На правах рукописи

Притула Михаил Николаевич

ОТОБРАЖЕНИЕ DVMH-ПРОГРАММ НА КЛАСТЕРЫ С ГРАФИЧЕСКИМИ ПРОЦЕССОРАМИ

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

А в т о р е ф е р а т

диссертации на соискание ученой степени

кандидата физико-математических наук

Москва – 2013

Работа выполнена в Институте прикладной математики
им. М.В. Келдыша РАН.

Научный руководитель:
Крюков Виктор Алексеевич, доктор физико-математических наук, профессор.

Официальные оппоненты:
Якобовский Михаил Владимирович, доктор физико-математических наук, профессор, заведующий сектором в Институте прикладной математики им. М.В. Келдыша РАН.

Антонов Александр Сергеевич, кандидат физико-математических наук, ведущий научный сотрудник Лаборатории Параллельных информационных технологий НИВЦ МГУ.

Ведущая организация: Институт системного программирования РАН

Защита состоится «17» декабря 2013 г. в 11 часов на заседании Диссертационного совета Д 002.024.01 в Институте прикладной математики им. М.В. Келдыша РАН по адресу: 125047, Москва, Миусская пл., 4.

С диссертацией можно ознакомиться в библиотеке Института прикладной математики им. М.В. Келдыша РАН.
Автореферат разослан «15» ноября 2013 г.

Ученый секретарь
диссертационного совета

Т.А.Полилова

доктор физико-математических наук

Общая характеристика работы

Объект исследования и актуальность темы

В настоящее время большое количество параллельных программ для кластеров разрабатываются с использованием низкоуровневых средств передачи сообщений (MPI). MPI-программы трудно разрабатывать, сопровождать и повторно использовать при создании новых программ. Данная проблема осложняется тем, что в последнее время появляется много вычислительных кластеров с установленными в их узлах ускорителями. В основном, это графические процессоры. Программисту требуется теперь освоение на достаточном уровне сразу нескольких моделей и языков программирования. Традиционным подходом можно назвать использование технологии MPI для разделения работы между узлами кластера, а затем технологий OpenMP и CUDA или OpenCL для загрузки всех ядер центрального и графического процессоров.

Поэтому программисту необходимы высокоуровневые языки параллельного программирования, обеспечивающие эффективное использование современных параллельных систем. Такие языки могут быть построены путем расширения языков DVM-системы [http://www.keldysh.ru/dvm/], базирующихся на разработанной в ИПМ им.М.В.Келдыша РАН высокоуровневой модели параллельного программирования, получившей название DVM (Distributed Virtual Machine). Эти языки (Фортран-DVM и Си-DVM) в течение многих лет использовались для создания параллельных программ для кластеров.

Цели работы

Целями данной диссертационной работы являлись:

- исследование возможностей отображения DVM-программ на кластер с ускорителями, обеспечивающих распределение вычислений между универсальными многоядерными процессорами (ЦПУ) и несколькими графическими процессорами (ГПУ);
- разработка алгоритмов распределения вычислений между ЦПУ и ГПУ и алгоритмов управления перемещением данных между памятью ЦПУ и памятью ГПУ;
- разработка алгоритма распределения подзадач между узлами кластера.

Научная новизна работы

1. Разработаны алгоритмы распределения подзадач между узлами кластера, позволяющие балансировать загрузку вычислительных узлов кластера при выполнении многоблочных программ.
2. Разработаны алгоритмы распределения витков параллельных циклов внутри узлов между ядрами ЦПУ и несколькими ГПУ.

3. Разработаны алгоритмы автоматического перемещения требуемых актуальных данных между памятью ЦПУ и памятями нескольких ГПУ.
4. Созданы средства сравнительной отладки DVMH-программ, базирующиеся на сопоставлении результатов одновременного выполнения на ЦПУ и на ГПУ одних и тех же фрагментов программы.

Проведенные эксперименты с тестами и реальными приложениями показали, что для класса задач, при решении которых используются разностные методы на статических структурных сетках, можно писать программы на языке Фортран-DVMH, которые эффективно выполняются на кластерах с ускорителями.

