Московский государственный университет
им. М.В. Ломоносова

На правах рукописи

Сальников Алексей Николаевич

СИСТЕМА РАЗРАБОТКИ И ПОДДЕРЖКИ ИСПОЛНЕНИЯ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

Специальность 05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат диссертации на соискание учёной степени
кандидата физико-математических наук

Москва – 2007

Работа выполнена на кафедре автоматизации систем вычислительных комплексов факультета вычислительной математики и кибернетики Московского государственного университета им. М.В. Ломоносова.

Научные руководители: член-корреспондент РАН, профессор
Королёв Лев Николаевич;

к.ф.-м.н., доцент
Попова Нина Николаевна.

Официальные оппоненты: д.ф. - м.н. Крюков Виктор Алексеевич,
Институт прикладной математики РАН;

к. ф. -м.н. Антонов Александр Сергеевич,
Научно-исследовательский
вычислительный центр МГУ
им. М.В. Ломоносова

Ведущая организация: Институт системного программирования РАН

Защита диссертации состоится «16» февраля 2007 г. в 11.00 часов на заседании диссертационного совета Д 501.001.44 в Московском государственном университете им. М.В. Ломоносова по адресу: 119992 ГСП-2 Москва, Ленинские горы, МГУ им. М.В. Ломоносова, 2-й учебный корпус, факультет ВМиК, аудитория 685.

С диссертацией можно ознакомиться в библиотеке факультета ВМиК МГУ. С текстом автореферата можно ознакомиться на официальном сайте ВМиК МГУ им. М.В. Ломоносова http://www.cmc.msu.ru в разделе «Наука» – «Работа диссертационных советов» – «Д 501.001.44».

Автореферат разослан «__» января 2007 г.

Учёный секретарь
диссертационного совета

профессор Н.П. Трифонов

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы

Современные многопроцессорные вычислительные системы параллельной обработки весьма разнообразны в своих архитектурных особенностях. В связи с этим возникает необходимость в программах, которые умеют подстраиваться под различные архитектурные особенности этих машин. Для каждой архитектуры многопроцессорной системы существует своя специфика написания эффективных параллельных программ. Программы, эффективные для одной архитектуры, могут быть неэффективны для другой. Предложенный в работе подход преследует цель снижения потерь в эффективности при переносе программы с одной архитектуры на другую.

Наиболее популярный способ написания параллельных программ – это создание параллельного программного кода с использованием таких библиотек, как MPI для кластеров и систем с распределённой памятью, а также OpenMP для SMP систем. Эти библиотеки позволяют создавать параллельные программы, которые способны выполняться на многопроцессорных системах с различными архитектурами. Однако добиться переносимости параллельной программы, с точки зрения сохранения эффективности, значительно сложнее. Для этого требуются некоторые навыки в «искусстве программирования» и трудоёмкий предварительный анализ исходного алгоритма для выявления возможностей по распараллеливанию на определённой архитектуре.

Всё сказанное выше и сложность учёта всех особенностей многопроцессорной системы оправдывает создание высокоуровневых средств параллельного программирования, базирующихся на MPI, OpenMP, pthread и облегчающих создание параллельных программ, с учётом особенностей многопроцессорных систем. К таким средствам можно отнести DVM, Cilk, PETSc, mpC и предлагаемый в данной работе «PARUS».

Цель работы

Целью диссертационной работы является создание среды программирования для разработки и исполнения параллельных программ в гетерогенной среде. Создаваемая среда программирования должна отличаться от других высокоуровневых сред программирования ориентированностью на представление программы как графа зависимостей по данным и не требовать от пользователя знания архитектуры вычислительной системы. Для достижения основной цели диссертационной работы сформулируем несколько подцелей:

1. Разработать алгоритмы планирования вычислений для гетерогенных многопроцессорных систем, учитывающие нелинейность задержек от размера сообщения при передаче данных в многопроцессорной системе. Эти алгоритмы должны работать в статическом и динамическом режимах, а также в режиме с учётом подсказок пользователя.

2. Разработать алгоритмы тестирования коммуникационной среды многопроцессорной системы, определяющие задержки при передаче сообщений и имитирующие при

помощи фоновых шумовых сообщений наличие задач других пользователей.

