На правах рукописи

Аль-Марди Мохаммед Хайдар Авадх

МОДЕЛИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ С УЧЁТОМ СХЕМЫ ОБМЕНА И ОБЪЁМА ПЕРЕДАВАЕМЫХ СООБЩЕНИЙ

Специальность 05.13.11 - Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени
кандидата технических наук

Санкт-Петербург – 2019

Работа выполнена в Федеральном государственном автономном образовательном учреждении высшего образования «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» имени В. И. Ульянова (Ленина)» (СПбГЭТУ «ЛЭТИ») на кафедре вычислительной техники.

Научный руководитель: **Шичкина Юлия Александровна**
доктор технических наук, профессор кафедры вычислительной техники СПбГЭТУ «ЛЭТИ», г.Санкт-Петербург

Официальные оппоненты: **Богатырев Владимир Анатольевич**
доктор технических наук, профессор, профессор кафедры вычислительной техники, ФГАОУ ВО «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики», г.Санкт-Петербург

Гордеев Александр Владимирович
доктор технических наук, профессор, профессор кафедры вычислительных систем и сетей, ФГАОУ ВО «Санкт-Петербургский государственный университет аэрокосмического приборостроения», г.Санкт-Петербург

Ведущая организация: **ФГБОУ ВО Санкт-Петербургский государственный университет,** г.Санкт-Петербург

Защита диссертации состоится 29 мая 2019 года в 15.30 часов на заседании диссертационного совета Д212.238.01 Санкт-Петербургского государственного электротехнического университета «ЛЭТИ» имени В. И. Ульянова (Ленина) по адресу: 197376, Санкт-Петербург, ул. Профессора Попова, д. 5.

С диссертацией можно ознакомиться в библиотеке ФГАОУ ВО «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)» и на сайте университета www.etu.ru в разделе «Подготовки кадров высшей квалификации» - «Объявление о защитах»

Отзывы на автореферат в двух экземплярах, заверенные печатью, просим направлять по адресу: 197376, Санкт-Петербург, улица Профессора Попова, д. 5.

Автореферат разослан « 28 » марта 2019 г.

Ученый секретарь
диссертационного совета Д 212.238.01

кандидат технических наук Пазников А.А.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы исследования. Актуальность формируется из следующих позиций:

- Рост числа задач, требующих применение высокопроизводительных систем. Требования науки и производства стимулируют развитие высокопроизводительных систем и программных средств для них. Последние в свою очередь открывают новые возможности для повышения точности математических моделей, охвата большего числа параметров и т.д., что опять же приводит к новым задачам, для которых уже недостаточно последних достижений в области вычислительных систем и т.д.
- Рост объемов данных. Развитие веб-технологий, распространение мобильных устройств, возникновение «Интернета вещей» привели к появлению проблем обработки данных и их хранения в традиционных реляционных базах данных. Объём информации стал насколько велик, что обрабатывать её традиционными способами стало нецелесообразно, в некоторых случаях и вовсе невозможно. Самой основной проблемой данных большого объёма являются затраты на их обработку, поскольку она требует дорогостоящего оборудования и высокой квалификации от специалистов. Второй проблемой является фильтрация данных, поскольку из большого объёма структурированной, слабоструктурированной и неструктурированной информации необходимо выделить только те данные, которые необходимы для обработки. Третьей проблемой является конфиденциальность данных. Четвёртой проблемой является потеря информации. С увеличением объёмов информации растёт и сложность ее хранения. Постоянно растущие объёмы информации требуют не только изобретения высокопроизводительных вычислительных систем, но и новых подходов к обработке данных, основанных на параллельных вычислениях.
- Неэффективность использования средств вычислительной техники. Как показали множественные публикации, можно значительно ускорить работу параллельной программы, если перестроить ее должным образом. При этом часто наблюдается не только ускорение работы самой программы, но и уменьшение требуемых вычислительных ресурсов, таких как ядер, памяти и т.д.
- Быстрое устаревание существующих программ и невозможность их применения на новых вычислительных мощностях.

В настоящей диссертационной работе рассмотрены подходы, позволяющие уменьшить проблемы применения высокопроизводительной техники при решении сложных задач с большим объемом обрабатываемых данных за счет преобразования параллельных алгоритмов к форме, допускающей эффективное использование вычислительных ресурсов.

В связи с этим **целью** диссертационной работы является разработка методов организации параллельных вычислений с учетом времени выполнения операций алгоритма и схемы обмена данными при межпроцессном взаимодействии применительно к вычислительным системам с распределенной памятью.

Для достижения поставленной цели в работе решались следующие **задачи:**

1. Разработать методы построения расписания параллельного алгоритма с учетом схемы обмена данными при межпроцессном взаимодействии.
2. Проанализировать эффективность применения разработанных методов в совокупности с другими методами построения расписания параллельного алгоритма.

3. Разработать методику применения методов построения расписания параллельного алгоритма с учетом схемы обмена данными при межпроцессном взаимодействии и дополнительных свойств алгоритма и вычислительной системы.

4. Адаптировать разработанные методы под применение в распределённой вычислительной среде и в запросах к базам данных.

Объектом исследования являются параллельные и распределённые вычисления.

Предметом исследования являются методы оптимизации параллельных алгоритмов по ряду параметров, таких как время и объем вычислительных ресурсов.

Методы исследования. Для решения поставленных задач в диссертационном исследовании использовались методы теории графов, теории множеств, теории оптимизации, теории параллельных вычислений и матричной алгебры.

Основные научные положения, выносимые на защиту:

1. Метод оптимизации параллельных алгоритмов с учетом межпроцессного обмена данными и без учета времени выполнения операций.
2. Формула, определяющая минимальное время выполнения алгоритма; формулы для расчета числа проекций информационного графа на граф взаимосвязей узлов; формула расчета числа узлов информационного графа запроса к базе данных.
3. Метод оптимизации параллельных алгоритмов с учетом межпроцессного обмена данными и времени выполнения операций алгоритма.
4. Методика применения разработанных методов при добавлении нового параметра (например, объёма данных) исследуемого алгоритма.
5. Метод организации распределенных вычислений.

