Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

УДК 681.3.06 : 681.324

КАРАСИК
Олег Николаевич

КООПЕРАТИВНЫЙ МНОГОПОТОЧНЫЙ ПЛАНИРОВЩИК И БЛОЧНО-ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧ НА МНОГОЯДЕРНЫХ СИСТЕМАХ

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

по специальности 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Минск 2019

Работа выполнена в Белорусском национальном техническом университете.

Научный руководитель: **Прихожий Анатолий Алексеевич**, доктор технических наук, профессор, профессор кафедры «Программное обеспечение вычислительной техники и автоматизированных систем» Белорусского национального технического университета

Официальные оппоненты: **Татур Михаил Михайлович**, доктор технических наук, профессор, профессор кафедры «Электронных вычислительных машин» Белорусского государственного университета информатики и радиоэлектроники

Воротницкий Юрий Иосифович, кандидат физико-математических наук, доцент, заведующий кафедрой «Телекоммуникаций и информационных технологий» Белорусского государственного университета

Оппонирующая организация: Государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси»

Защита состоится «27» июня 2019 г. в 10.00 на заседании совета по защите диссертаций Д 02.15.04 при учреждении образования «Белорусский государственный университет информатики и радиоэлектроники» по адресу: 220013, г. Минск, ул. П. Бровки 6, корп. 1, ауд. 232, тел. (017) 293-89-89, e-mail: dissovet@bsuir.by.

С диссертацией можно ознакомиться в библиотеке учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»

Автореферат разослан «__» _____ 2019 г.

И. о. ученого секретаря
совета по защите диссертаций
доктор технических наук, профессор М. М. Татур

КРАТКОЕ ВВЕДЕНИЕ

Современные многоядерные процессоры обладают высокой тактовой частотой, большим объемом высокоскоростной иерархической памяти, специальными векторными регистрами и инструкциями для работы с ними, позволяющими выполнять одновременно операции над многими данными. Широкое распространение многоядерных систем в совокупности с появлением эффективных и доступных сетевых средств высокоскоростной передачи данных, объединяющих многоядерные системы в кластеры, привело к настоятельной необходимости распараллеливания последовательных алгоритмов и разработке эффективных параллельных алгоритмов решения разнообразных прикладных задач с учетом архитектурных особенностей вычислительных систем. Ввиду большого разнообразия аппаратных решений и высокой сложности учета всех факторов, влияющих на характеристики параллельных алгоритмов, зачастую не представляется возможным точно учесть особенности взаимодействия частей программы на конкретной архитектуре, особенности функционирования иерархической памяти, возможности векторизации вычислений и т. д.

Разработка параллельных алгоритмов предъявляет высокие требования к профессиональным качествам разработчиков и требует серьезных затрат времени. По этим причинам высокоуровневые средства разработки OpenMP, MPI, Cilk++, Threading Building Blocks и др., предоставляющие высокоуровневый интерфейс описания параллелизма и абстрагирующие разработчика от деталей многопоточной модели операционной системы и особенностей аппаратной архитектуры, получили широкое распространение.

Несмотря на преимущества средств разработки и операционных систем с универсальными возможностями реализации параллельных вычислений, остается открытой проблема сближения моделей и средств планирования и управления выполнением многопоточных приложений с моделями и средствами построения и функционирования параллельных алгоритмов и программ, улучшающих характеристики многопоточных реализаций на многоядерных системах.

При кооперативной многозадачности операционная система не инициирует переключение контекста из одного процесса в другой, как при вытесняющей многозадачности, но сами процессы устанавливают порядок передачи управления или логически блокируют друг друга посредством кооперативного планировщика. Разрабатываемый в диссертации новый кооперативный многопоточный планировщик и известные блочно-параллельные алгоритмы решения прикладных задач положены в основу нового класса кооперативных блочно-параллельных алгоритмов, которые, в свою очередь, усиливают преимущества нового кооперативного планировщика.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Связь работы с крупными научными программами, темами

Диссертационная работа выполнена в соответствии с научно-техническими заданиями и планами работ кафедры «Программное обеспечение вычислительной техники и автоматизированных систем» Белорусского национального технического университета и в соответствии с государственными научными темами в рамках бюджетного финансирования Республики Беларусь:

1. Математическое и программное обеспечение систем обработки информации в образовании и автоматизированных системах управления техническими объектами. – Минск, БНТУ, 2014 г. – ГБ № 11-254.
2. Модели, методы, технологии создания программных комплексов для инженерных и компьютерных систем. – Минск, БНТУ, 2016–2018 гг. – ГБ № 16-274.

Тема диссертации соответствует направлениям государственных программ научных исследований на 2016–2020 гг. (Постановление Совета Министров Республики Беларусь от 10 июня 2015 г. № 483), включающим разработку методов высокопроизводительных вычислений и «облачных» технологий, математического и компьютерного моделирования и проектирования высокотехнологичных изделий и процессов; оптимального планирования и логистики.

Цель и задачи исследования

Целью исследования явилась разработка архитектуры кооперативного планировщика выполнения потоков, блочно-параллельных алгоритмов и програмного обеспечения, повышающих производительность многоядерной системы при решении задач большого размера.

Поставленная цель определяет следующие *задачи исследования*:

1. Выполнить анализ и экспериментальное исследование существующих механизмов реализации многозадачности в операционных системах.
2. Разработать архитектуру кооперативного планировщика выполнения потоков, повышающую эффективность использования вычислительных ресурсов при исполнении многопоточных приложений на многоядерных системах.
3. Разработать способ построения потоковых блочно-параллельных алгоритмов, использующих преимущества кооперативного планировщика.
4. Разработать эффективные кооперативные потоковые блочно-параллельные алгоритмы решения конкретных прикладных задач.

5. Выполнить программную реализацию кооперативного планировщика и блочно-параллельных алгоритмов, оценить их эффективность в сопоставлении с лучшими аналогами.

Объектом исследования являются многопоточные реализации параллельных алгоритмов и средства управления их выполнением на многоядерных системах.

Предметом исследования являются архитектуры и алгоритмы планирования выполнения потоков, потоковые блочно-параллельные алгоритмы решения прикладных задач на многоядерных системах.

Научная новизна

1. Разработанные новый кооперативный планировщик выполнения многопоточного приложения и способ построения кооперативных потоковых блочно-параллельных алгоритмов предназначены для эффективного совместного использования на многоядерных архитектурах.

2. Предложен новый кооперативный планировщик выполнения потоков, позволяющий учесть особенности решаемой задачи и специфику многоядерной системы. Разработана архитектура планировщика, построенная на базе нового примитива синхронизации, исключающая взаимодействие пользовательских потоков с операционной системой, изменяющая механизм работы с очередью потоков, ускоряющая процессы их блокировки и разблокировки. Для задач с небольшим временем вычислений и доминирующим временем управления потоками новый планировщик выигрывает по производительности у базовых средств кооперативного планирования в среднем 187 %.

3. Предложен способ построения кооперативных потоковых блочно-параллельных алгоритмов, эффективно распараллеливающих решение прикладных задач, декомпозируемых на части. Кооперативные потоковые алгоритмы отличаются повышенными возможностями балансирования распределения данных и вычислительных операций на потоках и ядрах, изменением порядка выполнения потоков на каждом ядре с целью сокращения слотов времени на ожидание и простаивание, сокращением обменов данными между кэш памятью ядер и между уровнями памяти, возможностью поиска оптимального размера блока данных при учете всех факторов, влияющих на параметры многопоточного приложения.

4. Эффективность способа продемонстрирована на двух новых кооперативных потоковых блочно-параллельных алгоритмах решения систем линейных алгебраичесих уравнений (СЛАУ), построенных на базе блочно-параллельного метода Гаусса и представленных высокоуровневыми конечными автоматами. Алгоритмы равномерно распределяют вычислительную нагрузку между потоками и повышают загрузку ядер, уменьшают число передач управления между потоками, сокращают критический путь и повышают коэффициент распараллеленности на

графе вычислений. Имитационное моделирование показало преимущество предложенных кооперативных потоковых блочно-параллельных алгоритмов над известными блочно-параллельными алгоритмами на СЛАУ большого размера, на большом количестве используемых потоков, а также на многоядерных системах с большим количеством ядер. Проведенные эксперименты на больших СЛАУ демонстрируют ускорение над известными блочно-параллельными алгоритмами на 29.62 % в среднем по всем конфигурациям потоков.

