевые слова*: связанные данные, онтология, обогащение онтологии, семантический веб*

ВВЕДЕНИЕ

Термин Linked Data, или «связанные данные», обозначает широкий набор конкретных методов, подходов и технологий для публикации структурированных данных в Вебе. Процесс публикации технологически обеспечивается: представлением данных в виде триплетов «субъект-предикат-объект» на языке RDF, идентифи-

фикацией данных с помощью URI, механизмом доступа по протоколу HTTP и спецификацией контролируемых словарей, или онтологий на языках RDFS и OWL. Как правило, множества триплетов логически объединяются и хранятся в виде наборов данных (data sets) или баз знаний. Обзор содержит описание методов и алгоритмов повышения выразительности связанных данных в парадигме Linked Open Data (LOD).

Общая схема жизненного цикла данных в LOD содержит следующие этапы:

- извлечение RDF-триплетов из неструктурированных источников (Extraction);
- сохранение RDF-триплетов в постоянном хранилище с обеспечением доступа (Storage/Querying);
- интерактивное создание и редактирование данных (Manual Revision/Authoring);
- обнаружение и связывание идентичных или семантически связанных данных (Interlinking/Fusing);
- улучшение классификации данных (Classification/Enrichment);
- анализ качества данных (Quality Analysis);
- поиск и просмотр данных (Search/Browsing/Exploration).

Методы и алгоритмы повышения выразительности связанных данных используются на этапах анализа качества данных и улучшения классификации данных.

1. МЕТОДЫ ОБОГАЩЕНИЯ ДАННЫХ

Онтологии Linked Data в основном наполнены экземплярами классов, в результате чего наблюдается нехватка классификации и структурной информации [1]. Данная проблема решается путем обогащения существующих онтологий высокоуровневыми структурами для облегчения задач агрегирования данных и построения запросов.

Под термином «обогащение» понимается автоматическое или полуавтоматическое расширение базовой схемы знаний [1]. Этот термин описывает процесс увеличения выразительности и семантического богатства базы знаний. Обычно это достигается путем добавления или совершенствования терминологических аксиом.

Как правило, методы обогащения могут быть массово применены для создания баз знаний. При таком подходе онтологическая структура не создается сразу напрямую, а постепенно эволюционирует вместе с данными в базе знаний. В идеале это способствует более быстрому созданию баз знаний. В частности, в контексте Web of Linked Data такой подход кажется интересной альтернативой традиционным методам разработки онтологий.

Обогащение базы знаний может быть рассмотрено как подпроблема задачи машинного обучения онтологий. Обучение онтологий отличается тем, что при этом возможно использование внешних источников информации, например, письменного текста. В то же время обогащение предполагает анализ уже существующей базы знаний для улучшения ее схемы.

Методы обогащения опираются на методы, применяемые в различных областях, таких как представление знаний, машинное обучение, статистика, обработка текстов на естественном языке, анализ формальных понятий и теория игр.

Далее будут рассмотрены основные подходы к обогащению, описаны методы, на которых они базируются, а также приведен инструментарий, реализующий данные подходы.

2. ОСНОВНЫЕ ПОДХОДЫ К ОБОГАЩЕНИЮ

Обогащение онтологии использует методы машинного обучения и эвристические методы для добавления дополнительных аксиом к существующей онтологии, при этом применение методов зависит от целевого типа аксиом.

Можно выделить следующие основные направления применения методов обогащения:

- нахождение определений классов (definitions of classes);
- построение аксиом надклассов (super class axioms);
- гибридный подход с использованием жестких правил;
- изучение несвязностей (disjointness);
- завершение (completion);
- нахождение параметров свойств (properties of properties);
- построение отображений (ontology mapping).

2.1. Нахождение определений классов

Нахождение определений классов – одна из самых сложных задач обогащения онтологий. Решение этой задачи сильно связано с методами индуктивного логического программирования (Inductive Logic Programming) [2] и требует применения дескрипционной логики.

Разработка проблем обучения в рамках дескрипционной логики начинается с работ [3], [4], в которых использовались так называемые наименьшие общие вышестоящие (least common subsumers) для решения проблемы обучения (несколько модифицированный вариант проблемы нахождения определений классов). Позже в [5] был описан оператор усовершенствования (refinement operator) для дискрипционной логики *ALER* (модификация DL, Attributive language, расширенная некоторыми дополнительными ограничениями), а также предложен к использованию нисходящий (top-down) подход. Оператор усовершенствования – метод индуктивного логического программирования, который используется для поиска в пространстве выражений. Нисходящий подход предполагает применение оператора усовершенствования, начиная с наиболее общего класса, в результате чего строится описание этого класса из отображений его подклассов.

Позже оба вышеописанных подхода были объединены и осуществлены в инструменте YINYANG [6]. Однако эти алгоритмы имеют тенденцию производить описания классов, очень длинные и сложные для понимания. Алгоритмы, используемые в DL-learner [7], преодолевают эту проблему, а также значительно прорабатывают методы обучения и нисходящего усовершенствования (top-down refinement). DL-FOIL [8] – это похожий подход, но он основан на смеси восходящего и нисходящего методов усовершенствования описаний классов. Используются альтернативные меры оценки, которые, в том числе, принимают во внимание предположение об открытости мира, которое не было сделано в ILP ранее.

Другим подходом к обучению определениям именованных классов является вычисление так называемого наиболее определенного понятия (most specific concept) для всех экземпляров класса. Наиболее определенное понятие – это самое определенное описание класса, такого, что объект есть экземпляр описания класса. В таком случае можно вычислить наименьшее общее вышестоящее (least common subsumer) [9] этих выражений, чтобы получить описание класса. Однако

в выразительной дескрипционной логике наиболее определенное понятие может не существовать, и тогда наименьшее общее вышестоящее – просто дизъюнкция всех выражений. Для простых логик, таких как, например, *EL*, подход кажется многообещающим.

Инструментарий для обучения определениям классов

DL-Learner [8]. DL-Learner – приложение обучения концептам дескрипционной логики на основе примеров, что также позволяет обучаться классам в OWL-онтологиях. Приложение предоставляет инструментарий с основами машинного обучения для DL/OWL для решения задач обучения с помощью экспертов и изучения уже имеющихся данных. Возможности:

- Обучение определениям классов: на основе существующих экземпляров OWL-классов DL-Learner делает предположения для определений классов о включении в них owl:equivalentClass или rdfs:subClassOf Axiom. Так как алгоритм смещен в сторону коротких и легко понимаемых человеком предложений, то эксперту может быть оказана помощь при редактировании Tbox онтологии;
- Поиск похожих экземпляров: описания классов, предложенные DL-Learner, могут быть использованы для нахождения похожих экземпляров с помощью поиска. Масштабируемость методов позволяет проводить генерацию рекомендаций на лету, например, в веб-сценарии;
- Классификация экземпляров: полученные во время обучения описания классов могут быть использованы для определения, к какому конкретному классу принадлежат новые добавляемые экземпляры.

DL-FOIL. Это адаптация алгоритма FOIL (First Order Inductive Learner) [10] для обучения DL-представлениям, с поддержкой языка OWL-DL. В основе системы лежит множество операторов улучшения, позаимствованных из аналогичных систем [11], [12] и с различными целевыми функциями, принимающими во внимание предположение об открытости мира.

YINYANG (Yet another INduction Yields to ANother Generalization) [6]. Эта система содержит реализацию методологии управляемого обучения концептам и дескрипционной логики. Система позволяет перепроверить результаты, рассматривая часть из них как новое обучающее множество.

ORE (Ontology Repair and Enrichment) [13]. ORE – инструментарий, позволяющий инженерам по знаниям улучшать OWL-онтологии путем исправления противоречий и добавления новых аксиом. ORE использует фреймворк DL-Learner для добавления новых определений классов и надклассов для классов, уже существующих в базе знаний.

2.2. Построение аксиом надклассов

В [14] применены приспосабливающиеся эвристики и осуществлена адаптация методов обучения к определениям в онтологиях. Работа направлена на достижение максимальной эффективности и применение методов обучения. Эта статья описывает плагины для двух редакторов онтологий (Protege и OntoWiki), а также стохастические методы, улучшающие результаты работы методов машинного обучения на порядок. Кроме того, алгоритмы, представленные в статье, могут быть применены для построения аксиом надклассов.

2.3. Гибридный подход с использованием жестких правил

Ряд подходов, например, [15], разработан для обучения на гибридных базах знаний, объединяющих онтологии и правила. Обычно гибридные подходы – это обобщение стандартных методов обучения, которые допускают использование жестких правил за счет потери эффективности (из-за большей области поиска). Подобно ситуации с представлением знания требуется очень тщательный выбор баланса между выразительностью языка и эффективностью алгоритмов обучения.

Инструментарий для полуавтоматического обогащения онтологий

Blomqvist [16]. Blomqvist предоставляет фреймворк для полуавтоматического построения и обогащения онтологий. Применяется подход, основанный на шаблонах. Система предназначена для обогащения «легких» онтологий путем работы с логической сложностью и выразительностью, а не для получения богатой аксиоматизации. По этой причине шаблоны используются не для нахождения новых аксиом, а для автоматического размещения новых концептов в контексте уже имеющихся более общих сущностей.

2.4. Изучение несвязностей

Исследование, проведенное в работе [17], направлено на изучение несвязностей между классами в онтологии, чтобы более тщательно проверить рассуждения и выводы. Для достижения этой цели могут использоваться как непосредственно сама онтология, так и внешние тексты, например, статьи Wikipedia, соответствующие рассматриваемому понятию. Статья включает обширное исследование, которое показывает, что задача обнаружения несвязностей является трудной, но может быть упрощена путем обогащения онтологии.

2.5. Завершение

Другая задача обогащения – завершение базы знаний. Целью решения такой задачи является приведение базы знаний к законченной форме (в некотором заданном смысле). Например, целью может быть возможность гарантировать, что все отношения включения как подкласса между именованными классами могут быть выведены. В работах [18], [9] исследованы возможности применения формального анализа понятий (FCA) для завершения баз знаний. Этот подход выглядит многообещающим, хотя, возможно, не в состоянии обрабатывать шумы так же хорошо, как методы машинного обучения. Доступно соответствующее дополнение к Protege [19]. Подход, описанный в [20], предлагает обогащать базы знаний путем исследования взаимосвязей в них, что было реализовано в RELExO framework [21]. Подход работает с простыми отношениями и задает эксперту серии вопросов. Эксперт или должен положительно ответить на вопрос, или обеспечить контрпример.

Инструментарий для задачи завершения

RELExO (Relational Exploration for Learning Expressive Ontologies) [21]. RELExO – инструментарий для расширения связей в OWL/DL-онтологиях, который был разработан для поддержки дополнения и совершенствования сложных описаний классов. RELExO комбинирует обучение сложным описаниям классов из текстовых источников с подходом к обработке связей, основанным на FCA, с целью нахождения связей нижестоящих классов для концептов онтологии.

Генерируются гипотезы расширения связей классов, которые не могут быть

выведены или опровергнуты, исходя из аксиом, уже имеющихся в онтологии. После этого осуществляется поиск контрпримеров среди экземпляров онтологии. В случае, если это не удается, запрашивается помощь эксперта, который должен предоставить контрпример или подтвердить предложенную гипотезу.

RoLExO [20]. RoLExO опирается на тот же тип верификации гипотез и взаимодействия с пользователем, что и RELExO, но генерирует гипотезы об области ограничений для данных.

2.6. Нахождение параметров свойств

Существуют более «легковесные» методы обогащения онтологий, чем описанные ранее. Например, параметры свойств могут быть получены путем простого статистического анализа. Под этим подразумевается возможность того или иного свойства быть симметричным, функциональным, рефлексивным, обратно функциональным и т. д. Точно так же области и диапазоны значений свойств могут быть определены по существующим данным.

2.7. Построение отображений онтологий

Весьма популярным подходом к обогащению онтологий является построение онтологических отображений [1]. Часто задачи, связанные с онтологиями, требуют доступа сразу к нескольким из них. Распределенная природа разработки онтологий привела к тому, что существуют различные онтологии одной или нескольких пересекающихся предметных областей. В таком случае значительно затрудняется понимание одной онтологии с точки зрения другой. Именно для разрешения данной проблемы строятся отображения между онтологиями.

Выделяются три основных вида онтологических отображений [22]. Одним из ключевых различий между ними является то, как данное отображение строится и поддерживается. Каждое отображение имеет свои преимущества и недостатки.

Построение отображения между интегрированной и частной онтологиями. В этом случае производится отображение некоторого концепта из одной онтологии в представление или запрос над другими онтологиями. Это отображение используется для поддержки интеграции онтологий в Semantic Web. Из локальных онтологий извлекается информация в интегрированную онтологию Sematic

Web, обеспечивая единое представление, которое позволяет пользователям получать результаты на запросы к локальным онтологиям. Также возможно создание специализированных интегрированных онтологий из нескольких локальных онтологий, например, для нужд управления знаниями крупного предприятия.

Инструментарий:

LSD (Learning Source Description) [23]. Полуавтоматически создаются семантические отображения с применением подхода множественных стратегий для машинного обучения. Этот подход предполагает использование нескольких модулей обучения, при этом каждый модуль использует различную информацию в исходных схемах данных.

LSD использует следующие базовые модули:

- Обучение по именам: сопоставляет XML-элементы, имеющие соответствующий тэг;
- Обучение по содержимому: сопоставляет XML-элементы по данным в схеме (хорошо работает для текстовых данных);
- Наивная байесовская модель: использует значения данных, но не работает для коротких и числовых полей;
- XML-обучение: работает с иерархической структурой входных сущностей.

Множественная стратегия обучения имеет два этапа: подготовка и согласование. На фазе подготовки небольшое количество исходных данных вручную отображаются в промежуточную схему, и эта информация используется для обучения модулей. На фазе согласования система автоматически строит отображения на основе информации, полученной на подготовительной фазе. Также LSD учитывает соответствие ограничений, использует обратную связь с пользователем, вложенные структуры в XML-данных для улучшения точности отображений.

MOMIS (Mediator Environment for Multiple Information Sources) [24]. MOMIS создает глобальные виртуальные представления источников информации, независимо от их происхождения или разнородности данных. Онтология строится за 5 фаз:

- Извлечение схемы из локального источника программным модулем

(wrapper);

- Аннотация локального источника с помощью WordNet;
- Генерация общего тезауруса: отношения внутри и между схемами знаний о классах и атрибутах;
- Генерация глобального виртуального представления: создание глобальной схемы и отображений между атрибутами глобальной и исходных схем с использованием общего тезауруса и схем, построенных на первой фазе;
- Построение аннотаций глобального виртуального представления с использованием аннотации исходных схем и отображения между глобальной и локальными схемами.

MOMIS строит онтологию, которая более точно описывает домены и предоставляет легко понимаемый смысл содержимого, а также способ расширения ранее созданных концепций путем добавления новых источников.

A Framework for OIS (Ontology Integration System) [25]. Отображения между интегрированной онтологией и локальными онтологиями представляются как запросы, а онтологии – как дескрипционные логики. Предлагается два подхода к построению отображений:

- Концепты глобальной онтологии отображаются в запросы к локальным онтологиям;
- Концепты локальных онтологий отображаются в запросы к глобальной онтологии.

Отличием этого подхода является возможность построения отображений глобальной онтологии на локальную, в отличие от систем, описанных выше, которые строят отображение локальной онтологии в представление глобальной.

Построение отображения между двумя частными онтологиями [26]. Сущности одной онтологии преобразуются в сущности другой онтологии, базируясь на семантических отношениях. Исходная и целевая онтологии должны быть связаны на концептуальном уровне.

Данное отображение позволяет онтологиям оставаться привязанными к некоторому контексту, при этом, по-прежнему, оставаясь локальными. Возможно построение связей между онтологиями, когда они не могут быть объединены по причине несовместимости информации, заложенной в них.

Нахождение связей между локальными онтологиями может оказаться не проще, чем между локальной и интегрированной онтологией по причине недостатка общих словарей.

Инструментарий:

Context OWL (Contextualizing Ontologies) [27]. Используются расширенные синтаксис и семантика OWL. Онтологии не могут быть связаны или объединены, если две из них содержат взаимно несовместимые понятия. Однако эти две онтологии могут быть связаны через специальные промежуточные правила, которые содержат базовые понятия о контексте отображения. Такое отображение представляется как множество промежуточных правил, использующих операнды $\supseteq$, $\subseteq$, $\equiv$, $*$(связаны) и $\perp$ (не связаны), например, если k является более общим понятием, чем k1 ($k \supseteq k1$), k – менее общее понятие k1 ($k \subseteq k1$), k эквивалентно k1 ($k \equiv k1$), k сравнимо с k1 ($k * k1$), k несравнимо с k1 ($k \perp k1$).

CTXMATCH [28]. CTXMATCH – это алгоритм обнаружения семантических связей всей иерархической классификации с помощью правил логического вывода. CTXMATCH на входе получает классы из иерархической классификации и для каждой пары концептов этих классов (узлы соответствующих знаний, включая смысл в иерархической классификации) возвращает их семантическую связь ($\supseteq$, $\subseteq$, $\equiv$,$*$, или $\perp$).

Достоинством CTXMTCH является то, что отображениям могут быть приписаны четко определенные модели семантики, и эта структурная, лексическая и контекстная информация принимается в расчет.

GLUE [29]. GLUE полуавтоматически создает связи между онтологиями, используя методы машинного обучения. GLUE находит наиболее схожие концепты между онтологиями и просчитывает распределение совместной вероятности этих концептов, используя обучение с множественными стратегиями для оценки степени близости. Первая стратегия основывается на частоте слов в контексте рассматриваемой сущности (например, объединение всех атрибутов сущности) и использует наивный Байесовский подход. Вторая стратегия основана на полном имени входной сущности, а третья комбинирует предсказания первых двух, назначая им веса в соответствии с оценкой доверия.

MAFRA (Ontology MAapping FRAmework for distributed ontologies in the Semantic Web) [30]**.** MAFRA обеспечивает распределенный процесс построения связей, состоящий из пяти вертикальных и четырех горизонтальных модулей.

Горизонтальные модули:

- Нормализация: работает с языковой и лексической разнородностями между исходной и целевой онтологиями;
- Обнаружение общего: находит и устанавливает похожие сущности обеих онтологий;
- Семантическое связывание: строит отображения для изменяемых сущностей исходной онтологии в наиболее близкие сущности целевой онтологии;
- Исполнение: трансформирует сущности исходной онтологии в целевую, основываясь на ранее построенных семантических связях;
- Пост-обработка: перепроверяет результаты трансформации для улучшения качества результатов.

Вертикальные модули:

- Эволюция: обслуживает семантические связи, синхронизируя связи в обеих онтологиях;
- Совместное достижение консенсуса: отвечает за установление консенсуса в семантических связях между обеими онтологиями;
- Ограничения предметной области: улучшает семантические связи, используя тезаурус WordNet соответствующей предметной области;
- Интерфейс: позволяет вмешательство человека для достижения лучших результатов.