3. Разработать алгоритмы тестирования, учитывающие гетерогенность коммуникаций, включая физическую структуру коммуникационной среды, но на этапе планирования вычислений позволяющие учитывать только результаты тестирования, с тем, чтобы достичь независимости этапа планирования от архитектурных особенностей многопроцессорной системы.

4. Создать анализатор зависимостей по данным в исходном коде на языке программирования C, адаптированный к нуждам системы, и разработать способ описания алгоритма решения задачи как набора действий, взаимодействующих между собой через передачу сообщений.

Научная новизна

Предложен новый метод построения параллельных программ и язык их описания как граф-схемы потока данных, позволяющие абстрагироваться от конкретной технологии передачи сообщений.

Разработаны и реализованы новые алгоритмы управления процессом исполнения параллельной программы с учётом структуры программы, динамики обменов данными и текущего состояния многопроцессорной системы.

Практическая ценность

На основе предложенных метода, языка и алгоритмов создана система поддержки этапов разработки и исполнения параллельных программ.

Практическая применимость и эффективность созданной системы разработки и исполнения параллельных программ «PARUS» показана на примере решения ряда модельных и актуальных практических задач, в том числе задач биоинформатики, на современных параллельных вычислительных системах. (IBM pSeries 690, МВС-1000м и др.)

Система «PARUS», построенная на основе технологии MPI, оформлена как проект с открытым исходным кодом и доступна в Internet (**http://parus.sf.net**).

Методы исследования

В диссертации применялись методы анализа и представления программ как графа зависимостей по данным. Применялись генетические и списочные алгоритмы планирования вычислений на многопроцессорной системе, методы кластеризации и динамического программирования, частотные методы фильтрации сигналов.

Апробация работы и публикации

По теме диссертации опубликовано 12 печатных работ. Результаты диссертации докладывались на объединённом научно-исследовательском семинаре кафедр

Автоматизации систем вычислительных комплексов, Алгоритмических языков и Системного программирования ВМиК МГУ под руководством проф. Шура-Бура М.Р., на семинарах факультета ВМиК и научных конференциях:

1. Международный семинар «Супервычисления и математическое моделирование», Российский федеральный ядерный центр – Всероссийской НИИ экспериментальной физики, Саров, 17-21 июня 2002.

2. Всероссийская научная конференция «Научный сервис в сети Интернет», Новороссийск, Россия, 2004.

3. The Fourth International Conference on Bioinformatics of Genome Regulation and Structure (BGRS'2004). Novosibirsk, Russia, July 25-30, 2004.

4. Международная научная конференция студентов, аспирантов и молодых ученых «Ломоносов – 2004», Москва, Россия.

5. Всероссийская научная конференция «Методы и средства обработки информации», Москва, Россия, 2005.

6. Всероссийская научная конференция «Научный сервис в сети Интернет», г. Новороссийск, 19-24 сентября 2005.

7. 13th European PVMMPI Users' Group Meeting (Euro PVM/MPI 2006). Bonn, Germany, September 17-20, 2006.

Структура и объём работы

Диссертация состоит из семи глав, где первая глава – введение, списка литературы и четырёх приложений. Объём диссертации – 90 страниц. Список цитируемой литературы включает 81 наименование.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В **первой главе** обосновывается актуальность темы и определяется направление исследования, а также определяется основная терминология.

Обосновывается необходимость разработки высокоуровневых средств для создания параллельных программ, кратко обсуждается проблема оптимизации последовательной части параллельной программы, проблема тестирования производительности процессоров многопроцессорной системы, тестирование производительности внутренней коммуникационной среды многопроцессорной системы.

Вторая глава посвящена обзору некоторых известных высокоуровневых средств параллельного программирования. Здесь обсуждаются: DVM-система**,** созданная в Институте прикладной математики им. М.В. Келдыша РАН, T-система, основывающаяся на парадигме функционального программирования для обеспечения динамического

распараллеливания программ и язык программирования mpC, созданный в институте системного программирования РАН.