5. Предложен новый кооперативный потоковый блочно-параллельный алгоритм поиска кратчайших путей между всеми парами вершин графа, который в шести режимах работы потоков учитывает особенности решаемой задачи, изменяет порядок вычисления блоков с целью увеличения загрузки ядер и сокращения обмена данными между уровнями иерархической памяти, обладает свойством масштабируемости. Он сокращает время решения задачи на 22.54 % по сравнению с блочно-параллельным алгоритмом Флойда – Уоршелла. Разработан неоднородный блочно-параллельный алгоритм поиска кратчайших путей, выделяющий четыре типа блоков, использующий преимущества векторизации и конвейеризации.

Положения, выносимые на защиту

1. Три версии архитектуры кооперативного планировщика потоков: базовая архитектура, состоящая из менеджера памяти, потоков планирования и пользовательских потоков; модифицированная архитектура с новым примитивом синхронизации и отдельной очередью заблокированных потоков в каждом потоке планирования; усовершенствованная архитектура с новым механизмом работы с очередью потоков в примитиве синхронизации и новыми алгоритмами блокировки и разблокировки потоков.

2. Способ построения кооперативных потоковых блочно-параллельных алгоритмов, балансирующих распределение блоков данных по потокам и потоков по ядрам, изменяющих порядок выполнения потоков на каждом ядре, сокращающих время простаивания потоков, уменьшающих обмен данными между кэшем ядер.

3. Два кооперативных потоковых блочно-параллельных алгоритма решения СЛАУ, представленных высокоуровневыми конечными автоматами, балансирующих вычислительную нагрузку между потоками и ядрами, уменьшающих число передач управления между потоками, сокращающих критический путь и повышающих коэффициент распараллеленности на графе вычислений.

4. Кооперативный потоковый блочно-параллельный алгоритм поиска кратчайших путей на графе, использующий шесть режимов работы в каждом

потоке и устанавливающий такой порядок вычисления блоков, который увеличивает полезную загрузку ядер. Неоднородный блочно-параллельный алгоритм поиска кратчайших путей, различающий четыре типа блоков и ускоряющий вычисление блоков каждого типа с использованием преимуществ векторизации и конвейеризации.

5. Программное обеспечение, реализующее разработанный планировщик, средства управления выполнением потоков и предложенные кооперативные потоковые блочно-параллельные алгоритмы решения прикладных задач на многоядерных системах.

Личный вклад соискателя ученой степени

Положения и результаты, выносимые на защиту, принадлежат лично автору. Научный руководитель доктор технических наук, профессор А. А. Прихожий, соавтор основных публикаций, принимал участие в формулировании целей исследования, постановке задач, определении возможных путей их решения и обсуждении полученных результатов. В публикациях с соавторами вклад соискателя определяется рамками излагаемых в диссертации результатов.

Апробация диссертации и информация об использовании ее результатов

Основные положения и результаты диссертационной работы докладывались и обсуждались на 16 научно-технических конференциях: Информационные технологии в технических и социально-экономических системах (Минск, Беларусь, 2013, 2014, 2015, 2016, 2017, 2018); Наука – образованию, производству, экономике: секция «Информационные технологии и автоматизация» (Минск, Беларусь, 2013, 2014, 2015, 2016, 2017, 2018); Open Semantic Technologies for Intelligent Systems (Minsk, Belarus, 2017); Информационные технологии и системы: проблемы, методы, решения: ИТС–2017 (Минск, Беларусь, 2017); Математические методы в технике и технологиях (Санкт-Петербург, Россия, 2017); BIG DATA and Advanced Analytics (Минск, Беларусь, 2018).

Результаты диссертационной работы внедрены и используются: на иностранном производственном унитарном предприятии «ИССОФТ СОЛЮШЕНЗ» (ПВТ, г. Минск); в учебном процессе Белорусского национального технического университета.

Опубликование результатов диссертации

По материалам диссертации опубликованы 24 печатные работы, в том числе 6 статей в рецензируемых научных журналах, 1 статья в научном журнале, 10 статей

в сборниках материалов научных конференций, 7 тезисов докладов на научных конференциях. Результаты диссертационной работы включены в отчет по НИР.

Общий объем публикаций по теме диссертации, соответствующий пункту 18 Положения о присуждении ученых степеней и присвоении ученых званий в Республике Беларусь, составляет около 7.1 авторского листа.

Структура и объем диссертации

Диссертация состоит из введения, общей характеристики работы, пяти глав, заключения, библиографического списка, списка публикаций автора, двух приложений. Общий объем диссертационной работы составляет 200 страницы, из них 96 страниц основного текста, 80 рисунков на 42 страницах, 16 таблиц на 7 страницах, библиография из 196 наименований на 15 страницах, включающая 24 публикации автора, и два приложения на 40 страницах.

ОСНОВНАЯ ЧАСТЬ

Во ***введении*** определена область исследования, обоснована актуальность темы диссертационной работы, сформулирована решаемая проблема.

В ***первой главе*** выполнен анализ тенденций развития многоядерных систем, моделей параллелизма в программах, технологий разработки параллельных алгоритмов, методов планирования параллельных вычислений, средств разработки параллельных программ.

На протяжении десятилетий число транзисторов на кристалле удваивалось каждые 24 месяца, а тактовая частота процессора – каждые 18 месяцев. Однако, в последние годы тенденция изменилась из-за проблемы отвода тепла, а рост производительности системы стал возможным благодаря производству процессоров со многими исполняемыми модулями (многоядерных – multi-core и мультиядерных – many-core). Так современный мультиядерный процессор Intel Xeon Phi состоит из 57–72 ядер с частотой 1.1–1.7 гигагерца.

Широкое применение многоядерных систем сдерживается двумя главными факторами: сложностью и трудоемкостью процесса разработки параллельных приложений; трудностями в существенном повышении производительности из-за проблем выявления параллелизма в алгоритмах и проблем учета архитектурных особенностей многоядерных систем при реализации этого параллелизма. Разработка методов, алгоритмов и средств решения этих проблем представляется перспективным направлением исследований и является целью настоящей работы.

Вторая глава посвящена методам и средствам организации, планирования и выполнения многопоточных приложений на многоядерных системах. Исследованы возможности операционной системы с вытесняющей многозадачностью,

рассмотрены основные средства назначения потоков на логические процессоры и физические ядра, а также способы учета особенностей архитектуры для выбора предпочтительного варианта назначения. Исследование проведено на двух известных блочно-параллельных алгоритмах μ1 и μ2, решающих СЛАУ методом Гаусса на архитектуре из двух процессоров Intel®Xeon®CPUE5530 с четырьмя ядрами, обслуживающих по два потока посредством технологии Hyper-Threading Technology. Аппаратная платформа включает 16 логических процессоров, выполняющих 16 потоков одновременно.

Исследованы два механизма ОС Windows Server 2008 R2 назначения потоков на логические процессоры (маска привязанности и идеальный процессор) и пять вариантов их сочетания: мягкая привязка, мягкая привязка NUMA, групповая привязка NUMA, жесткая привязка и жесткая привязка NUMA (рисунок 1). Наилучшие результаты для однопоточного приложения получены при использовании групповой привязки NUMA. Наилучшие результаты для 16-поточного приложения, реализующего алгоритмы μ1 и μ2, в сравнении с лучшим однопоточным вариантом получены для жесткой привязки NUMA (3.88 раза) и групповой привязки NUMA (4.74 раза) соответственно.

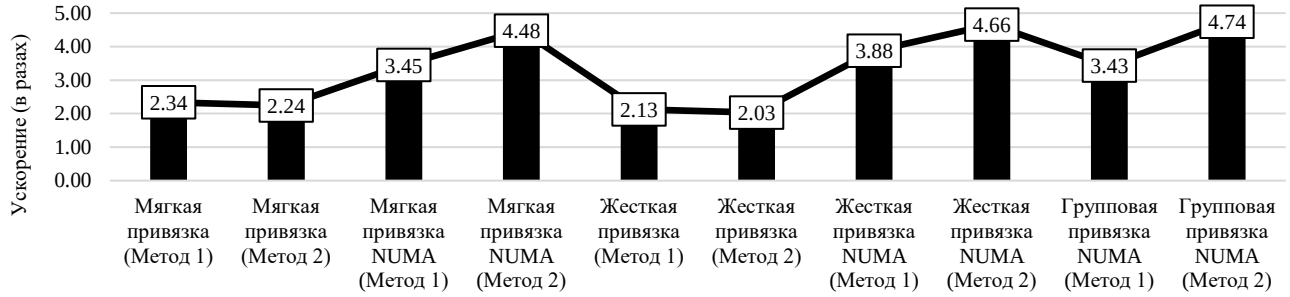


Рисунок 1. – Ускорение выполнения 16-поточных приложений над лучшим однопоточным приложением для алгоритмов μ1 и μ2 и 17920 переменных в СЛАУ

В диссертации производительность многопоточного приложения повышается двумя способами: 1) разработкой нового набора примитивов и программных средств построения и управления кооперативно выполняющимися потоками планировщика; 2) созданием кооперативной модели и новых блочно-параллельных алгоритмов решения задач. Разработка планировщика выполнена на примере Windows Server 2012 R2 с использованием механизма User-Mode Scheduling (UMS).