Практическая значимость

С использованием разработанных алгоритмов создана система поддержки выполнения DVMH-программ, являющаяся неотъемлемой частью компиляторов DVMH-программ. Разработка компилятора с языка Фортран-DVMH существенно упростила создание эффективных программ для кластеров с ускорителями, способных автоматически настраиваться на целевую конфигурацию вычислительной системы. Компилятор с языка Фортран-DVMH, включающий в себя систему поддержки выполнения DVMH-программ, входит в состав DVM-системы и используется на факультете ВМК МГУ при проведении практикума по технологиям параллельного программирования. С использованием этого компилятора был распараллелен на кластер с ускорителями ряд прикладных вычислительных задач.

Апробация работы и публикации

Основные результаты диссертации были доложены на российских и международных научных конференциях и семинарах:

1. Международной научной конференции “Научный сервис в сети Интернет: суперкомпьютерные центры и задачи”, сентябрь 2010 г., г. Новороссийск.
2. Международной научной конференции “Научный сервис в сети Интернет: экзафлопсное будущее”, сентябрь 2011 г., г. Новороссийск.
3. XIII международном семинаре “Супервычисления и математическое моделирование”, октябрь 2011 г., г. Саров.
4. International Research Conference on Information Technology. Seventh International PhD&DLA Symposium, октябрь 2011 г., г. Pecs, Hungary.
5. Международной суперкомпьютерной конференции “Научный сервис в сети Интернет: поиск новых решений”, сентябрь 2012 г., г. Новороссийск.
6. APOS/HOPSA Workshop on Exploiting Heterogeneous HPC Platforms, январь 2013 г., г. Berlin.

7. Международной конференции "Параллельные вычислительные технологии (ПаВТ'2013)", апрель 2013 г., г. Челябинск.
8. Международной суперкомпьютерной конференции “Научный сервис в сети Интернет: все грани параллелизма”, сентябрь 2013 г., г. Новороссийск.

Имеется 12 публикаций, из которых три [5,7,11] – в журналах из списка ВАК.

Структура и объем работы

Диссертация состоит из введения, семи глав, заключения, списка литературы (39 наименований) и одного приложения. Общий объем работы составляет 105 страниц, работа содержит 1 иллюстрацию и 9 таблиц.

Содержание работы

Во **введении** обосновывается актуальность темы исследования, формулируются цель и задачи диссертации. Приводятся основные результаты работы, их практическая значимость. Перечисляются выступления на конференциях и семинарах, кратко излагается содержание работы.

В **первой главе** приводится обзор языков и технологий программирования для кластеров и графических процессоров.

Появление многопроцессорных ЭВМ с распределенной памятью значительно расширило возможности решения больших задач, однако резко усложнило разработку программ. Программист должен распределить данные между процессорами, а программу представить в виде совокупности процессов, взаимодействующих между собой посредством обмена сообщениями.

С развитием графических процессоров стало возможным использовать их и для вычислений общего назначения, а в последние годы их стали устанавливать в суперкомпьютеры для ускорения вычислений.

Для программирования графических процессоров производители аппаратного обеспечения предложили две довольно низкоуровневые технологии – CUDA и OpenCL. Относительная сложность и необычность (в сравнении с работой на ЦПУ) использования графических процессоров для вычислений общего назначения удерживает их от повсеместного использования.

В то же время успех OpenMP, предназначенного для систем с общей памятью, давал прикладным программистам надежду на появление подобных средств для использования графических процессоров, а исследовательским группам – направление для разработок.

Однако распространить стандарт OpenMP на графические процессоры не получалось.

В ИПМ им. М. В. Келдыша РАН в 2011 году в рамках данного исследования было предложено расширение DVM-модели, позволяющее разрабатывать программы для кластеров с графическими процессорами. Эта расширенная модель названа DVMH (DVM for Heterogeneous systems).

В ноябре 2011 года был предложен стандарт OpenACC для программирования графических процессоров с помощью директив, его вторая версия вышла в июне 2013 года.