В той же главе рассказывается о преимуществах подхода, изложенного в диссертации. В DVM исходный код программы дополняется специальными конструкциями, которые оформлены невидимым образом для компилятора с «чистого» языка программирования, следовательно, предполагается, что для решения соответствующей задачи должна быть создана последовательная программа. В «PARUS» не требуется реализация последовательной программы. Рассмотренные выше системы учитывают гетерогенность с точки зрения производительности процессоров, однако производительность коммуникаций учитывается в недостаточной степени. В разделе диссертации (глава 6), посвящённом тестированию коммуникационной среды, показано, насколько сложно могут себя вести коммуникации в зависимости от варьирования некоторых параметров при передаче данных. Рассмотренные выше системы не обладают рядом полезных средств, имеющихся в системе «PARUS», по тестированию коммуникационной среды многопроцессорной системы.

На основе результатов тестирования, проведённого с использованием предложенных в «PARUS» средств тестирования, осуществляется балансировка нагрузки на отдельные компоненты многопроцессорной системы с учётом нелинейности скорости передачи данных по коммуникациям от объёма передаваемых данных.

В «PARUS» присутствует анализатор зависимостей по данным. Этот анализатор можно считать прототипом средства для автоматического распараллеливания С-программы. В Т-системе допускается наличие только так называемых «чистых функций», то есть функций без побочных эффектов. В отличие от Т-системы, где граф строится неявно, в «PARUS» граф строится явно. Достоинство заключается в том, что разработчик параллельной программы может определять степень параллелизма в программе; определять, насколько вычислительно сложной делать отдельную вершину.

В mpC для задания гетерогенности нужно явно указывать долю вычислений, которая будет помещена на процессор. В «PARUS» производительность процессоров определяется на этапе тестирования многопроцессорной системы, и доля вычислений определяется автоматически в процессе назначения заданий процессорам.

Третья глава посвящена обзору алгоритмов планирования вычислений на многопроцессорных системах. Здесь приводится постановка задачи планирования вычислений. Рассматриваются списочные алгоритмы; алгоритм, основанный на множестве очередей; алгоритм имитации отжига; генетический алгоритм; алгоритм поиска критического пути; алгоритм обратного заполнения; алгоритм управления группами работ с прерываниями. Анализ стратегий планирования показал, что в «PARUS» целесообразно использовать списочные алгоритмы и генетический алгоритм. В текущей реализации в «PARUS» представлено 3 алгоритма планирования вычислений. Все три алгоритма реализованы таким образом, чтобы учитывать гетерогенность многопроцессорной системы как с точки зрения производительности процессоров, так и с точки зрения коммуникаций.

Первый алгоритм статически строит расписание исполнения внутренних работ программы на многопроцессорной системе и построен на основе генетического алгоритма.

Основное отличие реализованного генетического алгоритма от других реализаций генетических алгоритмов заключается в задании функции качества, которая более тщательно определяет, сколько времени атомарная работа параллельной программы будет исполняться процессором с учётом времени, затрачиваемого на передачу сообщений между процессорами.

Второй алгоритм динамически, уже в процессе исполнения параллельной программы, распределяет работы по MPI-процессам. Данный алгоритм можно отнести к классу списочных алгоритмов. Он является совмещением RR алгоритма с алгоритмом назначения кратчайшей работы вперёд. Основное преимущество такого алгоритма заключается в скорости его срабатывания и низкой вычислительной сложности, что очень важно для назначения работ на процессоры в динамике. Идея этого алгоритма близка к идее алгоритма Backfill. Также как и Backfill, он стремится полностью заполнить многопроцессорную систему работами.

Третий алгоритм, так же как и второй, работает в динамическом режиме. Однако, в отличие от предыдущего алгоритма, данный алгоритм способен учитывать подсказки пользователя по назначению работ на процессоры.

В **четвёртой главе** обсуждаются принципы организации системы «PARUS». На вход системе «PARUS» поступает либо ориентированный ациклический граф, либо C-программа, которая затем всё равно преобразуется в ориентированный граф. Алгоритм решаемой задачи представляется в виде ориентированного графа, где в вершинах сосредоточены вычислительные операции (действия над данными), а рёбра задают зависимость по данным. Таким образом, программа описывается как сеть из вершин-истоков, в которых чаще всего происходит чтение входных данных из файлов, внутренних вершин, где происходит обработка данных, а также вершин-стоков, в которых обычно происходит запись результатов в файлы. С точки зрения эффективности выполнения построенный таким образом граф должен обладать некоторыми особенностями. Вершины графа должны являться полновесными, «тяжёлыми» в вычислительном смысле фрагментами кода. Граф выстраивается по уровням. Номер уровня вершины в графе – максимальная длина пути от вершин истоков до текущей вершины. Вершины графа на каждом уровне независимы между собой и могут быть исполнены параллельно. Таким образом, определённый выше граф задаёт граф-программу.

