На правах рукописи

Лебедев Артем Сергеевич

Методы и средства распараллеливания программ линейного класса для выполнения на многопроцессорных вычислительных системах

Специальность 2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Автореферат
диссертации на соискание учёной степени
кандидата технических наук

Одинцово — 2024

Работа выполнена в федеральном государственном бюджетном учреждении науки Центр информационных технологий в проектировании Российской академии наук.

Научный руководитель:	**Солодовников Владимир Игоревич** кандидат технических наук
Официальные оппоненты:	**Портнов Евгений Михайлович** доктор технических наук, профессор, Национальный исследовательский университет «МИЭТ», г. Москва, г. Зеленоград, профессор
	Левченко Вадим Дмитриевич кандидат физико-математических наук, ИПМ им. М.В. Келдыша РАН, г. Москва, ведущий научный сотрудник
Ведущая организация:	ООО «НИЦ супер-ЭВМ и нейрокомпьютеров», г. Таганрог

Защита состоится «18» июня 2024 года в 14:00 часов на заседании диссертационного совета 24.2.417.03, созданного на базе ФГБОУ ВО «Тульский государственный университет» (300012, г. Тула, пр. Ленина, 92, 9-101).

С диссертацией можно ознакомиться в библиотеке и на сайте ФГБОУ ВО «Тульский государственный университет».

https://tulsu.ru/science/dissertation/diss-24-2-417-03/lebedev-as-24-2-417-03

Автореферат разослан «23» апреля 2024 года.

Ученый секретарь диссертационного совета — Маслова Анна Александровна

Общая характеристика работы

Актуальность темы. Повсеместное распространение многоядерных процессоров в мобильных устройствах, рабочих станциях, серверных решениях сделало концепцию параллелизма массовой и доступной. Практика последних лет показала эффективность специализированных вычислителей и ускорителей на основе программируемых логических интегральных схем (ПЛИС) и графических процессорных устройств (ГПУ) применительно к высокопроизводительным вычислениям, однако вычислительные системы, построенные на основе универсальных многоядерных процессоров, в частности вычислительные машины с архитектурой NUMA (Non Uniform Memory Access) и построенные на их основе кластеры, остаются наиболее распространенными и востребованными благодаря их универсальности, наличию развитых инструментов программирования и многообразию библиотек функций.

Эффективное программирование параллельных архитектур всегда было сложной задачей, и особенно усложняется при возрастающих требованиях к быстродействию современного программного обеспечения с объемной кодовой базой, созданной в процессе многолетней разработки. Кроме того, специалисты предметных областей, разрабатывающие программный код для проведения собственных исследований, предпочитают фокусироваться на решении прикладной задачи, а не на технических аспектах параллельного программирования конкретной вычислительной системы, что может повлечь неэффективное использование ресурсов оборудования. Одним из подходов к решению обозначенных проблем является автоматическое распараллеливание программ, исключающее усилия программиста по анализу потока данных в программе, выявлению параллелизма, синтезу параллельного вычислительного кода. Задача автоматического распараллеливания программного кода была сформулирована с момента появления первых параллельных отечественных вычислителей (например, ПС2000). К настоящему времени разработаны языки, модели и инструменты программирования, которые упрощают труд разработчика (DVM-система, САПФОР, НОРМА, OPS, T-система, Erlang, Go), но не делают распараллеливание автоматическим.

Наибольшая вычислительная трудоемкость сосредоточена в циклических конструкциях. Особый интерес для исследователей представляют программы линейного класса, встречающиеся в научных и инженерных приложениях, которые тратят значительную часть времени именно на исполнение гнезд циклов. Модель многогранников предоставляет мощный математический аппарат, упрощающий анализ и преобразование таких программ с целью улучшения их быстродействия путем распараллеливания гнезд циклов и улучшения локальности использования данных при вычислениях. Методы модели многогранников развивались в исследовательских проектах LooPo, PIPS, Pluto, PPCG, C-to-CUDA и были применены разработчиками компиляторов в компонентах GCC Graphite, LLVM Polly. Несмотря на обилие работ в этом направлении, задача

построения оптимальных преобразований, улучшающих быстродействие программы, до конца не решена. Поэтому тема настоящей диссертационной работы, связанной с разработкой методов автоматического распараллеливания линейных программ для вычислительных систем, построенных на основе универсальных многоядерных процессоров, является востребованной и актуальной.

Степень разработанности темы диссертационного исследования. В диссертационном исследовании был выполнен обзор существующих методов и средств преобразования линейных программ для распараллеливания гнезд циклов. Теоретическую базу диссертации в области методов выпуклого анализа и линейного целочисленного программирования составили работы таких ученых, как Черникова Н.В. (алгоритм для нахождения общей формулы неотрицательных решений системы линейных неравенств), Филипп Клаусс (метод подсчета точек с целочисленными координатами внутри многогранника через полиномы Эрхарта), Манфред Падберг и Джованни Ринальди (метод ветвей и отсечений), в области автоматического распараллеливания программ — Воеводин В.В. и Воеводин Вл.В. (анализ программ линейного класса), Кристиан Ленгауэр (модель многогранников), Поль Футриер (таймирование), Удай Бондхугула (локальность использования данных), Мартин Грибль (распределение операций и данных между процессорами), Седрик Бастуль (кодогенерация). Задачи разбиения операций программы на зерна вычислений и построения распараллеливающих трансляторов рассматривались в работах Левченко В.Д., Перепелкиной А.Ю., Крюкова В.А., Бахтина В.А. Отдельные вопросы прогнозирования времени выполнения вычислений рассматривались в работе Ларкина Е.В. и Ивутина А.Н. В настоящей диссертации развиты методы П. Футриера, М. Грибля, У. Бондхугулы для нахождения пространственных и временных отображений программ линейного класса, ориентированные на распараллеливание гнезд циклов вместе с улучшением локальности использования данных.

Целью работы является повышение быстродействия программ, получаемых в результате применения средств автоматического распараллеливания.

Объектом исследований является транслятор, выполняющий автоматическое распараллеливание программ линейного класса для многопроцессорных вычислительных систем.

Предметом исследований являются методы нахождения пространственных и временных отображений программ линейного класса для организации их параллельного выполнения на многопроцессорных вычислительных системах.