LOM (Lexicon-based Ontology Mapping) [31]. LOM находит автоморфизмы между словарями онтологий с целью уменьшения вклада человеческого труда в процесс построения отображения. Применяются методы, основанные на использовании: всего термина, частей слова, соответствующего синсета WordNet и типа соответствия. LOM не гарантирует точности и аккуратности построенного отображения и имеет существенные ограничения при работе с абстрактными символами или обозначениями в математике, химии или медицине. Преимуществом данного подхода является почти полная автоматизированность.

QOM (Quick Ontology Mapping) [32]**.** QOM – эффективный метод обнаружения связей между онтологиями, так как имеет очень низкую временную сложность. QOM использует подход, основанный на динамическом программировании. Строятся структуры, рассматривающие отображения-кандидаты, классифицируются по перспективности, в результате чего отсекаются наихудшие. Система применяется в тех случаях, когда критично время работы.

ONION (ONtology compositION system) [33]. ONION разрешает терминологическую гетерогенность и предоставляет соединяющие правила для отображений. Лингвистический модуль определяет все возможные пары терминов в онтологиях и приписывает им соответствующие веса близости. Если вес близости превышает определенный порог, то такая пара принимается, и генерируется соответствующее соединяющее правило. После этого дополнительные соответствия ищутся структурным модулем. Построение связей между онтологиями проводится за несколько циклов. Эксперт может удалять, добавлять или изменять соединяющие правила, предложенные системой через специальный интерфейс.

OKMS (Ontology-based knowledge management system) [34]. В OKMS отображения используются для комбинирования либо распределенных, либо гетерогенных онтологий. Процесс осуществляется в 5 шагов:

- Нормализация: если исходная информация не является онтологией, то осуществляется соответствующее преобразование программным модулем;
- Нахождение близких сущностей: генерируется матрица, содержащая меры близостей сущностей разных онтологий;
- Построение семантического отображения: на этом шаге генерируется отображение, описывающее преобразование концептов исходной онтологии в концепты целевой;
- Исполнение: выполнение преобразований семантического отображения;
- Пост-процессинг: дополнительная обработка для улучшения качества результатов.

OMEN (Ontology Mapping Enhancer) [35]. OMEN – это вероятностный инструментарий построения отображений онтологий, который улучшает качество уже

построенного отображения, используя Байесовские сети. Байесовские сети используют множество мета-правил, представляющих то, как каждое отображение онтологии влияет на другие отображения, базируясь на структуре онтологии и семантике онтологических связей. Существующие отображения между двумя концептами могут быть использованы для улучшения отображений между соседними концептами.

P2P ontology mapping (Peer-to-peer ontology mapping) [36]. Данный фреймворк создает эффективное взаимодействие между онтологиями, участвующими в построении отображения, базируясь на динамических отображениях только части онтологии, релевантной данному взаимодействию. Работа осуществляется в три шага:

- Генерация гипотез;
- Фильтрация гипотез;
- Выбор лучшей гипотезы.

Отображение онтологий в их объединение [37]. Устанавливается соответствие между исходными онтологиями для объединения или согласования. При этом определяются пересекающиеся концепты, синонимы, аналоги и уникальные записи в этих двух онтологиях. Выявляются общие структуры и конфликты локальных онтологий перед их сведением или объединением.

Инструментарий:

SMART [38]. SMART – это инструментарий полуавтоматического объединения и сведения онтологий. Он находит лингвистически близкие имена классов и, основываясь на этих эквивалентностях, создает список начальной лингвистической близости (синонимия, единая подстрока, общий суффикс или префикс). Изучается структура отношений между объединенными концептами и строятся эквивалентности между именами свойств и их типами. Инструментарий проверяет результат работы на возможные конфликты, а также предлагает возможные пути их разрешения пользователю.

OntoMorph [39]. OntoMorph предоставляет мощный язык правил для определения отображений, способствуя слиянию онтологий и быстрой генерации

базы знаний преобразования. Он комбинирует два мощных механизма для трансформации баз знаний: синтаксическое и семантическое преобразования. Синтаксическое преобразование осуществляется по специальным шаблонам на уровне предложений, семантическое – на основе семантических моделей и правил логического вывода.

HICAL (Hierarchical Concept Alignment system) [40]. HICAL предоставляет управление иерархией концептов для объединения/сведения онтологий (одна иерархия концептов сводится к другой), используя методы машинного обучения для сведения множества иерархий концептов, а также использует данные о сущностях в пересечении таксономий для выведения отображений. Иерархия используется для категоризации и получения синтаксической информации, а не вычисления близости между словами, что позволяет категорировать различные слова в рамках одного концепта.

Anchor-PROMPT [41]. Система Anchor-PROMPT берет пары связных терминов из исходных онтологий и проходит по кратчайшему пути, связывающему эти пары. Проводится сравнение терминов на этом пути с целью определения одинаковых терминов и последующей генерации множества пар семантически идентичных концептов.

CMS (CROSI Mapping System) [42]. CMS – это система сведения онтологий, основанная на структурном сравнении и использующая богатую семантику OWL-конструкций. Модульная архитектура системы позволяет использовать внешние лингвистические ресурсы, а также множественные стратегии при нахождении меры близости сущностей онтологий.

FCA-Merge [43]. FCA-Merge – это метод слияния онтологий, базирующийся на анализе формальных концептов Гантера и Вилли [44]. Весь процесс состоит из трех шагов:

- Извлечение сущностей и генерация формального контекста для каждой онтологии;
- Вычисление решетки концептов с помощью алгоритма TITANIC [45];
- Ручная генерация единой онтологии путем взаимодействия с пользователем, базирующаяся на ранее построенной решетке концептов.

CHIMAERA [46]. CHIMAERA – это интерактивный инструментарий для объединения онтологий, а также онтолого-лингвистический редактор. Он позволяет вмешательство пользователя в любой момент в течение процесса объединения. Если находятся лингвистические соответствия, то они обрабатываются автоматически, в противном же случае – принять решение предлагается пользователю.

ЗАКЛЮЧЕНИЕ

Обогащение онтологий является широко изученной проблемой. Ведется разработка многочисленных подходов, направленных на обогащение различных видов аксиом онтологий. Наиболее разработанными направлениями являются нахождение определений классов и построение отображений онтологий.

Выбор того или иного инструментария в значительной степени зависит от того, какую именно задачу предполагается решать.

Для задачи нахождения определений классов наиболее разработанным инструментом является ORE. Дополнительным преимуществом этой системы является наличие плагинов к системам Protege и OntoWiki.

Для выбора инструментария для решения задачи построения отображения между интегрированной и частной онтологиями требуются дополнительные исследования. Представленные системы используют различные подходы, и их эффективность зависит от онтологий, между которыми строится отображение.

Рассматривая задачу построения отображения между двумя частными онтологиями, можно выделить LOM как систему, требующую меньшего вклада труда экспертов, и QOM по причине низкой ресурсоемкости по времени используемого алгоритма.

Благодарности

Исследование выполнено за счет гранта Российского научного фонда, проект № 19-71-10056.

СПИСОК ЛИТЕРАТУРЫ

1. *Auer S., Lehmann J., Ngonga-Ngomo A.-C.* Introduction to Linked Data and Its Lifecycle on the Web // Reasoning Web 2011. Lecture Notes in Computer Science. 2011. V. 6848. Springer, Heidelberg. P. 1–75.

2. *Nienhuys-Cheng S.-H., de Wolf R.* Foundations of Inductive Logic Programming // Lecture Notes in Computer Science. V. 1228. Springer, Heidelberg, 1997. 248 p.

3. *Cohen W.W., Borgida A., Hirsh H.* // Computing Least Common Subsumers in Expressive Description Logics. In: Foo N. (eds). Advanced Topics in Artificial Intelligence. AI 1999. Lecture Notes in Computer Science. Vol. 1747. Springer, Berlin, Heidelberg. P. 754–760.

4. *Cohen W.W., Hirsh H.* Learning the CLASSIC description logic: Theoretical and experimental results // Proceedings of the 4th International Conference on the Principles of Knowledge Representation and Reasoning (KR-94). San Francisco: Morgan Kaufmann, 1994. P. 121–133.

5. *Badea L., Nienhuys-Cheng S.-H.* A refinement operator for description logics // In: Cussens J., Frisch A.M. (eds). Inductive Logic Programming: 10th International Conference, ILP 2000. Lecture Notes in Computer Science (LNAI). 2000. Vol. 1866. Springer, Heidelberg. P. 40–59.

6. *Esposito F., Fanizzi N., Iannone L.* Knowledge-intensive induction of terminologies from metadata // Proceedings of The Third International Semantic Web Conference Proceedings, ISWC 2004. Lecture Notes in Computer Science. 2004. V. 3298. Springer-Verlag, Heidelberg. P. 411–426.

7. *Lehmann J., Hitzler P.* Foundations of Refinement Operators for Description Logics // In: Blockeel H., Ramon J., Shavlik J., Tadepalli P. (eds). Inductive Logic Programming: 17th International Conference, ILP 2007. Lecture Notes in Computer Science (LNAI). 2008. Vol. 4894. Springer, Heidelberg. P. 161–174.

8. *Fanizzi N., d'Amato C., Esposito F.* DL-FOIL Concept Learning in Description Logics // Inductive Logic Programming: 18th International Conference, ILP 2008. Lecture Notes in Computer Science. 2008. V. 5194. Springer, Heidelberg. P. 107–121.

9. *Baader F., Sertkaya B., Turhan A.Y.* Computing the Least Common Subsumer w.r.t. a Background Terminology. In: Alferes J.J., Leite J. (eds). Logics in

Artificial Intelligence. JELIA 2004. Lecture Notes in Computer Science. 2004. Vol. 3229. Springer, Berlin, Heidelberg. P. 400–412.

10. *Quinlan J.R.* Learning Logical Definitions from Relations // Machine Learning. 1990. V. 5. P. 239–266.

11. *Iannone L., Palmisano I., Fanizzi N.* An algorithm based on counterfactuals for concept learning in the Semantic Web // Applied Intelligence. 2007. Vol. 26. P. 139–159.

12. *Lehmann J., Hitzler P.* A refinement operator based learning algorithm for the ALC description logic // In: Blockeel H., Ramon J., Shavlik J., Tadepalli P. (eds). Inductive Logic Programming: 17th International Conference, ILP 2007. Lecture Notes in Computer Science (LNAI). 2008. Vol. 4894. Springer, Heidelberg. P. 147–160.

13. *Lehmann J., Bühmann L.* ORE – A Tool for Repairing and Enriching Knowledge Bases // In: Patel-Schneider P.F. et al. (eds) The Semantic Web – ISWC 2010. ISWC 2010. Lecture Notes in Computer Science. 2010. Vol. 6497. Springer, Berlin, Heidelberg. P. 177–193.

14. *Lehmann J., Auer S., Bühmann L*. Class expression learning for ontology engineering // Journal of Web Semantics. 2011. Vol. 9. P. 71–81.

15. *Lisi F.A.* Building rules on top of ontologies for the semantic web with inductive logic programming // Theory and Practice of Logic Programming. 2008. Vol. 8(3). P. 271–300.

16. *Blomqvist E.* Semi-automatic Ontology Construction based on Patterns. Ph.D. thesis. Linkoping University. 2009.

17. *Volker J., Vrandecic D., Sure Y.* Learning Disjointness // In: Franconi E., Kifer M., May W. (eds). 4th European Semantic Web Conference, ESWC 2007. Lecture Notes in Computer Science (LNAI). 2007. Vol. 4894. Springer, Heidelberg. P. 175–189.

18. *Rudolph S.* Exploring relational structures via FLE // In: Wolff K.E., Pfeiffer H.D., Delugach H.S. (eds) // 12th International Conference on Conceptual Structures, ICCS 2004. Lecture Notes in Computer Science (LNAI). 2004. Vol. 3127. Springer, Heidelberg. P. 196–212.

19. *Sertkaya B.* OntocomP system description // In: Grau B.C., Horrocks I., Motik B., Sattler U. (eds.) Proceedings of the 22nd International Workshop on

Description Logics (DL 2009), Oxford, UK, July 27-30. CEUR Workshop Proceedings. Vol. 477. CEUR-WS.org (2009).

20. *Völker J., Rudolph S.* Fostering Web Intelligence by Semi-automatic OWL Ontology Refinement // IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Sydney, NSW. 2008. P. 454–460. doi: 10.1109/WIIAT.2008.36.

21. RELExO (Relational Exploration for Learning Expressive Ontologies). URL: http://code.google.com/p/relexo/.

22. *Choi N., Song I.-Y., Han H.* A survey on ontology mapping // SIGMOD Record. 2006. Vol. 35(3). P. 34–41.

23. *Doan A., Domingos P., Halevy A.* Learning to Match the Schemas of Data Sources: A Multistrategy Approach // Machine Learning. 2003. Vol. 50 (3). P. 279–301.

24. *Beneventano D., Bergamaschi S., Guerra F.* Synthesizing an Integrated Ontology // IEEE Internet Computing. 2003. Vol. 7(5). P.42–51.

25. *Calvanese D., De Giacomo G., Lenzerini M.* A Framework for Ontology Integration // Proceedings of the 1st International Semantic Web Working Symposium (SWWS). 2001. P. 303–317.

26. *Silva N., Rocha J.* Ontology Mapping for Interoperability in Semantic Web // Proceedings of the IADIS International Conference WWW/Internet 2003, ICWI 2003. 2003. P. 603–610.

27. *Bouquet P., Giunchiglia F., van Harmelen F.* C-OWL: Contextualizing Ontologies // The Semantic Web – ISWC 2003, Second International Semantic Web Conference, 2003. Lecture Notes in Computer Science. 2003. Vol. 2870. P. 164–179.

28. *Bouquet P., Serafini L., Zanobini S.* Semantic Coordination: A New Approach and an Application // The Semantic Web - ISWC 2003, Second International Semantic Web Conference, 2003. Lecture Notes in Computer Science. 2003. Vol. 2870. P.130–145.

29. *Doan A., Madhavan J., Dhamankar R. et al.* Learning to match ontologies on the Semantic Web // VLDB. 2003. Vol. 12. P. 303–319. https://doi.org/10.1007/s00778-003-0104-2.

30. *Silva N., Rocha J.* Semantic Web complex ontology mapping // Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003).2003. P. 82-88. doi: 10.1109/WI.2003.1241177.

31. *Li J.* LOM: A Lexicon-based Ontology Mapping Tool // Proceedings of the Performance Metrics for Intelligent Systems. 2004. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.5379&rep=rep1&type=pdf

32. *Ehrig M., Staab S.* QOM – Quick OntologyMapping // Proceedings of The Third International Semantic Web Conference Proceedings, ISWC 2004. Lecture Notes in Computer Science. 2004. Vol. 3298. P. 683–697.

33. *Mitra P., Wiederhold G.* Resolving Terminological Heterogeneity in Ontologies // Proceedings of the ECAI'02 workshop on Ontologies and Semantic Interoperability. 2002. P. 45–50.

34. *Maedche A., Motik B., Stojanovic L., Studer R., Volz R.* Ontologies for Enterprise Knowledge Management // IEEE Intelligent Systems. 2003. Vol. 18(2). P. 26–33.

35. *Mitra P., Noy N. F., Jaiswals A.* OMEN: A Probabilistic Ontology Mapping Tool // Proceedings of International Semantic Web Conference, ISWC- 2005. 2005. P. 537–547.

36. *Besana P., Robertson D., Rovatsos M.* Exploiting interaction contexts in P2P ontology mapping // Proceedings of 2nd International Workshop on Peer to Peer Knowledge Management. 2005. CEUR Workshop Proceedings. P. 1613–1673. CEUR-WS.org/Vol-139/2.pdf.

37. *Noy N.F., Musen M.A.* PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment // Proceedings of the National Conference on Artificial Intelligence (AAAI/IAAI). 2000. P.450–455.

38. *Noy N.F., Musen M.A.* Smart: Automated Support for Oontology Merging and Alignment // Proceedings of the 12th Workshop on Knowledge, Acquistion Modeling and Management (IKAW 1999). 1999. P. 1–20.

39. *Chalupsky H.* Ontomorph: A Translation System for Symbolic Knowledge // Proceedings of the 7th international Conference on Principles of Knowledge Representation and Reasoning. 2000. P. 471–482.

40. *Ichise R., Takeda H., Honiden S.* Rule Induction for Concept Hierarchy Alignment // Proceedings of the Workshop on Ontology Learning at the 17th International Joint Conference on Artificial Intelligence (IJCAI). 2001. http://ceur-ws.org/Vol-38/ichise_IJICAI-OL.pdf.

41. *Noy N.F., Musen M.A.* Anchor-PROMPT: Using Non-Local Context for Semantic Matching // Proceedings of the Workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI). 2001. http://dit.unitn.it/~accord/RelatedWork/Matching/noy.pdf.

42. *Kalfoglou Y., Hu B.* CROSI Mapping System (CMS) Results of the 2005 Ontology Alignment Contest // Proceedings of the K-CAP 2005 Workshop of Integrating Ontologies. 2005. P. 77–85.

43. *Stumme G., Maedche A.* FCA-Merge: Bottom-Up Merging of Ontologies // Proceeding of the International Joint Conference on Artificial Intelligence IJCAI-01. Seattle: 2001. P. 205–234.

44. *Ganter B., Wille R.* Formal Concept Analysis. Mathematical Foundations. Berlin: Springer, 1999. 284 p.

45. Titanic: Machine Learning Algorithms. URL: https://www.kaggle.com/berhag/titanic-machine-learning-algorithms.

46. *McGuinness D., Fikes R., Rice J., Wilder S.* The Chimaera Ontology Environment // Proceedings of the 17th National Conference on Artificial Intelligence (AAAI). 2000. P. 1123–1124.

METHODS AND ALGORITHMS FOR INCREASING LINKED DATA EXPRESSIVENESS (OVERVIEW)

O. A. Nevzorova

Kazan Federal University

onevzoro@gmail.com

Abstract

This review discusses methods and algorithms for increasing linked data expressiveness which are prepared for Web publication. The main approaches to the enrichment of ontologies are considered, the methods on which they are based and the tools for implementing the corresponding methods are described.

The main stage in the general scheme of the related data life cycle in a cloud of Linked Open Data is the stage of building a set of related RDF- triples. To improve the classification of data and the analysis of their quality, various methods are used to increase the expressiveness of related data. The main ideas of these methods are concerned with the enrichment of existing ontologies (an expansion of the basic scheme of knowledge) by adding or improving terminological axioms. Enrichment methods are based on methods used in various fields, such as knowledge representation, machine learning, statistics, natural language processing, analysis of formal concepts, and game theory.

Keywords*: linked data, ontology, ontology enrichment, semantic web*

REFERENCES

1. *Auer S., Lehmann J., Ngonga-Ngomo A.-C.* Introduction to Linked Data and Its Lifecycle on the Web // Reasoning Web 2011. Lecture Notes in Computer Science. 2011. V. 6848. Springer, Heidelberg. P. 1–75.