Разработаны три архитектуры A0, A1 и A2 кооперативного планировщика. Базовая архитектура A0 является реализацией программного интерфейса UMS и состоит из менеджера памяти, потоков пользователя и потоков планирования. Модифицированная архитектура A1 расширяет архитектуру A0 посредством разработки нового примитива синхронизации для исключения взаимодействия потока пользователя с ОС, а также посредством добавления своей очереди

заблокированных потоков пользователя в каждый поток планировщика. Усовершенствованная архитектура А2 расширяет архитектуру А1 посредством изменения механизма работы с очередью заблокированных потоков, а также посредством модификации и ускорения процессов блокировки и разблокировки потоков пользователя. Каждое ядро и все назначенные на него потоки пользователя обслуживается одним потоком планировщика. Порядок выполнения потоков пользователя определяется кооперативным алгоритмом и реализуется посредством прямой передачи управления между потоками пользователя. Синхронизация потоков пользователя, назначенных на разные ядра в архитектуре А0, выполняется при помощи потока планировщика и примитива синхронизации ОС (рисунок 2, *а*). Синхронизация в архитектуре А1 выполняется посредством предложенного примитива синхронизации и очереди заблокированных потоков пользователя, введенной в поток планировщика (рисунок 2, *б*). В архитектуре А2 управление процессами блокировки-разблокировки становиться более распределенным и быстрым за счет переноса очереди заблокированных потоков пользователя в примитив синхронизации (рисунок 2, *в*). Эффективность планировщика возрастает от А0 к А1 и от А1 к А2. В приложении к диссертации приведены программные коды планировщика на языке C/C++.

В ***третьей главе*** предложен способ построения кооперативных потоковых блочно-параллельных алгоритмов решения задач, базирующийся на возможностях планировщика, разработанного во второй главе.

Способ предусматривает: размещение одного блока данных алгоритма в одном потоке; группирование потоков на ядрах; загрузку ядер близко к 100 % полезными вычислениями за счет сокращения времени ожидания и простаивания потоков; установление минимально последовательного и максимально параллельного порядка выполнения и синхронизации потоков; поиск предельного уменьшения размера блока, за которым общее время работы алгоритма начинает возрастать из-за частых передач управления и частых посылок данных между потоками; моделирование алгоритма высокоуровневым конечным автоматом.

Разработаны два кооперативных потоковых блочно-параллельных алгоритма μ1κ (рисунок 3) и μ2κ (рисунок 4) решения СЛАУ, улучшающих параметры известных алгоритмов μ1 и μ2. Один алгоритм описывает поведение потока, находящегося в текущем состоянии и функционирующего в одном из четырех режимов: *start*, *active*, *passive* и *finish*. Поток выполняет вычисления, а также операции передачи данных, ожидания, и приема данных. Преимущество μ1κ состоит в увеличении возможности изменения порядка обработки строк матрицы СЛАУ при увеличении количества потоков, что позволяет строкам с более высоким индексом начать обрабатываться в более ранний момент времени. Преимущество μ2κ состоит в минимизации числа переходов от пересчета коэффициентов одной строки к пересчету коэффициентов другой строки, что сокращает число передач

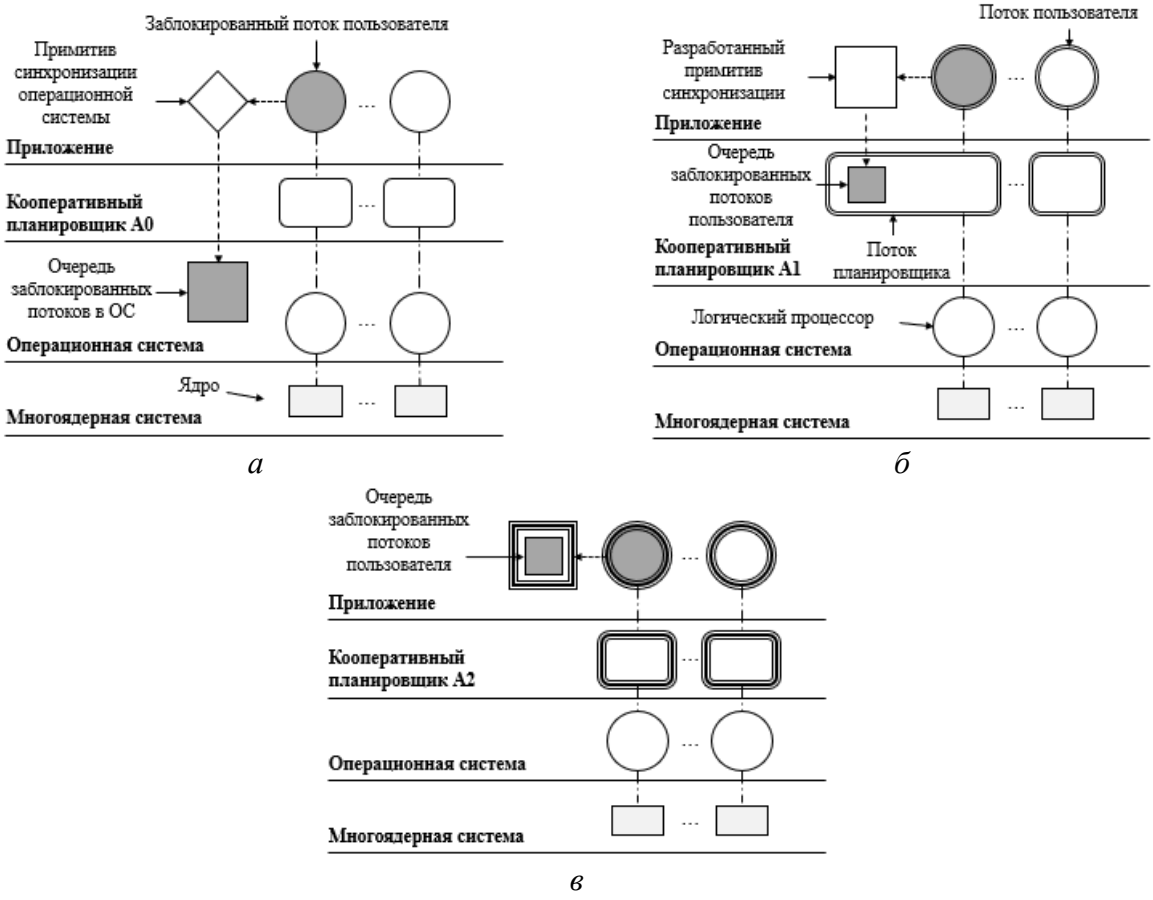
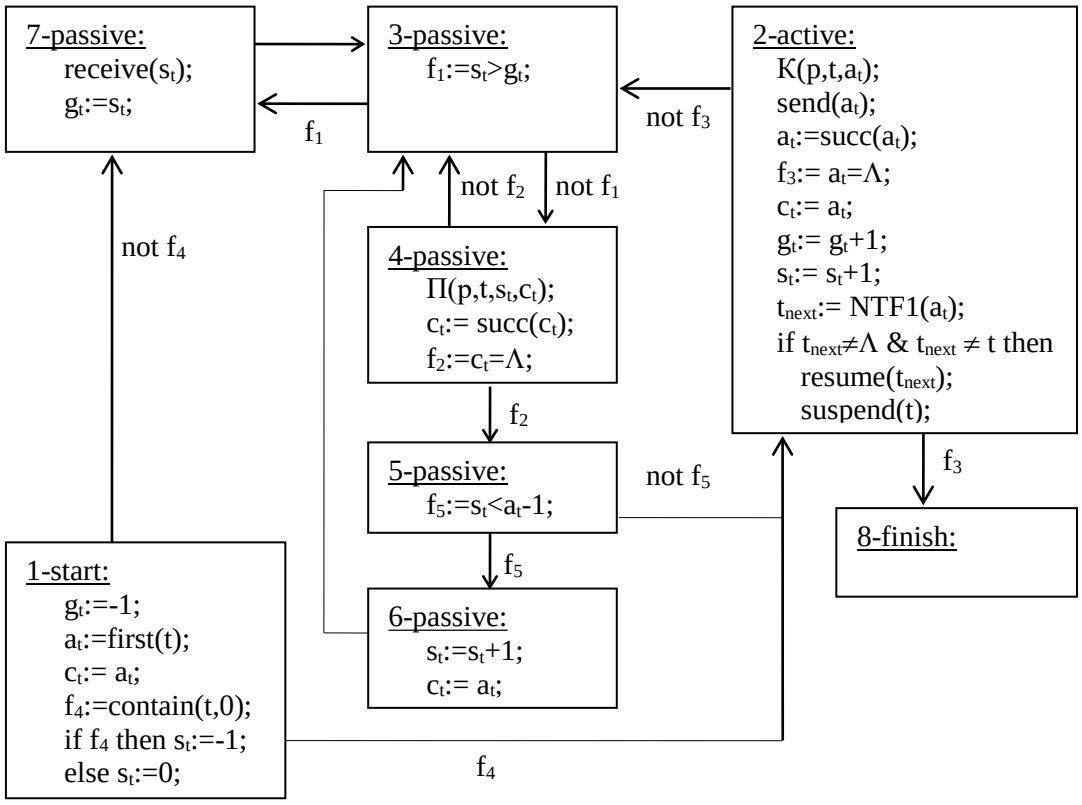