В июле 2013 года опубликован стандарт OpenMP 4.0, в который вошли средства для работы с подключаемыми вычислительными устройствами, обладающие собственной памятью (сопроцессоры, ускорители).

Приводится сравнение этих трех предлагаемых подходов.

Главное отличие этих подходов заключается в том, что DVMH предназначен для написания программ для кластеров, в узлах которых установлены многоядерные процессоры и ускорители разных архитектур, отличающиеся как по типу памяти (общая или раздельная), так и по скорости обменов между памятью ускорителей и памятью ЦПУ. Стандарты OpenACC и OpenMP 4.0 предназначены для написания программ, выполняющихся в отдельных узлах такого кластера.

В стандартах OpenACC и OpenMP 4.0 применяется методика управления перемещением данных, основанная на указании каждого перемещения пользователем и определяемых пользователем интервалах (для OpenMP 4.0 – только статически определенных) существования экземпляра переменной на ускорителе. В DVMH-языках применена иная методика, основанная на указаниях входных и выходных данных для фрагментов кода, предназначенных для выполнения на ускорителях (такие фрагменты называются *вычислительными регионами* или просто *регионами*), и позволяющая автоматически определять необходимые перемещения данных в зависимости от того, на каком устройстве (ускорителе или многоядерном процессоре) был выполнен тот или иной вычислительный регион.

В стандартах OpenACC и OpenMP 4.0 управление тем, исполнять ли данный регион на ускорителе или нет, задается вычисляемыми во время выполнения выражениями. В DVMH такой возможности нет, однако имеется набор режимов работы, задающих на каких устройствах выполнять регионы.

Модели DVMH и OpenMP 4.0 предусматривают использование нескольких ускорителей для одной программы. В OpenACC одновременная работа с несколькими ускорителями не предусмотрена.

По части поддержки циклов с зависимостями DVMH имеет поддержку редукционных зависимостей, регулярных зависимостей; OpenACC имеет поддержку только редукционных зависимостей; OpenMP 4.0 имеет поддержку редукционных зависимостей и предоставляет средства синхронизации для распараллеливания циклов и с другими типами зависимостей.

Вторая глава посвящена описанию схемы построения компилятора с языка Фортран-DVMH, функций системы поддержки выполнения DVMH-программ.