Научная новизна

1. Предложены новые методы оптимизации параллельных алгоритмов с учетом межпроцессного обмена данными.
2. Выведены формулы расчета различных показателей алгоритма, таких как минимальное время выполнения алгоритма, число узлов информационного графа запроса к базе данных, число проекций информационного графа на граф взаимосвязей узлов.
3. Показана возможность комбинации разработанных методов с другими методами оптимизации параллельных алгоритмов.
4. Разработана методика применения методов оптимизации параллельных алгоритмов в распределённой вычислительной среде.
5. Показана возможность применения разработанных методов к запросам в базах данных.

Теоретическая значимость результатов исследования заключается в:

1) разработке метода оптимизации параллельных алгоритмов с учетом межпроцессного обмена данными;
2) выведении формул расчета различных показателей алгоритма, таких как минимальное время выполнения алгоритма, число узлов информационного графа запроса к базе данных, число проекций информационного графа на граф взаимосвязей узлов;
3) разработке метода оптимизации алгоритма на взвешенных графах с учетом времени выполнения операций алгоритма;

4) обосновании возможности добавлении новых параметров для расширения методов и их улучшения;

5) разработке метода организации распределенных вычислений на основе полученных новых методов для параллельных вычислений.

Практическая значимость результатов исследования заключается в применимости:

1) разработанных методов для параллельных и распределенных вычислений;

2) разработанных программных модулей как по отдельности, так и в совокупности с другими модулями оптимизации алгоритмов;

3) разработанных методов не только к классическим алгоритмам, но и к запросам в базах данных.

Внедрение результатов работы. Теоретические и практические результаты диссертационной работы используются в программах ООО ЭК «СТИ» и ООО ПО «Энергоресурс», что подтверждено актами о применении. Результаты работы используются в учебном процессе кафедры вычислительной техники ФГАОУ ВПО «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)» при преподавании курсов «Параллельные алгоритмы и системы», «Распределенные базы данных».

Апробация результатов работы. Основные положения диссертационной работы докладывались и обсуждались на конференциях различного уровня: 69-я научно-техническая конференция профессорско-преподавательского состава ЛЭТИ, 16th International Conference NEW2AN 2016, 9th Conference ruSMART 2016, 18th International Conference on Computational Science and Its Applications, ICCSA 2018, 18th International Conference on Next Generation Wired/Wireless Advanced Networks and Systems NEW2AN 2018.

Публикации. Полученные основные теоретические и практические результаты диссертационного исследования опубликованы в 8 трудах, среди них: 4 научные статьи, опубликованные в журналах, входящих в перечень ВАК, 2 научные публикации в журналах, входящих в базы цитирования Web of Science и Scopus, 2 публикации в сборниках конференций. Также получено свидетельство о государственной регистрации программы для ЭВМ «Multithreaded execution of SQL-queries» № 2018619271, дата гос. рег. 02.08.18.

Личный вклад автора во всех работах, выполненных в соавторстве, составляет более 75% и включает постановку задачи, разработку основных методов и средств для проведения исследований, обработку и анализ результатов. Автор является непосредственным исполнителем теоретических и экспериментальных исследований.

Структура и объём диссертации.

Диссертация состоит из введения, 4 глав, заключения, списка литературных источников, состоящего из 124 наименований, 2 приложений. Работа изложена на 180 машинописных страницах, включает 57 рисунков и 7 таблиц.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обоснована актуальность темы диссертационной работы, освещены объект и предмет исследования, сформулированы цель и задачи исследования, основные положения, выносимые на защиту, прописана научная

новизна, описаны теоритическая и практическая значимость результатов, а также сведения о публикациях по тематике работы.

В первой главе приводится обзор основного инструментария, применяемого при проводимом исследовании.

Проведен анализ способов представления алгоритмов, рассмотрены достоинства и недостатки каждого способа. Установлено, что наиболее подходящим способом для оптимизации параллельных алгоритмов является способ представления алгоритмов с помощью ориентированных графов. Проведен анализ способов формализованного представления графов, в котором показано, что ориентированные разреженные графы эффективнее всего обрабатывать с помощью списков связности или ребер. Алгоритмов обработки информационных графов с помощью списков очень мало, в то время как эти алгоритмы являются более эффективными по скорости выполнения по сравнению с остальными графовыми алгоритмами.

Обосновано решение применять для оптимизации параллельных алгоритмов информационные графы.

Проведен анализ существующих методов оптимизации и модификации параллельных алгоритмов, представленных информационными графами. Выявлены основные особенности, достоинства и недостатки этих методов. Установлено, что эти методы не учитывают межпроцессное взаимодействие между вычислительными узлами. Также установлено, что часть из них может послужить основой для создания методов оптимизации алгоритмов с учетом схемы передачи данных между процессами.

Во второй главе рассмотрены виды параллелизма. Предлагаемые в работе методы оптимизации алгоритмов по времени выполнения основываются на параллелизме задач, методы оптимизации запросов по времени выполнения основываются на параллелизме задач и данных. Далее в главе приводятся в форме схемы основные этапы разработки методов оптимизации параллельного алгоритма по объему коммуникаций. Представленная схема отображает связь между разработанными методами. Построенная концептуальная модель предметной области исследования отображает взаимосвязь между объектом исследования, задачами, используемыми моделями и применяемым математическим инструментарием.

Проведенные исследования типов связей вершин информационного графа и разделение их на бинарные и множественные связи позволило вывести формулу оценки минимального общего времени выполнения алгоритма (1):

$$t_{min} = \sum_{k=1}^{m} M + 2 \sum_{k=1}^{m} M_c \qquad (1)$$

где:

t_{min} - минимальное общее время алгоритма, которое может быть достигнуто при оптимизации расписания по объему межпроцессорных коммуникаций;

$k = 1,2, \ldots, m$; m – число групп графа;

M – число групп, содержащих вершины только с бинарными связями входных данных;

M_c – число групп, в которых хотя бы одна вершина имеет более одного входящего в нее ребра (множественные связи).

Данная формула позволяет оценить необходимость оптимизации данного алгоритма.