Описание графа содержится в текстовых файлах, которые можно подать на вход набору программных компонент, осуществляющих преобразование граф-программы в исходный код на C++ с вызовами MPI-функций. Код на C++ с MPI-вызовами компилируется совместно с кодом управляющего MPI-процесса. Управляющий процесс осуществляет непосредственное управление назначениями вершин графа на MPI-процессы, а также осуществляет контроль передаваемых по рёбрам данных во время исполнения получившейся параллельной MPI-программы на многопроцессорной системе.

Каждой вершине и ребру графа приписаны веса. Вес вершины характеризует вычислительную сложность вершины, выраженную как отношение объёма вычислений к эталонным операциям. Под эталонной операцией можно понимать, например, время решения эталонной системы линейных алгебраических уравнений. Вес ребра характеризует

объём данных, которые нужно передать по данному ребру от одной вершины графа к другой. Эта информация используется при назначении вершины на процессор.

В той же главе описана разработанная система тестирования многопроцессорной системы. Создана система тестов, помогающая прогнозировать поведение коммуникационной среды многопроцессорной системы относительно размера передаваемого сообщения, номера процессора и уровня «шума». Создан тест для определения производительности процессоров многопроцессорной системы, построенный на основе измерения времени, затрачиваемого на перемножение матриц эталонного размера.

Там же описано разработанное графическое средство для визуального представления граф-программы, расписания исполнения вершин граф-программы и их редактирования.

Создано специальное приложение, которое анализирует зависимости по данным в С-программе и строит по ним граф зависимости по данным в формате, понимаемом системой «PARUS». Данное программное средство можно считать прототипом автораспараллеливающего компилятора.

Компоненты, отвечающие за создание параллельной программы, и их взаимосвязь показаны на рисунке 1.

После компиляции C++ кода с вызовами MPI-функций граф-программа становится MPI-программой. MPI-программа строится как набор MPI-процессов, взаимодействующих между собой через передачу сообщений. Обычно каждый MPI-процесс, составляющий MPI-программу, привязывается к своему процессору многопроцессорной системы. MPI-процесс может быть также назначен на свой узел в кластере. Под узлом кластера будем понимать отдельный компьютер, возможно, состоящий из нескольких микропроцессоров, который работает под управлением своей собственной операционной системы и который объединён сетью с другими узлами кластера. Среди всех MPI-процессов выделяется один, который в дальнейшем будет выполнять роль координатора в граф-программе. Остальные MPI-процессы исполняют код, приписанный вершинам графа.

В MPI-программе вершины граф-программы становятся функциями, в которых могут быть описаны свои переменные и которые могут использовать описанные глобальные переменные. Каждый MPI-процесс хранит свою собственную копию глобальной переменной. Вершины могут менять переменные и передавать изменённые значения другим вершинам граф-программы по ребрам графа.

Рёбра, по существу, содержат информацию о синхронизируемых ребром значениях переменных, где переменные описаны либо глобально, либо в связанных ребром графа вершинах. С ребром связана весовая метка, характеризующая объём передаваемой по ребру информации в байтах. Для каждого ребра указывается посылающая вершина — источник данных и принимающая вершина — приёмник данных.