Научная задача, решаемая в диссертации — разработка методов нахождения пространственных и временных отображений программ линейного класса, обеспечивающих локальность использования данных при их параллельном выполнении на многопроцессорных вычислительных системах.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Провести анализ существующих методов и средств нахождения пространственных и временных отображений программ линейного класса, ориентированных на распараллеливание гнезд циклов.
2. Разработать критерии оптимальности пространственных и временных отображений программ линейного класса с точки зрения локальности использования данных.
3. Разработать метод нахождения оптимальных пространственных и временных отображений программ линейного класса для организации их параллельного выполнения на многопроцессорных вычислительных системах.
4. Разработать метод генерации параллельной MPI-программы, позволяющий организовать информационный обмен между параллельными процессами в случае явно заданного распределения данных между процессорами.
5. Провести анализ быстродействия параллельных программ, полученных применением разработанных методов и средств к тестам из набора PolyBench.

Методология и методы исследования. В работе применяются методы теории множеств, теории графов, линейной алгебры, выпуклого анализа, дискретного программирования, описательной статистики. Практические исследования проведены на кластерной вычислительной системе.

Научная новизна диссертации заключается в том, что в ней разработаны:

1. новые критерии оптимальности пространственных и временных отображений программ линейного класса, **отличающиеся** возможностью ранжировать информационные зависимости и доступы к данным *для более гибкого количественного описания локальности использования данных*;
2. новый метод нахождения оптимальных пространственных и временных отображений программ линейного класса, **отличающийся** применением взвешенной суммы показателей качества решения *для повышения производительности программ при параллельном выполнении*;
3. новый метод генерации параллельной MPI-программы, **не требующий** дублирования входных данных во всех исполняющихся процессах *для сокращения накладных расходов памяти на поддержку распределенных вычислений*.

Положения, выносимые на защиту:

1. Критерии оптимальности пространственных и временных отображений программ линейного класса, предоставляющие более гибкое количественное описание локальности использования данных по сравнению с целевыми функциями на основе лексикографического упорядочивания.

2. Метод нахождения оптимальных пространственных и временных отображений программ линейного класса, позволяющий избежать полного перебора решений с индивидуальной трудоемкой оценкой их качества.
3. Метод генерации параллельной MPI-программы, позволяющий организовать информационный обмен между параллельными процессами в случае явно заданного распределения данных между процессорами, не требуя дублирования входных данных во всех исполняющихся процессах.

Теоретическая значимость работы состоит в развитии подходов к исследованию влияния локальности использования данных на быстродействие программ линейного класса при их параллельном выполнении. Эти подходы могут быть применены при разработке методов и методик повышения быстродействия программ, затрачивающих большую часть времени выполнения на участки с циклическими конструкциями.

Практическая значимость работы заключается в разработанных компонентах, которые могут быть использованы в автоматически распараллеливающем трансляторе для поддержки распараллеливания программ, написанных на языке Си: компонент нахождения пространственных и временных отображений программ линейного класса, скрипт расстановки директив OpenMP, библиотека макросов для организации информационного обмена и скрипт постобработки параллельных циклов для реализации блочной схемы распределения процессоров в MPI-программе.

Реализация и внедрение результатов работы. Разработанное автором программное обеспечение для распараллеливания программ линейного класса внедрено ООО «НПП САТЭК плюс», а полученные теоретические результаты использованы в учебном процессе Федерального государственного бюджетного образовательного учреждения высшего образования «МИРЭА — Российский технологический университет», что подтверждено соответствующими актами.

Отдельные направления диссертационного исследования были поддержаны грантом РФФИ №14-37-50316 мол_нр «Исследование и разработка методов автоматического распараллеливания программ для гетерогенных вычислительных систем и систем с распределенной памятью» (2014 г.) и грантом Фонда содействия развитию малых форм предприятий в научно-технической сфере (Фонда содействия инновациям) в рамках проекта «Разработка платформы автоматического распараллеливания программ для гетерогенных высокопроизводительных вычислительных систем» (договор №1546ГС1/24323 от 28.09.2016).

Апробация результатов работы. Результаты диссертации докладывались и обсуждались на конференциях: Третий Национальный Суперкомпьютерный Форум (НСКФ-2014) (Переславль-Залесский, Институт программных систем имени А.К. Айламазяна Российской академии наук, 2014), Пятый Национальный Суперкомпьютерный Форум (НСКФ-2016) (Переславль-Залесский, Институт программных систем имени А.К. Айламазяна Российской академии наук, 2016), 11th IEEE International Conference on Application of Information

and Communication Technologies (AICT) (Moscow, V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, 2017), 3rd International Conference «Futuristic Trends in Networks and Computing Technologies» (FTNCT) (Taganrog, Southern Federal University, 2020), Всероссийская научно-техническая конференция «Многопроцессорные вычислительный и управляющие системы» (МВУС-2022) (Таганрог, Южный федеральный университет, 2022).

Достоверность и обоснованность научных результатов, полученных соискателем, подтверждены корректностью и непротиворечивостью математических выкладок и доказательств, результатами машинных экспериментов, а также внедрениями и использованием в различных организациях, что подтверждается соответствующими актами.

Публикации. Основные результаты по теме диссертации изложены в 12 печатных работах общим объемом 11,5 п.л., авторский вклад 9,9 п.л., 7 из которых опубликованы в рецензируемых научных журналах, входящих в Перечень ВАК РФ, 2 — в сборниках трудов конференций, индексируемых Web of Science и Scopus, 3 — в иных сборниках тезисов докладов. Зарегистрированы 2 программы для ЭВМ.

Соответствие паспорту специальности. Диссертация соответствует п. 8 («Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования») в части «методов создания программ для параллельной и распределенной обработки данных» и «инструментальных средств параллельного программирования» паспорта специальности 2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей, технические науки.

Содержание работы

Во **введении** обоснована актуальность и значимость исследуемой проблематики, дан обзор литературы, сформулированы цель и задачи исследования, приведены данные о структуре и объеме диссертационной работы.

В **первой главе** рассмотрен класс линейных программ и современные методы распараллеливания гнезд циклов с применением модели многогранников, также приводится их классификация вместе с разработанными в работе методами (рисунок 1): выявленные недостатки существующих методов определили направления их совершенствования, а преимущества сохранены в основе методов, разработанных в диссертации. Даются постановки задач таймирования (нахождения временных отображений, составляющих *расписание программы*) и распределения операций и данных между процессорами (нахождения пространственных отображений, задающих *размещение вычислений и данных*) для программ линейного класса.