Рисунок 2. –Архитектуры планировщика А0 (*а*), А1 (*б*) и А2 (*в*); двойными и тройными линиями отмечены потоки пользователя и планировщика с усовершенствованными алгоритмами работы

управления между потоками по сравнению с алгоритмом μ1к. Имитационное моделирование известных μ1 и μ2 и кооперативных потоковых блочно-параллельных алгоритмов μ1к и μ2к показало преимущество последних по большинству параметров, важнейшими из которых (таблица 1) являются длина критического пути (CP), коэффициент распараллеливания (Prl) и средняя загрузка процессора (Ld).

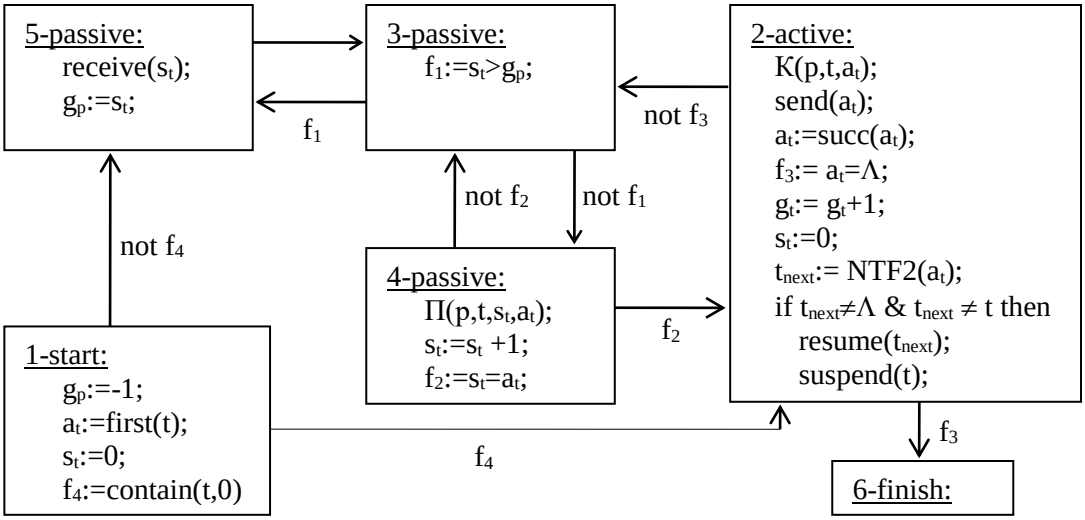
Таблица 1. – Зависимость параметров *CP*, *Prl* и *Ld* от числа строк в матрице СЛАУ для алгоритмов μ1, μ2, μ1к и μ2к, 8 потоков и 4 процессоров

Число строк СЛАУ	Алгоритмы / Параметры											
	μ1			μ2			μ1к			μ2к		
	CP	*Prl*	*Ld* %	*CP*	*Prl*	*Ld* %	*CP*	*Prl*	*Ld* %	*CP*	*Prl*	*Ld* %
8	36	2.00	50.0	36	2.00	50.0	30	2.40	60.0	30	2.40	60.0
16	104	2.62	65.4	104	2.62	65.4	98	2.78	69.4	86	3.16	79.1
32	372	2.84	71.0	336	3.14	78.6	366	2.89	72.1	294	3.59	89.8
64	1436	2.90	72.4	1184	3.51	87.8	1430	2.91	72.7	1094	3.80	95.1
128	5676	2.91	72.7	4416	3.74	93.5	5670	2.91	72.8	4230	3.90	97.6



t –поток; p –процессор; a_t – активная строка матрицы в t; g_t – глобально активная строка; c_t – пассивная строка в t; s_t – строка для пересчета c_t; К(p,t,a_t) – операция расчета активной строки; П(p,t,s_t,c_t) – операция расчета пассивной строки; NTF1(a_t) – следующий активный поток на p;

Рисунок 3. – Алгоритм μ1к решения СЛАУ (прямой ход)



t –поток; p –процессор; a_t – активная строка матрицы в t; g_t – глобально активная строка; c_t – пассивная строка в t; s_t – строка для пересчета c_t; К(p,t,a_t) – операция расчета активной строки; П(p,t,s_t,c_t) – операция расчета пассивной строки; NTF2(a_t) – следующий активный поток на p;

Рисунок 4. – Алгоритм μ2к решения СЛАУ (прямой ход)

В ***четвертой главе*** разработан новый кооперативный потоковый блочно-параллельный алгоритм (КПБПА) решения задачи поиска кратчайших путей между всеми парами вершин графа. Он изменяет порядок расчета блоков по сравнению с известным блочно-параллельным алгоритмом Флойда – Уоршелла (БФУ), максимально загружает ядра процессора и улучшает работу с многоуровневой памятью. Он использует новое понятие «уровень пересчета блока» матрицы кратчайших путей с целью изменения порядка вычислений блоков и сокращения при этом обмена данными между кэшем ядер, количества блокировок и зависимостей по данным между потоками, интервала синхронизации готовности данных, а также повышения локализации использования данных на каждом из ядер.

КПБПА вводит шесть режимов работы потока: ведущий, ведомый, дополняющий, пассивный типа А, пассивный типа Б и пассивный типа С. Для каждого режима он определяет свой набор рассчитываемых блоков, а также диапазон уровней расчета и порядок расчета блоков, принадлежащих потоку. Диаграмма переходов потока из одного режима в другой показана на рисунке 5.

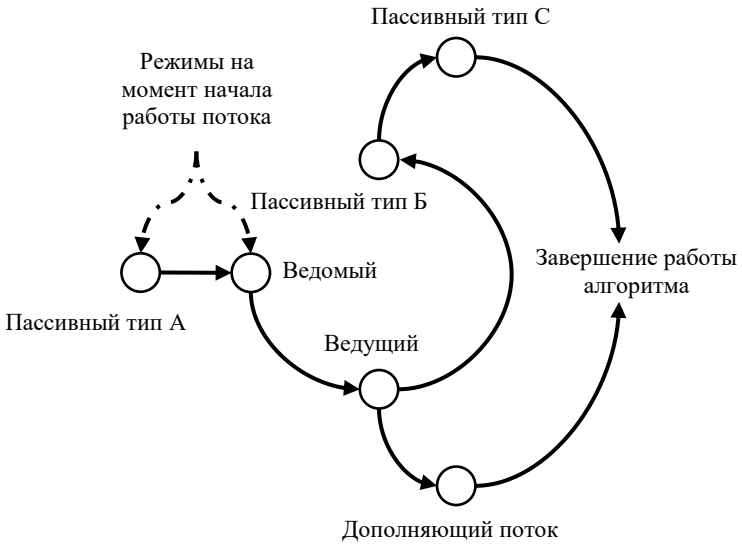


Рисунок 5. – Диаграмма переходов режимов потока

Взаимодействие потоков (рисунок 6) и параллельно-последовательный порядок расчета блоков в КПБПА проиллюстрируем на примере матрицы блоков размером 4×4, вычисляемой на двух процессорах. Процесс развернут во времени, измеряемом в числе вычисленных потоками блоков. На процессоре P_1 работают потоки t_0, t_2, на процессоре P_2 – потоки t_1, t_3. КПБПА увеличивает производительность по сравнению с БФУ до 22.54 % для всех размеров графа и всех размеров блока. Увеличение обусловлено сокращением критического пути в алгоритме КПБПА, которое достигнуто изменением порядка вычисления блоков. Оно обусловлено также сокращением числа обменов данными между ядрами и между уровнями иерархической памяти. КПБПА обладает свойством