Далее в главе представлен разработанный метод оптимизации параллельного алгоритма с учетом схемы передачи данных между процессами при одинаковом времени выполнения операций. Суть метода заключается в переборе вершин по каждому ярусу, начиная с предпоследнего, и перемещении вершин графа на просматриваемом ярусе таким образом, чтобы операции, зависимые по данным, выполнялись на одном вычислительном узле. Метод реализуется при следующих ограничениях на оптимизируемый алгоритм и вычислительную систему: вычислительная система неограниченна по количеству вычислительных узлов; каждая операция обладает одинаковым объемом входных данных; каждая операция выполняется за равное время, условно равное 1 ед.; время передачи данных между двумя любыми процессами является постоянным, условно равным 1 ед.

Рассмотрены четыре направления реализации данного метода. Ниже приведены описания методов в направлениях перебора вершин «**в ширину с конца граф**» и «**в глубину с конца графа**».

Метод оптимизации параллельного алгоритма в ширину с конца графа

Обозначения: Пусть m – общее число групп вершин, полученное произвольным способом, например, методом оптимизации информационного графа по ширине с помощью матрицы или списка смежности, k –номер текущей группы вершин, n_k – число вершин графа в группе с номером k, i – номер текущей вершины графа в группе вершин с номером k, j – номер текущей вершины графа в группе вершин с номером k -1.

1. Процесс перестановки вершин графа начинается с последней группы. Номер очередной группы вершин $k=m$.

В первую очередь необходимо в расписание поставить вершины графа, которые находятся в смежных группах и имеют бинарную связь, и только потом расставлять вершины графа с множественными связями. Это поможет избежать лишних простоев вычислительных узлов.

2. В k-й группе вершин выбрать первую вершину. Считать номер позиции этой вершины графа в группе вершин равным 0: $i=0$.

3. Сравнить вершину графа Mk_i последовательно с вершинами предыдущей (k - 1)-й группы: $Mk\text{-}1_j$, $j=i..n_{k-1}$. Если в (k - 1)-й группе существует j-я вершина, напрямую связанная в информационном графе ребром с данной вершиной ($Mk\text{-}1_j$, где j – номер позиции вершины графа в группе вершин, $j>=i$), то вершину графа $Mk\text{-}1_j$, необходимо переместить в своей группе вершин в i-ю позицию.

4. Если в k-й группе вершин перебраны еще не все вершины графа, то $i = i+1$ и шаг 3. Иначе перейти на шаг 5.

5. Если в k-й группе вершин перебраны все вершины графа и $k >2$, то $k= k$-1 и шаг 3. Если $k = 2$, то шаг 6.

6. Конец метода.

Данный метод оптимизации параллельного алгоритма можно применять в другом направлении: в ширину с начала. Основное отличие этого направления обхода вершин графа заключается в том, что процесс перестановки вершин графа начинается с первой группы вершин.

Метод оптимизации параллельного алгоритма в глубину с конца графа

Обозначения: Пусть m – общее число групп вершин, полученное произвольным способом, например, методом оптимизации информационного графа по ширине с помощью матрицы или списка смежности, k –номер текущей группы вершин, n_k – число вершин графа в группе вершин с номером k, i – номер текущей

вершины графа в группе вершин с номером k, r – номер группы вершин в которой ищется связанная напрямую вершина, j – номер текущей вершины графа в группе вершин с номером r.

1. Процесс перестановки вершин графа начинается с последней группы вершин: $k=m$, $r=m$-1.
2. Как и в предыдущих методах в первую очередь необходимо в расписание поставить вершины, которые находятся в смежных группах и имеют бинарную связь. В k-й группе выбрать первую вершину графа. Считать номер позиции этой вершины в группе вершин равным 0: $i=0$.
3. Сравнить вершину графа Mk_i (где i – номер позиции вершины графа в группе вершин) последовательно с непомеченными вершинами r-й группы, Mr_j, где $j=0..n_r$. Если в r-й группе существует вершина, напрямую связанная в информационном графе ребром с данной вершиной, Mr_j, то вершину графа Mr_j, необходимо переместить в своей группе вершин в i-ю позицию и пометить ее. Если в r-й группе вершин нет вершины, связанной с вершиной графа Mk_i, то шаг 5.
4. Если $k>2$, то $k=k$-1, $r=r$-1 и шаг 3, иначе шаг 6
5. Взять для сравнения следующую группу вершин. Если $r>2$, то $r=r$-1 и шаг 3. Иначе шаг6.
6. Если $m>2$, то $m=m$-1 и шаг 7. Иначе шаг 8.
7. Если в m-й группе есть непомеченная вершина графа, то i = номер непомеченной вершины графа, $k=m$, $r=m$-1 и шаг 3. Иначе шаг 6.
8. Конец метода.

Метод оптимизации параллельного алгоритма в глубину, также, как и в ширину, можно проводить начиная не с последней группы вершин графа, а с первой. В диссертации подробно изложен метод оптимизации параллельного алгоритма в глубину с начала.

С целью определения наилучшего направления для сокращения времени выполнения алгоритма и числа простоев вычислительной системы были проанализировали четыре направления перебора вершин графа: в ширину с начала, в ширину с конца, в глубину с начала и в глубину с конца алгоритма.

Разработанный метод оптимизации параллельного алгоритма по объему передач между процессами был протестирован на различных по сложности и числу вершин графах с помощью специального программного обеспечения, написанного на Java. Результаты тестирования метода приведены на рис. 1.

Рисунок 1 - Результаты тестирования применения метода в различных направлениях перебора вершин информационного графа

Установлено, что, отдать предпочтение одному из направлений перебора вершин графа невозможно. В зависимости от связей между вершинами в графе наилучший результат может быть достигнут по разным направлениям. В целом при тестировании направлений перебора вершин на 100 графах различной структуры и объема связей и вершин было установлено, что в случае перебора вершин графа **в глубину с конца** чаще всего удается достичь наибольшего сокращения простоев вычислительной системы.

Третья глава посвящена методам оптимизации алгоритма по времени выполнения с учетом схемы обменов данными при следующих ограничениях на оптимизируемый алгоритм и вычислительную систему: вычислительная система неограниченна по количеству вычислительных узлов; каждая операция обладает одинаковым объемом входных данных; время выполнения каждой операции разное; время передачи данных между двумя любыми процессами является постоянным, условно равным 1 ед.

В главе выведены правила расчета параметров операций, соответствующих вершинам информационного графа, в зависимости от различных этапов выполнения метода оптимизации алгоритма.

Правило 1.