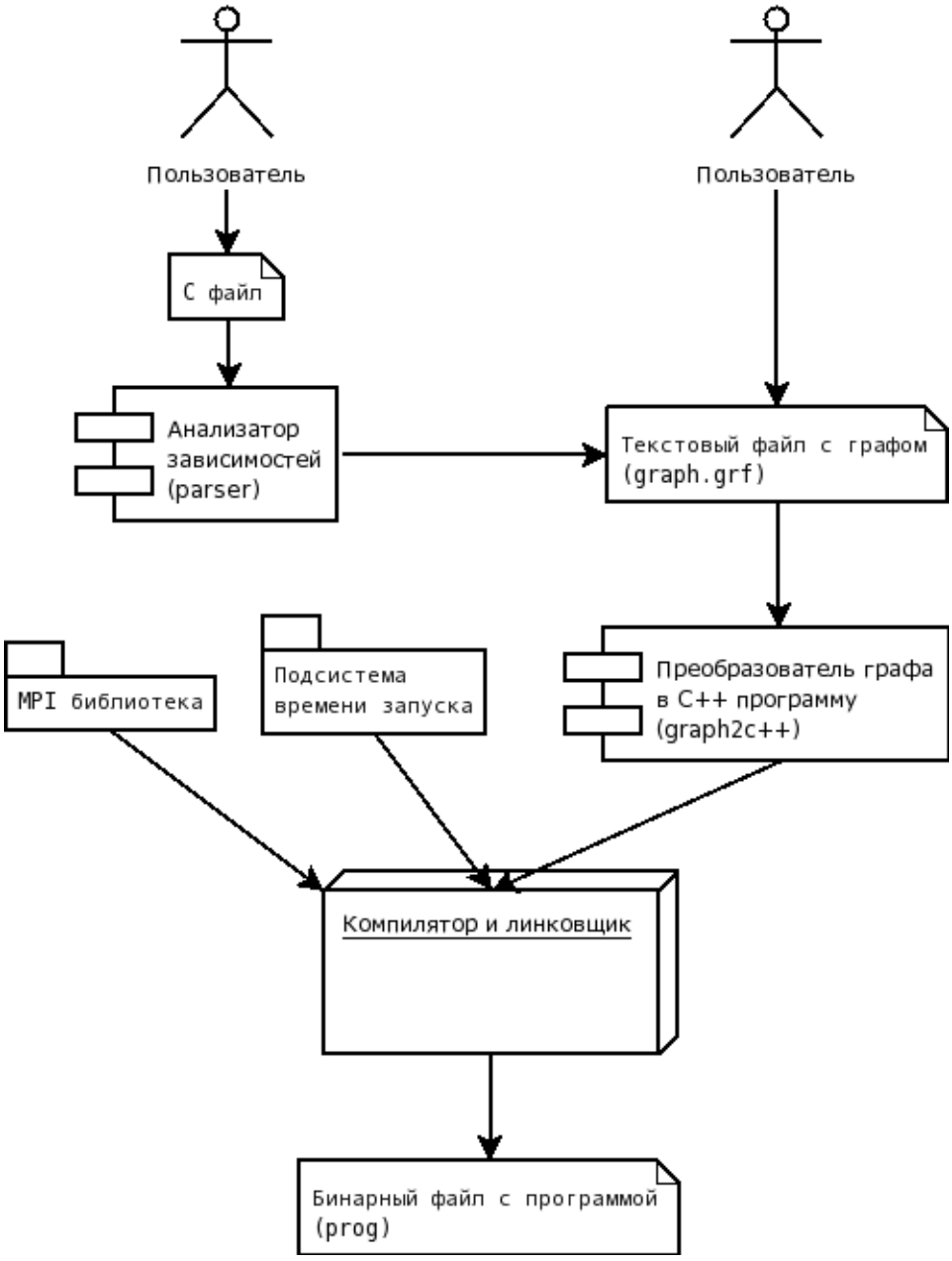

Рисунок 1. Компоненты и связи (преобразование в C++ программу)