2. *Nienhuys-Cheng S.-H., de Wolf R.* Foundations of Inductive Logic Programming // Lecture Notes in Computer Science. V. 1228. Springer, Heidelberg, 1997. 248 p.

3. *Cohen W.W., Borgida A., Hirsh H.* // Computing Least Common Subsumers in Expressive Description Logics. In: Foo N. (eds). Advanced Topics in Artificial

Intelligence. AI 1999. Lecture Notes in Computer Science. Vol. 1747. Springer, Berlin, Heidelberg. P. 754–760.

4. *Cohen W.W., Hirsh H.* Learning the CLASSIC description logic: Theoretical and experimental results // Proceedings of the 4th International Conference on the Principles of Knowledge Representation and Reasoning (KR-94). San Francisco: Morgan Kaufmann, 1994. P. 121–133.

5. *Badea L., Nienhuys-Cheng S.-H.* A refinement operator for description logics // In: Cussens J., Frisch A.M. (eds). Inductive Logic Programming: 10th International Conference, ILP 2000. Lecture Notes in Computer Science (LNAI). 2000. Vol. 1866. Springer, Heidelberg. P. 40–59.

6. *Esposito F., Fanizzi N., Iannone L.* Knowledge-intensive induction of terminologies from metadata // Proceedings of The Third International Semantic Web Conference Proceedings, ISWC 2004. Lecture Notes in Computer Science. 2004. V. 3298. Springer-Verlag, Heidelberg. P. 411–426.

7. *Lehmann J., Hitzler P.* Foundations of Refinement Operators for Description Logics // In: Blockeel H., Ramon J., Shavlik J., Tadepalli P. (eds). Inductive Logic Programming: 17th International Conference, ILP 2007. Lecture Notes in Computer Science (LNAI). 2008. Vol. 4894. Springer, Heidelberg. P. 161–174.

8. *Fanizzi N., d'Amato C., Esposito F.* DL-FOIL Concept Learning in Description Logics // Inductive Logic Programming: 18th International Conference, ILP 2008. Lecture Notes in Computer Science. 2008. V. 5194. Springer, Heidelberg. P. 107–121.

9. *Baader F., Sertkaya B., Turhan A.Y.* Computing the Least Common Subsumer w.r.t. a Background Terminology. In: Alferes J.J., Leite J. (eds). Logics in Artificial Intelligence. JELIA 2004. Lecture Notes in Computer Science. 2004. Vol. 3229. Springer, Berlin, Heidelberg. P. 400–412.

10. *Quinlan J.R.* Learning Logical Definitions from Relations // Machine Learning. 1990. V. 5. P. 239–266.

11. *Iannone L., Palmisano I., Fanizzi N.* An algorithm based on counterfactuals for concept learning in the Semantic Web // Applied Intelligence. 2007. Vol. 26. P. 139–159.

12. *Lehmann J., Hitzler P.* A refinement operator based learning algorithm for the ALC description logic // In: Blockeel H., Ramon J., Shavlik J., Tadepalli P. (eds).

Inductive Logic Programming: 17th International Conference, ILP 2007. Lecture Notes in Computer Science (LNAI). 2008. Vol. 4894. Springer, Heidelberg. P. 147–160.

13. *Lehmann J., Bühmann L.* ORE – A Tool for Repairing and Enriching Knowledge Bases // In: Patel-Schneider P.F. et al. (eds) The Semantic Web – ISWC 2010. ISWC 2010. Lecture Notes in Computer Science. 2010. Vol. 6497. Springer, Berlin, Heidelberg. P. 177–193.

14. *Lehmann J., Auer S., Bühmann L*. Class expression learning for ontology engineering // Journal of Web Semantics. 2011. Vol. 9. P. 71–81.

15. *Lisi F.A.* Building rules on top of ontologies for the semantic web with inductive logic programming // Theory and Practice of Logic Programming. 2008. Vol. 8(3). P. 271–300.

16. *Blomqvist E.* Semi-automatic Ontology Construction based on Patterns. Ph.D. thesis. Linkoping University. 2009.

17. *Volker J., Vrandecic D., Sure Y.* Learning Disjointness // In: Franconi E., Kifer M., May W. (eds). 4th European Semantic Web Conference, ESWC 2007. Lecture Notes in Computer Science (LNAI). 2007. Vol. 4894. Springer, Heidelberg. P. 175–189.

18. *Rudolph S.* Exploring relational structures via FLE // In: Wolff K.E., Pfeiffer H.D., Delugach H.S. (eds) // 12th International Conference on Conceptual Structures, ICCS 2004. Lecture Notes in Computer Science (LNAI). 2004. Vol. 3127. Springer, Heidelberg. P. 196–212.

19. *Sertkaya B.* OntocomP system description // In: Grau B.C., Horrocks I., Motik B., Sattler U. (eds.) Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30. CEUR Workshop Proceedings. Vol. 477. CEUR-WS.org (2009).

20. *Völker J., Rudolph S.* Fostering Web Intelligence by Semi-automatic OWL Ontology Refinement // IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Sydney, NSW. 2008. P. 454–460. doi: 10.1109/WIIAT.2008.36.

21. RELExO (Relational Exploration for Learning Expressive Ontologies). URL: http://code.google.com/p/relexo/.

22. *Choi N., Song I.-Y., Han H.* A survey on ontology mapping // SIGMOD Record. 2006. Vol. 35(3). P. 34–41.

23. *Doan A., Domingos P., Halevy A.* Learning to Match the Schemas of Data Sources: A Multistrategy Approach // Machine Learning. 2003. Vol. 50 (3). P. 279–301.

24. *Beneventano D., Bergamaschi S., Guerra F.* Synthesizing an Integrated Ontology // IEEE Internet Computing. 2003. Vol. 7(5). P.42–51.

25. *Calvanese D., De Giacomo G., Lenzerini M.* A Framework for Ontology Integration // Proceedings of the 1st International Semantic Web Working Symposium (SWWS). 2001. P. 303–317.

26. *Silva N., Rocha J.* Ontology Mapping for Interoperability in Semantic Web // Proceedings of the IADIS International Conference WWW/Internet 2003, ICWI 2003. 2003. P. 603–610.

27. *Bouquet P., Giunchiglia F., van Harmelen F.* C-OWL: Contextualizing Ontologies // The Semantic Web – ISWC 2003, Second International Semantic Web Conference, 2003. Lecture Notes in Computer Science. 2003. Vol. 2870. P. 164–179.

28. *Bouquet P., Serafini L., Zanobini S.* Semantic Coordination: A New Approach and an Application // The Semantic Web - ISWC 2003, Second International Semantic Web Conference, 2003. Lecture Notes in Computer Science. 2003. Vol. 2870. P.130–145.

29. *Doan A., Madhavan J., Dhamankar R. et al.* Learning to match ontologies on the Semantic Web // VLDB. 2003. Vol. 12. P. 303–319. https://doi.org/10.1007/s00778-003-0104-2.

30. *Silva N., Rocha J.* Semantic Web complex ontology mapping // Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003).2003. P. 82-88. doi: 10.1109/WI.2003.1241177.

31. *Li J.* LOM: A Lexicon-based Ontology Mapping Tool // Proceedings of the Performance Metrics for Intelligent Systems. 2004. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.5379&rep=rep1&type=pdf

32. *Ehrig M., Staab S.* QOM – Quick OntologyMapping // Proceedings of The Third International Semantic Web Conference Proceedings, ISWC 2004. Lecture Notes in Computer Science. 2004. Vol. 3298. P. 683–697.

33. *Mitra P., Wiederhold G.* Resolving Terminological Heterogeneity in Ontologies // Proceedings of the ECAI'02 workshop on Ontologies and Semantic Interoperability. 2002. P. 45–50.

34. *Maedche A., Motik B., Stojanovic L., Studer R., Volz R.* Ontologies for Enterprise Knowledge Management // IEEE Intelligent Systems. 2003. Vol. 18(2). P. 26–33.

35. *Mitra P., Noy N. F., Jaiswals A.* OMEN: A Probabilistic Ontology Mapping Tool // Proceedings of International Semantic Web Conference, ISWC- 2005. 2005. P. 537–547.

36. *Besana P., Robertson D., Rovatsos M.* Exploiting interaction contexts in P2P ontology mapping // Proceedings of 2nd International Workshop on Peer to Peer Knowledge Management. 2005. CEUR Workshop Proceedings. P. 1613–1673. CEUR-WS.org/Vol-139/2.pdf.

37. *Noy N.F., Musen M.A.* PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment // Proceedings of the National Conference on Artificial Intelligence (AAAI/IAAI). 2000. P.450–455.

38. *Noy N.F., Musen M.A.* Smart: Automated Support for Oontology Merging and Alignment // Proceedings of the 12th Workshop on Knowledge, Acquistion Modeling and Management (IKAW 1999). 1999. P. 1–20.

39. *Chalupsky H.* Ontomorph: A Translation System for Symbolic Knowledge // Proceedings of the 7th international Conference on Principles of Knowledge Representation and Reasoning. 2000. P. 471–482.

40. *Ichise R., Takeda H., Honiden S.* Rule Induction for Concept Hierarchy Alignment // Proceedings of the Workshop on Ontology Learning at the 17th International Joint Conference on Artificial Intelligence (IJCAI). 2001. http://ceur-ws.org/Vol-38/ichise_IJICAI-OL.pdf.

41. *Noy N.F., Musen M.A.* Anchor-PROMPT: Using Non-Local Context for Semantic Matching // Proceedings of the Workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI). 2001. http://dit.unitn.it/~accord/RelatedWork/Matching/noy.pdf.

42. *Kalfoglou Y., Hu B.* CROSI Mapping System (CMS) Results of the 2005 Ontology Alignment Contest // Proceedings of the K-CAP 2005 Workshop of Integrating Ontologies. 2005. P. 77–85.

43. *Stumme G., Maedche A.* FCA-Merge: Bottom-Up Merging of Ontologies // Proceeding of the International Joint Conference on Artificial Intelligence IJCAI-01. Seattle: 2001. P. 205–234.

44. *Ganter B., Wille R.* Formal Concept Analysis. Mathematical Foundations. Berlin: Springer, 1999. 284 p.

45. Titanic: Machine Learning Algorithms. URL: https://www.kaggle.com/berhag/titanic-machine-learning-algorithms.

46. *McGuinness D., Fikes R., Rice J., Wilder S.* The Chimaera Ontology Environment // Proceedings of the 17th National Conference on Artificial Intelligence (AAAI). 2000. P. 1123–1124.

СВЕДЕНИЯ ОБ АВТОРЕ

НЕВЗОРОВА Ольга Авенировна – доцент кафедры информационных систем Института вычислительной математики и информационных технологий Казанского федерального университета, к. т. н. Основные направления научных исследований: обработка естественного языка, искусственный интеллект.

Olga Avenirovna NEVZOROVA – Kazan Federal University, Institute of Computer Mathematics and Information Technologies, Associated Professor the Department of Information Systems, PhD. Major Fields of Scientific Research: Natural language processing, Artificial intelligence.

e-mail: onevzoro@gmail.com

Материал поступил в редакцию 27 апреля 2020 года

УДК 004.78

ОСОБЕННОСТИ МОНИТОРИНГА МОБИЛЬНЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

О. Панарин[1], И. Захаров[2]

[1,2] *Сколковский институт науки и технологий, г. Москва*

[1]O.panarin@skoltech.ru, [2]i.zacharov@skoltech.ru

Аннотация

Рассмотрена реализация подсистемы мониторинга систем обработки информации на мобильных платформах и ее применение на беспилотных автомобилях. В условиях беспилотной эксплуатации автомобиля предъявляются наиболее жёсткие требования к надежности систем обработки информации, принятию решения о готовности этих систем к эксплуатации и обеспечению анализа их возможных сбоев. Представленная система мониторинга pLOG сочетает в себе функционал записи событий операционной системы устройств и измерений параметров систем в реальном времени, при этом запись производится как файлы, так и в базу данных временных рядов (TSDB). При этом каждый сервер в составе системы обработки информации на мобильных платформах дублирует запись обо всех событиях в системе.

Ключевые слова: *сенсорные данные, распределенные системы, мониторинг*

MONITORING MOBILE INFORMATION PROCESSING SYSTEMS

O. Panarin[1], I. Zacharov[1]

[1]*Skolkovo Institute of Science and Technology, Moscow*

[1]O.panarin@skoltech.ru, [2]i.zacharov@skoltech.ru

Abstract

We describe the implementation of the monitoring for the IT systems at the core of the autonomous driving vehicle. The role of the monitoring is to assist in decision to

start the driving cycle and continuous assessment for the fitness to drive the vehicle. The requirements for the monitoring system with the increased resiliency and data replication make it sufficiently different from standard monitoring systems and warrant a unique implementation tuned for the autonomous driving requirements. The monitoring system combines the OS events and real-time measurements of sensor data. The information is stored in flat files for emergency access as well as in a Time Series Data Base (TSDB).

Keywords: *mobile systems, distributed systems, sensor data, monitoring*

INTRODUCTION

The autonomous driving is facilitated with information processing system analyzing data coming from many different sources (for example the LIDARs for the distance measurement, stereo-vision, global positioning, etc.) in combination with the embedded maps, real time object classification and decision taking (Neural networks, expert systems). The information processing system is built with modern high performance servers in ruggedized enclosure with water cooling. Combined performance of such system in autonomous vehicle may exceed 100 Tflop/s (e.g. for the single precision floating point operations) while the power consumption is in the order of 5 KW with contemporary technology. The performance is distributed between several systems usually serving one kind of data source for analysis each. For example, there is the LIDAR analysis system, system for stereo-vision of the right wing and so on, as well as a system that combines all the information and taking the steering decisions.

While specific implementation is vehicle dependent the common element in the contemporary systems is the high performance server running Linux OS with GPU accelerator to speed up the neural network operations. The information processing system in an autonomous vehicle is computational cluster consisting from such servers with a number of networks. In particular, there is a high speed network that shares all raw and/or analyzed sensor data and allows to store the data in a logging operation at a speed of 5 GB/s. The given performance is required and sufficient at current technological level (and with the current sensor resolution) for vehicle autonomous operation on one hand and to store the data for transfer to the laboratory and analysis off line. We continue to analyze the data storage requirements.

For completeness we mention another type of the event storage system in an autonomous vehicle to store data related to an emergency and/or crash analysis. This type of storage system should provide an absolute data integrity that may be required to analyze vehicle movement from the liability perspective. The speed or size of storage should be sufficient to store the last 30 seconds of vehicle movement before an aversive event. This subsystem is required for the SAE vehicle readiness level 3 and above [1] to cater for the insurance claims.

The main subject of this article is the subsystem dedicated to monitoring the servers and the networks they connect in an autonomous vehicle coupled with the storage and analysis of the monitoring data. We have reported previously an advanced implementation of the monitoring in autonomous vehicle [14] and extend the study in this article.

Before an autonomous vehicle may be allowed to move there should be a clear understanding that all subsystems work as expected, the sensors are collecting the data, servers are up and running and the vehicle is ready for the operation. During the operation any deviation from the expected behavior should be reported to the operator to reclaim the vehicle (level 2, 3 of SAE) or it should lead to an emergency/controlled stop (level 4, 5 SAE). During the regular operation monitoring systems is not critical and is necessary only if problems are detected. If there is a problem the collected monitoring information should be sufficient to analyze the situation.

Monitoring of servers in Datacenter setting is well understood with a number of successful implementations. For example, there are the most flexible and adoptable Nagios [2] or Zabbix [3] monitoring systems with highest deployment track record for Datacenters. The autonomous system monitoring has a similar task with a number of peculiarities.

Here is the list of features unique to the monitoring in autonomous vehicles:

- The reliability requirement must exclude any single point of failure and allow for multiple failures in the monitoring system. Of course, even a single failure in a non-critical component, such as the monitoring system may lead to the decision to stop the vehicle. However, the analysis of the failure has to be performed with the same monitoring system; therefore, the monitoring must be especially robust and allow for multiple points of failure. Using this monitoring system to analyze failures

may lead to the resolution of the problem “in the field” and allow to continue vehicle operation

- There are non-standard (i.e. not typically found in the Datacenter) sensors in the system. In the typical Datacenter monitoring systems the sensors are attached with IMPI and SNMP protocol; in a vehicle there are additional standards (such as the COM-Bus and networks like 100Base-T1) that must be integrated in the monitoring system
- The autonomous vehicle cluster has small number of servers (nodes). The Datacenter monitoring systems scale to 100s and 1000s of nodes while the autonomous vehicle cluster is limited to about 10 servers. This peculiarity allows to implement monitoring systems with smaller complexity and aids to increase the robustness
- There is typically no need to keep the monitoring data for duration beyond the immediate run or a few runs of the autonomous vehicle. This relaxes the storage requirement and allows to increase the robustness by duplication of the incoming monitoring data on all the nodes
- There is a need to provide the monitoring data to the operator through different interfaces, such as the GUI or WEB-interface for a comfortable analysis by operator without special training, as well as harder to interpret data with an interface not featuring any graphics. This is necessary for casual analysis of the monitoring data, as well as for the emergency analysis to restore system functionality after a crash
- The monitoring system should adjust itself to the running environment without the need for a special configuration or tuning in an autonomous vehicle. We expect that there is a great number of autonomous vehicles where the monitoring system is deployed and it is not possible to follow up and tune each separate installation manually. With other words the monitoring system should run “out-of-the-box” and only in special conditions require analysis and tuning.

In what follows we consider how a monitoring system that fulfills all these requirements may be implemented for monitoring servers in an autonomous vehicle. Further we provide examples running this system. Before that we analyze competing strategies for building such monitoring system.

STRATEGIES FOR THE MONITORING SYSTEM

System under study — high performance cluster with nodes using OS Linux. There are a number of tools exist to monitor each individual system. For example, top, htop, nmon, vmstat, iotop, iftop, netstat, bmon, glances and so on (see the list in ref. [4]). All these utility programs use Linux statistics that is provided through the (pseudo-) file system /proc. For example, the processor load statistics can be read from the file /proc/stat, the memory usage from /proc/meminfo, network /proc/net/dev, storage /proc/diskstats and so on. During each read from these files Linux kernel driver is providing internal counters data. The only difference between the utility programs is how each structures the output and how fast the programs are refreshing the view for the user.

There several solution to monitor the whole cluster consisting of several servers. For example, Nagios, ganglia, zabbix, cacti and so on (see the list in ref. [5]). These programs obtain the data from the same interface /proc, but structure it in a way comfortable to judge the state of the whole cluster. Also, some of these programs (eg. zabbix) store the data in a database.

Typically, the autonomous vehicle would integrate monitoring solution based on the listed programs. The contemporary computational base uses Linux servers (clusters) for the implementation of the autonomy function in this functional testing phase, which is considered as an intermediate phase on the way to broad deployment of autonomous vehicles that will use a different computing hardware. Current Linux servers running in the vehicles should be replaced by specialized hardware suitable for massive application. Therefore, investment into the monitoring software at this stage may seem premature.