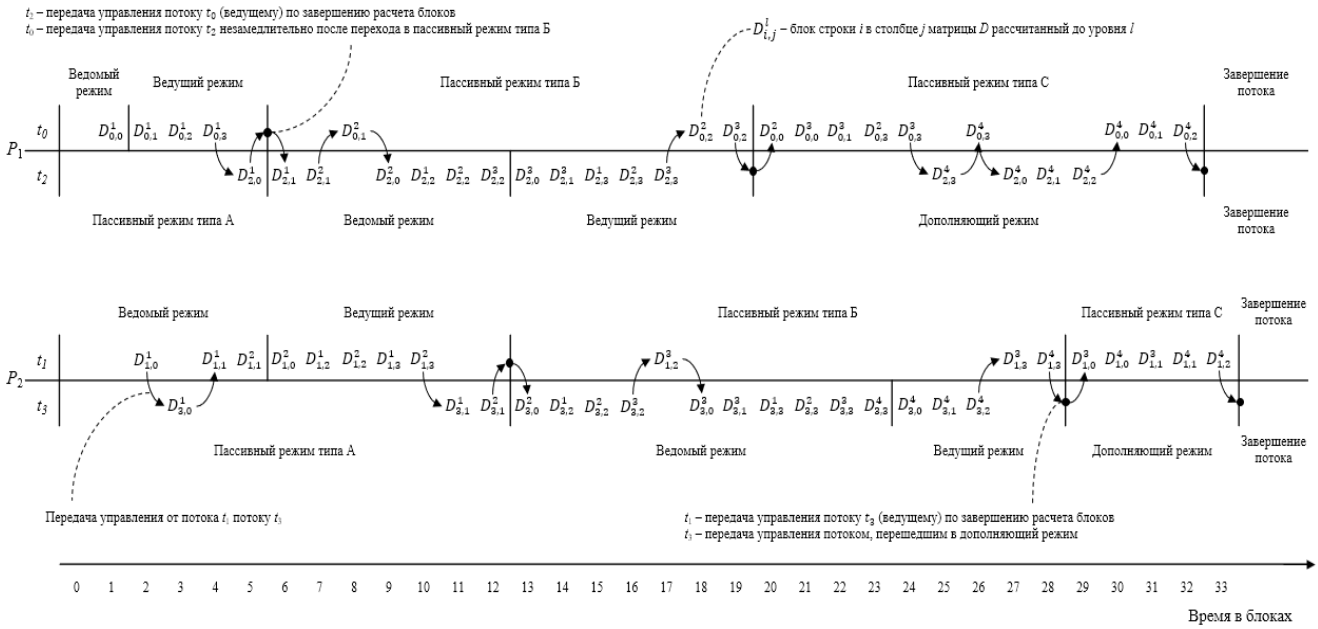


Рисунок 6. – Временная диаграмма работы КПБПА на матрице блоков 4×4 на двух процессорах

масштабируемости при настройке его на различные многоядерные архитектуры и при увеличении размера СЛАУ, числа потоков и числа ядер.

Анализ алгоритмов КПБПА и БФУ показал, что в них присутствует неоднородность по типам и способам вычисления блоков, которая использована для улучшения параметров алгоритмов путем настройки на многоядерную архитектуру. В работе идентифицированы четыре типа блоков, для каждого из них разработаны свои алгоритмы расчета, эффективно применены средства векторизации и конвейеризации вычислений, встроенные в ядра процессора.

В ***пятой главе*** представлены результаты экспериментального исследования, сравнительного анализа и практического применения разработанного кооперативного планировщика потоков и предложенных потоковых блочно-параллельных алгоритмов.

Сравнение архитектур планировщика выполнено на 8-ядерной системе с 16 логическими процессорами на алгоритмах μ1κ и μ2κ решения СЛАУ, включающей 15872 переменных, с числом потоков от 32 до 7936. Важнейшим параметром, характеризующим свойства планировщика и потоковых блочно-параллельных алгоритмов, является ускорение, получаемое многопоточным приложением, выполняемым на многоядерной системе посредством разных планировщиков. При исполнении обратного хода метода Гаусса, где время управления потоками доминирует над временем выполнения вычислительных операций, на алгоритме μ1κ планировщик A2 выиграл у A1 до 166.03 %, планировщик A1 выиграл у A0 до 125.47 %, а планировщик A2 выиграл у A0 до 270.46 %. На алгоритме μ2κ A2 выиграл у A1 до 411.11 %, A1 выиграл у A0 до 17,81 %, а A2 выиграл у A0 до 436.59 %.

Графики ускорения многопоточных приложений, реализующих алгоритмы μ1, μ2, исполняемые вытесняющим планировщиком ОС, и алгоритмы μ1k и μ2k, исполняемые разработанным кооперативным планировщиком, над однопоточным приложением, реализующим метод Гаусса, построенные на 8-ядерной системе с 16 логическими процессорами в зависимости от числа потоков, показывают, что оба алгоритма μ1k и μ2k имеют похожие характеристики (рисунок 7). На 992 потоках μ2k превзошел μ1k на 8.76 %, а на 7936 потоках уже μ1k превзошел μ2k на 3.32 %. На 3968 потоках алгоритмы μ1k и μ2k показали близкие максимальные ускорения 12.67 и 12.60 соответственно над однопоточной реализацией. Они на 21.7 % превосходят алгоритм μ1 (максимальное ускорение 10.41 на 1984 потоках) и еще больше превосходят (на 140.0 %) алгоритм μ2 (максимальное ускорение 5.28).

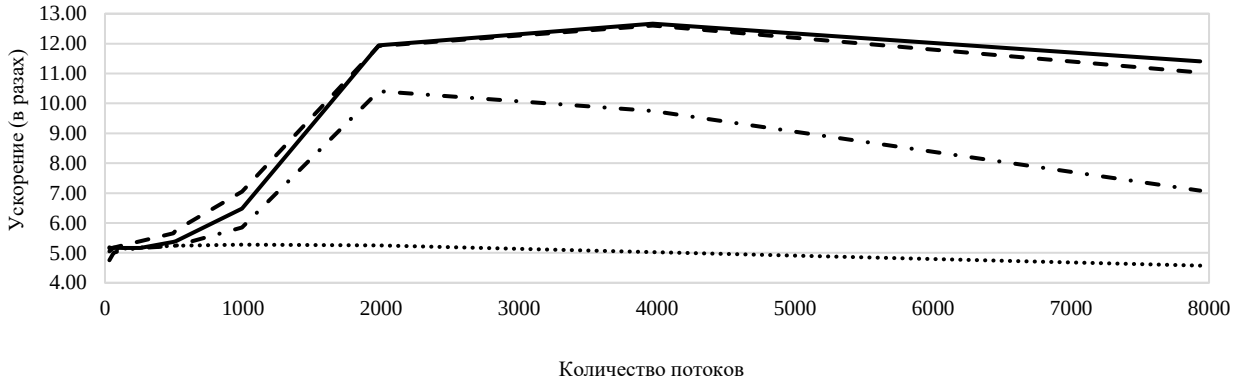
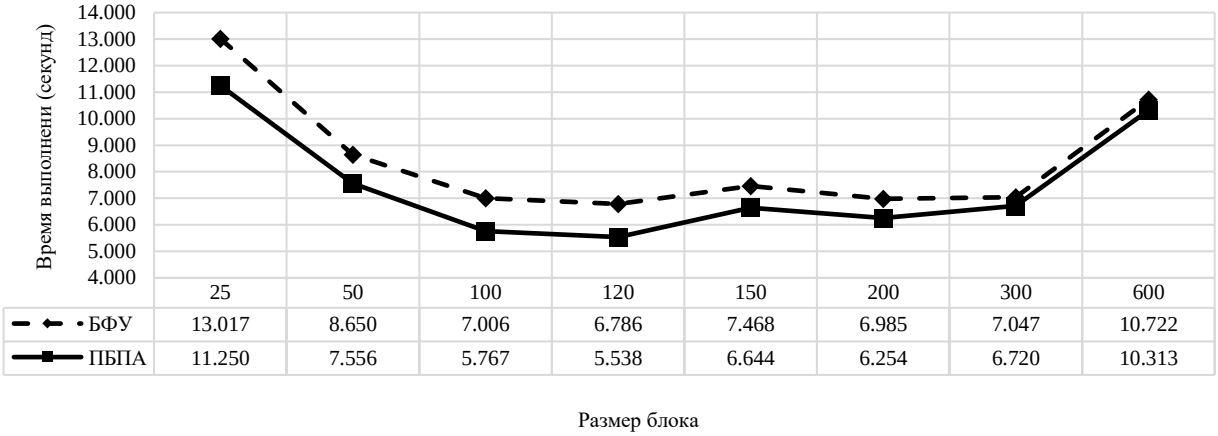