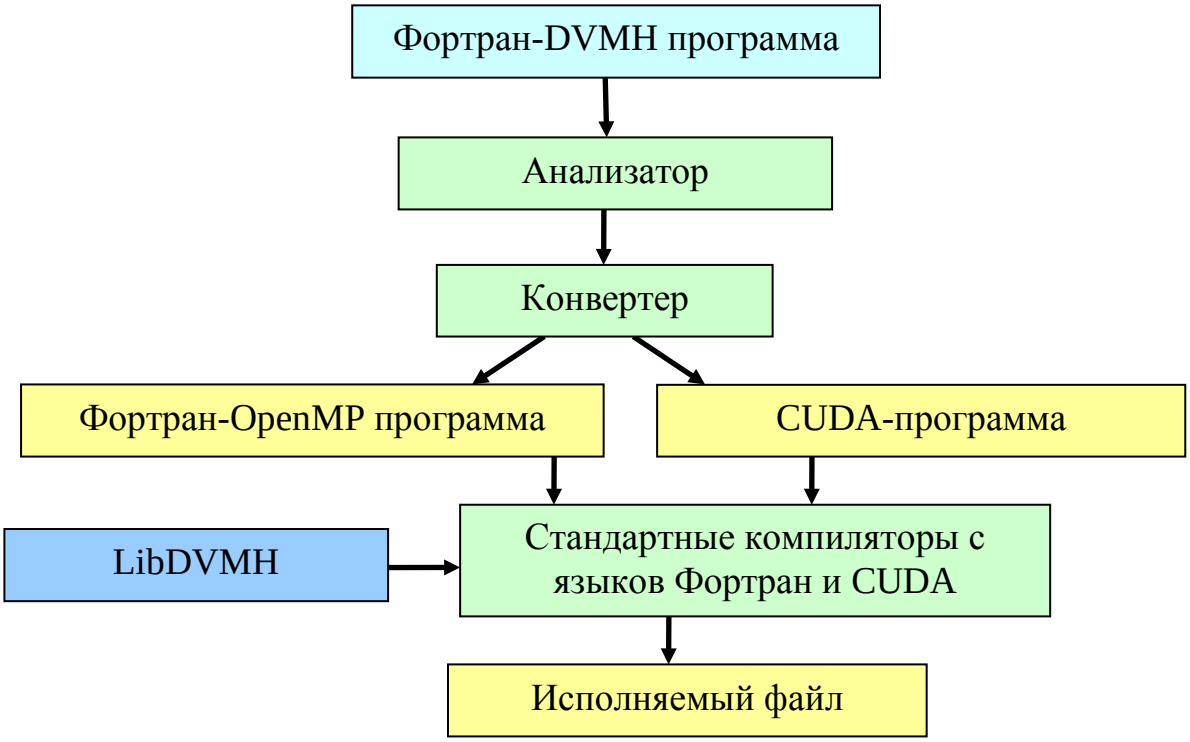

Рис. 1. Схема работы компилятора с языка Фортран-DVMH.

Компилятор с языка Фортран-DVMH состоит из блоков анализа программы, блока конвертации, системы поддержки выполнения DVMH-программ (библиотека LibDVMH).

Анализатор строит структуру исходной программы и дополняет заданные программистом указания о режимах использования данных, для которых есть правила автоматического определения.

Конвертер преобразует исходную программу в эквивалентную ей пару программ, которые опираются на систему поддержки выполнения DVMH-программ и уже могут быть скомпилированы в машинный код стандартными компиляторами.

LibDVMH является библиотекой функций, которая обеспечивает выполнение DVMH-программ, основные ее функции включают:

- отображение абстрактной параллельной машины (идеальной машины, наиболее подходящей для выполнения программы) на заданную при запуске решетку процессов и имеющийся в каждом узле набор вычислительных устройств;
- отображение распределенного массива на абстрактную параллельную машину;
- отображение параллельного цикла и параллельных задач на абстрактную параллельную машину;
- создание и загрузка буферов для доступа к удаленным данным;
- подготовку и организацию параллельного выполнения вычислительных регионов на разнородных вычислительных устройствах;

- балансировку загрузки узлов кластера и вычислительных устройств каждого узла (статическую и динамическую);
- управление текущим состоянием и местонахождением данных, их перемещениями;
- возможности для функциональной сравнительной отладки корректности работы на ускорителях;
- возможности для отладки производительности.

При обработке большинства директив конвертер генерирует один или несколько вызовов процедур LibDVMH с параметрами, отражающими указания пользователя в данной директиве.

При компиляции вычислительных регионов происходит их разделение на составные части – параллельные гнезда циклов и последовательные операторы. Эти части выделяются в отдельные подпрограммы с использованием соответствующих технологий программирования: для ГПУ – CUDA, для ЦПУ – OpenMP. Использование и настройка таких подготовленных подпрограмм ведется напрямую из LibDVMH, что позволяет их гибко комбинировать, многократно запускать, откладывать их выполнение (асинхронный режим).

В **третьей главе** описывается набор алгоритмов учета актуального состояния переменных. Эти алгоритмы описывают механизмы обработки всех ситуаций в программе, потенциально приводящих к перемещениям данных.

Ручное управление программистом всеми перемещениями данных имеет ряд недостатков:

- Программисту приходится быть четко ориентированным на то, сколько и каких ускорителей используется в программе, а также предполагается ли одновременное использование ЦПУ для дополнительного разделения вычислений. Это может приводить к утере ее универсальности в плане используемой целевой ЭВМ, или же значительному усложнению программы.
- Для программ с достаточно большой степенью разветвленности, а также программ, имеющих многократно выполняемые части, причем при различных входных данных и различных путях выполнения, становится серьезной проблемой оптимизация перемещений данных, что может приводить как к излишним перемещениям, так и к сложно находимым ошибкам, проявляющимся только при некоторых наборах входных данных.