Our strategy is different. We suggest that the evolution of Linux clusters in autonomous vehicles will follow the specialization and miniaturization trend but will keep the current software interfaces. Following this hardware evolution the monitoring software will demand health management function which is not part and cannot be expected from the standard Linux monitoring programs. This transformation will go in hand with the integration of multitude of sensors that will provide the data for the health management. With the use of standard Linux interfaces the general purpose monitoring systems like Zabbix can be extended with additional functionality but the

complexity will grow substantially which is conflicting with the stated requirement. Therefore, on the specialized future computational platforms used to run autonomous vehicles the monitoring and health management software will also be special, even if it will continue use todays software interfaces. The proposed monitoring software is expected to evolve in this direction.

On a timeline of 5–7 years the monitoring system transformed and integrated into the health management system of an autonomous vehicle will follow same milestones of security provisioning in the maned aviation [6]. In aviation this process is far from over [7], but this experience influences our planning in preparing software for autonomous vehicles.

IMPLEMENTATION OF THE MONITORING SYSTEM

The implementation is based on a symmetric deployment of the software on all nodes of the cluster (servers) independent of the intended usage or characteristics of the server or application running on it. There should be however a dedicated node that collects and stores all data from the cluster, including the sensor data. Naturally the regular access to the monitoring data will provided from this node, but because of the symmetry all other nodes collect and store same monitoring data. In case of network failure and node separation the monitoring data may be obtained from any surviving node. It will contain information about the working of the cluster before the separation and each server will have information about its own state after the separation and it can be used to analyze the crash.

The analysis of cluster events is aided by time marks recorded with the data. The cluster is fully time-synchronized with the PTP (IEEE 1588 [8]) protocol having resolution of about 1 micro-second. In case of network separation the clock drift is not considered critical, since in this emergency mode the cluster should work only few (tens) of minutes before the communication is restored.

The information is collected by daemons that are started when OS is booted on the cluster nodes. The logical interaction scheme is shown in Fig. 1 and each cluster node implementing the same scheme.

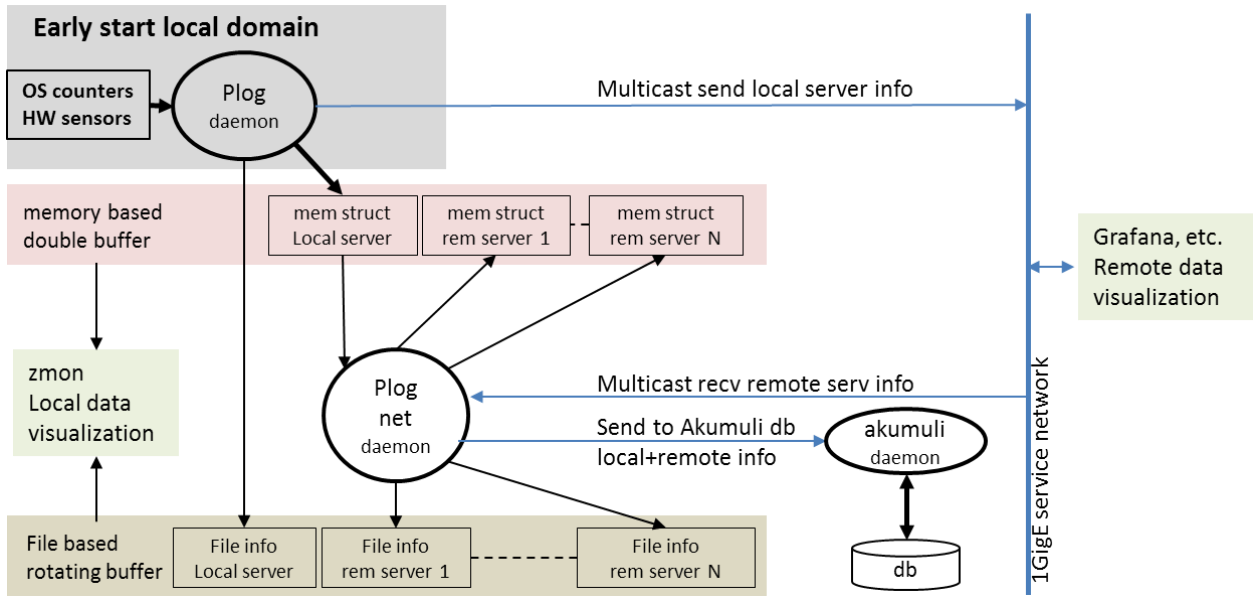


Fig. 1. The architecture of the monitoring system in autonomous vehicle for each node

On each node we start the OS Linux, Plog daemon, Plog-net daemon and Akumuli daemon. The basic Plog daemon starts first when Linux boots, initializes memory structures, collects the sensor and OS data, writes them to into the memory and an asci file. Plog is refreshing this data every 2 seconds (tunable) and sends this data in a multicast packet if network is available. After Plog – the Plog-net daemon starts, it collects the data packets sent by Plog daemon running on other servers on the network and adds this data to the structure in memory and writes an asci file for each remote server. The dump to file is important, as it increases the chances that post mortem analysis may be done after a crash; to limit the size of the asci file it contains data of last 20 minutes of running (size can be adjusted).

The functional splitting between Plog and Plog-net is determined by the requirement of monitoring without the network. Local monitoring collects data from the server and sensors that monitor external condition of each server. These sensors are attached to each server and network router to monitor the power and cooling conditions. Typically, each server collects data from its own sensor. The sensors from the network devices are attached to (one or several) server(s) that process that data in addition to their own monitoring. If network is separated it is possible to analyze the monitoring data from network devices on these servers.

Users watch each node (and all sensors attached to it) with command zmon that displays data on a textual console. A column for each server is displayed. The zmon

command is modeled after the well-known Linux utility nmon [4], build with ncursors library [9]. Command zmon with –R flag will show historical data stored in asci files and displays them circularly with tunable speed.

The Plog-net daemon implements full functionality by collecting packets sent by Plog from other nodes. In this way each node has the full picture of the working cluster.

In addition to the asci files where last 20 minutes of data is stored – all data is sent to the local time-series database Akumuli [10]. It maintains a larger data collection of about 1 month worth data (tunable) that results in about 4 GB of storage size. Akumuli can supply this data in format json to the Grafana [11] plotting package. Each cluster node has its own akumuli agent and grafana-server daemon. If graphics head is available the monitoring data can be analyzed with graphics tools where trends are clearly visible.

This implementation is fully symmetrical and does not need a specialization or per server tuning. When the test fleet of autonomous vehicles grows it will save time and resources in setting up the system and eliminates configuration mistakes.

The usage of the Time Series Data Base (TSDB) Akumuli for monitoring in Data-centers has been proposed earlier (see for example [12]). Akumuli – is developed by Evgeny Lazin [13] and its distinctive feature is speed and compactness.

The known Akumuli limitations, such as the strict locality of application (not supporting distributed data collection) and chronological data entry order do not play a role in our setup, since the Plog-net daemon will utilize local Akumuli agent with strict data entry with the local time mark. In addition, all servers support and run time synchronization protocol, therefore a discrepancy between the “local” and “remote” time mark is a symptom of dysfunction of the system and a reason to stop the vehicle. On the other hand the speed and compactness were decisive criteria when choosing this TSDB for autonomous vehicle monitoring data.

All accesses to the TSDB Akumuli write node id and other static information that allows performing a search and selection of data related to a specific node. In this way we can analyze the work of each and single node as well as cluster as a whole starting from the information stored on each node in an autonomous vehicle cluster.

THE METRICS SELECTION

Generally all sensor and OS data are stored in the TSDB and therefore available for the retrieval and graphical analysis with Grafana. For the zmon tool that has to fit the information on a single console display we are limited to a generic 224x256 (columns x rows) letters and numbers that has to be structured in a most intuitive way. For the first implementation in zmon we have chosen to present all of the environmental data for each server, i.e. the power, temperatures (in/out) and flow rate of the coolant, air temperatures measured inside the server enclosure and dew point temperatures. We also present integral characteristics of the processing, such as the total idle, user and system time, total network and total storage performance, GPU power draw and occupancy. The GPU data should give a quick assessment if the application (using GPU for neural network computation) is started and running on the system.

This selection is driven by the need to quickly assess the fitness of the system for the field run and allows analyzing failures. Sensor data is connected to the safety system that will protect servers against any adverse conditions. However, it may not be immediately obvious what has triggered the alarm and/or the safety shutdown, especially if one or more systems are down and only few consoles are operating. The present selection of signals for the display should help in this analysis.

A significant step from monitoring to a health management system has been done in collecting the per-process data. This is not displayed in the zmon console view, but stored in the TSDB and can be retrieved from there in a separate analysis program which is planned for later. The per-process monitoring will narrow down cases when the processing slows down or changes its characteristics from known values which may be a signature of an imminent failure. We are tuning this feature on the Zhores cluster at Skoltech [15] where we also do analysis for the failure signatures in user programs processing data.

IMPLEMENTATION RESULTS

The monitoring activity loads the network at about 4 kB/s (or 13 MB/hour). There is compression in the Akumuli TSDB therefore the database retains data of about one month worth. After that period the new data replaces the oldest data and the monitoring does not need manual maintenance related to the uncontrolled use of storage.

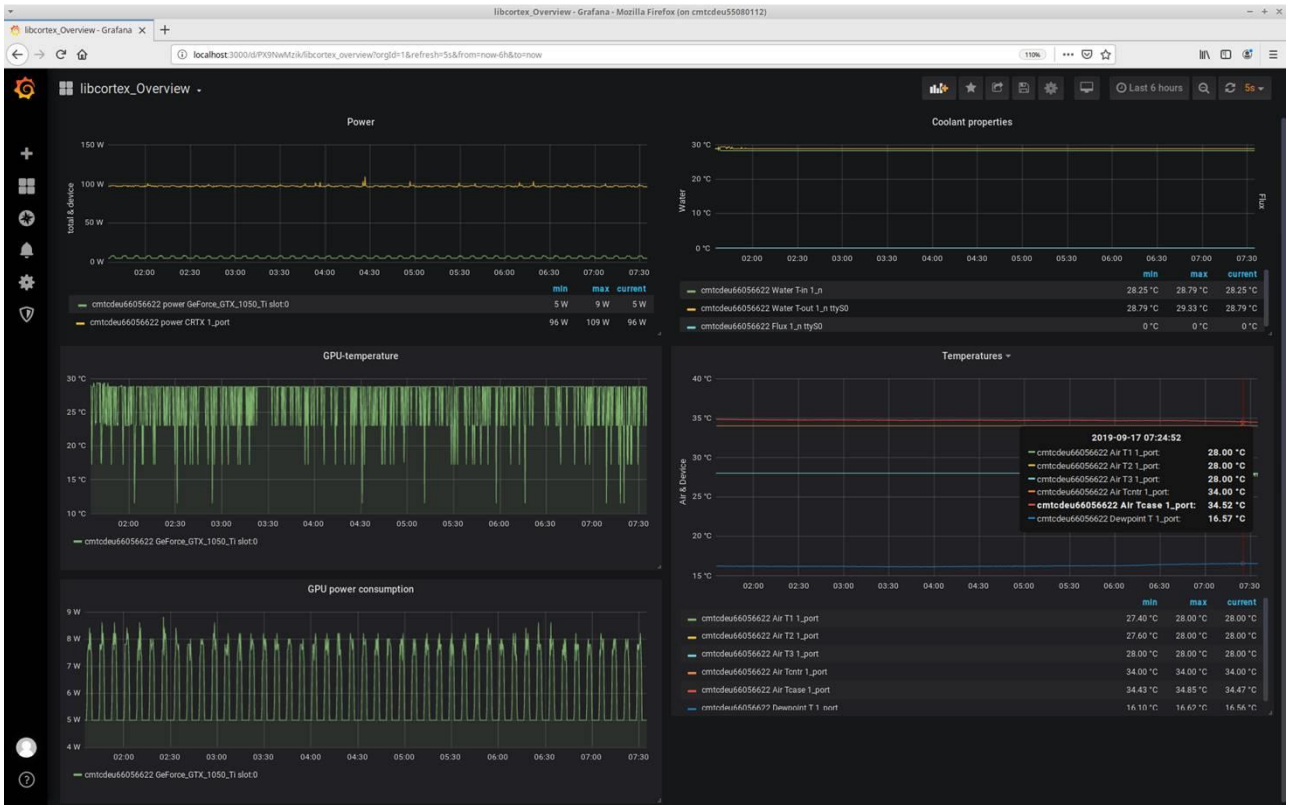


Fig. 2. Typical Grafana view of the monitoring results in a web brouser

The graphical analysis Grafana shows a typical representation, see Fig. 2. That figure illustrates the upper Dashboard for one server, where we show (left to right and up to down) the overall power consumption, temperature of the coolant, temperature of the GPU, a series of air temperatures and the GPU power draw. Note that the graphical representation is not finalized yet and will change as more experience is collected using the system. The Grafana plotting package is flexible to make many different representations of the same data, therefore it is more to the users of the system to define the proper views. The actual monitoring implementation does not depend on the Grafana graphics views. This is not the case for the zmon tool, which must be reprogrammed if a different view on the system is required.

Fig. 3 shows a typical console display from the zmon tool. When analyzing the system with the zmon program it is possible to switch on and off the information panels represented with the header in inverse video. It is also possible to select and unselect servers to view as part of the cluster configuration. In historic view this may serve as a way to focus on certain configurations.

Fig. 3. Typical zmon (-R) display on a console terminal. Each column shows one node in cluster

CONCLUSION

We have discussed the requirements and the implementation methodology of the autonomous vehicle cluster monitoring. These requirements and the implementation are distinctly different from the monitoring in the Datacenters, although similar devices (servers, routers) are used in these clusters. The project has been implemented for autonomous vehicle testing fleet.

Further development will encompass the monitoring of the monitoring system self, as well as the information from the applications that govern the data processing in the autonomous vehicle.

REFERENCES / СПИСОК ЛИТЕРАТУРЫ

1. SAE J3016 “Levels of Driving Automation”. URL: https://www.sae. org/news/2019/01/sae-updates-j3016-automated-driving-graphic (accessed: 12/05/2019)

2. Nagios. URL: https://www.nagios.org (accessed: 12/05/2019)

3. Zabbix. URL: https://www.zabbix.com (accessed: 12/05/2019)

4. Most Comprehensive List of Linux Monitoring Tools For SysAdmin. URL: https://www.ubuntupit.com/most-comprehensive-list-of-linux-monitoring-tools-for-sysadmin (accessed: 12/05/2019)

5. Top FREE Server Monitoring Tools. URL: https://www.dnsstuff.com/free-server-monitoring-tools (accessed: 12/05/2019)

6. In-Time Aviation Safety Management: Challenges and Research for an Evolving Aviation System (2018). ISBN 978-0-309-46880-0. DOI 10.17226/24962

7. Aircracft Health Monitoring – FAA Perspective, presented to IATA Conference by Tim Shaver, 13 November 2017. URL: https://www.iata.org/whatwedo/workgroups/Documents/Paperless_Conference_2017/Day1/1130-1200_AircraftHealthMonitoring_FAA.pdf (accessed: 12/05/2019)

8. NIST Intelligent Systems Division. URL: https://www.nist.gov /el/intelligent-systems-division-73500/ieee-1588 (accessed: 12/05/2019)

9. NCURSES Programming HOWTO. URL: https://www.tldp.org/HOWTO/NCURSES-Programming-HOWTO/intro.html (accessed: 12/05/2019).

10. Akumuli. URL: https://akumuli.org/ (дата обращения: 12/05/2019)

11. The open platform for beautiful analytics and monitoring. URL: https://grafana.com (accessed: 12/05/2019)

12. *Живчикова Н.С., Шевчук Ю.В.* Подсистема архивации данных системы мониторинга Botikmon3 // Научный сервис в сети Интернет: труды XX Всероссийской научной конференции (17–22 сентября 2018 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2018. С. 223–229.
URL: http://keldysh.ru/abrau/2018/theses/26.pdf doi:10.20948/abrau-2018-26

13. *Лазин* E. Numeric B+tree reference. URL: https://akumuli.org/akumuli/2017/ 04/29/nbplustree/ (accessed: 20/11/2019)

14. *Панарин О., Захаров И.* Особенности мониторинга мобильных систем обработки информации// Научный сервис в сети Интернет: труды XXI Всероссийской научной конференции (23–28 сентября 2019 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2019. С. 551–560. URL: http://keldysh.ru/abrau/2019/ theses/81.pdf (accessed 16/11/2019). doi: 10.20948/abrau-2019-81

15. *Zacharov I., Arslanov R., Gunin M., Stefonishin D., Bykov A., Pavlov S., Panarin O., Maliutin A., Rykovanov S., Fedorov M.,* “Zhores” – Petaflops supercomputer

for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology// Open Engineering. Published 2019-10-26, V. 9. Issue 1. doi: https://doi.org/10.1515/eng-2019-0059

СВЕДЕНИЯ ОБ АВТОРАХ

ПАНАРИН Олег Анатольевич – менеджер по информационным сервисам и обработке данных Сколковского института науки и технологий, специалист в области высокопроизводительных компьютерных систем и систем мониторинга.

Oleg PANARIN – Manager of Data and Information Services at Skolkovo institute for Science and Technology specializing in High Performance Computing and System Monitoring.

e-mail: o.panarin@skoltech.ru

ЗАХАРОВ Игорь Евгеньевич – старший научный сотрудник Сколковского института науки и технологий, специалист в области высокопроизводительных компьютерных систем и системного программирования.

Igor ZACHAROV – Senior researcher at Skolkovo institute for Science and Technology specializing in High Performance Computing and System Programming.

e-mail: i.zacharov@skoltech.ru

Материал поступил в редакцию 15 ноября 2019 года

УДК 004.42+004.021+519.688

ПРИМЕНЕНИЕ СУПЕРКОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ ДЛЯ ДОЛГОСРОЧНОГО МОДЕЛИРОВАНИЯ ГРАНИЦ ЗАЛЕГАНИЯ ВЕЧНОЙ МЕРЗЛОТЫ НА НЕФТЕГАЗОВЫХ МЕСТОРОЖДЕНИЯХ АРКТИКИ

М. Ю. Филимонов[1], Н. А. Ваганова[2], Е. Н. Акимова[3], В. Е. Мисилов[4]

Уральский федеральный университет им. первого Президента России Б.Н. Ельцина;
Институт математики и механики им. Н.Н. Красовского УрО РАН

[1]fmy@imm.uran.ru, [2]vna@imm.uran.ru, [3]aen15@yandex.ru, [4]v.e.misilov@urfu.ru

Аннотация

Рассмотрена модель распространения тепловых полей в многолетнемерзлых породах от различных инженерных объектов, функционирующих в Арктических районах. В предложенную модель включены наиболее существенные технические и климатические параметры, влияющие на формирование тепловых полей в приповерхностном слое грунта. Основной целью исследования является долгосрочное прогнозирование изменения динамики границы залегания вечной мерзлоты при эксплуатации кустовой площадки северного нефтегазового месторождения. Такой прогноз получается при моделировании сложной системы, состоящей из источников тепла (холода) и мерзлого грунта, растепление которого может приводить к потере его несущей способности и возможным техногенным и экологическим авариям. Например, источниками тепла могут выступать добывающие скважины, а источниками холода – сезоннодействующие охлаждающие устройства, которые используются для термостабилизации грунта. Для минимизации воздействия источников тепла на вечную мерзлоту использованы различные варианты теплоизоляции, а для сохранения первоначального температурного режима верхнего слоя грунта – отсыпки, состоящие из песка, бетона, пеноплекса или другого теплоизолирующего материала. Разработанный комплекс программ был использован при проектировании 12 северных нефтегазовых месторождений. Для решения описанной задачи в сложной трехмерной области

требуются существенные вычислительные ресурсы. Время расчета одного варианта часто может превышать 10–20 часов машинного времени на суперЭВМ. Для ускорения проведения численных расчетов были использованы многоядерные процессоры. Приведены численные расчеты, которые иллюстрируют возможности разработанного комплекса программ для проведения долгосрочных прогнозов по определению изменения границ распространения зон вечной мерзлоты, а также показывают, что на многоядерных процессорах можно достичь ускорения, близкого к теоретическому.