Рисунок 7. – Ускорение в разах алгоритмов μ1k (сплошная), μ2k (пунктирная), реализованных на архитектуре A2 планировщика (кооперативная многозадачность), и алгоритмов μ1 (штрихпунктирная), μ2 (точечная), реализованных планировщиком вытесняющей многозадачности, по сравнению с лучшим однопоточным приложением в зависимости от числа потоков

Приведем результаты экспериментальных исследований кооперативного потокового блочно-параллельного алгоритма КПБПА поиска кратчайших путей на графе, полученные на 4-ядерной системе, построенной на базе процессора Intel(R) Core (TM) i5-3450 CPU, архитектура IvyBridge. Многоуровневая кэш-память включает локальный уровень L1 емкостью 256 KB, локальный уровень L2 емкостью 1.0 MB и разделяемый уровень L3 емкостью 6.0 MB. Рисунок 8 показывает зависимость времени выполнения программных реализаций алгоритмов БФУ (реализован на OpenMP 3.0) и КПБПА (реализован кооперативным планировщиком) от размера блока на графе из 4800 вершин.

КПБПА сокращает время выполнения по сравнению с БФУ на всех восьми размерах блока. Выигрыш составил до 22.54 %, причем максимальный выигрыш получен при минимальном времени работы алгоритмов. Главным фактором успеха явилась способность КПБПА загружать ядра процессора близко к 100% и сокращать обмены данными между кэшем ядер и между уровнями иерархической



	25	50	100	120	150	200	300	600
БФУ	13.017	8.650	7.006	6.786	7.468	6.985	7.047	10.722
ПБПА	11.250	7.556	5.767	5.538	6.644	6.254	6.720	10.313

Рисунок 8. – Зависимость CPU-времени поиска кратчайших путей алгоритмами БФУ (пунктирная линия) и КПБПА (сплошная линия) от размера блока для графа из 4800 вершин

памяти. Предложенные однородные кооперативные потоковые блочно-параллельные алгоритмы хорошо векторизуется. Так КПБПА на графе размером 4800 вершин с блоком 120×120 выполняется с векторизацией в 3.72 раза быстрее чем без векторизации.

Выполнено также экспериментальное сравнение однородного БФУ с двумя версиями неоднородного потокового блочно-параллельного алгоритма. При малых размерах 200 блока, первая версия выигрывает у БФУ до 16.8 % процессорного времени. При увеличении размера блока до 600 уже вторая версия разнородного алгоритма начинает выигрывать у БФУ до 50 %. Неоднородные алгоритмы поддерживают эффективную конвейерную реализацию. По среднему числу тактов процессора, затраченных на выполнение одной инструкции, первая версия неоднородного алгоритма выиграла 15.8 % у БФУ на графе с 4800 вершинами, а вторая версия выиграла 54.9 % на графе с 9600 вершинами.

ЗАКЛЮЧЕНИЕ

Основные научные результаты диссертации

1. Установлено, что известные методы и средства организации и управления выполнением многопоточных приложений в ОС с вытесняющей многозадачностью, включая такие механизмы привязки потоков к логическим процессорам как маска привязанности, идеальный процессор и их комбинации, позволяют получить заметное ускорение многопоточной реализации при учете архитектуры многоядерной системы, однако не позволяют учесть особенности решаемой задачи и характер параллелизма, присутствующего в алгоритме ее решения, не обеспечивают достаточно эффективное управление вычислительными ресурсами [1, 8, 15, 18].

2. Системы с кооперативной многозадачностью позволяют учесть особенности решаемой задачи и специфику многоядерной системы. В диссертации разработаны три версии архитектуры кооперативного планировщика потоков. Базовая архитектура A0 состоит из менеджера памяти, потоков планировщика и пользовательских потоков. Модифицированная архитектура A1 вводит новый примитив синхронизации с целью исключения взаимодействия потоков пользователя с ОС и добавляет отдельную очередь заблокированных потоков в каждый поток планировщика. Усовершенствованная архитектура A2 изменяет механизм работы с очередью заблокированных потоков, модифицирует и ускоряет процессы блокировки и разблокировки пользовательских потоков за счет их переноса в примитив синхронизации. На задачах с малой долей вычислительных операций и большой долей времени на управление потоками модифицированная архитектура планировщика выигрывает в среднем у базовой архитектура 31.16 %, а усовершенствованная архитектура выигрывает в среднем у модифицированной 131.08 % и у базовой 186.66 % [4, 10, 15, 22, 24].

3. Предложен способ построения кооперативных потоковых блочно-параллельных алгоритмов, обеспечивающих эффективное параллельное на ядрах решение прикладных задач, декомпозируемых на части. Алгоритмы балансируют распределение блоков данных по потокам и потоков по ядрам; устанавливают порядок выполнения потоков на каждом ядре, минимизирующий слоты времени на управление, ожидание и простаивание потоков; повышают полезную загрузку ядер близко к 100%; позволяют находить оптимальный размер блока; сокращают обмен данными между кэшем ядер и между уровнями иерархической памяти [2, 6, 7, 9, 15, 17, 19, 21, 22].

4. Разработаны два новых кооперативных потоковых блочно-параллельных алгоритма решения СЛАУ, построенных на блочно-параллельном методе Гаусса и представленных высокоуровневыми конечными автоматами. Алгоритмы балансируют вычислительную нагрузку между потоками и повышают загрузку ядер, уменьшают число передач управления между потоками, сокращают критический путь и повышают коэффициент распараллеленности на графе вычислений. Выполнено имитационное моделирование алгоритмов, показывающее, что предложенные потоковые блочно-параллельные алгоритмы сильнее проявляют свои преимущества перед блочно-параллельными алгоритмами Гаусса для СЛАУ большого размера, для большого количества используемых потоков, а также для многоядерных систем с большим количеством ядер. Проведенные эксперименты показали ускорение предложенных потоковых блочно-параллельных алгоритмов над однопоточным методом Гаусса до 12.67 раз на 8-ядерной системе с 16 логическими процессорами и показали ускорение над блочно-параллельными алгоритмами Гаусса на 29.62 % в среднем по всем конфигурациям потоков [2, 4, 7, 15, 17, 20].

5. Задача поиска кратчайших путей на больших графах имеет широчайшее применение во многих прикладных областях: в базах данных для оптимизации обработки запросов, системах автоматизированного проектирования, микроэлектронике, инструментальных средствах оптимизации конвейеров, компьютерных играх, биоинформатике для анализа кластеров генов, планировании работы многоагентных систем, распознавании речи и т. д. Для реальных больших графов, известные методы и алгоритмы не решают эту задачу за приемлемое время. В диссертации предложен новый кооперативный потоковый блочно-параллельный алгоритм поиска кратчайших путей, который использует шесть режимов работы в каждом потоке, учитывает специфические особенности задачи, включая неоднородность вычисления блоков, изменяет порядок вычисления блоков по сравнению с известным блочным алгоритмом Флойда – Уоршелла, увеличивает загрузку ядер и сокращает обмен данными между кэш-памятью отдельных ядер и между уровнями иерархической памяти вычислительной системы, обладает свойством масштабируемости в отношении размера графа и размера многоядерной системы. Он сокращает время решения задачи на 22.54 % по сравнению с блочно-параллельными алгоритмами Флойда – Уоршелла, выгодно используя средства векторизации и конвейеризации вычислений. Предложенный разнородный алгоритм поиска кратчайших путей превосходит известный однородный алгоритм на 16.04 % для размера блока 200 и на 50.84 % для размера блока 600. [3, 5, 6, 11, 12, 13, 14, 15, 16, 17, 23].