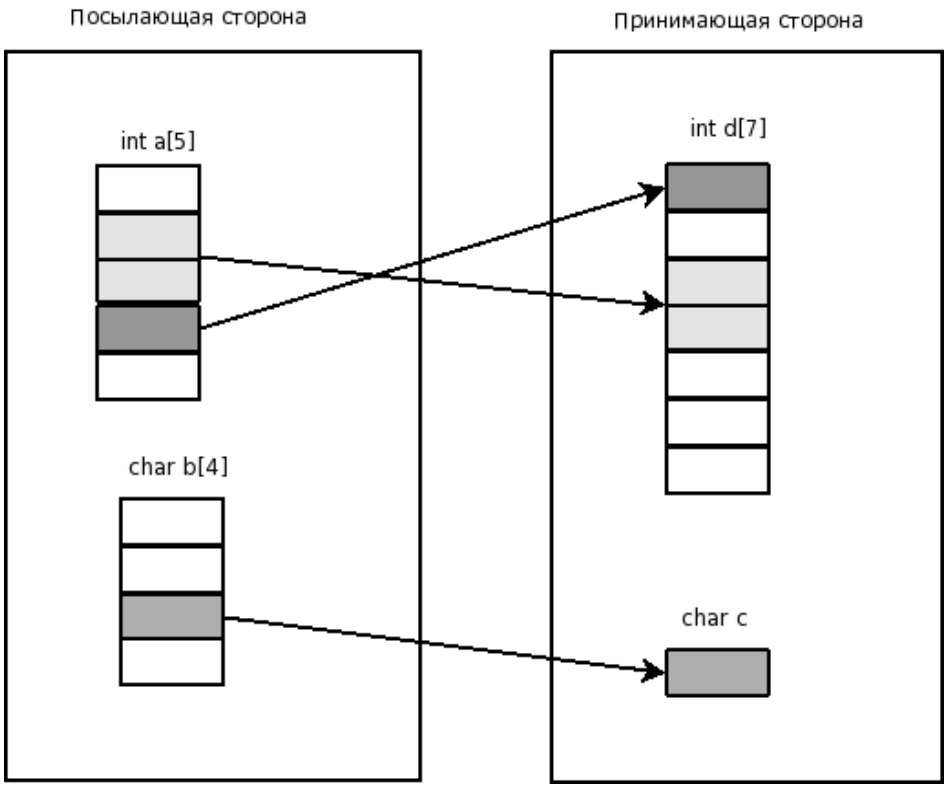

Рисунок 2. Внутренняя структура ребра граф-программы

С каждым ребром графа сопоставляется набор фрагментов данных («чанков»). На рисунке 2 представлена иллюстрация процесса выделения чанков на множестве переменных MPI-программы для ребра графа.

Чанк задаёт фрагмент в области памяти MPI-процесса, который будет сопоставлен с аналогичным фрагментом в памяти другого MPI-процесса. Один чанк связан с определённой переменной в вершине — источнике данных или в вершине — приёмнике данных. Несколько чанков могут быть связаны с одной и той же переменной. Чанк задаётся указателем и смещениями, которые определяют левую и правую границы фрагмента в области памяти. Имена переменных, которые синхронизируются через ребро графа, могут отличаться с принимающей и посылающей сторон. Всё множество чанков, описанных в текстовом файле для ребра графа, образуют одно ребро графа.

Действия, связанные непосредственно с передачей данных и управлением вызовами вершин на процессорах, не видны пользователю на этапе создания текстового файла с граф-программой. Они появляются только на этапе автоматического преобразования текстового файла с граф-программой в C++ код с вызовами MPI-функций. Тем самым достигается независимость описания граф-программы от особенностей многопроцессорной системы.

В **пятой главе** приведены примеры использования системы «PARUS». Работоспособность системы «PARUS» была протестирована на решении нескольких модельных задач. Система «PARUS» используется для решения конкретных прикладных задач. Как модельные задачи

были рассмотрены: распределённая операция над массивом и параллельная реализация перцептрона. В качестве прикладных задач в главе рассмотрены: задача создания параллельной реализации частотного фильтра звуковых сигналов и задача параллельного построения множественного выравнивания нуклеотидных и белковых последовательностей.

В **шестой главе** обсуждаются результаты тестирования системы «PARUS» на различных многопроцессорных системах. Система «PARUS» была установлена на следующие многопроцессорные вычислительные системы: МВС-1000М, IBM eServer pSeries 690 Regatta и Fujitsu Sun PRIMEPOWER 850. Исследовался характер влияния длины сообщения на время его доставки для многопроцессорных систем различной архитектуры, влияние «фоновых» сообщений на задержки.

Перцептрон был протестирован на машине Regatta. Параллельная реализация перцептрона написана как граф зависимости по данным. Для тестирования был выбран трехслойный перцептрон с большим количеством вершин (до 18500 в одном слое). Для генерации граф-программы было написано специальное приложение на языке С, которое по заданным характеристикам нейронной сети, а также параметрам группировки нейронов создаёт текстовый файл с описанием графа. Результаты тестирования этой задачи на системе Regatta показали, что при некоторых приёмах группирования нейронов достигается семикратное ускорение, и при увеличении числа нейронов оно стабилизируется.