Для каждой i-й вершины графа в k-й группе вершин M_{ki} время выполнения j-й операции в (k+1)–й группе вершин $M_{(k+1)j}$ на i-м процессе с учетом передачи данных следует рассчитывать следующим образом:

> если операция $M_{(k+1)j}$ при перестановке на i-й процесс должна будет получать информацию с другого процесса, то ко времени ее выполнения необходимо прибавить 1.

Формализовано это можно описать так:

Пусть $M_{(k+1)} = \{v_1, \ldots, v_j, \ldots, v_{n(k+1)}\}$ – группа вершин, в которой происходит перестановка вершин графа, $T_{(k+1)} = \{t_1, \ldots, t_j, \ldots, t_{n(k+1)}\}$ – непосредственное время выполнения каждой операции из группы вершин $M_{(k+1)}$.

Тогда время выполнения каждой операции из группы вершин $M_{(k+1)}$ с учетом передачи данных на i-м процессе равно:

$$T'=\{T'_{(k+1)}\}_i = \{t_j+\alpha\},\ j=1..n_{(k+1)},\ i=1..n_k,$$

$$\text{где } \alpha = \begin{cases} 1, \text{ если } \exists\left(M_{ki}, M_{(k+1)j}\right) \\ 0, \text{ если } \nexists\left(M_{ki}, M_{(k+1)j}\right) \end{cases} \quad (2)$$

Правило 2.

Для каждой i-й вершины графа в k-й группе вершин M_{ki} ранний срок выполнения j-й операции в (k+1)–й группе вершин $M_{(k+1)j}$ на i-м процессе с учетом времени выполнения j-й операции, полученном на i -м шаге, следует рассчитывать следующим образом:

> ко времени выполнения группы вершин $M_{(k+1)j}$ на i-м процессе необходимо прибавить максимальный ранний срок выполнения операции M_{ki} и операций k-й группы вершин, бинарно-связанных с операцией $M_{(k+1)j}$.

Формализовано это можно описать так:

Пусть $M_{(k+1)} = \{v_1, \ldots, v_j, \ldots, v_{n(k+1)}\}$ – группа вершин, в которой происходит перестановка вершин графа, $M_k = \{u_1, \ldots, u_j, \ldots, u_{nk}\}$ – группа вершин графа на закреплённом (фиксированном) ярусе, $T'=\{T'_{(k+1)}\}_i$ (i=1..n_k) – время выполнения каждой операции из группы вершин $M_{(k+1)}$ с учетом передачи данных на i-й процесс.

Тогда ранний срок выполнения на i-м процессе каждой операции из группы вершин $M_{(k+1)}$ с учетом передачи данных равно:

$$\tau'=\{\tau'_{(k+1)}\}_i = \{T'_{(k+1)i} + \max(\tau_i, \tau_r\},$$
$$\text{для } \forall r: \exists\left(M_{kr}, M_{(k+1)_j}\right),\ r=1..n_k,\ j=1..n_{(k+1)} \qquad (3)$$

На основе этих правил в главе описан разработанный **Метод поиска оптимального расписания для группы вершин $M(k+1)$:**

а) Составить множество P всех возможных расписаний выполнения операций группы вершин $M_{(k+1)}$.

б) В каждом расписании P_r, $r=1..n_p$ найти максимальный ранний срок выполнения операции:

$$P_{rmax}=\max(\tau'_{rj}),\ r=1..n_p,\ j=1..n. \qquad (4)$$

в) Среди всех P_{rmax} найти минимальное значение. Расписание, которому будет соответствовать это значение, будет оптимальным расписанием.

Формализовано это можно описать так:

Правило 3.

$$P_{min}=P_s:\ P_{smax}=\min(\max(\tau'_{rj}))$$
$$r=1..n_p,\ j=1..n,\ n_p=n(n!),\ n=\max(n_k, n_{(k+1)}) \qquad (5)$$

г) Расписаний, удовлетворяющих условию (4) может быть несколько. В этом случае из них необходимо выбрать то расписание, суммарное значение ранних сроков выполнения которого наименьшее.

Формализовано это можно описать так:

Правило 4.

$$P_{min}=P_s:\ P_{smax}=\min(\max(\tau'_{rj})),\ \sum_{k=1}^{n}\tau'_{sj} = \min_s\left(\sum_{j=1}^{n}\tau'_{rj}\right)$$
$$r=1..n_p,\ j=1..n,\ n_p=n(n!),\ n=\max(n_k, n_{(k+1)}) \qquad (6)$$

Перебирать все $n(n!)$ сочетаний – это очень трудоемкий процесс. С учетом того, что необходимо найти расписание, которое выполняется за минимальное время, то значительно сократить процесс перебора расписаний можно в соответствии со следующим методом.

Оптимизированный метод перебора расписаний

Обозначения. Пусть R – номер процесса, с которого начинается составление расписания, r – номер текущего процесса, j – номер операции в группе вершин $M_{(k+1)}$, P – множество всех выбранных расписаний, s – номер текущего расписания, p_{sr} –операция, выполняемая на r-м процессе по расписанию с номером s, τ'_{sr} – ранний срок выполнения p_{sr} -й операции на r-м процессе по расписанию с номером s.

1. Пусть $R=1$, $s=1$.
2. Пусть $r=1$, $p_{sr}=0$ для всех $r=1..n_{(k+1)}$.
3. Найти минимальный ранний срок во множестве τ'_r: $\tau'_{rj}=\min(\tau'_{rl})$, $l=1..n_{(k+1)}/\{p_{sr}\}$, $r=1..n_{(k+1)}$. Положить $p_{sr}=j$, $\tau'_{sr}=\tau'_{rj}$.
4. Если $r<n_k$, то положить $r=r+1$ и шаг 3. Если на шаге 3 было найдено несколько минимальных ранних сроков, то шаг 4 выполняется для каждого из них. Если $r=n_k$, то шаг 5.
5. Если $R<n_k$, то положить $R=R+1$ и шаг 2. Иначе шаг 6.
6. Конец метода.

В результате трудоемкость всего алгоритма по поиску возможных расписаний составит $O(n_k n_{(k+1)})= O(n^2)$, где $n = \max(n_k)$, $k=1..m$, m – число групп вершин (ярусов).