Подход к таймированию и размещению вычислений, предложенный Бондхугулой, демонстрирует эффективность на практике, но имеет недостатки,

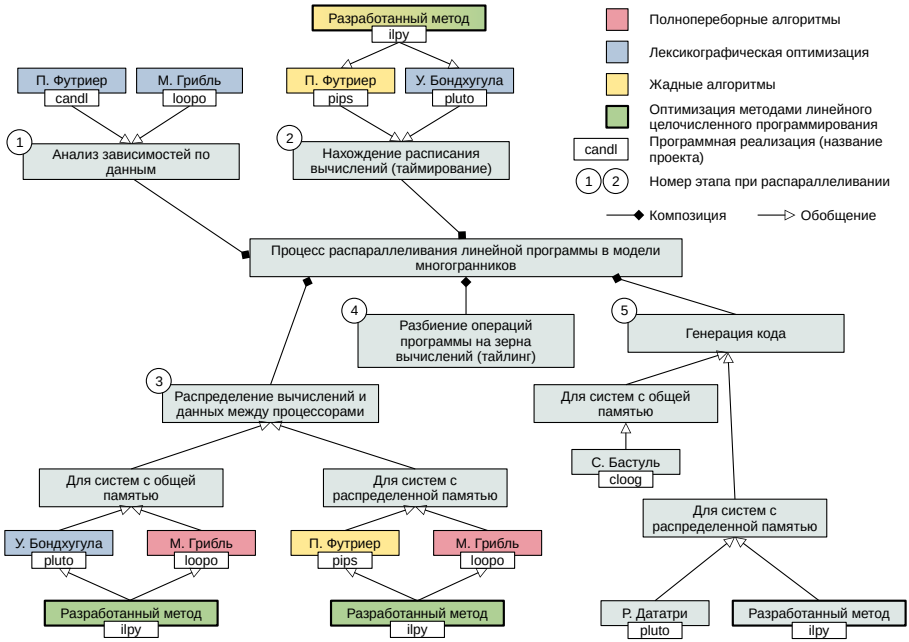

Рис. 1 — Разработанные методы на основе модели многогранников.

обусловленные тем, что поиск решений сводится к поиску лексикографического минимума на многограннике. В постановке задачи оптимизации невозможно задать одинаковый приоритет различным зависимостям, поскольку невозможно поставить две переменные на одну и ту же позицию в целевом векторе. Формальных рекомендаций к упорядочиванию переменных в целевом векторе также не приводится. Бондхугула в проекте pluto выполняет построение набора линейно независимых отображений, оптимизирующих локальность использования данных, сохраняя размерность пространств итераций инструкций по отношению к исходной программе. Грибль предлагает полный перебор решений для нахождения пространственных отображений программы для систем с общей и распределенной памятью: выбор лучшего решения выполняется на усмотрение исследователя, поскольку не приводится формального критерия качества.

Разработанный в диссертации метод нахождения оптимальных пространственных и временных отображений программы минимизирует задержку и расстояние использования данных, основываясь на более гибких средствах количественного описания локальности использования данных (взвешенная сумма показателей качества решения), и применяя аппарат линейного целочисленного программирования вместо лексикографической оптимизации. Это позволяет эффективнее оптимизировать временную и пространственную локальность использования данных по сравнению с методом Бондхугулы, и устранить необходимость полного перебора решений по сравнению с подходом Грибля. Для

систем с распределенной памятью пространственная локальность использования данных оптимизируется согласно ослабленным требованиям Футриера: выполняется сокращение расстояния коммуникаций, а не полное их исключение для как можно большего количества информационных зависимостей. Жадный алгоритм Футриера для построения многомерных расписаний вычислений используется совместно с разработанным методом, применяемым при нахождении одного компонента.

Для систем с общей памятью задача генерации параллельной программы исчерпывающе решается методом Бастуля. Для систем с распределенной памятью задача генерации параллельной программы усложняется организацией информационного обмена между параллельными процессами. Дататри и соавторами был предложен и реализован в проекте pluto метод, определяющий поток данных между итерациями распределенных циклов вдоль векторов информационных зависимостей, и исключающий дублирование информации при пересылке сообщений. При этом входные данные размещаются на всех вычислительных устройствах. Разработанный в диссертации метод учитывает оптимальное размещение вычислений и данных, вводит понятие многогранников коммуникаций, и преобразует параллельную программу, сгенерированную по методу Бастуля, в параллельную MPI-программу с двусторонней коммуникацией процессов, не требуя размещать данные на всех вычислительных устройствах.

Во **второй главе** введены и рассмотрены такие характеристики программы, как задержка и расстояние использования данных, а также разработаны: критерии оптимальности пространственных и временных отображений программ линейного класса, отличающиеся возможностью ранжировать информационные зависимости и доступы к данным для описания локальности использования данных; метод нахождения оптимальных пространственных и временных отображений программ линейного класса, отличающийся применением взвешенной суммы показателей качества решения. Изложенные во второй главе результаты опубликованы в [1—3; 8—12].

Рассмотрена функция $\chi_e(\vec{y},\vec{z}) = \varphi_{\delta(e)}(\vec{y},\vec{z}) - \varphi_{\sigma(e)}(h_e(\vec{y}),\vec{z})$, $\vec{y} \in P_e$ как мера повторного использования данных для ребра e в обобщенном графе зависимостей, где φ — одномерное аффинное отображение инструкции, h-преобразование h_e и многогранник P_e описывают информационную зависимость $\sigma(e) \to \delta(e)$, $\vec{z}$ — вектор внешних переменных программы. Если $\varphi_{\delta(e)}$ и $\varphi_{\sigma(e)}$ представляют расписание вычислений, то $\chi_e(\vec{y},\vec{z})$ — количество отсчетов логического времени, проходящего между исполнением инструкций $\sigma(e)$ и $\delta(e)$, то есть *мера задержки использования данных* $d_e^{\tau}(\vec{z})$. Если отображения представляют размещение вычислений, то $\chi_e(\vec{y},\vec{z})$ — расстояние в пространстве виртуальных процессоров между процессорами, выполняющими инструкции $\sigma(e)$ и $\delta(e)$, то есть *мера расстояния использования данных* $d_e^{\rho}(\vec{z})$. Если размещение данных задается явно, то *мера расстояния использования данных* $d_a^{\rho}(\vec{z})$ определяется функцией $\nu_a(\vec{i},\vec{z}) = |\pi_X(\vec{i},\vec{z}) - \eta_{A_a}(g_a(\vec{i}),\vec{z})|, \quad \vec{i} \in D_X$ для доступа к данным $\langle g_a,\vec{z};A_a\rangle$ в некоторой позиции a инструкции X. $\nu_a(\vec{i},\vec{z})$ —