Использование метода управления перемещениями данных, основанного на задании входных и выходных данных регионов, и, как следствие, полное информирование системы поддержки выполнения программ о выполняемых модификациях переменных, позволяет избавиться от недостатков полностью ручного управления перемещениями данных, а также добавляет полезные возможности такие, как:

- сравнительная функциональная отладка (выполнение региона с одними и теми же исходными данными на ЦПУ и на ускорителях, а затем сравнение значений полученных выходных данных);

- многократное выполнение регионов с одними и теми же исходными данными с целью поиска значений оптимизационных параметров.

Разработанные алгоритмы основываются на следующих понятиях:

- представитель некоторой переменной на некотором устройстве – экземпляр переменной, размещенный в памяти устройства;
- состояние актуальности представителя – задание для всех элементов представителя, являются ли они (элементы) актуальными, т.е. имеют ли они последнее присвоенное этому элементу значение;

Для манипуляций с состояниями актуальности представителей переменных вводится особый вид функций – PCS_n. Вводятся базовые операции над ними: объединение двух PCS, разность двух PCS, понижение одного PCS на уровень другого PCS, горизонтальная разность двух PCS, пересечение двух PCS, пересечение двух PCS с повышением.

В процессе работы DVMH-программы система поддержки выполнения DVMH-программ следит за состоянием актуальности представителей переменных на всех используемых устройствах, модифицирует его в соответствии с указаниями направления использования данных в вычислительных регионах, происходящими теневыми обменами, указаниями директив актуализации и другими происходящими событиями, затрагивающими данные.

Приводятся алгоритмы учета актуального состояния данных при различных ситуациях, в том числе:

- Вход в вычислительный регион. При входе в вычислительный регион системе поддержки выполнения сообщается о каждой переменной направления ее использования для ее подмассивов.

  После определения распределения данных по устройствам система поддержки выполнения для каждой используемой в регионе переменной выполняет следующие действия:

  1. Для каждого используемого устройства:
     1. Определение части переменной, которая должна находиться на устройстве.
     2. Если на нем не находится необходимая часть переменной, то расширить имеющийся представитель.
     3. Получение запрошенных актуальных данных с других устройств.
     4. Для каждого используемого устройства объявление актуальности для выходных частей переменной на данном устройстве.
- Запрос актуализации. Это одна из основополагающих операций, которая в обобщенном виде лежит в основе почти всех манипуляций с данными. Она для заданных элементов переменной и для заданного устройства обновляет те элементы, которые не

являются в требуемой степени актуальными, находя в достаточной степени актуальные элементы переменной на других устройствах.

Четвертая глава посвящена описанию режимов распределения данных и вычислений между вычислительными устройствами.

Одним из важных аспектов функционирования такой программной модели, как DVMH является вопрос отображения исходной программы на все уровни параллелизма и разнородные вычислительные устройства. Важными задачами механизма отображения является обеспечение корректного исполнения всех поддерживаемых языком конструкций на разнородных вычислительных устройствах, балансировка нагрузки между вычислительными устройствами, а также выбор оптимального способа исполнения каждого фрагмента кода на том или ином устройстве.

Выполнение DVMH-программы можно представить, как выполнение последовательности вычислительных регионов и участков между ними, которые будем называть внерегионным пространством. Код во внерегионном пространстве выполняется на центральном процессоре, тогда как для вычислительных регионов возможно их исполнение на разнородных вычислительных устройствах. Внутри, равно как и вне регионов могут быть как параллельные циклы, так и последовательные участки программы.

Параллелизм в DVMH-программах проявляется на нескольких уровнях:

- Распределение данных и вычислений по MPI-процессам. Этот уровень задается специальными директивами распределения или перераспределения данных и спецификациями параллельных подзадач и циклов.
- Распределение данных и вычислений по вычислительным устройствам при входе в вычислительный регион.
- Параллельная обработка в рамках конкретного вычислительного устройства. Этот уровень появляется при входе в параллельный цикл, находящийся внутри вычислительного региона.