Следующая задача, эффективность реализации которой на «PARUS» была исследована, – это модельная задача одновременного поиска максимума, минимума и суммы элементов массива чисел с плавающей точкой. Для машины МВС-1000М было получено практически линейное ускорение. Параметры, по которым строилась граф-программа, следующие: размерность массива – 1 миллиард ячеек, размер одного фрагмента – 1 миллион ячеек. До 30 процессоров ускорение можно считать линейным, причём с коэффициентом наклона больше 0,7. Однако далее кривая становится более пологой: хотя ускорение растёт, но значительно медленнее. Максимальное ускорение, которое было получено, – это 42,7 на 100 процессорах.

В человеческом геноме существуют неоднократно повторяющиеся слабо отличающиеся по последовательности нуклеотидов участки – повторы. Повторы занимают приблизительно 50% от всего размера генома. Как часть работ по совместному проекту Института физико-химической биологии им. А.Н. Белозерского МГУ и Людвиговского института раковых исследований "Ludwig Institute for Cancer Research" (грант CRDF RB01277-MO-2), проводилось исследование одного из типов повторов в человеческом геноме, а именно так называемого LTR класса 5. Важным инструментом для классификации повторов и выявления их свойств, моделирования их эволюции – является инструмент множественных выравниваний. С целью изучения LTR5 и их классификации на машине PRIMEPOWER 850 была установлена система «PARUS». Затем с её помощью была создана параллельная программная реализация для выравнивания последовательностей.

Параллельная реализация строилась на основе последовательного алгоритма MUSCLE. С целью распараллеливания алгоритма MUSCLE была внесена модификация в оригинальный алгоритм на стадии построения профилей выравнивания по кластерному дереву. На основе кластерного дерева строится граф-программа. Вершинам-истокам соответствуют одна или несколько последовательностей. Эти вершины создают профиль попавшего к ним

подмножества последовательностей. От них направляется дуга к внутренним вершинам. Внутренние вершины выравнивают пару профилей между собой, в результате получается новый профиль, который затем отправляется очередной внутренней вершине или в вершину-сток. Для повышения эффективности работы параллельной программы производится сжатие нижних уровней кластерного дерева (близких к листьям) в один уровень граф-программы. Для этого в алгоритм вводится специальный параметр, который задаёт число сжимаемых уровней кластерного дерева. Выигрыш во времени работы программы достигается за счёт того, что каждую вершину граф-программы можно исполнять на своём процессоре. Максимальное количество одновременно обрабатывающихся вершин будет ограничиваться числом вершин, приписанных одному слою граф-программы.

В однопроцессорном варианте выравнивание всех известных LTR5 в человеческом геноме занимает 1 час 8 минут. В многопроцессорном варианте на двенадцати процессорах оно же занимает 28 минут, что в 2,4 раза быстрее, чем последовательный вариант. На первый взгляд, эффективность параллельной реализации не очень высокая. Однако, ускорение параллельной реализации для конкретного множества выравниваемых последовательностей будет очень сильно зависеть от сбалансированности построенного кластерного или филогенетического дерева. Кроме всего прочего, в случае уточнения информации о последовательности LTR5 или обнаружения нового экземпляра LTR5 в геноме придётся перестраивать выравнивание. С большим количеством последовательностей или с последовательностями большой длины алгоритм MUSCLE, изначально не параллельный, может работать значительный промежуток времени – до нескольких суток; даже незначительное ускорение для параллельной программы в этом случае может значительно ускорить работу исследователя.

В **седьмой главе** приведены основные результаты и выводы диссертационной работы.

Основные результаты диссертационной работы

Разработан метод построения параллельных программ и язык их описания как граф-схемы потока данных, позволяющие абстрагироваться от конкретной технологии передачи сообщений.

Разработаны и реализованы алгоритмы управления процессом исполнения параллельной программы с учётом структуры программы, динамики обменов данными и текущего состояния многопроцессорной системы.

На основе предложенных метода, языка и алгоритмов создана система поддержки этапов разработки и исполнения параллельных программ.

Публикации автора по теме диссертации

1. Сальников А.Н. Некоторые технические аспекты инструментальной системы для динамической балансировки загрузки процессоров и каналов связи // Программные системы и инструменты. Тематический сборник № 3 факультета ВМиК МГУ им. Ломоносова. 2002 г., стр. 152-164, ISBN: 5-89407-149-6.