Суть метода оптимизации алгоритма по времени выполнения с учетом схемы передачи данных между процессами с учетом времени выполнения каждой операции заключается в перестановке вершин графа внутри яруса и между ярусами таким образом, чтобы минимизировать общее время выполнения алгоритма.

В методах, рассмотренных в главе 2, все операции имели одинаковое время выполнения и поэтому вершины графа переставлялись только внутри яруса. В данных методах время выполнения операций разное и это сразу в несколько раз усложняет методы. С одной стороны – это позволяет сократить общее время работы алгоритма не только за счет ликвидации простоев процессов, но и за счет более плотного распределения операций, с другой стороны необходимо отслеживать не только прямые, но и транзитивные связи.

Еще одним отличием методов оптимизации алгоритма с учетом времени выполнения операций является важность направления перебора вершин графа.

Метод с перебором вершин графа с начала алгоритма в ширину

1. Процесс перестановки вершин графа начинается с первой группы. Номер текущей группы вершин $k=1$. Все процессы в группе вершин полагаются не отмеченными.
2. Для M_k найти множество ранних сроков выполнения операций из группы вершин $M_{(k+1)}$ с учетом передачи данных по правилам (1, 2).
3. Найти оптимальное расписание выполнения операций группы вершин $M_{(k+1)}$. Для поиска оптимального расписания необходимо:
   - Составить множество P всех возможных расписаний выполнения операций группы вершин $M_{(k+1)}$.
   - В каждом расписании P_r, $r=1..n_p$ найти максимальный ранний срок выполнения операции:
     - $P_{rmax}=\max(\tau'_{rj})$, $r=1..n_p$, $j=1..n$.
     - Среди всех P_{rmax} найти минимальное значение.

   Расписание, которому будет соответствовать это значение, будет оптимальным расписанием.
4. Выбрать то расписание, суммарное значение ранних сроков выполнения которого наименьшее по правилу 3.
5. Переставить вершины графа в соответствии с найденным расписанием.
6. Если $k<m$, то $k=k+1$ и шаг 2. Иначе шаг 6.
7. Конец метода.

Прямое применение методов оптимизации алгоритма с конца невозможно при условии учета времени выполнения операций. Но применять тем не менее эти методы можно и очень успешно, если внести некоторую поправку.

Модификация метода оптимизации алгоритма с конца в ширину

Обозначения: Пусть m – общее число групп вершин, полученное формализованным способом, например, методом оптимизации информационного графа по ширине с помощью матрицы или списка смежности, k –номер текущей группы вершин, n_k – число вершин графа в группе вершин с номером k, i – номер текущей вершины графа в группе вершин с номером k, j – номер текущей вершины графа в группе с номером k-1, $p0_i$ – время простоев в i-й группе вершин начиная с момента t_0 и до начала выполнения первой операции.

1. Процесс перестановки вершин графа начинается с последней группы. Номер очередной группы вершин $k=m$.

В первую очередь необходимо в расписание поставить вершины графа, которые находятся в смежных группах и имеют бинарную связь, и только потом

расставлять вершины графа с множественными связями. Это поможет избежать лишних простоев вычислительных узлов.

2. Для M_k найти множество ранних сроков выполнения операций из группы вершин $M_{(k-1)}$ с учетом передачи данных по формулам (1, 2).

3. В соответствии с методом поиска оптимального расписания для группы вершин $M_{(k-1)}$ найти оптимальное расписание.

4. Переставить вершины графа в соответствии с найденным расписанием.

5. Если $k>1$, то $k=k-1$ и шаг 2. Иначе шаг 6.

6. Рассчитать время простоев в каждой группе вершин начиная со времени $t0$ и до момента начала выполнения первой операции в этой группе вершин: $p0_i$.

7. Найти минимальное время среди всех начальных простоев: $p_{min}=\min(p0_i)$, $i=1..m$.

8. Уменьшить время начала каждой операции в алгоритме на величину p_{min}.

9. Перестроить граф с учетом закрепления операций за вычислительными узлами.

10. Конец метода.

Модификация заключается в дополнении метода четырьмя шагами 6-9. Эти шаги равносильны сдвигу временной диаграммы на максимально возможное время влево.

В четвёртой главе описана методика оптимизации алгоритма в случае наличия дополнительных (кроме времени и схемы взаимосвязи операций по данным). Все эти параметры можно разделить на два класса: параметры алгоритма и параметры вычислительной среды. Все параметры алгоритма могут быть сведены к одному суммарному параметру. Чаще всего этим суммарным параметром является параметр «время». В этом случае возможно применение разработанных методов оптимизации алгоритмов без каких-либо модификаций. При наличии параметров вычислительной среды необходимо применять методы оптимизации алгоритмов по методике, разработанной для распределенной вычислительной среды, описанной в этой же главе. Таким образом методика дополнения информационного графа новыми параметрами алгоритма может представлена в виде схемы с рис. 2.

Далее в главе рассмотрены особенности распределённых вычислений, которые необходимо учесть при применении методов оптимизации алгоритмов с учетом схемы передачи данных между процессами.

Основной отличительной чертой разработанных для параллельных вычислений методов было условие, что все вычислительные узлы работают с одинаковой мощностью и поэтому время срабатывания одной и той же операции на разных узлах предполагалось одинаковым. В распределенных вычислениях часто это далеко не так. И поэтому для алгоритмов в распределенных вычислениях добавляется сразу ряд дополнительных параметров. Эти параметры могут быть разными в зависимости от архитектуры сети. Неважно сколько и какие это будут параметры. Суть проблемы в том, что их будет больше одного параметра, и они будут числовыми.