Рекомендации по практическому использованию результатов

Разработанные кооперативный планировщик потоков, блочно-параллельные алгоритмы и программные средства ориентированы на практическое применение при решении задач большой вычислительной сложности на многоядерных системах.

Потоковые блочно-параллельные алгоритмы могут быть использованы для оптимизации обработки запросов в базах данных, при навигации мобильных объектов, в системах автоматизированного проектирования в микроэлектронике, при оптимизации вычислительных конвейеров, в компьютерных играх, для анализа кластеров генов в биоинформатике, планирования работы многоагентных систем, распознавания речи и т. д.

Научные и практические результаты диссертации могут быть использованы в процессе создания высокопроизводительных параллельных алгоритмов и масштабируемых многопоточных приложений для решения разнообразных прикладных задач большого размера, высокой вычислительной сложности, на перспективных многоядерных архитектурах.

СПИСОК ПУБЛИКАЦИЙ СОИСКАТЕЛЯ

Статьи в изданиях согласно перечню ВАК

1. Прихожий, А. А. Исследование методов реализации многопоточных приложений на многоядерных системах / А. А. Прихожий, О. Н. Карасик // Информатизация образования. – 2014. – № 1. – С. 43–62.

2. Прыхожы, А. А. Кааператыўныя блочна-паралельныя алгарытмы рашэння задач на шмат'ядравых сістэмах / А. А. Прыхожы, А. М. Карасік // Сістэмны аналіз і прыкладная інфарматыка. – 2015. – № 2. – С. 10–18.

3. Прихожий, А. А. Эвристический генетический алгоритм оптимизации вычислительных конвейеров / А. А. Прихожий, А. М. Ждановский, О.Н. Карасик, М. Маттавелли // Доклады БГУИР. – 2017. – № 1. – С. 34–41.

4. Карасик, О. Н. Усовершенствованный планировщик кооперативного выполнения потоков на многоядерной системе / О. Н. Карасик, А. А. Прихожий // Системный анализ и прикладная математика. – 2017. – № 1. – С. 4–11.

5. Прихожий, А. А. Разнородный блочный алгоритм поиска кратчайших путей между всеми парами вершин графа / А. А. Прихожий, О. Н. Карасик // Системный анализ и прикладная информатика. – № 3. – 2017. – С. 68–75.

6. Карасик, О. Н. Потоковый блочно-параллельный алгоритм поиска кратчайших путей на графе / О. Н. Карасик, А. А. Прихожий // Доклады БГУИР. – 2018. – № 2. – С. 77–84.

Статьи в других журналах

7. Прихожий, А. А. Кооперативная модель оптимизации выполнения потоков на многоядерной системе / А. А. Прихожий, О. Н. Карасик // Системный анализ и прикладная информатика. – 2014. – № 4. – С. 13–20.

Материалы конференций

8. Карасик, О. Н. Исследование влияния архитектуры многоядерной системы и ОС Windows на параметры многопоточного приложения / О. Н. Карасик, А. А. Прихожий // Информационные технологии в технических и социально-экономических системах: материалы междунар. науч.-практ. конф., Минск, 22 апреля 2014 г. / РИВШ. – Минск, 2014. – С. 19–23.

9. Карасик, О. Н. Экспериментальное исследование кооперативной модели выполнения потоков / О. Н. Карасик, А. А. Прихожий // Информационные технологии в технических и социально-экономических системах: материалы междунар. науч.-практ. конф., Минск, 22 апреля 2015 г. / РИВШ. – Минск, 2015. – С. 51–54.

10. Карасик, О. Н. Исследование реализации многопоточного приложения с учетом состояния ожидания потоков выполнения / О. Н. Карасик, А. А. Прихожий // Информационные технологии в технических и социально-экономических системах: материалы междунар. науч.-практ. конф., Минск, 22 апреля 2016 г. / РИВШ. – Минск, 2016. – С. 14–16.

11. Prihozhy, A. A. Semantic model for high-level synthesis of dataflow pipelines / A. A. Prihozhy, O. N. Karasik, O. M. Frolov // Open Semantic Technologies for Intelligent Systems: Proceedings of International Conference, Minsk, 16–18 February 2017 / BSUIR. – Minsk, 2017. – P. 415–420.

12. Карасик, О. Н. Экспериментальные исследования распараллеливания на многоядерной системе задачи поиска кратчайших путей на графе / О. Н. Карасик, А. А. Прихожий // Информационные технологии в технических и социально-экономических системах: материалы междунар. науч.-практ. конф., Минск, 20 апреля 2017 г. / РИВШ. – Минск, 2017. – С. 9–11.

13. Прихожий, А. А. Матричный многопоточный параллельный алгоритм поиска кратчайших путей на графе / А. А. Прихожий, О.Н. Карасик // Информационные технологии в технических и социально-экономических системах: материалы междунар. науч.-практ. конф., Минск, 20 апреля 2017 г. / РИВШ. – Минск, 2017. – С. 15–18.

14. Карасик, О. Н. Экспериментальное исследование разнородного блочно-параллельного алгоритма поиска кратчайших путей на графе / О. Н. Карасик, А. А. Прихожий // Информационные технологии и системы: проблемы, методы, решения (ИТС – 2017): материалы Республиканской науч.-техн. конф., Минск, 23–24 ноября 2017 г. / Минск: Четыре четверти, 2018. – С. 200–205.

15. Карасик, О. Н. Средства параллельного многопоточного решения инженерно-математических задач на многоядерных системах / О. Н. Карасик // Математические методы в технике и технологиях: сб. тр. XXX-й междунар. науч. конф., 10–12 октября 2017 г. / СПб: Изд-во политех. ун-та, 2017. – Т. 12, Ч. 1. – С. 138–141.

16. Прихожий, А. А. Разнородный блочно-параллельный алгоритм учитывает особенности многоядерной архитектуры / А. А. Прихожий, О. Н. Карасик // Информационные технологии в технических, политических и социально-экономических системах: материалы междунар. науч.-техн. конф., Минск, 19 апреля 2018 г. / Белорусский национальный технический университет. – Минск: БНТУ, 2018. – С. 6–7.

17. Прихожий, А. А. Кооперативная потоковая модель решения задач большой размерности на многоядерных системах / А. А. Прихожий, О. Н. Карасик // BIG DATA and Advanced Analytics: материалы 4-й Междунар. науч.-практ. конф., Минск, 3–4 мая 2018 г. / БГУИР. – Минск, 2018. – С. 378–383.

Тезисы докладов

18. Прихожий, А. А. Технология разработки параллельных приложений для многоядерных систем / А. А. Прихожий, О. Н. Карасик // Информационные технологии в технических и социально-экономических системах: материалы междунар. науч.-практ. конф., Минск, 25 апреля 2013 г. / РИВШ. – Минск, 2013. – С. 18.

19. Прихожий, А. А. Построение модели многопоточного параллельного приложения / А. А. Прихожий, О. Н. Карасик // Наука – образованию, производству, экономике: материалы 11-й междунар. науч.-техн. конф., секция «Информационные технологии и автоматизация», Минск, 2013 г.: в 4 т. / БНТУ. – Минск, 2013. – Т. 1. – С. 266–267.

20. Карасик, О. Н. Экспериментальные исследования методов реализации многопоточных приложений на многоядерной системе / О. Н. Карасик, А. А. Прихожий // Наука – образованию, производству, экономике: материалы 12-й междунар. науч.-техн. конф., секция «Информационные технологии и автоматизация», Минск, 2014 г.: в 4 т. / БНТУ. – Минск, 2014. – Т. 1. – С. 282–283.

21. Карасик, О. Н. Кооперативная модель реализации многопоточных приложений на многоядерных системах / О. Н. Карасик // Наука – образованию, производству, экономике: материалы 13-й междунар. науч.-техн. конф., секция «Информационные технологии и автоматизация», Минск, 2015 г.: в 4 т. / БНТУ. – Минск, 2015. – Т. 1. – С. 277–278.

22. Карасик, О. Н. Разработка многопоточных приложений с учетом неравномерного распределения нагрузки между потоками посредством состояния ожидания потоков выполнения / О. Н. Карасик, А. А. Прихожий // Наука – образованию, производству, экономике: материалы 14-й междунар. науч.-техн. конф., секция «Информационные технологии и автоматизация», Минск, 2016 г.: в 4 т. / БНТУ. – Минск, 2016. – Т. 1. – С. 300.