2. Сальников А.Н. Разработка инструментальной системы для динамической балансировки загрузки процессоров и каналов связи // Высокопроизводительные параллельные вычисления на кластерных системах. Материалы Международного научно-практического семинара. Изд-во Нижегородского университета, 2002 г., стр. 159-167, ISBN: 5-85746-681-4.

3. Булочникова Н.М., Сальников А.Н. Разработка прототипа CASE средства создания программ для гетерогенных многопроцессорных систем «PARUS» // Программные системы и инструменты. Тематический сборник № 4 факультета ВМиК МГУ им. Ломоносова, 2003 г. Издательский отдел факультета ВМиК МГУ. Стр. 203-209, ISBN: 5-89407-170-4.

4. Сальников А.Н., Сазонов А.Н., Карев М.В. Прототип системы разработки приложений и автоматического распараллеливания программ для гетерогенных многопроцессорных систем // Вопросы Атомной Науки и Техники. Серия: Математическое моделирование физических процессов. Министерство Российской Федерации по атомной энергии ФГУП, Российский федеральный ядерный центр – ВНИИЭФ. Научно-технический сборник, выпуск № 1. 2003 г., стр. 61-68.

5. Булочникова Н.М., Горицкая В.Ю., Сальников А.Н. Методы тестирования производительности сети с точки зрения организации вычислений // Труды Всероссийской научной конференции «Научный сервис в сети Интернет 2004». Издательство Московского университета 2004 г., стр. 221-223, ISBN 5-211-05007-X.

6. Alexeevski A.V., Lukina E.N., Salnikov A.N., Spirin S.A. Database of long terminal repeats in human genome: structure and synchronization with main genome archives // Proceedings of the fours international conference on bioinformatics of genome regulation and structure. Volume 1. BGRS 2004, Novosibirsk, редакционно-издательский отдел ИЦиГ СО РАН, стр. 28-29.

7. Алексеевский А.В., Лукина Е.Н., Спирин С.А., Сальников А.Н. Структура базы данных длинных терминальных повторов в человеческом геноме и синхронизация данных с основными архивами генной информации // Труды Всероссийской научной конференции «Научный сервис в сети Интернет 2004». Издательство Московского университета, стр. 219 , 2004 г., ISBN: 5-211-05007-X.

8. Сальников А.Н. Разработка структуры хранения и организация синхронизации информации по повторяющимся нуклеотидным последовательностям в человеческом геноме // Сборник тезисов Международной научной конференции студентов,

аспирантов и молодых ученых «Ломоносов - 2004». Том 1. 12-15 апреля 2004 г. МГУ им. М.В. Ломоносова, стр. 28.

9. Булочникова Н.М., Горицкая В.Ю., Сальников А.Н. Система поддержки сбора и анализа статистики о работе вычислительной системы // Методы и средства обработки информации. Труды Второй всероссийской научной конференции. – Москва, издательский отдел факультета ВМиК МГУ. 2005 г., стр. 136-141, ISBN: 5-89407-230-1.

10. Булочникова Н.М., Горицкая В.Ю., Сальников А.Н. Некоторые аспекты тестирования многопроцессорных систем // Программные системы и инструменты. Тематический сборник № 5 факультета ВМиК МГУ им. М.В. Ломоносова. Москва, издательский отдел факультета ВМиК МГУ, 2005 г., стр. 73-82, ISBN: 5-89407-216-6.

11. Колпаков Р.В., Сальников А.Н. Аспекты параллельного программирования при задании программы как графа зависимости по данным на примере написания тестов для системы «PARUS» // Методы и средства обработки информации. Труды Второй всероссийской научной конференции. – Москва, издательский отдел факультета ВМиК МГУ, 2005 г., стр. 269-275, ISBN: 5-89407-230-1.

12. Alexey N. Salnikov PARUS: A Parallel Programming Framework for Heterogeneous Multiprocessor Systems // Lecture Notes in Computer Science. Recent Advantages in Parallel Virtual Machine and Message Passing Interface. 2006, Volume 4192, pp. 408-409, ISSN: 0302-9743, ISBN-10: 3-540-39110-X, ISBN-13: 978-3-540-39110-4.