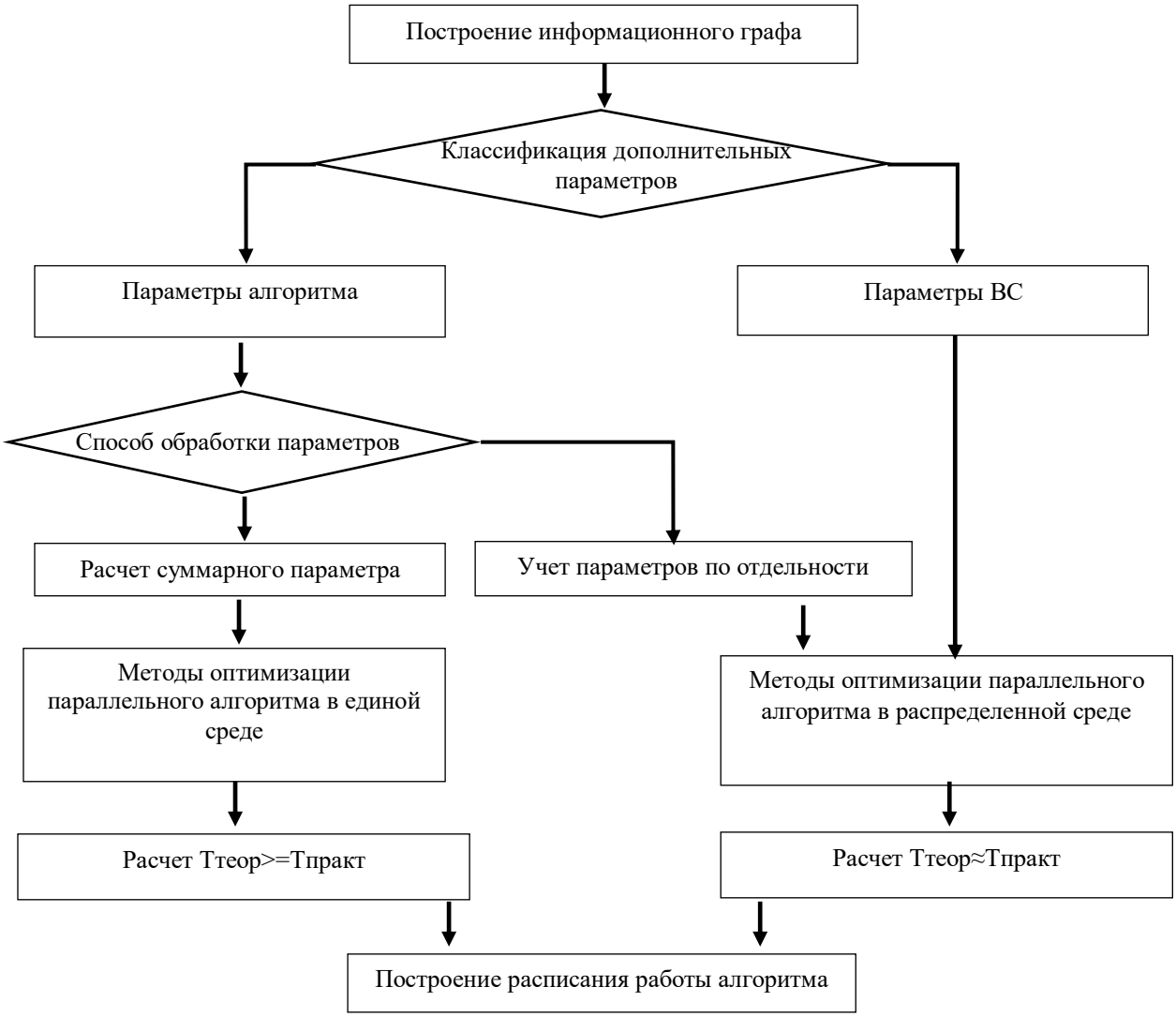

Рисунок 2 - Методика оптимизации алгоритма с дополнительными параметрами

Для построения расписания выполнения алгоритма по заданному информационному графу необходимо построить 2 графа:

1. Граф взаимосвязей распределённой системы вычислений. Этот граф является неориентированным взвешенным. Пример такого графа приведен на рис.3, где Yi – узлы, $Pi=\{p_i, l_i, d_i\}$, p_i – производительность узла, l_i – пропускная способность, d_i – длина пути

2. Информационный граф алгоритма в параллельной форме, по которой можно однозначно воспроизвести расписание выполнения алгоритма.

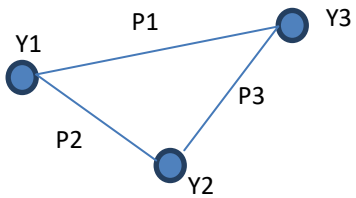

Рисунок 3 - Пример взвешенного графа системы распределенных вычислений, состоящей из трех узлов.

Вся сложность этого подхода заключается в том, что, для построения второго графа (информационного графа алгоритма в параллельной форме) необходимо произвести многократную проекцию начального информационного графа алгоритма на граф взаимосвязей.

Пример проекции информационного графа на граф взаимосвязей узлов (рис.4)

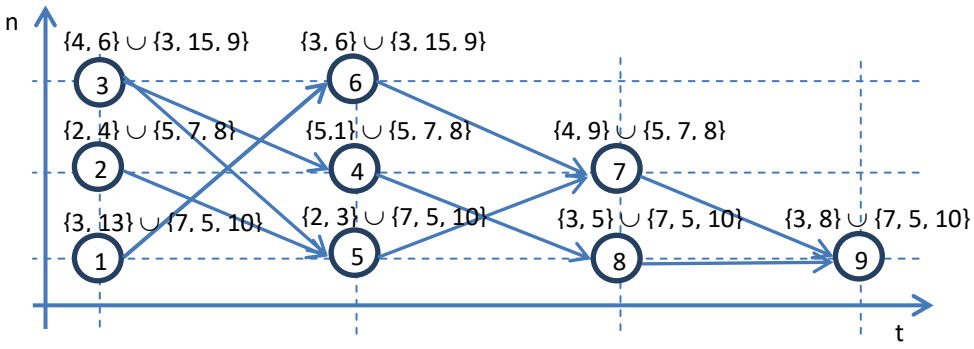

Рисунок 4 - Пример проекции информационного графа алгоритма на граф взаимосвязей узлов

В главе приведены формулы расчета общего числа вершин графа проекций.

Пусть на *j*-м ярусе *nj* вершин, граф взаимосвязей состоит из *ny* узлов.

Вариант 1. Пусть *nj*< *ny*. Тогда на каждом ярусе возможно $A_{ny}^{nj} = \frac{ny!}{(ny-nj)!}$ различных комбинаций из узлов и операций. Тогда общее количество проекций информационного графа на граф взаимосвязей будет составлять:

$$Gp = \prod_{j=1}^{m} A_{ny}^{nj} \qquad (7)$$

где *m* – число ярусов.