23. Карасик, О. Н. Экспериментальное исследование параллельных алгоритмов поиска кратчайших путей на графе в неоднородной многопроцессорной системе / О. Н. Карасик, А. А. Прихожий // Наука – образованию, производству, экономике: материалы 15-й междунар. науч.-техн. конф., секция «Информационные технологии и автоматизация», Минск, 2017 г.: в 4 т. / БНТУ. – Минск, 2017. – Т. 1. – С. 302.

24. Карасик, О. Н. Планировщик выполнения потоков на многоядерной системе / О. Н. Карасик // Наука – образованию, производству, экономике: материалы 16-й междунар. науч.-техн. конф., секция «Информационные технологии и автоматизация», Минск, 2018 г.: в 4 т. / БНТУ. – Минск, 2018. – Т. 1. – С. 234.

РЭЗЮМЭ

Карасік Алег Мікалаевіч

Кааператыўны шмат’паточны планавальнік і блочна-паралельныя алгарытмы рашэння задач на шмат’ядровых сістэмах

Ключавыя словы: кааператыўная шмат’задачнасць, шмат’ядровыя сістэмы, шмат’паточнасць, планавальнік, блочна-паралельныя алгарытмы.

Мэта працы: распрацоўка мадэлі і сродкаў планавання і выканання шмат’паточных праграм, стварэння паралельных алгарытмаў і праграм с палепшанымі параметрамі для шмат’ядровых сістэм.

Метады даследавання: тэорыя алгарытмаў, тэорыя паралельных праграм, метады сінхранізацыі праграмных патокаў, тэорыя графаў, тэорыя аўтаматаў.

Атрыманыя вынікі і іх навізна: прапанаваны тры версіі архітэктуры кааператыўнага планавальніка выканання шматпаточнага прыкладання: базавая архітэктура, якая складаецца з мэнэджара памяці, патокаў планавання і патокаў карыстальніка; мадыфікаваная архітэктура з новым прымітывам сінхранізацыі і асобнай чаргой заблакаваных патокаў карыстальніка ў кожным патоку планавання; ўдасканаленая архітэктура з новым механізмам працы з чаргой патокаў карыстальніка ў прымітывах сінхранізацыі і новымі алгарытмамі блакавання і разблакоўкі патокаў. Распрацавана кааператыўная мадэль пабудовы патокавых блочна-паралельных алгарытмаў, якая балансуе размеркаванне блокаў дадзеных па патокаў і патокаў па ядрах, якая змяняе парадак выканання патокаў, скарачае час чакання патокаў і памяншае абмен дадзенымі паміж кэшам ядраў. Распрацаваны два кааператыўная патокавыя блочна-паралельных алгарытмы рашэння СЛАУ і патокавы блочна-паралельны алгарытм пошуку найкароткіх шляхоў на графе, які выкарыстоўвае шэсць рэжымаў працы патоку і ўстанаўлівае такі парадак вылічэння блокаў, які павялічвае карысную загрузку ядраў, выкарыстоўвае перавагі векторизации і конвейеризации. Створана праграмнае забеспячэнне, якое рэалізуе распрацаваныя планавальнік, сродкі планавання выканання патокаў і прапанаваныя кааператыўныя патокавыя алгарытмы рашэння прыкладных задач на шмат'ядравых сістэмах. Праведзеныя шматлікія вылічальныя эксперыменты паказалі перавагу распрацаваных алгарытмаў і праграмных сродкаў над вядомымі.

Вобласць ужывання: атрыманыя вынікі могуць выкарыстоўвацца ў складзе складаных праграмных комплексаў, якія выконваюцца на шмат'ядравых сістэмах, а таксама для распрацоўкі кааператыўных паточных блочна-паралельных алгарытмаў рашэння разнастайных прыкладных задач на шматпрацэсарных сістэмах.

РЕЗЮМЕ

Карасик Олег Николаевич

Кооперативный многопоточный планировщик и блочно-параллельные алгоритмы решения задач на многоядерных системах

Ключевые слова: кооперативная многозадачность, многоядерные системы, многопоточность, планировщик, блочно-параллельные алгоритмы.

Цель исследования: разработка модели и средств планирования и выполнения многопоточных приложений, создание параллельных алгоритмов и программ с улучшенными параметрами для многоядерных систем.

Методы исследования: теория алгоритмов, теория параллельных программ, методы синхронизации потоков, теория графов, теория автоматов.

Полученные результаты и их новизна: предложены три версии архитектуры кооперативного планировщика выполнения многопоточного приложения: базовая архитектура, состоящая из менеджера памяти, потоков планирования и потоков пользователя; модифицированная архитектура с новым примитивом синхронизации и отдельной очередью заблокированных потоков пользователя в каждом потоке планирования; усовершенствованная архитектура с новым механизмом работы с очередью потоков пользователя в примитиве синхронизации и новыми алгоритмами блокировки и разблокировки потоков. Разработана кооперативная модель построения потоковых блочно-параллельных алгоритмов, балансирующих распределение блоков данных по потокам и потоков по ядрам, изменяющая порядок выполнения потоков, сокращающая время простаивания потоков и обмен данными между кэшем ядер. Разработаны два потоковых блочно-параллельных алгоритма решения СЛАУ и потоковый блочно-параллельный алгоритм поиска кратчайших путей на графе, использующий шесть режимов работы потока и устанавливающий такой порядок вычисления блоков, который увеличивает полезную загрузку ядер, использует преимущества векторизации и конвейеризации. Создано программное обеспечение, реализующее разработанные планировщик, средства планирования выполнения потоков и предложенные кооперативные потоковые алгоритмы решения прикладных задач на многоядерных системах. Проведенные многочисленные вычислительные эксперименты показали преимущество разработанных алгоритмов и средств над известными.

Область применения: полученные результаты могут использоваться в составе сложных программных комплексов, исполняющихся на многоядерных системах, а также для разработки кооперативных потоковых блочно-параллельных алгоритмов решения разнообразных прикладных задач на многопроцессорных системах.

SUMMARY

Karasik Oleg Nikolaevich

Cooperative multithreading scheduler and block-parallel algorithms for solving tasks on multi-core systems

Keywords: cooperative multitasking, multi-core systems, multithreading, scheduler, block-parallel algorithms, throughput.

The purpose of this thesis: development of a model and tools for scheduling and executing multi-threaded applications, creating block-parallel algorithms and programs with improved parameters for multi-core systems.

Research methods: theory of algorithms, theory of parallel programs, methods of synchronization of program threads, graph theory, theory of automata.

The obtained results and their novelty: three versions of the architecture for cooperative scheduling and execution of a multi-threaded application are proposed:

1. basic architecture consisting of a memory manager, scheduling threads and user threads;

2. modified architecture with a new synchronization primitive and a separate queue of blocked user threads in each scheduling flow;

3. improved architecture with a new mechanism for processing the user threads queue in the synchronization primitive and new algorithms for blocking and unblocking threads.

A cooperative model for constructing threaded block-parallel algorithms has been developed. The model balances the distribution of data blocks among threads and balances threads across cores, changes the order in which the threads execute, reduces the idle time of threads and reduces data exchanges between the cores' caches. Two threading block-parallel algorithms for solving the SLAE and one threading block-parallel algorithm for finding all vertex-pairs shortest paths on a graph, that uses six modes of thread operation, are proposed. The algorithms establish such an order of calculating blocks that increases cores payload and takes advantage of vectorization and pipelining in cores. Program tools, which implement the developed scheduler of multi-threaded applications, perform scheduling of the threads execution, and implement the proposed cooperative threaded block-parallel algorithms for solving applied problems on multi-core systems are created. The numerous computational experiments performed show the advantages of the developed algorithms and tools over the known ones.

Field of application: The obtained results can be used as part of complex software products running on multi-core systems, as well as for developing cooperative multithreading block-parallel algorithms, which solve various applied tasks on multiprocessor systems.

Научное издание

Карасик Олег Николаевич

АВТОРЕФЕРАТ

диссертации на соискание ученой степени
кандидата технических наук

Подписано в печать __.__.2019	Формат 60x84 1/16.	Бумага офсетная.
Гарнитура «Таймс»	Отпечатано на ризографе.	Усл. печ. л. ___
Уч. изд. л. __	Тираж __ экз.	Заказ _.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
Распространителя печатных изданий №1/238 от 24.03.2019
№ 2/113 от 07.04.2014, № 3/615 от 07.04.2019.
ЛП № 02330/264 от 14.04.2019
220013, Минск, П. Бровки, 6.