расстояние в пространстве виртуальных процессоров межу процессором, выполняющим инструкцию X, и процессором, владеющим элементом данных, к которому осуществляется доступ в позиции a инструкции X. Минимизация задержки и расстояния использования данных приводит к улучшению временной и пространственной локальности использования данных соответственно. Величины $d_e^{\tau}(\vec{z})$, $d_e^{\rho}(\vec{z})$, $d_a^{\rho}(\vec{z})$ желательно ограничить постоянным значением, что не всегда возможно. Доказаны утверждения:

1. Если домены $D_{\delta(e)}$ и $D_{\sigma(e)}$ являются ограниченными, и заданы два аффинных отображения $\varphi_{\delta(e)}(\vec{y},\vec{z})$ и $\varphi_{\sigma(e)}(h_e(\vec{y}),\vec{z})$, то найдется аффинная функция $L_e(\vec{z})$, зависящая только от внешних переменных программы, такая, что $L_e(\vec{z}) - \chi_e(\vec{y},\vec{z}) \geqslant 0, \quad \vec{y} \in P_e$.
2. Если домен D_X является ограниченным, и заданы два аффинных отображения: $\pi_X(\vec{i},\vec{z})$ и $\eta_{A_a}(g_a(\vec{i}),\vec{z})$ для некоторой позиции a инструкции X, то найдется аффинная функция $L_a(\vec{z})$, зависящая только от внешних переменных программы, такая, что $L_a(\vec{z}) - \nu_a(\vec{i},\vec{z}) \geqslant 0, \quad \vec{i} \in D_X$.

Следовательно, найдутся функции $L_e^{\tau}(\vec{z}) \geqslant d_e^{\tau}(\vec{z})$, $L_e^{\rho}(\vec{z}) \geqslant d_e^{\rho}(\vec{z})$, $L_a^{\rho}(\vec{z}) \geqslant d_a^{\rho}(\vec{z})$. Степень временной (пространственной) локальности использования данных тем выше, чем меньше величина $L_e^{\tau}(\vec{z})$ ($L_e^{\rho}(\vec{z})$ или $L_a^{\rho}(\vec{z})$). *Оптимальное расписание программы* минимизирует целевую функцию $\sum_{e\in E} \alpha_e L_e^{\tau}(\vec{z})$, где $\alpha_e > 0$ — весовые коэффициенты, рассматриваемые как показатели относительной значимости зависимостей по данным. Аналогично, *оптимальное размещение программы* минимизирует $\sum_{e\in E} \alpha_e L_e^{\rho}(\vec{z})$. *Оптимальное размещение вычислений и данных*, вычисленное совместно, минимизирует целевую функцию $\sum_a \alpha_a L_a^{\rho}(\vec{z})$, где $\alpha_a > 0$ — весовые коэффициенты, рассматриваемые как показатели относительной значимости доступов к данным.

Определение Пусть $\{\varphi\}_{E'}$ — множество допустимых наборов φ одномерных аффинных отображений φ_X, $X \in \bigcup\limits_{e\in E'} \{\sigma(e),\delta(e)\}$ для заданного подмножества ребер $E' \subseteq E$ обобщенного графа зависимостей. Набор $\varphi \in \{\varphi\}_{E'}$ называется оптимальным для E', если он минимизирует функционалы $f_e(\varphi) = L_e(\vec{z}) = \vec{l_e^{\,}} \cdot \vec{z} + l_e^0$, $e \in E'$ так, что

$$\nexists \varphi' \in \{\varphi\}_{E'} \left(\forall e \in E' \left(f_e(\varphi') \leqslant f_e(\varphi)\right) \wedge \exists e \in E' \left(f_e(\varphi') < f_e(\varphi)\right)\right).$$

Расписание θ вычисляется как оптимальный набор аффинных отображений $\theta_{S_i}(\vec{x},\vec{z}), \quad i = 1,\ldots,m$ для E. Для нахождения многомерных расписаний вычислений расширен классический алгоритм Футриера: каждый компонент многомерного расписания вычисляется как одномерное аффинное расписание для подмножества ребер $E_1' \subseteq E$, соответствующих удовлетворенным зависимостям на текущем шаге алгоритма. Размещение π вычисляется как оптимальный набор аффинных отображений $\pi_{S_i}(\vec{x},\vec{z}), \quad i = 1,\ldots,m$ для E.

Во всех случаях формулируется задача оптимизации, где $\{\varphi\}_{E'}$ — множество допустимых решений, $f_e(\varphi), e \in E'$ — показатели качества решения

φ, которые требуется минимизировать. Для нахождения расписаний вычислений ($\varphi = \theta$) рассматривается $f_e(\varphi) = L_e^{\tau}(\vec{z})$, а для размещений вычислений ($\varphi = \pi$) рассматривается $f_e(\varphi) = L_e^{\rho}(\vec{z})$. Формулировка задачи уточняется путем задания набора ограничений, описывающих множество допустимых решений. Минимизируется взвешенная сумма $\sum_{e \in E} \alpha_e f_e = \sum_{e \in E} \alpha_e \left(\vec{l}_e \cdot \vec{z} + l_e^0 \right), \quad \alpha_e > 0$, где $\alpha_e = \# P_e$ — количество точек с целочисленными координатами внутри многогранника зависимостей P_e.