Наличие этих уровней дает возможность органично отобразить программу на кластер с многоядерными процессорами и ускорителями в узлах.

Системой поддержки выполнения DVMH-программ поддерживаются три режима распределения данных и вычислений по вычислительным устройствам в точках входа в регионы:

1. Простой статический режим.
2. Динамический режим с подбором схемы распределения.
3. Динамический режим с использованием подобранной схемы распределения.

В простом статическом режиме в каждом регионе распределение производится одинаково. Пользователем задается вектор относительных производительностей вычислительных устройств, имеющихся в каждом узле кластера, затем эти производительности накладываются на параметры межпроцессного распределения данных. В таком режиме сводятся к минимуму перемещения данных, связанные с их перераспределением, но не учитывается

различное соотношение производительности вычислительных устройств на разных фрагментах кода.

Динамический режим с подбором распределения

В этом режиме в каждом вычислительном регионе распределение данных и вычислений выбирается на основе постоянно пополняющейся истории запусков данного региона и его соседей, определяющихся динамически.

Каждый вычислительный регион в данном режиме рассматривается в виде нескольких родственных вариантов запуска, каждый из которых определяется парой (вычислительный регион, соответствие данных), где под соответствием данных понимается соответствие используемых в исходном коде вычислительного региона локальных переменных реальным переменным.

В этом режиме периодически включается построение субоптимальной схемы распределения данных во всех вариантах запуска вычислительных регионов на основе накопленных сведений в целом для программы, анализируя целиком последовательность вариантов запуска регионов и их характеристики выполнения. При построении таких схем учитываются как внутренние показатели вариантов запуска регионов в виде зависимости времени работы от распределения данных, так и последовательность исполнения вариантов запуска вычислительных регионов с целью минимизации в том числе и временных затрат на перераспределение данных.

В динамическом режиме с использованием подобранной схемы распределения в каждом вычислительном регионе распределение выбирается на основе предоставленной схемы распределения, построенной при работе программы во втором режиме, причем есть возможность как перейти в этот режим непосредственно из второго, так и использовать схему распределения из файла, полученного в результате работы программы во втором режиме. При использовании схемы распределения из файла не гарантируется ее корректное использование в случае, если параметры программы были изменены, в особенности это касается тех параметров, которые влияют на путь выполнения программы (например, добавление или удаление этапов расчета).

В пятой главе описывается алгоритм распределения подзадач между узлами кластера.

Одним из достоинств DVM-системы является то, что получаемые параллельные программы могут настраиваться при запуске на количество выделенных для них процессоров. Такое свойство программ позволяет запускать их на произвольном числе процессоров, компоновать сложные программы из имеющихся простых программ, повысить эффективность использования параллельных систем коллективного пользования за счет более гибкого распределения процессоров между отдельными программами. Этим свойством не обладают DVM-программы, использующие механизм подзадач. Распределение подзадач по процессорам производится вручную и оно зачастую затруднено вследствие большого количества подзадач, заметного разброса их сложности, необходимости осуществлять распределение на различное количество процессоров.

В данной работе предлагается эвристический алгоритм распределения подзадач для составления расписания прохождения подзадач, где для каждой подзадачи будет указано стартовое время, группа процессоров для ее счета. Подзадачи не могут считаться одновременно используя один и тот же процессор. Для всех процессоров из группы, назначенной для счета подзадачи, времена старта счета этой подзадачи совпадают, равно как и времена счета (продолжительность) подзадачи – синхронное выполнение одной подзадачи.

Как известно, задача составления многопроцессорного расписания NP-полна, а, значит, исследуемая задача NP-трудна, так как является обобщением NP-полной задачи.