Вариант 2. Пусть *nj*=*ny*. Тогда на каждом ярусе возможно $P_{ny} = ny!$ различных комбинаций из узлов и операций. Тогда общее количество проекций информационного графа на граф взаимосвязей будет составлять:

$$Gp = \prod_{j=1}^{m} ny! = (ny!)^m \qquad (8)$$

где *m* – число ярусов.

Вариант 3. Пусть *nj*>*ny*. Тогда необходимо применить метод оптимизации параллельного алгоритма с учетом ограничений на вычислительные ресурсы и привести параллельную форму алгоритма к виду для случая *nj*=*ny*.

Анализ большого числа проекций простым перебором даже с помощью вычислительной техники займет очень много времени.

В главе представлено несколько вариантов решений этой проблемы.

1. Сокращение числа проекций путем уменьшения для каждой вершины графа списка возможных узлов.
2. Предварительная обработка информационного графа без учета времени выполнения операций, но с учетом передачи данных между процессами. Это позволит сократить число передач данных и убрать из рассмотрения все проекции, на которых данные не передаются от узла к узлу.
3. Разработка метода оптимизации расписания поярусно. Подбор узлов на первых двух ярусах. А далее выбор узлов с учетом тех, что уже зафиксированы для вершин предшествующих ярусов.

Последний метод будет более эффективным по трудоемкости по сравнению с методом перебора, но его недостатком является также отсутствие гарантии, что найденное расписание будет глобальным оптимумом.

Пусть дан информационный граф алгоритма G (u, v, s), где u – совокупность вершин графа, соответствующих операциям алгоритма, v – совокупность ребер графа, соответствующих связям между операциями по данным, s – совокупность весов, соответствующих вершинам графа (параметров операциям алгоритма). При этом s может содержать значения не только одного параметра, а сразу нескольких. Например, $s=T=\{t_i\}$, $i=1..n$, где t_i – время выполнения i-й операции, n – число вершин графа. Другим примером s является $s=\{T, A\}=\{(t_i, a_i)\}$, $i=1..n$, где t_i – время выполнения i-й операции, a_i – объем данных, которые являются результатом операции, n – число вершин графа.

Поярусный метод построения расписания распределенного алгоритма

Пусть дан граф взаимосвязей узлов вычислительной системы $G'(u',v',s')$, где u' – совокупность вершин графа, соответствующих узлам, v' – совокупность ребер графа, соответствующих связям между узлами, s' – совокупность весов, соответствующих вершинам графа (параметров сети), j – номер яруса.

1. Положить $j = 1$.
2. Построить все возможные графы проекций для двух ярусов j и $(j + 1)$.
3. Для каждого графа применить метод оптимизации параллельного алгоритма с учетом передач данными между процессами и времени выполнения операции. В данном случае время выполнения будет рассчитываться отдельно для каждой вершин графа исходя из времени выполнения операции с учетом производительности узла и времени на передачу данных по сети при необходимости.
4. Из всех полученных на 3 шаге расписаний выбрать то, у которого время выполнения алгоритма по двум рассматриваемым ярусам будет минимальным.
5. Зафиксировать для $(j + 1)$-го яруса операции и узлы в соответствии с выбранным оптимальным расписанием на шаге 4.
6. Если $j < m$, то положить $j = j + 1$ и перейти к шагу 2. Иначе шаг 7.
7. Конец метода.

Одной из областей применения разработанных методов являются запросы к реляционным базам данных.

Исследования показали, что все методы оптимизации информационного графа по различным параметрам (времени, вычислительным узлам, объему передач данными между процессами и т.д.), разработанные для классических алгоритмов, полностью подходят для запросов.

Но, при всех положительных качествах и возможностях этих методов, запросы к реляционным базам данных имеют свою специфику и поэтому непосредственное применение перечисленных методов в лучшем случае даст некоторое ускорение, а в худшем случае не даст никакого эффекта. В первую очередь это происходит из-за того, что входными данными для запросов являются целые таблицы, и они содержат очень большой объем данных. Во-вторых, информационный граф дает информацию только о параллелизме по задачам, а запросы, которые бы содержали очень много независимых подзапросов, на практике встречаются редко. Поэтому часто по информационному графу в первом приближении запас внутреннего параллелизма равен нулю.

Если же использовать параллелизм по данным и делить каждую таблицу на части, то изменится информационный граф (рис.5):

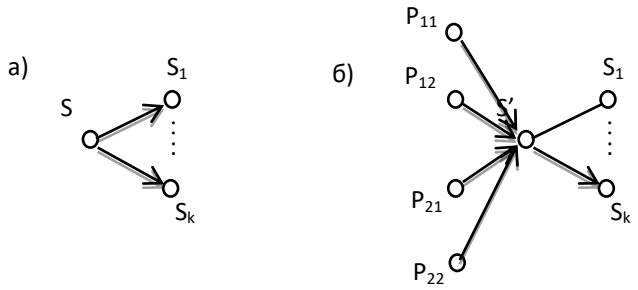

Рисунок 5 – Начальный и модифицированный граф запроса

При этом число вершин графа, на которые увеличивается исходный граф, не пропорционально числу делимых таблиц и числу частей, на которые они делятся. Пусть k – число делимых таблиц, t – число ярусов в информационном графе запроса, r – текущий ярус графа, nr – число запросов на ярусе r, m_r – число запросов варианта 1, k_i – число частей в i-й таблице, k_g – число вершин графа, добавляемых к исходному графу. Тогда:

$$k_g = \sum_{r=1}^{t}\left(\sum_{i=1}^{m_r} k_{ri} + \sum_{i=m_r+1}^{n_r}\sum_{j=m_r+1}^{n_r} k_{ri}k_{rj}\right) \quad (9)$$

С ростом сложности начального графа число вершин в модифицированном графе с учетом разделенных на части таблиц очень сильно возрастает. Вручную построить такой граф очень трудоемко, а проанализировать его на предмет построения оптимального параллельного плана выполнения запроса еще сложнее.

В этом случае на помощь приходят методы параллельных вычислений, разработанные в диссертации и описанные в главах 2 и 3.