Определение Пусть $\left\{ \varphi \middle| \varphi = \left\langle \bigcup\limits_{a \in \{a\}} \{\pi_{S_a}\}, \bigcup\limits_{a \in \{a\}} \{\eta_{A_a}\} \right\rangle \right\}_{\{a\}}$ — множество допустимых наборов φ одномерных аффинных отображений π_{S_a} и η_{A_a} для всех инструкций S_a и массивов A_a в программе, порождаемое множеством $\{a\}$ доступов к данным a. Набор φ называется оптимальным для программы, если он минимизирует функционалы $f_e(\varphi) = L_a(\vec{z}) = \vec{l}_a \cdot \vec{z} + l_a^0$, для всех доступов к данным a так, что

$$\nexists \varphi' \in \{\varphi\}_{\{a\}} \left(\forall a \in \{a\} \left(f_a(\varphi') \leqslant f_a(\varphi) \right) \wedge \exists a \in \{a\} \left(f_e(\varphi') < f_e(\varphi) \right) \right).$$

В случае явно заданного распределения данных между процессорами нахождение размещений вычислений и данных выполняется совместно. Размещение вычислений и данных вычисляется как оптимальный набор аффинных отображений $\langle \{\pi_{S_i}, i = 1, \ldots, m\}, \{\eta_{A_j}, j = 1, \ldots l\} \rangle$. Формулируется задача оптимизации, где $\{\varphi\}_{\{a\}}$ — множество допустимых решений, $f_a(\varphi), a \in \{a\}$ — показатели качества решения φ, которые требуется минимизировать. Формулировка задачи уточняется путем задания набора ограничений, описывающих множество допустимых решений. Минимизируется взвешенная сумма $\sum_a \alpha_a f_a = \sum_a \alpha_a \left(\vec{l}_a \cdot \vec{z} + l_a^0 \right), \quad \alpha_a > 0$, где $\alpha_a = \# D_{S_a}$ — количество точек с целочисленными координатами внутри домена S_a.

В **третьей главе** разработан метод генерации параллельной MPI-программы, не требующий дублирования входных данных во всех исполняющихся процессах, а также специальное программное обеспечение для распараллеливания программ линейного класса, написанных на языке Си. Изложенные в третьей главе результаты опубликованы в [4—7; 11; 13; 14].

Определены многогранники коммуникаций и приведена схема конструирования параллельной MPI-программы с синхронным параллелизмом для блочной схемы распределения процессоров: $r(v) = \lfloor \frac{v}{Q} \rfloor$, где v ($v \geqslant 0$) — индекс виртуального процессора, $r(v)$ — индекс физического процессора, Q — количество физических процессоров, и каждый физический процессор r обрабатывает отрезок пространства виртуальных процессоров $l(r), \ldots, u(r)$.

Определение Множества $Q_{a_j}^{r/w-r/s}$, определяющие состав элементов массива A_{a_j}, которые должны быть переданы между физическими процессорами R и r для обеспечения операции удаленного доступа в текущий момент логического

времени t, являются параметрически заданными многогранниками и называются многогранниками коммуникаций. В первой части верхнего индекса r означает операцию удаленного чтения (read), w — операцию удаленной записи (write). Во второй части верхнего индекса r означает прием данных (receive), s — отправку (send).

Пусть $D_S^{'}$ — домен инструкции S в параллельной программе, $p_S^{'}$ — его размерность, $\vec{i}^{'}_S \in D_S^{'}$ — целочисленный вектор индекса итерации, $\vec{i}^{'(p)}_S$ — индекс цикла, итерирующего по пространству виртуальных процессоров.

Пусть C — многогранник, описывающий внешние переменные программы (далее — «контекст»), заданный p_C ограничениями вида $\vec{c}_k \cdot \vec{z} + d_k \geqslant 0$, $\vec{c}_k \in \mathbb{Z}^{q_z}$, $d_k \in \mathbb{Z}$, $k = 1, \ldots, p_C$.

Пусть $D_S^{'}$ задан $p_{D_S^{'}}$ ограничениями вида $\vec{b}_{S,k} \cdot \vec{i}^{'}_S + \vec{c}_{S,k} \cdot \vec{z} + d_{S,k} \geqslant 0$, $\vec{b}_{S,k} \in \mathbb{Z}^{p_S^{'}}$, $\vec{c}_{S,k} \in \mathbb{Z}^{q_z}$, $d_{S,k} \in \mathbb{Z}$, $k = 1, \ldots, p_{D_S^{'}}$.

Определение Параметрически заданный многогранник $D_{S,R}^{''}$ называется мгновенным доменом инструкции S для физического процессора R:

$$\vec{i}^{''}_S = \begin{bmatrix} \vec{i}^{'(p)}_S \\ \vdots \\ \vec{i}^{'(p_S^{'})}_S \end{bmatrix} \in D_{S,R}^{''} \Leftrightarrow \begin{cases} l(R) \leqslant \vec{i}^{'(p)}_S \leqslant u(R) \\ \vec{b}_{S,k} \cdot \vec{i}^{'}_S + \vec{c}_{S,k} \cdot \vec{z} + d_{S,k} \geqslant 0, k \in K_S^{D^{''}} \end{cases},$$

где $K_S^{D^{''}} = \{k | k = 1, \ldots, p_{D_S^{'}} \wedge \sum\limits_{l=p+1}^{p_S^{'}} |\vec{b}^{(l)}_{S,k}| > 0\}$ — множество индексов ограничений, включающих хотя бы один из компонентов $\vec{i}^{'(l)}_S$, $l = p+1, \ldots, p_S^{'}$.

Контекст $C_{S,R}^{''}$ определяет параметры $D_{S,R}^{''}$ и задается ограничениями:

$$\begin{cases} \vec{c}_k \cdot \vec{z} + d_k \geqslant 0, k = 1, \ldots, p_C \\ \vec{b}_{S,k} \cdot \vec{i}^{'}_S + \vec{c}_{S,k} \cdot \vec{z} + d_{S,k} \geqslant 0, k \in K_S^{C^{''}} \end{cases},$$

где $K_S^{C^{''}} = \{k | k = 1, \ldots, p_{D_S^{'}} \wedge \sum\limits_{l=p+1}^{p_S^{'}} |\vec{b}^{(l)}_{S,k}| = 0\}$ — множество индексов ограничений, не включающих ни один из компонентов $\vec{i}^{'(l)}_S$, $l = p+1, \ldots, p_S^{'}$.

Пусть $\widetilde{g}_a(\theta^{'}, D_{S,R}^{''})$ — множество индексов элементов массива A_a, к которым осуществляется доступ a в некоторой позиции инструкции S физическим процессором R в момент времени $\theta^{'} = \begin{bmatrix} \vec{i}^{'(1)}_S \\ \vdots \\ \vec{i}^{'(p-1)}_S \end{bmatrix}$.