За M обозначим количество процессоров, за N – количество подзадач. Для каждой подзадачи t известно:

1. минимальное используемое количество процессоров $K_{min}(t)$
2. максимальное используемое количество процессоров $K_{max}(t)$
3. функция зависимости времени исполнения от количества процессоров time(t,k)>0 такая, что для всех k из диапазона $[K_{min}(t), K_{max}(t) - 1]$ выполнено: time(t,k) * k <= time(t,k + 1) * (k + 1)

Для описания алгоритма вводятся дополнительные обозначения:

Proc(x, d) – множество всех процессоров, свободных с момента времени x и, как минимум, до момента (x + d)

Proc(x) – отображение такое, что Proc(x)(d) = Proc(x, d) для всех d и x

Procs(x) – множество всех процессоров, свободных в момент времени x

Tasks – множество всех подзадач

Алгоритм распределения подзадач можно описать следующим образом:

1. Положим tRestMin равным сумме по всем подзадачам t величин $time(t,K_{min}(t))*K_{min}(t)$

2. Упорядочить подзадачи по убыванию величины $time(t, K_{min(t)}) * K_{min}(t)$ – список sortedTasks, где t принимает все значения из Tasks

3. Положим tMax и tOccupied равными нулю

4. Взять задачу t из начала списка sortedTasks

5. Удалить задачу t из списка sortedTasks

6. Уменьшить tRestMin на величину $time(t, K_{min}(t)) * K_{min}(t)$

7. Положим множество notExamined равным $[K_{min}(t), K_{max}(t)]$

8. Для каждого момента времени x, являющегося либо началом отсчета времени, либо моментом изменения множества Procs(x) в порядке возрастания:

8.1. Для каждой длительности d, не меньшей time(t, k) для некоторого k, в порядке убывания выполнять:

8.1.1. Для каждого количества процессоров k, принадлежащему notExamined и не большему, чем мощность Proc(x, d), а также такому, что time(t, k) <= d в порядке возрастания выполнять:

8.1.1.1. Положим tSuggested(x, k) равным максимуму из трех величин: tMax, x + time(t, k), x + (tOccupied + tRestMin) / k

8.1.2. Исключить из множества notExamined рассмотренные в цикле 8.1.1 количества процессоров.

8.1.3. Если множество notExamined пусто, то перейти к пункту 9

9. Пусть x0 – такое минимальное, что минимизирует tSuggested(x0, k) для некоторого k. Пусть k0 – минимальное из таких, что минимизируют tSuggested(x0, k0)

10. Положим tMax максимуму из tMax и x0 + time(t, k0)

11. Увеличим tOccupied на величину time(t, k0) * k0

12. Зарезервировать группу из k0 процессоров на время [x0, x0 + time(t, k0)], выделив подмножество из Proc(x0, d) с таким максимальным d, что мощность Proc(x0, d) не менее k0

13. Если список sortedTasks не пуст, то перейти к пункту 4

В результате будет составлено полное расписание прохождения подзадач на многопроцессорной системе. Алгоритм имеет алгоритмическую сложность асимптотически равную O(N * (N + M)), затраты по памяти асимптотически равны O(N + M).

Разработанный алгоритм также может быть использован в планировщиках систем очередей для кластеров коллективного доступа.

В язык Fortran-DVM были добавлены конструкции для поддержки автоматического распределения подзадач, однако от программиста требуется указать относительные сложности подзадач, минимальное и максимальное число процессоров для каждой подзадачи, а также параметры зависимости времени выполнения каждой подзадачи от числа выделенных ей процессоров.

Шестая глава содержит описание дополнительных возможностей по функциональной отладке и отладке производительности.

Для DVMH-программ есть возможность сравнительной отладки вычислительных регионов, это специальный режим работы вычислительного региона, при котором все вычисления одновременно выполняются на ЦПУ и других вычислительных устройствах с целью сравнения значений выходных переменных при завершении вычислительного региона. Такой механизм позволяет выявлять и локализовывать ошибки, проявляющиеся при работе на ускорителях.

Реализация этого механизма основывается на том, что, во-первых, вычислительный регион может быть выполнен одновременно независимо на нескольких вычислительных устройствах и, во-вторых, системе поддержки выполнения DVMH-программ известен исчерпывающий набор входных и выходных данных региона, так как он ей необходим для управления перемещениями данных между устройствами.