***Ключевые слова**: компьютерные программы, тепломассоперенос, криолитозона, моделирование, задача Стефана, параллельные вычисления, OpenMP*

ВВЕДЕНИЕ

Термин «вечная мерзлота» был введён в научное употребление в 1927 году основателем школы советских мерзлотоведов М.И. Сумгиным. Впоследствии этот термин критиковали, и были предложены альтернативные термины: многолетнемёрзлые породы, многолетняя криолитозона и др. Общепринято, что многолетнемерзлыми породами называются породы, которые не менее двух лет находятся при отрицательных температурах. В работах, посвященных освоению северных нефтегазовых месторождений, чаще используется термин многолетнемёрзлые породы, сокращенно ММП.

Криолитозона занимает около 25% всей суши земного шара [1, 2] и располагается в северных или горных регионах ниже зоны сезонного протаивания грунта, которая определяется географическими координатами, интенсивностью солнечного излучения и другими климатическими факторами. Возможные изменения климата, связанные, например, с глобальным потеплением (особенно в приполярных районах), могут привести к серьезным изменениям в ММП, которые окажут влияние на окружающую среду в глобальном масштабе [2].

Освоение зон распространения ММП особенно актуально для России, поскольку в криолитозоне добывается около 93% российского природного газа и 80% нефти. Вместе с тем, освоение этих регионов приводит к негативным последствиям, связанным с растеплением ММП, которое приводит к образованию опасных геологических явлений, называемых термокарстом. Средняя толщина ММП изменяется в пределах от 10 до 800 метров, а слагающие ММП породы имеют

различные физико-химические свойства, которые могут изменяться по всем направлениям. В летнее время, в результате положительных температур и солнечного излучения, происходит сезонное оттаивание верхнего слоя грунта, в зимнее время наблюдается обратный процесс. Моделирование таких сезонных процессов описано в [3], а применение многоядерных процессоров и технологии OpenMP – в [4]. Большое влияние на формирование тепловых полей в грунте оказывает и техногенное воздействие, приводящее часто к более существенным изменениям границ ММП [5, 6]. На кустовых площадках северных нефтегазовых месторождений такие воздействия могут оказывать следующие технические системы, являющиеся также и источниками тепла: добывающие и нагнетательные скважины [7, 8], факельные системы [9], эксплуатируемые по специальным алгоритмам [10], и другие инженерные объекты. Для термостабилизации грунта (его охлаждения) используются охлаждающие устройства, которые позволяют уменьшить тепловые воздействия от источников тепла на окружающий грунт и могут быть использованы для подготовки строительной площадки в зоне распространения вечной мерзлоты путем охлаждения верхней части грунта [11].

Для решения описанных задач были разработаны математические модели и соответствующие комплексы прикладных задач для моделирования нестационарных тепловых полей в приповерхностном слое грунта в сложной трехмерной области. Время расчета таких задач, для которых требуется получить большое число решений с различными параметрами, очень часто могло превышать десятки часов машинного времени на суперЭВМ. Были рассмотрены некоторые параллельные подходы [12], а также использованы облачные технологии, позволяющие проводить удаленные вычисления при решении некоторых задач, возникающих при обустройстве и эксплуатации северных нефтегазовых месторождений [13, 14]. В настоящей работе исследовано применение многоядерных процессоров с использованием технологии OpenMP для решения задач долгосрочного прогнозирования изменения границ протаивания мерзлых пород вокруг добывающей скважины. Приведены результаты численных расчетов и оценка эффективности применения многоядерных процессоров.

ПОСТАНОВКА ЗАДАЧИ И МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Пусть $T=T(t,x,y,z)$ – распределение температуры в момент времени t, (x,y,z) – точка расчетной области $\Omega = \{(x,y,z): -L_x \le x \le L_x,\ -L_y \le y \le L_y,\ -L_z \le x \le 0\}$. Расчетная область представляет собой трехмерный параллелепипед (рис. 1), из которого удалены вертикальная цилиндрическая скважина Ω_1 и сезоннодействующие охлаждающие устройства (СОУ) Ω_2. Оси x и y расположены параллельно поверхности грунта, а ось z – внутрь Ω. На рис. 1 приведены два возможных варианта: отдельно расположенная скважина и скважина, вокруг которой расположены СОУ.

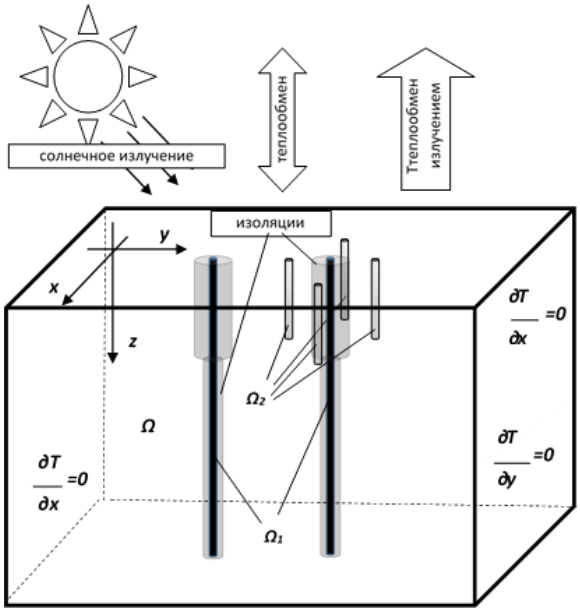


Рис. 1. Расчетная область Ω, скважина Ω_1, охлаждающие устройства Ω_2 и краевые условия

Процесс распространения тепла с учетом фазового перехода может быть описан следующим уравнением теплопроводности:

$$\rho\left(c_\nu(T) + k\delta(T - T^*)\right)\frac{\partial T}{\partial t} = \operatorname{div}\left(\lambda(T)\ \operatorname{grad} T\right) \qquad (1)$$

с учетом начального условия

$$T(0,x,y,z) = T_0(x,y,z), \qquad (2)$$

где ρ – плотность [кг/м^3], T* – температура фазового перехода,

$$c_\nu(T) = \begin{cases} c_1(x,y,z), & \text{при } T < T^*, \\ c_2(x,y,z), & \text{при } T > T^*, \end{cases}$$ – удельная теплоемкость [Дж/(м · К)],

$$\lambda(T)=\begin{cases}\lambda_1(\mathrm{x,y,z}),\ \text{при } T<T^*,\\ \lambda_2(\mathrm{x,y,z}),\ \text{при } T>T^*,\end{cases} - \text{коэффициент теплопроводности } [\text{Вт}/(\text{м}\cdot K)],$$

$k=k(x,y,z)$ – теплоемкость фазового перехода, δ – дельта-функция Дирака. Обоснование применимости этого уравнения для решения задач типа Стефана обосновано в [15, 16]. В [17] на основе баланса тепла на дневной поверхности выписано краевое условие

$$\alpha q(t)+b(T_{air}(t)-T\big|_{z=0})=\varepsilon\sigma(T^4-T_{air}^4(t))+\lambda\frac{\partial T}{\partial \mathrm{z}}\bigg|_{z=0} \qquad (3)$$

и описан алгоритм привязки математической модели к конкретному географическому месту. В условии (3) Через $T_{air}(t)$ обозначена температура воздуха в приповерхностном слое, которая изменяется периодически в соответствии с годичным температурным циклом, $\sigma = 5{,}67\cdot10^{-8}$ Вт/(м^{2}К^4) – постоянная Стефана–Больцмана, $b=b(t,x,y)$ – коэффициент теплообмена, $\varepsilon=\varepsilon(t,x,y)$ – коэффициент серости. Коэффициенты теплообмена и серости зависят от типа и состояния поверхности грунта. Суммарная солнечная радиация $q(t)$ состоит из суммы прямой солнечной радиации и рассеянной радиации. Грунтом поглощается только часть суммарной радиации, равной $\alpha q(t)$, где $\alpha=\alpha(t,x,y)$ – доля энергии, ушедшая на нагрев грунта, которая в общем случае зависит от состояния атмосферы, угла падения солнечных лучей, т. е. широты местности и времени суток.

Технические системы Ω_1 и Ω_2 являются дополнительными источниками тепловых полей в ММП. В связи с этим возникают дополнительные краевые условия для этих объектов. На поверхностях этих объектов $\partial\Omega_i$ зададим краевые условия

$$T\big|_{\partial\Omega_i}=T_i(t), \qquad i=1,2. \qquad (4)$$

Чтобы воспользоваться численными методами, необходимо на гранях расчетной области Ω задать краевые условия

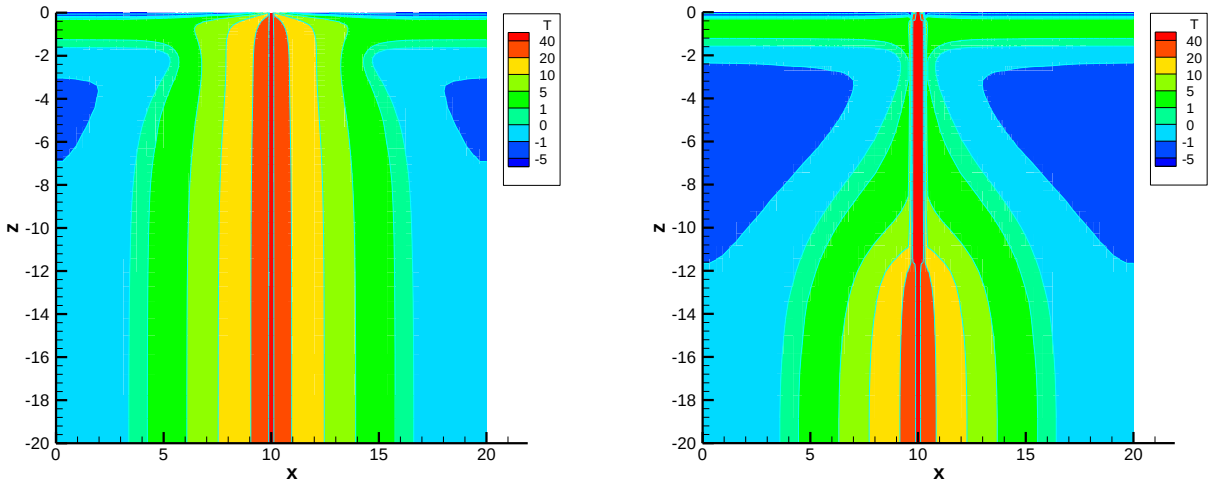
$$\frac{\partial T}{\partial x}\bigg|_{x=\pm L_x}=0,\quad \frac{\partial T}{\partial y}\bigg|_{y=\pm L_y}=0,\quad \frac{\partial T}{\partial \mathrm{z}}\bigg|_{\mathrm{z}=-L_z}=0. \qquad (5)$$

При этом расчетная область должна быть выбрана достаточно большой, чтобы избежать влияния краевых условий (5) на тепловые поля в расчетной области Ω, создаваемые объектами Ω_i.

ЧИСЛЕННЫЕ РАСЧЕТЫ И ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ

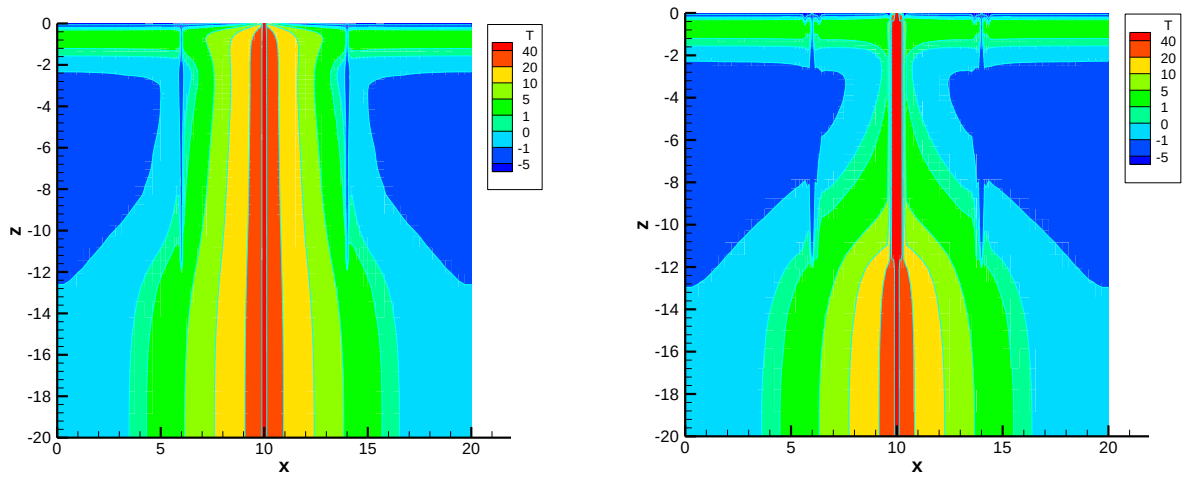
Для использования численного метода для решения задачи (1)–(5) необходимо определить параметры, входящие в краевое условие (3). В [17, 18] описан итерационный алгоритм по определению этих параметров. Выбор этих параметров позволяет опосредовано учесть влияние снежного покрова, климатические, и природные условия, связанные с географическими координатами конкретной кустовой площадке. При численной реализации решения задачи (1)–(5) применяется конечно-разностный метод, позволяющий использовать метод расщепления по пространственным переменным, для лучшей организации численных расчетов. Уравнение (1), следуя работам [15, 16], по каждому из пространственных направлений аппроксимируется неявной центрально-разностной трехточечной схемой, и методом прогонки решается система разностных линейных алгебраических уравнений, имеющая трехдиагональный вид. На поверхности грунта, ввиду условия (3), возникает алгебраическое уравнение 4-ой степени, для решения которого используется метод Ньютона [19]. В расчетах используется ортогональная сетка, равномерная или сгущающаяся по определенному закону вблизи поверхности грунта либо к поверхностям Ω_i. Описанная методика реализована в сертифицированном пакете программ «Wellfrost» и различных его модификациях, которые были апробированы на 12 северных нефтегазовых месторождениях. Также было проведено сравнение экспериментальных данных и результатов численных расчетов, связанных с определением границы нахождения нулевой изотермы, которая определяет границу области растепления грунта вокруг добывающей скважины. Точность численных расчетов была проверена в 2012 году для российского нефтяного месторождения «Русское», для которого полученные численные результаты отличались от экспериментальных менее 5% через 3 года после начала эксплуатации месторождения.

На рис. 2 и 3 представлены рассчитанные тепловые поля за 5 лет эксплуатации вокруг скважины без теплоизолирующих оболочек и с комбинированной теплоизолирующей оболочкой, а также с системой из 8 СОУ.



(а) (б)

Рис. 2. Тепловые поля после 5 лет эксплуатации неизолированной скважины (а) и скважины с комбинированной оболочкой (б)



(а) (б)

Рис. 3. Тепловые поля после 5 лет эксплуатации неизолированной скважины (а) и скважины с комбинированной оболочкой (б) и системы СОУ

Фоновая температура многолетнемерзлых пород –0.7°С, температура флюида в скважине 50°С. Расчеты позволяют оценить взаимное влияние источников тепла или холода, зону сезонного протаивания и промерзания верхнего слоя грунта. Для оценки долгосрочного влияния расчеты проводятся на временной период до 50 лет, шаг по времени не превышает 24 часа. Время расчета зависит как от размера расчетной сетки, так и от сложности внедряемых в расчетную область

объектов. В таблице 1 приведены описания вариантов расчета. На рис. 4 представлен график изменения времени расчета для каждого из вариантов. Расчеты проводились на сетке 91x91x51 узлов.

Таблица 1. Описание вариантов расчета

Номер варианта	Описание варианта
1	Промерзание и оттаивание грунта без внедренных в него объектов.
2	Растепление грунта вокруг скважины без теплоизоляции.
3	Растепление грунта вокруг скважины с простой оболочкой.
4	Растепление грунта вокруг скважины с комбинированной оболочкой.
5	Промерзание грунта при наличии 9 СОУ.
6	Промерзание и растепление грунта вокруг скважины без теплоизоляции и 8 СОУ.
7	Промерзание и растепление грунта вокруг скважины с простой оболочкой и 8 СОУ.
8	Промерзание и растепление грунта вокруг скважины с комбинированной оболочкой и 8 СОУ.

Время расчета варианта растет нелинейно как при увеличении количества объектов, включенных в модель, так и от сложности конструкции объекта даже при условии, что размер расчетной сетки остается неизменным.

Численное моделирование сложных объектов для долгосрочных прогнозов требует значительных затрат времени. Одним из способов сокращения времени счета является использование параллельных вычислений [20, 21].

Для процедуры пересчета поля температуры методом переменных направлений в работе [22] авторами было проведено распараллеливание для многоядерных процессоров с использованием технологии OpenMP [23]. Трехдиагональные системы линейных уравнений могут решаться независимо друг от друга. В параллельной реализации они распределяются по OpenMP-потокам.