Участок массива A_a, который должен быть принят процессором R от процессора r перед выполнением яруса параллельной формы для реализации удаленного чтения a в некоторой позиции инструкции S, определяется следующими ограничениями на индексы массива $\vec{g}$:

$$Q_a^{r-r}(\vec{g}, R, r) : \begin{cases} \vec{g} \in \widetilde{g}_a(\theta', D''_{S,R}) \\ l(r) \leqslant \eta_{A_a}(\vec{g}, \vec{z}) \leqslant u(r) \end{cases} .$$

Участок массива A_a, который должен быть передан процессору r процессором R перед выполнением яруса параллельной формы для реализации удаленного чтения a в некоторой позиции инструкции S, определяется следующими ограничениями на индексы массива $\vec{g}$:

$$Q_a^{r-s}(\vec{g}, R, r) : \begin{cases} \vec{g} \in \widetilde{g}_a(\theta', D''_{S,r}) \\ l(R) \leqslant \eta_{A_a}(\vec{g}, \vec{z}) \leqslant u(R) \end{cases} .$$

Участок массива A_a, который должен быть передан процессору r процессором R после выполнения яруса параллельной формы для реализации удаленной записи a в некоторой позиции инструкции S, определяется следующими ограничениями на индексы массива $\vec{g}$:

$$Q_a^{w-s}(\vec{g}, R, r) : \begin{cases} \vec{g} \in \widetilde{g}_a(\theta', D''_{S,R}) \\ l(r) \leqslant \eta_{A_a}(\vec{g}, \vec{z}) \leqslant u(r) \end{cases} .$$

Участок массива A_a, который должен быть принят процессором R от процессора r после выполнения яруса параллельной формы для реализации удаленной записи a в некоторой позиции инструкции S, определяется следующими ограничениями на индексы массива $\vec{g}$:

$$Q_a^{w-r}(\vec{g}, R, r) : \begin{cases} \vec{g} \in \widetilde{g}_a(\theta', D''_{S,r}) \\ l(R) \leqslant \eta_{A_a}(\vec{g}, \vec{z}) \leqslant u(R) \end{cases} .$$

Если R — ранг MPI-процесса, то поддержка удаленного доступа реализуются согласно схеме:

```
// for remote reads exchange data
for (int j = 1; j <= n; j++) {
  if (a_j is read) {
    for (int r = 0; r < Q; r++) {
      MPI_Isend(data for Q^{r-s}_{a_j}(g_{A_{a_j}}, R, r));
    }
    for (int r = 0; r < Q; r++) {
      MPI_Recv(data for Q^{r-r}_{a_j}(g_{A_{a_j}}, R, r));
    }
  }
}
```

```
// for remote writes exchange data
for (int j = 1; j <= n; j++) {
  if (a_j is write) {
    for (int r = 0; r < Q; r++) {
      MPI_Isend(data for Q^{w-s}_{a_j}(g_{A_{a_j}}, R, r));
    }
    for (int r = 0; r < Q; r++) {
      MPI_Recv(data for Q^{w-r}_{a_j}(g_{A_{a_j}}, R, r));
    }
  }
}
```

На основе этой схемы разработаны макросы информационного обмена, поддерживающие многогранники коммуникаций специальной формы: участки линейных массивов, строки и столбцы матриц, линеаризованных по строкам. Программа с синхронным параллелизмом строится по следующей схеме:

```
1 do t = 0; L
2   for remote reads exchange data
3   pardo F(t)
4   barrier
5   for remote writes exchange data
6 end do
```

Приводятся методики постобработки исходного кода на языке Си: расстановка директив OpenMP, подготовка циклов для MPI, расстановка конструкций информационного обмена, вынос ветвлений за пределы тела цикла для сокращения накладных расходов на распараллеливание. Разработано специальное программное обеспечение ilpy, реализующее компоненты для поддержки распараллеливания программ, написанных на языке Си: компонент нахождения пространственных и временных отображений программ линейного класса, скрипт расстановки директив OpenMP, скрипт постобработки параллельных циклов для реализации блочной схемы распределения процессоров в MPI-программе. На рисунке 2 представлена диаграмма потоков данных процесса распараллеливания программ.

В **четвертой главе** приводятся экспериментальные результаты исследования производительности параллельных вариантов программ (lu, atax, syr2k, floyd, gramschmidt), полученных применением к исходным текстам PolyBench специального разработанного программного обеспечения ilpy и современного распараллеливающего транслятора pluto 0.11.4. Изложенные в четвертой главе результаты опубликованы в [6; 7].

Каждая программа была распараллелена с помощью OpenMP для выполнения на одном компьютере, и с помощью MPI для выполнения на кластере в трех вариантах запуска: все процессы на одной машине; процессы распределены поровну между двумя машинами; выделено по отдельной машине на процесс. Вычисления производились с двойной точностью на восьми машинах РТУ МИРЭА (таблица 1).

Таблица 1 — Конфигурация оборудования и программное окружение

Конфигурация узлов	Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz, 16GB RAM
Сеть передачи сообщений	InfiniBand 40Gb/sec
Программное обеспечение	ОС Linux CentOS 7.8 x64, компилятор gcc 4.8.5, реализация MPI openmpi-4.0.3rc4
Компиляция программ OpenMP/MPI	gcc -O3 -std=c99 -fopenmp / mpicc -O3 -std=c99 -fopenmp
Трансляция pluto для OpenMP	polycc --parallelize --lastwriter
Трансляция pluto-distmem для MPI	polycc --distmem --commopt_fop --isldep --lastwriter --cloogsh --timereport

Ускорение оценивалось относительно запуска последовательного варианта программы. Пусть S_p^{tool} — ускорение, полученное для запуска в p процессах MPI (или нитях OpenMP) программы, порожденной средством $tool$. Преимущество ilpy перед pluto оценивалось по формуле $(S_p^{ilpy}/S_p^{pluto} - 1) \cdot 100\%$ для пяти рассматриваемых программ (таблица 2).