Так запросу:

```
Select field1, field2, field3, (field1*2 as field4)
from table1, (select *
                from table2, table4
                where (field5<value1))
where field6 in (select (field7+2) as field8
                   from table2
                   where field8='string1'
      union
select field9, field10
from table2, table3
where (field9=value2) and (field11> value3))
      union
select *
from table2;
```

по методу построения расписания алгоритма на основе информационного графа будет изначально соответствовать временная диаграмма выполнения подзапросов (рис.6):

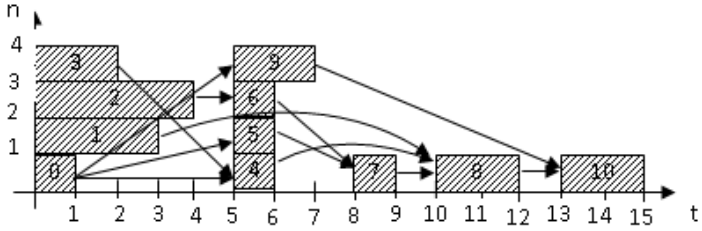

Рисунок 6 – Расписание выполнения запроса

где n – это число узлов, t- это время выполнения в условных единицах.

Время выполнения данного запроса в соответствии с диаграммой составит 15 условных единиц на 4-х вычислительных узлах.

После применения метода оптимизации алгоритма с учетом схемы обмена данными будет получено новое расписание, временная диаграмма которого представлена на рис.7:

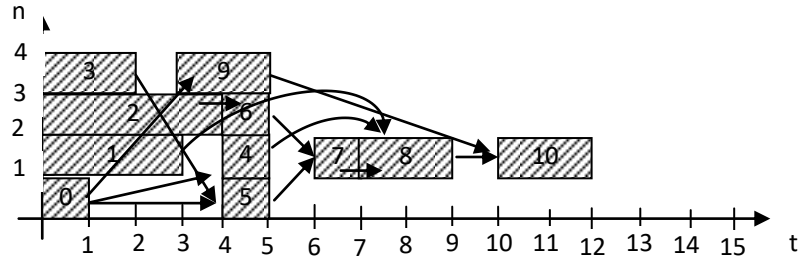

Рисунок 7 – Расписание выполнения запроса после оптимизации алгоритма по времени выполнения с учетом схемы обмена данными

Время при выполнении алгоритма по новому расписанию сократиться до 12 ед. Если под условными единицами понимаются часы, то это очень большая разница. Проведенное тестирование позволило удостовериться, что в новом расписании запрос работает более эффективно.

По результатам исследования возможности применения разработанных методов оптимизации параллельного алгоритма по времени выполнения на основе информационных зависимостей и схемы межпроцессного взаимодействия к запросам в базах данных была сформулирована следующая методика:

1. Провести эмпирический анализ времени выполнения подзапросов исследуемого запроса.
2. Построить информационный граф запроса.
3. Разбить большие отношения на *k* частей.
4. Изменить информационный граф с учетом декомпозиции отношений.
5. Применить методы оптимизации алгоритмов к полученному информационному графу.

Тестирование запросов проводилось с помощью специально разработанного программного обеспечения, которое также описано в главе 4.

В заключении сформулированы выводы и обобщены основные результаты проведённого исследования.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ

1. Методы оптимизации параллельных алгоритмов с учетом обмена данными между процессами, с учетом и без учёта времени выполнения операций.
2. Методика применения разработанных методов при расширенном наборе параметров исследуемых параллельных алгоритмов, таких как время передачи по сети, объем данных и других.
3. Формула, определяющая минимальное время выполнения алгоритма; формулы для расчета числа проекций информационного графа на граф взаимосвязей узлов; формула расчета числа узлов информационного графа запроса к базе данных.
4. Метод построения расписания выполнения алгоритма в распределенной вычислительной среде. Метод может быть применен в комплексе с другими методами оптимизации параллельных алгоритмов.
5. Методика применения разработанных методов к запросам в реляционных базах данных.

СПИСОК ОСНОВНЫХ ПУБЛИКАЦИЙ ПО ТЕМЕ ДИССЕРТАЦИИ

Публикации в изданиях, индексируемые в базах данных «Web of Science» и «Scopus»

1. Shichkina U. A., Al-Mardi M. H., Kupriyanov M. C. Optimization algorithm for an information graph for an amount of communications // Lecture Notes in Computer Science. 2016. Т. 9870 LNCS. С. 50-62.
2. Shichkina, Y., Al-Mardi M.H., Storublevtcev, N., Degtyarev, A. The construction of the parallel algorithm execution schedule taking into account the interprocessor data transfer// Lecture Notes in Computer Science. 2018. Т. 10963 LNCS. С. 61-77.

Публикации в изданиях, рекомендованных ВАК России:

3. Шичкина Ю. А., Аль-Марди М.Х. Метод оптимизации параллельного алгоритма за счет уменьшения объема межпроцессорной передачи информации // Известия СПбГЭТУ «ЛЭТИ». -2015. - № 10. - С. 15-28.
4. Шичкина Ю.А., Аль-Марди М.Х., М.С.Куприянов Оптимизация параллельного алгоритма путем сокращения времени на межпроцессорный обмен данными // Известия СПбГЭТУ «ЛЭТИ». -2017. - № 7. - С. 51-56.
5. Аль-Марди М.Х., Сравнительный анализ методов оптимизации параллельного алгоритма с учётом и без учёта времени выполнения операций // Компьютерные инструменты в образовании. -2018. – № 3 – С. 36-48.
6. Аль-Марди М.Х., Особенности распределённых вычислений, учитываемые в методах оптимизации алгоритмов по объему межпроцессорных передач // Компьютерные инструменты в образовании. -2018. – № 2 – С. 31-38.

Другие статьи и материалы конференций:

7. Аль-Марди М.Х., Шичкина Ю. А. Метод оптимизации параллельного алгоритма по объёму межпроцессорных коммуникаций на взвешенных графах // научный журнал "Chronos". -2016. - С. 46-52.
8. Аль-Марди М.Х., Шичкина Ю. А. Учёт межпроцессорных передач данных при распараллеливании алгоритмов // национальная ассоциация ученых (НАУ). -2016. - № 1(17). - С. 43-45.

Свидетельство о регистрации программы для ЭВМ

9. Шичкина Ю.А., Аль-Марди М.Х.А. Multithreaded execution of SQL-queries. ПрЭВМ, Свидетельство № 2018619271 дата гос. рег. 02.08.18