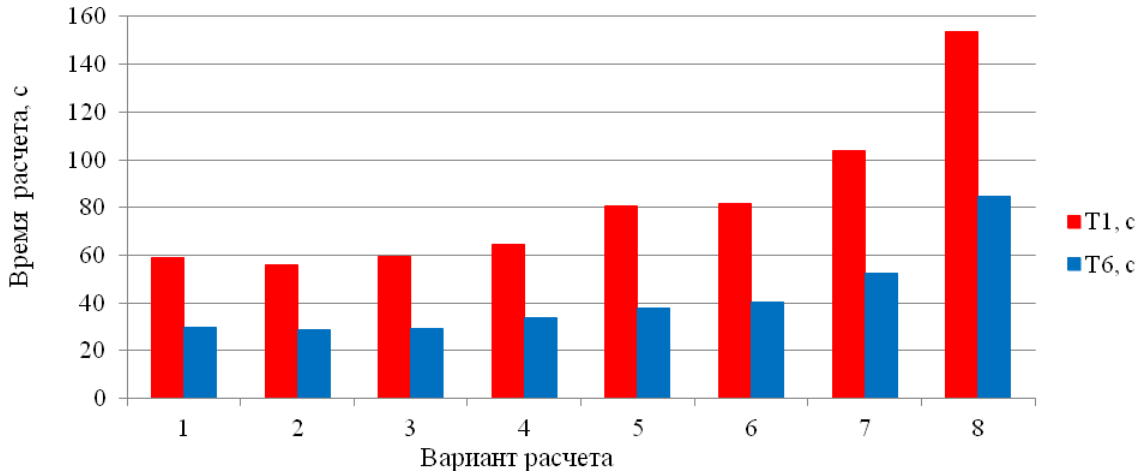


Рис. 4. Времена расчетов 1 года для различных вариантов: время выполнения последовательной программы (красный), время выполнения программы на 6 ядрах (синий)

В таблице 2 приведены времена выполнения параллельной программы для расчета 1 года на 6-ядерном процессоре AMD Ryzen 5 1600X: T_1 — время выполнения последовательной программы, T_6 — время выполнения программы на 6 ядрах.

Таблица 2. Времена выполнения параллельной программы для различных вариантов

Вариант расчета	T_1, с	T_6, с	Ускорение S_6	Эффективность E_6
1	59	29.7	1.98	0.33
2	56.2	28.8	1.95	0.32
3	59.6	29.5	2.01	0.34
4	64.3	33.6	1.91	0.32
5	80.8	38.1	2.12	0.35
6	81.8	40.5	2.01	0.34
7	103.7	52.4	1.98	0.33
8	153.8	84.9	1.81	0.30

Для исследования параллельного алгоритма используются показатели ускорения $S_n = T_1 / T_n$ и эффективности $E_n = S_n / n$ при выполнении программы на n ядрах. Полученные значения согласуются с теоретическим ускорением, рассчитанным по закону Амдала:

$\overline{S_n} = \frac{1}{\alpha + \frac{(1-\alpha)}{n}} = 2.2$, где $\alpha = 0.35$ — доля последовательного кода.

Для анализа производительности последовательной программы и измерения доли последовательного кода был использован инструмент для профилирования Intel Vtune Amplifier [24].

ЗАКЛЮЧЕНИЕ

Разработанные модели и алгоритмы позволяют моделировать распространение нестационарных тепловых полей в мерзлом грунте от добывающих скважин на кустовых площадках с учетом возможного размещения охлаждающих устройств вокруг скважины. Проведенные расчеты показали возможность получения долгосрочного прогноза о динамике изменения границы вечной мерзлоты для различных вариантов эксплуатации скважины с использованием дополнительных технических систем и теплоизоляционных материалов. Усложнения, вносимые в модель за счет учета различных дополнительных источников холода (применение СОУ) и использования теплоизоляции, приводят к увеличению времени расчета. Применение параллельных вычислений при решении таких задач позволило существенно сократить время расчета. Разработан комплекс параллельных программ для многоядерных процессоров с использованием технологии OpenMP.

Благодарности

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект 19-07-00435-а.

СПИСОК ЛИТЕРАТУРЫ

1. *Nelson F.E., Anisimov O.A., Shiklomanov N.I.* Subsidence risk from thawing permafrost // Nature. 2001. V. 410. P. 889–890.

2. *Nelson F.E., Anisimov O.A., Shiklomanov N.I.* Climate Change and Hazard Zonation in the Circum-Arctic Permafrost Regions // Natural Hazards. 2002. V. 26. P. 203–225.

3. *Akimova E.N, Filimonov M.Y., Misilov V.E., Vaganova N.A.* Supercomputer modelling of thermal stabilization processes of permafrost soils // 18th Intern. Conf. Geoinformatics: Theoretical and Applied Aspects, Geoinformatics 2019. Kyiv, Ukraine, 13–16 May 2019. P. 15482.

4. *Akimova E. N., Filimonov M.Yu., Misilov V.E., Vaganova N.A*. Simulation of thermal processes in permafrost: parallel implementation on multicore CPU // CEUR Workshop Proceedings. 2018. V. 2274. P. 1–9. URL: http://ceur-ws.org/Vol-2274/paper-01.pdf

5. *Vaganova N., Filimonov M.* Simulation of freezing and thawing of soil in Arctic regions // IOP Conf. Ser.: Earth Environ. Sci. 2017. V. 72. P. 012005. doi:10.1088/1755-1315/72/1/012005. URL: http://iopscience.iop.org/article/ 10.1088/1755-1315/72/1/012005/pdf

6. *Vaganova N.A., Filimonov M.Yu.* Computer simulation of nonstationary thermal fields in design and operation of northern oil and gas fields // AIP Conference Proceedings. 2015. V. 1690. P. 020016. doi: 10.1063/1.4936694

7. *Ваганова Н.А., Филимонов М.Ю.* Моделирование эксплуатации инженерных систем в условиях вечной мерзлоты // Вестник НГУ. Сер. Математика, механика, информатика. 2013. Т. 13. № 4. С. 37–42. URL: http://mathnet.ru/links/c888886340a471d696d18d1435e5eaf2/vngu312.pdf

8. *Filimonov M.Yu., Vaganova N.A.* Simulation of Technogenic and Climatic Influences in Permafrost // Lecture Notes in Computer Science. 2015. V. 9045. P. 178–185. doi:10.1007/978-3-319-20239-6_18. URL: https://link.springer.com/chapter/10.1007/978-3-319-20239-6_18

9. *Filimonov M., Vaganova N.* Short and Long Scale Regimes of Horizontal Flare System Exploitation in Permafrost // CEUR Workshop Proceedings. 2016. V. 1662. P. 253–260. URL: http://ceur-ws.org/Vol-1662/mod3.pdf

10. *Filimonov M.Yu., Vaganova N.A.* Simulation of Influence of Special Regimes of Horizontal Flare Systems on Permafrost // Lecture Notes in Computer Science. 2019. V. 11386. P. 233–240. doi:10.1007/978-3-030-11539-5_25

11. *Vaganova N.A., Filimonov M.Yu.* Simulation of Cooling Devices and Effect for Thermal Stabilization of Soil in a Cryolithozone with Anthropogenic Impact // Lecture Notes in Computer Science. 2019. V. 11386. P. 580–587. doi:10.1007/978-3-030-11539-5_68

12. *Vaganova N., Filimonov M.* Parallel splitting and decomposition method for computations of heat distribution in permafrost // CEUR Workshop Proceedings. 2015. V. 1513. P. 42–49. URL: http://ceur-ws.org/Vol-1513/paper-05.pdf

13. *Берсенев А.Ю., Ваганова Н.А., Васёв П.А., Игумнов А.С., Филимонов М.Ю.* Кластерные вычисления как сервис на примере задачи моделирования тепловых полей от скважин на северных нефтегазовых месторождениях // Научный сервис в сети Интернет: многообразие суперкомпьютерных миров: труды Международной суперкомпьютерной конференции (22–27 сентября 2014 г., г. Новороссийск). М.: Изд-во МГУ, 2014. С. 147–151. URL: http://agora.guru.ru/abrau2014/pdf/147.pdf

14. *Ваганова Н.А., Васев П.А., Гусарова В.В., Игумнов С.Т., Филимонов М.Ю.* Использование облачных технологий при моделировании эксплуатации северных нефтегазовых месторождений // Труды ИМех УрО РАН «Проблемы механики и материаловедения». Материалы конференции «Актуальные проблемы математики, механики, информатики». Ижевск, 3–5 марта 2014. 2014. Ижевск: ИМ УрО РАН. С. 23–28.

15. *Самарский А.А., Вабищевич П.Н.* Вычислительная теплопередача. М.: Едиториал УРСС, 2003. 784 с.

16. *Самарский А.А., Моисеенко Б.Д.* Экономическая схема сквозного счета для многомерной задачи Стефана // ЖВМиМФ. 1965. Т. 5. № 5. С. 816–827.

17. *Filimonov M.Yu., Vaganova N.A*. On Boundary Conditions Setting for Numerical Simulation of Thermal Fields Propagation in Permafrost Soils // CEUR-WS Proceedings. 2018. Vol. 2109. P. 115–122. URL: http://ceur-ws.org/Vol-2109/paper-04.pdf

18. *Ваганова Н.А., Филимонов М.Ю.* Долгосрочное прогнозирование динамики зон оттаивания многолетнемерзлых пород в устье куста добывающих сква-

жин // XXXI Сибирский теплофизический семинар, посвященный 100-летию академика С.С. Кутателадзе: сб. тр. Всероссийской конференции. Новосибирск: ИТ СО РАН, 2014. С. 42–48.

19. *Башуров Вл.В., Ваганова Н.А., Филимонов М.Ю.* Численное моделирование процессов теплообмена в грунте с учетом фильтрации жидкости // Вычислительные технологии. 2011. Т. 16. №. 4. С. 3–18.

20. *Voevodin V.V., Voevodin Vl.V.* Parallel computing // St. Petersburg: BHV-Petersburg, 2002. 608 p.

21. *Rodrigue G.* (Ed.) Parallel computations. Vol. 1. Elsevier. 2014.

22. *Akimova E.N., Filimonov M.Yu., Misilov V.E., Vaganova N.A.* Simulation of thermal processes in permafrost: parallel implementation on multicore CPU // CEUR Workshop Proceedings. 2018. Vol. 2274. P. 1–9. URL: http://ceur-ws.org/Vol-2274/paper-01.pdf

23. *Chandra R., Dagum L., Kohr D., Menon R., Maydan D., McDonald J.* Parallel programming in OpenMP. Morgan Kaufmann. 2001.

24. Intel Developer Zone. Intel VTune Amplifier. URL: https://software. intel. com/en-us/intel-vtune-amplifier-xe-support/documentation.

APPLICATION OF SUPERCOMPUTER TECHNOLOGIES FOR LONG-TERM MODELING OF PERMAFROST BOUNDARIES IN THE OIL AND GAS FIELDS OF THE ARCTIC

M. Yu. Filimonov[1], N. A. Vaganova [2], E. N. Akimova [3], V. E. Misilov[4]

Ural Federal University

N.N. Krasovskii Institute of Mathematics and Mechanics of UrB RAS

[1]fmy@imm.uran.ru, [2]vna@imm.uran.ru, [3]aen15@yandex.ru, [4]v.e.misilov@urfu.ru

Abstract

A model of propagation of thermal fields in permafrost from various engineering objects operating in Arctic regions is considered. The proposed model includes the most significant technical and climatic parameters affecting the formation of thermal fields in the surface layer of the soil. The main objective of the study is a long-term forecasting of changes in the dynamics of permafrost boundaries during operation of cluster sites of northern oil and gas fields. Such a forecast is obtained by simulation of complex system consisting of heat or cold sources and frozen soil, thawing of which can lead to the loss of the bearing capacity and possible technogenic and environmental accidents. For example, the sources of heat can be production wells, and the sources of cold can be seasonal cooling devices that are used to stabilize the soil. To minimize the impact of heat sources on permafrost, various options for thermal insulation are used, and to preserve the original temperature regime of the top layer of soil, riprap materials consisting of sand, concrete, foam concrete, or other heat insulating material are used. The developed set of programs was used in the design of 12 northern oil and gas fields. To solve the described problem in a complex three-dimensional area, substantial computational resources are required. The computing time of one variant can often exceed 10–20 hours of machine time on a supercomputer. To speed up the numerical calculations, multi-core processors are used. Numerical calculations illustrate the possibility of a developed set of programs for making long-term forecasts for determining changes in the boundaries of the permafrost zones, and show that on multi-core processors it is possible to achieve acceleration close to the theoretical one.

***Keywords**: computer software, heat and mass transfer, cryolithozone, simulation, parallel computing, Stefan problem, OpenMP*

REFERENCES

1. *Nelson F.E., Anisimov O.A., Shiklomanov N.I.* Subsidence risk from thawing permafrost // Nature. 2001. V. 410. P. 889–890.

2. *Nelson F.E., Anisimov O.A., Shiklomanov N.I.* Climate Change and Hazard Zonation in the Circum-Arctic Permafrost Regions // Natural Hazards. 2002.V. 26. P. 203–225.

3. *Akimova E.N, Filimonov M.Y., Misilov V.E., Vaganova N.A.* Supercomputer modelling of thermal stabilization processes of permafrost soils // 18th Intern. Conf. Geoinformatics: Theoretical and Applied Aspects, Geoinformatics 2019. Kyiv, Ukraine, 13–16 May 2019. P. 15482.

4. *Akimova E. N., Filimonov M.Yu., Misilov V.E., Vaganova N.A.* Simulation of thermal processes in permafrost: parallel implementation on multicore CPU // CEUR Workshop Proceedings. 2018. V. 2274. P. 1-9. URL:http://ceur-ws.org/Vol-2274/paper-01.pdf

5. *Vaganova N., Filimonov M.* Simulation of freezing and thawing of soil in Arctic regions // IOP Conf. Ser.: Earth Environ. Sci. 2017. V. 72. P. 012005. doi:10.1088/1755-1315/72/1/012005 URL: http://iopscience.iop.org/article/10.1088/1755-1315/72/1/012005/pdf

6. *Vaganova N.A., Filimonov M.Yu.* Computer simulation of nonstationary thermal fields in design and operation of northern oil and gas fields // AIP Conference Proceedings. 2015 V. 1690. P. 020016. doi: 10.1063/1.4936694

7. *Vaganova N.A., Filimonov M.Yu.* Modelirovanie ekspluatatsii inzhenernykh system v usloviyakh vechnoi merzloty // Vestnik NGU. Ser. Mathematics, mechanics, informatics. 2013. T. 13. № 4. C. 37–42.
URL: http://mathnet.ru/links/c888886340a471d696d18d1435e5eaf2/vngu312.pdf

8. *Filimonov M.Yu., Vaganova N.A.* Simulation of Technogenic and Climatic Influences in Permafrost // Lecture Notes in Computer Science. 2015. V. 9045. P. 178–185. doi:10.1007/978-3-319-20239-6_18. URL: https://link.springer.com/chapter/10.1007/978-3-319-20239-6_18

9. *Filimonov M., Vaganova N.* Short and Long Scale Regimes of Horizontal Flare System Exploitation in Permafrost // CEUR Workshop Proceedings. 2016. V. 1662. P. 253–260. URL: http://ceur-ws.org/Vol-1662/mod3.pdf

10. *Filimonov M.Yu., Vaganova N.A.* Simulation of Influence of Special Regimes of Horizontal Flare Systems on Permafrost // Lecture Notes in Computer Science. 2019. V. 11386. P. 233–240. doi:10.1007/978-3-030-11539-5_25

11. *Vaganova N.A., Filimonov M.Yu.* Simulation of Cooling Devices and Effect for Thermal Stabilization of Soil in a Cryolithozone with Anthropogenic Impact // Lecture Notes in Computer Science. 2019. V. 11386. P. 580–587. doi:10.1007/978-3-030-11539-5_68

12. *Vaganova N., Filimonov M.* Parallel splitting and decomposition method for computations of heat distribution in permafrost // CEUR Workshop Proceedings. 2015. V. 1513. P. 42–49. URL:http://ceur-ws.org/Vol-1513/paper-05.pdf

13. *Bersenev A.Iu., Vaganova N.A., Vasev P.A., Igumnov A.S., Filimonov M.Iu.* Klasternye vychisleniia kak servis na primere zadachi modelirovaniia teplovykh polei ot skvazhin na severnykh neftegazovykh mestorozhdeniiakh // Nauchnyi servis v seti Internet: mnogoobrazie superkompiuternykh mirov: trudy Mezhdunarodnoi superkompiuternoi konferentsii (22–27 sentiabria 2014 g., g. Novorossiisk). M.: Izd-vo MGU, 2014. S. 147–151. URL: http://agora.guru.ru/abrau2014/pdf/147.pdf

14. *Vaganova N.A., Vasev P.A., Gusarova V.V., Igumnov S.T., Filimonov M.Iu.* Ispolzovanie oblachnykh tekhnologii pri modelirovanii ekspluatatsii severnykh neftegazovykh mestorozhdenii // Trudy IMekh UrO RAN «Problemy mekhaniki i materialovedeniia». Materialy konferentsii «Aktualnye problemy matematiki, mekhaniki, informatiki». Izhevsk, 3–5 marta 2014. 2014. Izhevsk: IM UrO RAN. S. 23–28.

15. *Samarsky A.A., Vabishchevich P.N.* Computational Heat Transfer. Vol. 2. The Finite Difference Methodology. Chichester, Wiley, 1995. 432 p.

16. *Samarskii A.A., Moiseenko B.D.* Ekonomicheskaia skhema skvoznogo scheta dlia mnogomernoi zadachi Stefana // ZhVMiMF. 1965. V. 5. № 5. S. 816–827.

17. *Filimonov M.Yu., Vaganova N.A.* On Boundary Conditions Setting for Numerical Simulation of Thermal Fields Propagation in Permafrost Soils // CEUR-WS Proceedings. 2018. V. 2109. P. 115–122. URL: http://ceur-ws.org/Vol-2109/paper-04.pdf

18. *Vaganova N.A., Filimonov M.Yu.* Dolgosrochnoe prognozirovanie dinamiki zon ottaivaniia mnogoletnemerzlykh porod v uste kusta dobyvaiushchikh skvazhin // XXXI Sibirskii teplofizicheskii seminar, posviashchennyi 100-letiiu akademika S.S. Kutateladze: sb. tr. Vserossiiskoi konferentsii. Novosibirsk: IT SO RAN, 2014. S. 42–48.

19. *Bashurov Vl.V., Vaganova N.A., Filimonov M.Yu.* Chislennoe modelirovanie protsessov teploobmena v grunte s uchetom filtratsii zhidkosti // Vychislitelnye tekhnologii. 2011. V. 16. No 4. S. 3–18.

20. *Voevodin V.V., Voevodin Vl.V.* Parallel computing // St. Petersburg: BHV-Petersburg, 2002. 608 p.

21. *Rodrigue G.* (Ed.) Parallel computations. Vol. 1. Elsevier. 2014.

22. *Akimova E.N., Filimonov M.Yu., Misilov V.E., Vaganova N.A.* Simulation of thermal processes in permafrost: parallel implementation on multicore CPU // CEUR Workshop Proceedings. 2018. V. 2274. P. 1–9. URL: http://ceur-ws.org/Vol-2274/paper-01.pdf

23. *Chandra R., Dagum L., Kohr D., Menon R., Maydan D., McDonald, J.* Parallel programming in OpenMP. Morgan Kaufmann. 2001.