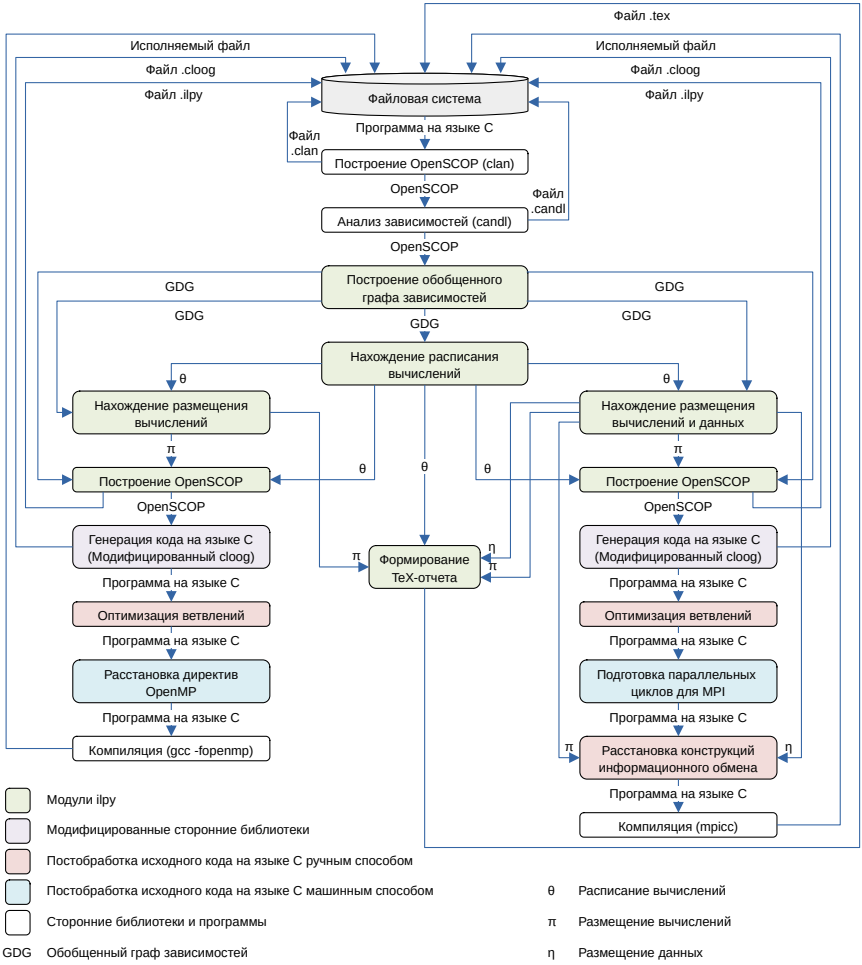

Рис. 2 — Диаграмма потоков данных процесса распараллеливания.

При распараллеливании с OpenMP ilpy показал преимущество для трех программ из пяти: для lu 25,05%, для syr2k 2,59%, для gramschmidt 5,37%. Идентичный результат был получен для программы floyd, а для программы atax был зафиксирован проигрыш на 18,53%. При распараллеливании с MPI ilpy показал преимущество для трех программ из пяти: для lu наблюдается преимущество в запусках на одной и восьми машинах, для atax и floyd наблюдается преимущество во всех вариантах запуска. Для программы syr2k зафиксирован проигрыш, составляющий до 2 раз. Для программы gramschmidt нельзя сделать заключение, поскольку транслятор pluto не смог завершить работу. Сравнение выполнялось с

Таблица 2 — Сравнение эффективности распараллеливания ilpy и pluto

Программа	lu	atax	syr2k	floyd	gramschmidt
Размер задачи	N=3072	M=8192, N=6144	N=1536, M=1536	N=1024	M=2048, N=1024
8 нитей (процессов)					
OpenMP, %	25,05	-18,53	2,59	-0,75	5,37
MPI (1 машина), %	26,81	302,82	-40,64	277,69	89,42
MPI (2 машины), %	-7,70	145,09	-43,79	155,39	-8,56
MPI (8 машин), %	18,14	77,57	-51,88	40,06	-57,42
1 нить (процесс)					
OpenMP, %	25,77	-3,00	-5,53	0,19	130,74
MPI (1 машина), %	30,43	252,60	-1,01	439,97	81,34

исходным последовательным вариантом. Результаты быстродействия параллельных программ, полученных применением ilpy, демонстрируют практическую применимость разработанного метода даже в условиях статического распределения нагрузки между MPI-процессами. Преимущество ilpy для lu и gramschmidt наблюдается и в запусках программ с одной нитью OpenMP. При увеличении количества нитей проигрыш на примере syr2k сокращается и исчезает, а на примере atax продолжает расти. Запуски с одним процессом MPI иллюстрируют преимущество ilpy для всех программ, кроме syr2k, для которой наблюдаются сходные результаты, но с увеличением количества процессов проигрыш растет за счет применения pluto динамического распределения нагрузки между MPI-процессами.

В **заключении** приведены основные результаты работы.

Основной научный результат диссертации заключается в решении актуальной научной задачи, заключающейся в разработке методов нахождения пространственных и временных отображений программ линейного класса, обеспечивающих локальность использования данных при их параллельном выполнении на многопроцессорных вычислительных системах.

При проведении исследований и разработок по теме настоящей работы получены следующие результаты.

1. Установлены ограничения и недостатки существующих методов и средств нахождения пространственных и временных отображений программ линейного класса, ориентированных на распараллеливание гнезд циклов, обозначившие направления их совершенствования.
2. Разработаны новые критерии оптимальности пространственных и временных отображений программ линейного класса, отличающиеся возможностью ранжировать информационные зависимости и доступы к данным для более гибкого количественного описания локальности использования данных, чем целевые функции на основе лексикографического упорядочивания в методах П. Футриера и У. Бондхугулы.
3. Разработан новый метод нахождения оптимальных пространственных и временных отображений программ линейного класса для распараллеливания гнезд циклов, отличающийся применением взвешенной суммы показателей качества решения. Разработанный метод устраняет необходимость полного перебора решений с их трудоемкой оценкой качества (М. Грибль), и при этом позволяет следовать идее П. Футриера, но при

ослабленных ограничениях, заключающихся в сокращении расстояния коммуникаций, а не полном их исключении, в меньшей степени ограничивая параллелизм.

4. Разработан метод генерации параллельной MPI-программы, позволяющий организовать информационный обмен между параллельными процессами в случае явно заданного распределения данных между процессорами. По сравнению с методом Р. Дататри, исключающим дублирование информации при пересылке информационных пакетов, нет необходимости размещать входные данные на всех вычислительных устройствах.
5. Получены экспериментальные результаты исследования производительности параллельных вариантов программ (lu, atax, syr2k, floyd, gramschmidt), свидетельствующие о выигрыше в эффективности распараллеливания по сравнению с современным транслятором Pluto: до 25% (с OpenMP) и до 302% (с MPI). Разработаны компоненты, которые могут быть использованы в автоматически распараллеливающем трансляторе для поддержки распараллеливания программ, написанных на языке Си.

Разработанные при решении научной задачи методы и средства позволили достигнуть поставленной цели, заключающейся в повышении быстродействия программ, получаемых в результате применения средств автоматического распараллеливания. Результаты диссертационного исследования подтверждены экспериментальными исследованиями и использованы в ООО «НПП САТЭК плюс», г. Рыбинск, и в учебном процессе кафедр КБ-4 и КБ-14 Института кибербезопасности и цифровых технологий Федерального государственного бюджетного образовательного учреждения высшего образования «МИРЭА — Российский технологический университет».