24. Intel Developer Zone. Intel VTune Amplifier. URL: https://software. intel. com/en-us/intel-vtune-amplifier-xe-support/documentation

СВЕДЕНИЯ ОБ АВТОРАХ

ФИЛИМОНОВ Михаил Юрьевич – заведующий кафедрой высокопроизводительных компьютерных *технологий Уральского федерального университета, ведущий научный сотрудник Института математики и механики им. Н.Н. Красовского УрО РАН. Сфера научных интересов – математическое моделирование и информатика.*

Mikhail Yurievich FILIMONOV – Head of the Department of High-Performance Computer Technologies of the Ural Federal University, Leading Researcher at N.N. Krasovskii Institute of Mathematics and Mechanics of the Ural Branch of the Russian Academy of Sciences. Research interests include mathematical modeling and computer science.

email: fmy@imm.uran.ru

ВАГАНОВА Наталия Анатольевна – старший научный сотрудник *Уральского федерального университета и Института математики и механики им. Н.Н. Красовского УрО РАН. Сфера научных интересов – информационные технологии и вычислительная математика.*

Nataliia Anatolyevna VAGANOVA – Senior Researcher, Ural Federal University and N.N. Krasovskii Institute of Mathematics and Mechanics of the Ural Branch of the Russian Academy of Sciences. Research interests include information technology and computational mathematics.

email: vna@imm.uran.ru

АКИМОВА Елена Николаевна – профессор кафедры Информационных технологий и систем управления Института радиоэлектроники и информационных технологий – РтФ Уральского федерального университета, ведущий научный сотрудник Института математики и механики им. Н.Н. Красовского УрО РАН. Сфера научных интересов – численные методы, некорректные задачи, параллельные вычисления.

Elena Nikolaevna AKIMOVA – Professor of Department of Information Technologies and Control System of the Institute of Radioelectronics and Information Technologies of the Ural Federal University; Leading researcher at N.N. Krasovskii Institute of Mathematics and Mechanic of the Ural Branch of Russian Academy of Sciences. Research interests include numerical methods, ill-posed problems, parallel computing.

email: aen15@yandex.ru

МИСИЛОВ Владимир Евгеньевич – научный сотрудник Института математики и механики им. Н.Н. Красовского УрО РАН, доцент Уральского федерального университета. Сфера научных интересов — обратные и некорректные задачи, численные методы, параллельные вычисления.

Vladimir Evgenevich MISILOV – Research scientist at the Krasovskii Institute of Mathematics and Mechanics, Ural Branch of Russian Academy of Sciences; docent at the Ural Federal University, Ekaterinburg, Russia. His research interests are in ill-posed problems, numerical methods, parallel computing.

email: v.e.misilov@urfu.ru

Материал поступил в редакцию 29 октября 2019 года

УДК 004.432

ОТЛАДКА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ В DVM-СИСТЕМЕ

В.А. Бахтин [1,2], Д.А. Захаров [1], А.А. Ермичев [1,2], В.А. Крюков [1,2]

[1] Институт прикладной математики им. М.В. Келдыша Российской академии наук, г. Москва;
[2] Московский государственный университет им. М.В. Ломоносова, г. Москва

bakhtin@keldysh.ru, s123-93@mail.ru, a.ermich@gmail.com, krukov@keldysh.ru

Аннотация

DVM-система предназначена для разработки параллельных программ научно-технических расчетов на языках C-DVMH и Fortran-DVMH. Эти языки используют единую DVMH-модель параллельного программирования и являются расширением стандартных языков Си и Фортран спецификациями параллелизма, оформленными в виде директив для компилятора. DVMH-модель позволяет создавать эффективные параллельные программы для гетерогенных вычислительных кластеров, в узлах которых в качестве вычислительных устройств наряду с универсальными многоядерными процессорами могут использоваться ускорители, графические процессоры или сопроцессоры Intel Xeon Phi. В статье описаны методика отладки параллельных программ в DVM-системе, а также новые возможности DVM-отладчика.

Ключевые слова*: автоматизация разработки параллельных программ, автоматизация отладки параллельных программ, динамический контроль, сравнительная отладка, DVM-система, ускоритель, ГПУ, Фортран, Си.*

ВВЕДЕНИЕ

Ручные методы отладки, такие, как пошаговое исследование процесса выполнения алгоритма в определенных точках останова или отладочная печать, не позволяют адекватно работать с реальными научными и инженерными параллельными программными комплексами, спроектированными на непрерывную работу в течение часов или даже дней. Параллельные алгоритмы обычно значительно сложнее последовательных вариантов решения тех же задач. Более того, параллельный код может содержать нетипичные для последовательной отладки

ошибки, связанные с некорректным использованием примитивов синхронизации, функций, обеспечивающих параллелизм.

Следует отметить сложность отладки параллельных программ из-за:

- необходимости отслеживать состояние нескольких (очень многих) параллельных процессов/нитей;
- сложности повторного воспроизведения ошибок, вызванной недетерминированностью выполнения;
- влияния отладочных средств на процесс выполнения (меняется время выполнения операторов, возможны внутренние синхронизации).

Для отладки параллельных программ разработаны автоматизированные методы, которые позволяют найти большинство ошибок в программе в автоматическом режиме с минимальным участием программиста. Одним из таких методов является динамический контроль корректности. Данный метод используется во многих инструментах отладки многопоточных программ: Helgrind[1], DRD[2], Intel Parallel Inspector[3]. В DVM-системе этот метод используется при отладке параллельных программ для кластеров с ускорителями.

Другим методом автоматизированной отладки является сравнительная отладка параллельных программ. Принцип работы данного метода заключается в сравнении процесса выполнения двух программ, посредством контроля значений переменных в определенных контролируемых точках. Сравнение может проводиться как между параллельно запущенными программами, так и с использованием файлов трассы, в которые заносятся все необходимые данные об операциях и значениях переменных в контролируемых точках. Подобные методы отладки были реализованы в отладчиках Guard[4] и Wizard[5], в описании которых впервые был использован термин «сравнительная отладка». Этот метод был реализован и в DVM-системе.

При использовании отладчиков Guard и Wizard контролируемые точки, в которых происходит сравнение значений переменных, задаются пользователем. В DVM-отладчике точки сравнения значений устанавливаются автоматически.

В статье кратко описана методика отладки DVMH-программ, представлены проблемы, возникающие при сравнительной отладке DVMH-программ, и предложены пути преодоления данных проблем.

МЕТОДИКА ОТЛАДКИ DVMH-ПРОГРАММ

Для применения автоматизированных методов отладки требуется инструментация параллельной программы. В программу добавляются специальные вызовы к отладчику, которые позволяют контролировать ход ее выполнения. Для выполнения инструментации возможны различные подходы: инструментация бинарного кода, инструментация исходного кода. Отладчики Helgrind, DRD, Intel Parallel Inspector основаны на бинарной инструментации. При использовании высокоуровневых моделей программирования (например, OpenMP [9]) применение бинарной инструментации приводит к определенным сложностям. Так, по бинарному коду программы инструментатор должен восстановить спецификации параллелизма, которые были в программе изначально, чтобы иметь возможность выдавать ошибки в терминах программы пользователя. Для этого инструментатор должен знать логику работы компилятора. Если компилятор и отладчик разработаны одной компанией (например, Intel), то такое восстановление возможно. Если компилятор разработан одной компанией (например, Microsoft), а отладчик другой (например, Intel), то восстановление спецификаций параллелизма может быть затруднено. Избежать данной проблемы можно, если использовать инструментацию исходного кода.

В DVM-системе инструментация отлаживаемых программ выполняется компиляторами с языков C-DVMH [6] и Fortran-DVMH [7], которые добавляют обращения к функциям DVM-отладчика в следующих точках программы:

- начало последовательного или параллельного циклов;
- завершение последовательного или параллельного циклов;
- начало нового витка цикла;
- обращение к переменной на чтение;
- перед обращением к переменной на запись;
- после обращения к переменной на запись.

Кроме того, обращения к функциям отладчика производятся также и при выполнении многих функций библиотеки Lib-DVMH [8].

Динамический контроль DVMH-указаний основан на анализе последовательности вызовов функций Lib-DVMH и обращений к переменным. Динамический контроль позволяет выявлять ошибки следующих типов:

- необъявленная зависимость по данным в параллельном цикле;
- некорректное использование приватных и редукционных переменных;
- необъявленный доступ к нелокальным элементам распределенного массива;
- некорректная работа с теневыми гранями редукционного массива
- модификация нелокального элемента распределенного массива в последовательной части программы;
- выход за пределы распределенного массива;
- запись в буфер удаленного доступа.

Следует отметить, что не все ошибки могут быть определены в результате динамического контроля. Например, с помощью динамического контроля не могут быть проанализированы процедуры и функции, для которых отсутствует исходный код. Для поиска ошибок в таких программах может быть использован метод сравнительной отладки, который более детально будет рассмотрен в следующем разделе.

СРАВНИТЕЛЬНАЯ ОТЛАДКА DVMH-ПРОГРАММ

Сравнение промежуточных результатов выполнения позволяет обнаруживать ошибки, возникающие из-за некорректных DVMH-указаний, а также ошибки программы, которые не проявлялись при ее последовательном выполнении и не были обнаружены методом динамического контроля.

Общая схема сравнительной отладки выглядит следующим образом:

1) Получение эталонной трассировки (команда **./dvm trc**). При трассировке выполняется сбор информации обо всех чтениях и модификациях переменных, о начале выполнения каждого витка цикла, о начале и конце выполнения параллельного цикла, о начале выполнения каждой параллельной задачи, о начале и конце выполнения группы задач. В качестве эталонной трассировки может выступать трасса последовательного выполнения программы (т. к. DVMH-программа одновременно может быть и последовательной, и параллельной); трасса параллельного выполнения программы, в т.ч. трасса, полученная на другом вычислительном кластере, на котором ошибка не проявляется.

2) Автоматическое сравнение результатов выполнения программы с накопленной ранее эталонной трассировкой (команда **./dvm dif**). Все целочисленные данные сравниваются на совпадение, а вещественные числа сравниваются с заданной точностью по абсолютной и относительной погрешностям. В случае нахождения расхождений выдается информация о найденных различиях.

Файл трассировки создается для каждого процесса и имеет текстовый формат. При накоплении трассировки в начало файла помещается специальный заголовок (рис. 1). Данный заголовок содержит параметры, которые использовались при накоплении трассировки. Также в заголовке сохраняется структура вложенности циклов и областей задач программы в том виде, в котором она представляется отладчику (она определяется по последовательности обращений к нему и передаваемым параметрам, таким, как номер конструкции).

```
<параметры запуска трассировки (архитектура системы, директории)>
MODE = <NONE | MINIMAL | MODIFY | FULL>
<SL | PL | TR> <номер конструкции> (<номер объемлющей конструкции>) [<ранг
цикла>] {<имя файла>, <номер строки>} = <NONE | MINIMAL | MODIFY | FULL>,
(<измерение>:<первый виток>, <последний виток>, <шаг>), …
EL: <номер конструкции>
<SL | PL | TR> <номер конструкции> (<номер объемлющей конструкции>) [<ранг
цикла>] {<имя файла>, <номер строки>} = <NONE | MINIMAL | MODIFY | FULL>,
(<измерение>:<первый виток>, <последний виток>, <шаг>), …
EL: <номер конструкции>
END_HEADER
```

Рис. 1. Формат заголовка файла трассы

Модифицируя значение MODE, можно контролировать глобальный уровень подробности трассировки программы, также можно задавать подробность трассировки для каждого цикла/параллельной области DVMH-программы, поставив соответствующий режим в строке, описывающей нужную конструкцию программы в заголовке.

После заголовка в файле содержится последовательность записей, формирующих собственно трассу. Эти записи определяют динамическую структуру программы, то есть последовательность операторов в процессе выполнения программы. На рис. 2 и 3 представлен полный список событий, которые заносятся в трассу.

```
- чтение переменной:
RD: [<тип переменной>] <имя переменной> = <значение>; {<файл>, <строка>}

- начало оператора присваивания нового значения переменной:
BW: [<тип переменной>] <имя переменной>; {<файл>, <строка>}

- запись нового значения переменной:
AW: [<тип переменной>] <имя переменной> = <значение>; {<файл>, <строка>}

- чтение редукционной переменной:
RV_RD: [<тип переменной >] <имя переменной> = <значение>; {<файл>, <строка>}

- начало оператора присваивания нового значения редукционной переменной:
RV_BW: [<тип переменной>] <имя переменной>; {<файл>, <строка>}

- запись нового значения редукционной переменной:
RV_AW: [<тип переменной>] <имя переменной> = <значение>; {<файл>, <строка>}

- результат вычисления редукции:
RV: [<тип переменной>] <значение>; {<файл>, <строка>}
```

Рис. 2. Формат записей событий взаимодействий с переменными

```
- начало параллельного цикла:
PL: <номер конструкции> (<номер объемлющей конструкции>) [<ранг цикла>];
{<файл>, <строка>}

- начало последовательного цикла:
SL: <номер конструкции> (<номер объемлющей конструкции>) [<ранг цикла
(всегда равен единице)>]; {<файл>, <строка>}

- начало области параллельных задач:
TR: <номер конструкции> (<номер объемлющей конструкции>) [<ранг области
 (всегда равен единице)>]; {<файл>, <строка>}

- начало витка или параллельной задачи:
IT: <абсолютный индекс витка (вычисляется из значений всех индексных
переменных цикла) или номер задачи>, (<значение индексной переменной 1>,<
значение индексной переменной 2>,…).

- конец параллельного или последовательного цикла или области задач:
EL: <номер конструкции>; {<файл>, <строка>}

- конец блока собственных вычислений в последовательной части программы:
SKP: {<файл>, <строка>}
```

Рис. 3. Формат записей, описывающих циклы и параллельные области

На рис. 5 показана трасса выполнения тестовой программы, листинг которой приведен на рис. 4. Данная программа состоит из одного параллельного цикла, который начинается на 7-ой строке файла test.c и выполняет инициализацию элементов распределенного массива A. В результате выполнения этого цикла элементы массива A[0] и A[2] получают значение «0», а элемент A[1] становится равным «3».

```
#define L 3
int main(int an, char **as)
{
    #pragma dvm array distribute[block]
    double A[L];
    #pragma dvm parallel([i] on A[i])
    for (int i = 0; i < L; i++)
    {
        if (i == 0 || i == L - 1)
            A[i] = 0;
        else
            A[i] = 2 + i;
    }
    return 0;
}
```

Рис. 4. Фрагмент тестовой программы на языке C-DVMH (файл test.c)

Текстовый формат для файла трассировки выбран не случайно. Такой формат позволяет переносить трассы, полученные на одной вычислительной системе, на другую вычислительную систему, не заботясь о необходимости преобразования данных (различия в размере типов данных, порядке байт в слове и т. п.), а также позволяет, при необходимости, вручную проанализировать ход выполнения отлаживаемой программы (вместо добавления в программу операторов печати).

```
# Begin trace header. Don't modify these records
TRACE_TIME = "Mon Nov 15 00:00:22 2019"
ARCHITECTURE = "Machine x86_64"
USER_HOST = "DVM-COREI7@DVM-COREI7"
WORK_DIR = "/DVM/dvm_current/dvm_sys/demo"
TASK_NAME = "test"
MODE = FULL
PL: 1() [1] {"test.c", 7} = #, (0:0,2,1)
EL: 1
END_HEADER
# End trace header
PL: 1() [1]; {"test.c", 7}, 1.B
  IT: 0, (0)
  BW: [4] "A[i]"; {"test.c", 10}
  AW: [4] "A[i]" = 0; {"test.c", 10}
  IT: 1, (1)
  BW: [4] "A[i]"; {"test.c", 12}
  AW: [4] "A[i]" = 3; {"test.c", 12}
  IT: 2, (2)
  BW: [4] "A[i]"; {"test.c", 10}
  AW: [4] "A[i]" = 0; {"test.c", 10}
EL: 1; {"test.c", 7}, 1.E
END_TRACE
```

Рис. 5. Файл трассы выполнения тестовой программы

Сравнительная отладка показала свою эффективность на простых модельных задачах, но натолкнулась на два препятствия: ресурсы и точность. Рассмотрим подробней суть этих проблем.

ПРОБЛЕМЫ СРАВНИТЕЛЬНОЙ ОТЛАДКИ

После полной инструментации каждый оператор окружается несколькими вызовами подпрограмм трассировщика, которые должны сформировать записи с читаемыми и модифицируемыми значениями переменных. Неудивительно, что при этом время выполнения программы может значительно увеличиться. Например, в результате экспериментов с программами из пакета NAS NPB [11] были получены следующие результаты:

- замедление только от отладочной инструментации (т. е. программа была скомпилирована для отладки, но выполнена без сбора/сравнения трассы) составило 50–100 раз;
- объемы трассы (в среднем несколько десятков байтов на каждый выполненный оператор присваивания) оказались совершенно неприемлемыми для реальных программ; средний размер трассы – порядка нескольких терабайт, и среднее время сбора трасы ~28000 сек. (при среднем времени выполнения исходных программ ~5 сек., т. е. замедление в ~4000 раз!).

Поэтому полная сравнительная отладка оказалась применимой только на «модельных» данных, которые не всегда доступны. Для преодоления данной проблемы были разработаны и реализованы в отладчике DVM-системы средства управления и оптимизации трассы [12]:

- выборочная трассировка, например: только запись, только распределенные массивы и т. п. (режимы **-d1**…**-d4**, которые можно задать при компиляции DVMH-программы);
- локализация инструментации, т. е. выделение отдельных фрагментов программы, инструментируемых для отладки (директивы **DEBUG**<режим отладки>/**END DEBUG**);
- предварительная оценка объема трассы, получение заголовка трассы и его ручная коррекция для отключения трассировки на определенных итерациях и т. д. (команда **./dvm size**);

- автоматический выбор итераций параллельных циклов для трассировки: «границы» и «уголки» (режимы **-dbif1**, **-dbif2**, которые можно задать при компляции DVMH-программы);
- генерация двух тел цикла: одно – без вызовов трассировщика, другое инструментировано; при этом на выбранных для трассировки итерациях работает инструментированное тело цикла, на остальных – исходная программа;
- условный вызов подпрограмм трассировщика (т. е. перед вызовом подпрограммы трассировщика проверяется необходимость ее вызова);
- трассировка контрольных сумм массивов по завершении цикла вместо трассировки элементов массивов при выполнении тела цикла (параметр **TraceOptions.CalcChecksums**).

Использование данных средств расширяет возможности применения сравнительной отладки для реальных приложений. Например, при обработке больших массивов данных в циклах наиболее вероятным местом возникновения и проявления многих ошибок (использование неинициализированных переменных, выход за границы массива) могут стать граничные итерации. Вычислительные алгоритмы часто таковы, что ошибка, возникшая на внутренней итерации цикла, проявляется и на граничной. Для таких задач можно использовать метод выборочной трассировки и сравнения граничных итераций цикла (опция – **dbif**<level>), который позволяет (таблица 1):

- существенно сократить размер трасс (в сотни – сотни тысяч раз);
- существенно сократить время выполнения программ с генерацией трасс (в десятки – тысячи раз);
- сохранить значительное покрытие операторов программы (более 99%).