Направлениями дальнейших исследований являются:

1. критерии оптимальности многомерного аффинного отображения, включающего измерения расписания и размещения вычислений;
2. техники передачи данных в информационных пакетах специального формата (например, разреженные матрицы);
3. методы построения программ с асинхронным параллелизмом, позволяющие отказаться от барьерной синхронизации;
4. обеспечение стабильного времени работы компонента нахождения размещений вычислений и данных;
5. применение разработанных компонентов для распараллеливания программ линейного класса в JIT-средах.

Публикации автора по теме диссертации

1. *Лебедев, А. С.* Оптимизация временной локальности данных при автоматическом распараллеливании линейных программ [Электронный ресурс] / А. С. Лебедев // Современные проблемы науки и образования. — 2014. — № 6. — С. 192—192. — URL: https://science-education.ru/ru/article/view?id=16255.
2. *Лебедев, А. С.* Пространственно-временные преобразования при распараллеливании линейных программ [Текст] / А. С. Лебедев // Информационные технологии и вычислительные системы. — 2015. — № 1. — С. 19—32.
3. *Лебедев, А. С.* Размещение данных при автоматическом распараллеливании линейных программ для систем с распределенной памятью [Текст] / А. С. Лебедев // Вестник Рыбинской государственной авиационной технологической академии им. П. А. Соловьева. — 2015. — № 3. — С. 92—99.
4. *Лебедев, А. С.* Организация информационного обмена между параллельными процессами при автоматическом распараллеливании линейных программ для кластерных систем с применением модели многогранников [Электронный ресурс] / А. С. Лебедев // Программные системы: теория и приложения. — 2017. — Т. 8, 4 (35). — С. 3—20. — URL: https://psta.psiras.ru/read/psta2017_4_3-20.pdf.
5. *Лебедев, А. С.* Система автоматического распараллеливания линейных программ для машин с общей и распределенной памятью [Электронный ресурс] / А. С. Лебедев, Ш. Г. Магомедов // Российский технологический журнал. — 2019. — Т. 7, № 5. — С. 7—19. — URL: https://www.rtj-mirea.ru/jour/article/view/168.
6. *Лебедев, А. С.* Трансляция программ линейного класса для параллельного выполнения на универсальных многоядерных процессорах [Текст] / А. С. Лебедев, В. И. Солодовников // Труды Института системного анализа Российской академии наук. — 2023. — Т. 73, № 4. — С. 36—47.
7. *Лебедев, А. С.* Трансляция программ линейного класса для параллельного выполнения на системах с распределенной памятью [Текст] / А. С. Лебедев // Системы высокой доступности. — 2023. — Т. 19, № 3. — С. 35—47.
8. *Лебедев, А. С.* Оптимизация локальности данных при автоматическом распараллеливании программ [Электронный ресурс] / А. С. Лебедев // Сборник тезисов докладов Третьего Национального суперкомпьютерного форума (НСКФ-2014) (Переславль-Залесский 25-27 ноября 2014 г.). — Переславль-Залесский: Институт программных систем имени А. К. Айламазяна Российской академии наук, 2014. — 2014. — URL: https://2014.nscf.ru/TesisAll/4_Systemnoe_i_promezhytochnoe_PO/11_179_LebedevAS.pdf.

9. *Лебедев, А. С.* Среда автоматического распараллеливания программ для систем с общей и распределенной памятью [Электронный ресурс] / А. С. Лебедев // Сборник тезисов докладов Пятого Национального суперкомпьютерного форума (НСКФ-2016) (Переславль-Залесский, 29 ноября — 02 декабря 2016 г.). — Переславль-Залесский: Институт программных систем имени А. К. Айламазяна Российской академии наук, 2016. — 2016. — URL: https://2016.nscf.ru/TesisAll/03_Systemnoe_i_promezhytochnoe_PO/732_LlebedevAS.pdf.

10. *Lebedev, A. S.* Construction of optimal space-time mappings for automatic parallelization of loop nests with static control flow [Text] / A. S. Lebedev // 2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT). — IEEE. 2017. — P. 1—7. — (Scopus, WoS).

11. *Lebedev, A. S.* Automatic Parallelization of Affine Programs for Distributed Memory Systems [Text] / A. S. Lebedev, S. G. Magomedov // Futuristic Trends in Network and Communication Technologies: Third International Conference, FTNCT 2020, Taganrog, Russia, October 14–16, 2020, Revised Selected Papers, Part II 3. — Springer. 2021. — P. 91—101. — (Scopus).

12. *Лебедев, А. С.* Построение пространственных и временных преобразований программ для распараллеливания вложенностей циклов [Текст] / А. С. Лебедев // Многопроцессорные вычислительные и управляющие системы : материалы Всероссийской научно-технической конференции (МВУС-2022), Таганрог, 27—30 июня 2022 г. — Ростов-на-Дону: Южный федеральный университет, 2022. — Издательство Южного федерального университета, 2022. — С. 90—95.

13. *Свидетельство о гос. регистрации программы для ЭВМ*. Модуль вычисления пространственно-временного преобразования линейной программы с целью ее распараллеливания и оптимизации локальности данных [Текст] : 2016618301 Рос. Федерация / А. С. Лебедев ; О. с ограниченной ответственностью Технологии высокопроизводительных вычислений. — № 2016612892 ; заявл. 30.03.2016 ; опубл. 20.08.2016.

14. *Свидетельство о гос. регистрации программы для ЭВМ*. Программа для построения аффинных преобразований программ линейного класса [Текст] : 2022667001 Рос. Федерация / А. С. Лебедев, С. В. И. ; Ф. государственное бюджетное учреждение науки Центр информационных технологий в проектировании Российской академии наук. — № 2022666032 ; заявл. 29.08.2022 ; опубл. 13.09.2022.

Лебедев Артем Сергеевич

Методы и средства распараллеливания программ линейного класса для выполнения на многопроцессорных вычислительных системах

Автореф. дис. на соискание ученой степени канд. тех. наук

Подписано в печать 15.04.2024.
Формат бумаги 70×100/16. Бумага офсетная. Усл. печ. л. 1,2. Тираж 100 экз. Заказ 010к
Отпечатано в Издательстве ТулГУ. 300012, г. Тула, просп. Ленина, 95