Таблица 1. Размер и время генерации трасс для тестов NAS NPB класса A при использовании различных режимов инструментации

	Полная инструментация	**«Уголки» ширины 1**	**«Уголки» ширины 2**	**«Грани» ширины 1**	**«Грани» ширины 2**
Среднее покрытие операторов	100%	99,4%	99,8%	99,8%	99,8%
Средний размер трассы, байт	6,57E+12 (~6 Tb)	4,6E+07 (~44 Mb)	9,4E+08 (~894 Mb)	1,7E+10 (~16 Gb)	6,0E+10 (~56 Gb)
Среднее время генерации трассы, сек.	27915 (7,75 ч.)	15	19	105	287

Тем не менее, сравнительная отладка научно-технических программных комплексов, особенно на реальных данных (а не искусственно подобранных «модельных» тестах небольшого размера) все еще не доступна в полном объеме.

Другая проблема, выявленная в процессе эксплуатации системы отладки, – «допустимое» несовпадение значений переменных. Такими переменными являются, например, редукционные переменные. При вычислении суммы элементов распределенного массива изменяется порядок операций: сначала вычисляются локальные суммы, а потом они суммируются в непредсказуемом порядке. Результаты при этом получаются разными (типично расхождение в 1–2 младших разрядах, хотя возможно и более значительное), но, с точки зрения программиста, эти результаты могут быть одинаково допустимыми. Редукционные операции создают четыре проблемы для сравнительной отладки («ложные тревоги»):

- значения редукционной переменной на промежуточных итерациях не совпадают, потому что вычисляются только частичные суммы или максимум ищется в другом диапазоне и т. п.;
- окончательное значение суммы, как сказано выше, может отличаться (недетерминизм в слабом смысле);
- отличие в одной переменной может сразу распространиться на многие другие (например, если посчитана норма вектора, и затем вектор нормируется);

- если значение переменной, на которую повлияло различие результатов редукции, используется в критерии окончания итераций или при выборе ветви вычислений, то дальше могут быть выбраны разные ветви, и трассы вообще могут оказаться несопоставимыми (недетерминизм в сильном смысле).

Данные проблемы были решены введением особого режима обработки редукции при отладке:

- редукционные переменные распознаются (т. к. они описаны в DVMH-директивах), и операции их чтения и модификации в теле цикла игнорируются сравнительной отладкой;
- после достижения конца параллельного цикла в трассу заносится итоговое значение редукционной переменной (конструкция RV), которое и участвует в сравнении;
- значения редукционных переменных сравниваются с некоторой точностью, которая может быть меньше точности сравнения значений обычных переменных; данная точность может быть задана программистом через специальный конфигурационный параметр;
- в процессе сравнения реального выполнения с трассой выполнения другого варианта реально вычисленное значение редукционной переменной (после успешного сравнения с заданной точностью) заменяется ее значением из «эталонной» трассы для ликвидации потенциально опасного расхождения.

Описанный подход позволил подавить «ложные тревоги», связанные с редукционными переменными. Однако в дальнейшем было обнаружено, что данная проблема может проявиться не только на редукционных переменных. При выполнении отлаживаемой программы на разных машинах или при использовании различных средств компиляции любая арифметическая операция, например, умножение или деление, может вернуть результаты, отличающиеся в одном-двух младших десятичных знаках мантиссы [13].

РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ СРАВНИТЕЛЬНОЙ ОТЛАДКИ

Для обнаружения ошибок, проявляющихся при выполнении программы на графических ускорителях, в DVM-системе реализован специальный режим работы DVMH-программы, при котором все вычисления в регионах одновременно выполняются на центральном процессоре и графическом ускорителе. Сравнение выходных данных, полученных в регионе при выполнении на ускорителе, с данными, полученными в регионе при выполнении на центральном процессоре, позволяет выявить и локализовать ошибки, которые могут возникать по следующим причинам:

1. Программистом произведено некорректное распараллеливание, не подходящее для массивно-параллельного выполнения в общей памяти.
2. Программист некорректно указал приватные или редукционные переменные в параллельном цикле.
3. Арифметические операции или математические функции на ускорителе отработали с результатом, иным по сравнению с работой на центральном процессоре. Это может происходить из-за различий в системе команд, приводящих к различным результатам (в пределах точности округлений).
4. Программист указал неверные директивы актуализации данных **get_actual** и **actual**, вследствие чего обрабатываемые данные на центральном процессоре и ускорителе оказались разными.

При сравнении все целочисленные данные сравниваются на совпадение, а вещественные числа сравниваются с заданной точностью по абсолютной и относительной погрешностям. В случае нахождения расхождений пользователю системы выдается информация о найденных расхождениях. После этого для дальнейшей работы программы берется версия данных, которая была получена при выполнении на центральном процессоре.

Включение и использование данного режима сравнительной отладки не требует от программиста вносить какие-либо изменения в программу, а также заново ее компилировать. Для включения данного режима сравнительной отладки необходимо установить значение переменной окружения

DVMH_COMPARE_DEBUG равным 1 либо использовать команду **./dvm cmph** для запуска программы на выполнение.

Данный режим сравнительной отладки стал очень востребован при разработке DVMH-программ. Одно из главных примуществ данного метода – возможность его использования для больших программных комплексов на реальных данных (т. к. не требуется память для хранения трасс).

В настоящее время в DVM-системе ведется разработка новой версии системы сравнительной отладки [14], которая основана на схожих принципах. Вместо использования файлов с трассами новая версия отладчика организует обмены отладочной информацией между двумя экземплярами программы, которые выполняются одновременно:

1) программа разделяется на блоки (например, в качестве блока могут рассматриваться параллельный цикл программы или вычислительный регион);

2) в процессе выполнения очередного блока происходит накопление очередного отрезка трассы каждым экземпляром отлаживаемой программы;

3) по завершению выполнения блока накопленные трассы пересылаются одному процессу, где и происходит их сравнение.

При использовании данного подхода остается возможность применения всех ранее реализованных оптимизаций, связанных с размером трассы (например, использование интегральных характеристик массивов), так как все изменения, вносимые данными режимами, влияют на трассу локально, внутри одного конкретного блока трассы.

Разбиение параллельной DVMH-программы на блоки позволяет выполнять вычисления внутри блока без дополнительных затрат на вызовы отладчика. Значения всех прочитанных и/или модифицированных переменных могут собираться в конце выполнения блока, а не в процессе его выполнения. Отказ от вызовов отладчика перед/после каждого обращения к переменной на чтение/запись внутри блока позволит существенно ускорить процесс отладки программы.

Для решения проблемы расхождения результатов операций с вещественными числами в новой версии отладчика будет реализован режим, расширяющий описанный ранее режим корректировки редукционных переменных на все

результаты вещественных операций. После успешного сравнения значения вещественной переменной с эталонным данное эталонное значение будет подставлено в выполняющуюся программу и использовано для дальнейших вычислений.

Реализация метода сравнения одновременно выполняющихся программ предполагает использование механизма передачи данных между независимо работающими задачами. Данный механизм был реализован на основе библиотеки, написанной для DVM-системы, предназначавшейся для контроля корректности выполнения программ через механизм контрольных точек.

Одной из особенностей данной библиотеки является модуль, предназначенный для обмена данными между программами по модели клиент-сервер, в том числе, и удаленно. Данный модуль был переработан для интеграции с существующей реализацией отладчика DVM-системы и подключен к ней.

Для реализации новых механизмов были реализованы/модифицированы существующие вызовы отладчика:

- добавлены обработчики, накапливающие список задействованных внутри блока переменных;
- в методы dbegpl_ и dendl_, вызываемые в начале и конце параллельных циклов, был добавлен вызов механизма передачи текущего блока другой программе;
- в структуры данных, использующиеся для накопления трассировки, была добавлена возможность хранения, обработки отдельных блоков трассы.

В таблице 1 представлены первые результаты тестирования новой версии системы сравнительной отладки на программе Якоби (одной из тестовых программ DVM-системы), где block – это максимальный размер одного блока, передаваемого между эталонной и отлаживаемой программами, а all – сумарный объем данных, переданных в процесса отладки.

Таблица 1. Сравнение экспериментальной реализации системы сравнительной отладки с существующей версией

	N = 10 100 итераций		N = 100 100 итераций		N = 1000 1000 итераций	
	Размер Трассы	Время работы	Размер трассы	Время работы	Размер трассы	Время работы
Без отладки	-	0.42 s	-	0.45 s	-	3.02 s
Существующая Реализация	4.6 Mb	0.5 s	694 Mb	9.35 s	~800 Gb	~12000 s
Сравнение одновременно выполняющихся программ	1.08 Mb (all) 7.4 K (block)	0.92 s	163 Mb (all) 1.1 Mb (block)	31.2 s	~180 Gb (all) 113 Mb (block)	~18000 s

Тестирование экспериментальной реализации нового режима показало в принципе успешный результат – максимальный объем памяти, требующийся отладчику для корректной работы, снизился до размеров одного блока. Время выполнения отладки увеличилось за счет необходимости постоянной передачи блоков данных, но это не является препятствием для отладки программных комплексов на реальных данных.

ЗАКЛЮЧЕНИЕ

Система автоматизации разработки параллельных программ (DVM-система) существенно упрощает процесс разработки параллельных программ для гетерогенных кластеров с ускорителями. Важным преимуществом DVM-системы является наличие мощных инструментов для отладки, получаемых в процессе распараллеливания программ. Поддерживаются различные варианты сравнительной отладки: с использованием трасс; сравнение результатов выполнения программы на центральном процессоре и графическом ускорителе «на лету».

В настоящее время ведется разработка новой версии системы сравнительной отладки, в которой и эталонная, и отлаживаемая программы будут выполняться одновременно, и сравнение результатов выполнения будет осуществляться «на лету» без трасс. Создаваемая версия системы отладки решит проблему точности сравнения, сделает процесс отладки более гибким и менее требовательным к памяти вычислительного комплекса. Новая версия системы сравнительной

отладки будет допускать как одновременный запуск эталонной и отлаживаемой программ на одной многопроцессорной машине, так и удаленную отладку – сравнение выполнения эталонной программы на одной машине с экспериментальной версией, работающей на другом вычислительном комплексе и использующей иные средства компиляции, при наличии технической возможности создания сетевого соединения между двумя машинами.

Разработка новой системы существенно расширит возможности применения сравнительной отладки для реальных приложений на реальных данных.

СПИСОК ЛИТЕРАТУРЫ

1. Helgrind: a thread error detector. URL: http://www.valgrind.org/docs/ manual/ hg-manual.html

2. DRD: a thread error detector. URL: http://www.valgrind.org/ docs/ manual/drd-manual.html

3. Intel Inspector. Memory and Thread Debugger. URL: https:// software.intel.com/en-us/intel-inspector.

4. Guard Parallel Relative Debugger. URL: http://sourceforge.net/ projects/guardsoft/

5. *Abramson D.A., Sosic R.* Relative Debugging using Multiple Program Versions // Intensional Programming I. Sydney: World Scientific. 1995.

6. Язык C-DVMH. C-DVMH компилятор. Компиляция, выполнение и отладка CDVMH-программ. URL: http://dvm-system.org/static_data/docs/CDVMH-reference-ru.pdf

7. Язык Fortran-DVMH. Fortran-DVMH компилятор. Компиляция, выполнение и отладка DVMH-программ. URL: http://dvm-system.org/static_data/docs/FDVMH-user-guide-ru.pdf

8. Система поддержки выполнения параллельных программ (библиотека Lib-DVM). URL: http://www.keldysh.ru/dvm/dvmhtm1107/rus/sys/libdvm/rtsDDr0.html

9. OpenMP Application Programming Interface. Version 5.0. November, 2018. URL: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf

10. The OpenACC Application Programming Interface. Version 2.6. November, 2017. URL: https://www.openacc.org/sites/default/files/inline-files/OpenACC.2.6.final.pdf

11. NAS Parallel Benchmarks, URL: http://www.nas.nasa.gov/publications/npb.html

12. *Крюков В.А., Кудрявцев М.В.* Автоматизация отладки параллельных программ // Вычислительные методы и программирование. 2006. Т. 7. Вып. 4. С. 102–110.

13. *David Monniaux.* The pitfalls of verifying floating-point computations // ACM Transactions on Programming Languages and Systems (TOPLAS), ACM. 2008. V. 30. No 3. 12 p.

14. *Ермичев А.А., Крюков В.А.* Развитие метода сравнительной отладки DVMH-программ // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18–23 сентября 2017г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2017. С. 150–156.

DEBUGGING PARALLEL PROGRAMS IN DVM-SYSTEM

***V.A. Bakhtin*[1,2], *D.A. Zakharov*[1], *A.A. Ermichev*[1,2], *V.A. Krukov*[1,2]**

[1] *Keldysh Institute of Applied Mathematics,*
[2] *Lomonosov Moscow State University*

bakhtin@keldysh.ru, a.ermich@gmail.com, s123-93@mail.ru, krukov@keldysh.ru

Abstract

DVM-system is designed for the development of parallel programs of scientific and technical calculations in the C-DVMH and Fortran-DVMH languages. These languages use a single DVMH-model of parallel programming model and are an extension of the standard C and Fortran languages with parallelism specifications in the form of compiler directives. The DVMH model makes it possible to create efficient parallel programs for heterogeneous computing clusters, in the nodes of which accelerators, graphic processors or Intel Xeon Phi coprocessors can be used as computing devices along with universal multi-core processors. The article describes the method of debugging parallel programs in DVM-system, as well as new features of DVM-debugger.

Keywords*: automation of development of parallel programs, automation of debugging of parallel programs,* dynamic control, relative debugger, *DVM-system, accelerator, GPU, Fortran, C*

REFERENCES

1. Helgrind: a thread error detector. URL: http://www.valgrind.org/docs/ manual/hg-manual.html

2. DRD: a thread error detector. URL: http://www.valgrind.org/docs/ manual/drd-manual.html

3. Intel Inspector. Memory and Thread Debugger. URL: https:// software.intel.com/en-us/intel-inspector

4. Guard Parallel Relative Debugger. URL: http://sourceforge.net/ projects/guardsoft/

5. *Abramson D.A., Sosic R.* Relative Debugging using Multiple Program Versions // Intensional Programming I. Sydney: World Scientific. 1995.

6. C-DVMH language, C-DVMH compiler, compilation, execution and debugging of DVMH programs. URL: http://dvm-system.org/static_data/docs/CDVMH-reference-en.pdf

7. Fortran DVMH langauge, Fortran DVMH compiler, compilation, execution and debugging of DVMH programs. URL: http://dvm-system.org/ static_data/docs/ FDVMH-user-guide-en.pdf

8. Run-time library Lib-DVM. URL: http://www.keldysh.ru/dvm/dvmhtm1107/ eng/sys/libdvm/rtsDDe0.html

9. OpenMP Application Programming Interface. Version 5.0. November, 2018. URL: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf

10. The OpenACC Application Programming Interface. Version 2.6. November, 2017. URL: https://www.openacc.org/sites/default/files/inline-files/ OpenACC.2.6.final.pdf

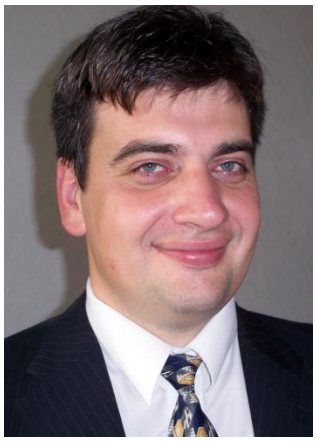
11. NAS Parallel Benchmarks, URL: http://www.nas.nasa.gov/publications/ npb.html

12. *Krukov V.A., Kudryavtsev M.B.* Automated debugging of parallel programs // Vychisl. Metody Programm. 2006. T. 4. S. 102–110.

13. *David Monniaux.* The pitfalls of verifying floating-point computations // ACM Transactions on Programming Languages and Systems (TOPLAS), ACM. 2008. V. 30. No 3. 12 p.

14. *Ermichev A.A., Krjukov V.A.* Razvitie metoda sravnitel'noj otladki DVMH-programm // Nauchnyj servis v seti Internet: trudy XIX Vserossijskoj nauchnoj konferencii (18–23 sentjabrja 2017 g., g. Novorossijsk). M.: IPM im. M.V. Keldysha, 2017. S. 150–156.

СВЕДЕНИЯ ОБ АВТОРАХ



БАХТИН Владимир Александрович – ведущий научный сотрудник ИПМ им. М.В. Келдыша РАН, доцент кафедры системного программирования факультета ВМК МГУ им. М.В. Ломоносова. Сфера научных интересов – математическое обеспечение, программные средства и системы для распределенных вычислений; параллельные алгоритмы; методы, средства и системы обработки данных большого объема.

Vladimir Aleksandrovich BAKHTIN – leading researcher of Keldysh Institute of Applied Mathematics, docent of the faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University. Research interests include mathematical software, software and systems for distributed computing; parallel algorithms; methods, tools and systems of large data processing.

email: bakhtin@keldysh.ru

ЗАХАРОВ Дмитрий Александрович – программист ИПМ им. М.В. Келдыша РАН. Сфера научных интересов – программные средства и системы для распределенных вычислений; параллельные алгоритмы; автоматизация параллельного программирования; распараллеливание программ, использующих неструктурные сетки.

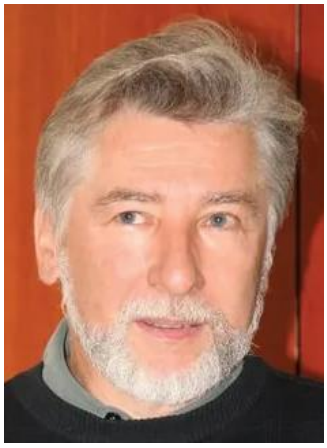
Dmitry Aleksandrovich ZAKHAROV – programmer of Keldysh Institute of Applied Mathematics. Research interests include mathematical software, software and systems for distributed computing; parallel algorithms; automatization of parallel programming; parallelization of unstructured grid applications.

email: s123-93@mail.ru

ЕРМИЧЕВ ***Александр Александрович*** – лаборант ИПМ им. М.В. Келдыша РАН, аспирант кафедры системного программирования факультета ВМК МГУ им. М.В. Ломоносова. Сфера научных интересов – программное обеспечение для распределенных вычислений; распределенные вычислительные системы; параллельные алгоритмы.

Aleksandr Aleksandrovich ERMICHEV – assistant of Keldysh Institute of Applied Mathematics, postgraduate of the Computational Mathematics and Cybernetics faculty of Lomonosov Moscow State University. Research interests include software for distributed computing; distributed computing systems; parallel algorithms.

email: a.ermich@gmail.com



КРЮКОВ Виктор Алексеевич – главный научный сотрудник ИПМ им. М.В. Келдыша РАН, профессор кафедры системного программирования факультета ВМК МГУ им. М.В. Ломоносова. Сфера научных интересов – математическое обеспечение, программные средства и системы для распределенных вычислений; параллельные алгоритмы; методы, средства и системы обработки данных большого объема.

Victor Alekseevich KRUKOV – chief researcher of Keldysh Institute of Applied Mathematics, professor of the faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University. Research interests include mathematical software, software and systems for distributed computing; parallel algorithms; methods, tools and systems of large data processing.

email: krukov@keldysh.ru

Материал поступил в редакцию 13 ноября 2019 года