В сравнение включаются все выходные данные вычислительного региона. При этом целочисленные данные сравниваются на совпадение, а вещественные числа сравниваются с заданной точностью по абсолютной и относительной погрешности. В случае нахождения расхождений пользователю выдается информация о них, и далее в программе используется та версия данных, которая была получена при счете на центральном процессоре.

Возможности по отладке производительности позволяют узнать временные затраты на выполнение параллельных циклов, последовательных участков; временные затраты и объемы перемещений данных при обновлении

теневых граней, выполнении редукционных операций, выполнении директив актуализации и перераспределении данных.

Седьмая глава содержит описание приложений и тестов, разработанных с использованием языка Фортран-DVMH.

Задача «Каверна»

Программа «Каверна» предназначена для моделирования циркуляционного течения в плоской квадратной каверне с движущейся верхней крышкой в двумерной постановке в широком диапазоне параметров задачи.

Последовательная версия программы занимает 496 строк.

В параллельной программе указаны директивы языка Фортран-DVMH для распределения данных и вычислений (18 распределенных массивов, 7 регионов, 28 параллельных циклов), организации доступа к удаленным данным (8 мест), актуализации (11 мест). Текст программы занимает 613 строк.

В таблицах 1 и 2 приведены времена выполнения 200 итераций программы «Каверна» на сетке 3200x3200 на суперкомпьютере «Ломоносов» на разном числе процессорных ядер и ГПУ (в секундах).

Таблица 1

Время программы «Каверна» на сетке 3200x3200 на разном числе ядер ЦПУ

1	**2**	**4**	**8**	**16**	**64**	**128**	**256**	**400**	**512**	**1024**
1241,83	631,47	332,36	182,95	100,05	40,20	21,33	11,74	7,11	6,44	3,48

При использовании 1024 ядер программа «Каверна» ускорилась в 357 раз по сравнению с 1 ядром. При использовании 1 ускорителя программа ускоряется в 17 раз по сравнению с выполнением программы на 1 ядре.

Таблица 2

Время программы «Каверна» на сетке 3200x3200 на разном числе ГПУ

1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**400**
73,07	39,34	19,94	11,65	7,17	4,80	3,96	3,45	3,32	3,19

Задача «Контейнер»

Программа «Контейнер» предназначена для численного моделирования течения вязкой тяжелой жидкости под действием силы тяжести в прямоугольном контейнере с открытой верхней стенкой и отверстием.

Последовательная версия программы занимает 828 строк.

В параллельной программе указаны директивы языка Фортран-DVMH для распределения данных и вычислений (26 распределенных массивов, 5 регионов, 21 параллельный цикл), организации доступа к удаленным данным (6 мест), актуализации (7 мест). Текст программы занимает 942 строки.

В таблицах 3 и 4 приведены времена выполнения 50 итераций программы «Контейнер» на сетке 800x800x800 на суперкомпьютере «Ломоносов» на разном числе процессорных ядер и ГПУ.

Таблица 3

Время выполнения программы «Контейнер» на разном числе ядер

Ядра ЦПУ	**4**	**8**	**64**	**128**	**256**	**512**	**1024**	**2048**
Время	-	-	576,16	318,75	151,78	79,68	41,26	21,91

Таблица 4

Время выполнения программы «Контейнер» на разном числе ГПУ

Кол-во ГПУ	**1**	**2**	**4**	**32**	**128**	**256**	**512**	**1024**	**1280**
Время	-	-	-	92,20	30,32	13,58	7,56	4,67	4,17

При использовании 2048 ядер программа «Контейнер» ускорилась в 26,3 раза по сравнению с выполнением на 64 ядрах. При использовании 64 ускорителей программа ускоряется в 11 раз по сравнению с выполнением программы на 64 ядрах. При использовании 1280 ускорителей программа ускоряется в 138 раз по сравнению с выполнением программы на 64 ядрах.

Задача «Состояния кубитов»

Программа «Состояния кубитов» предназначена для проведения трехмерных нестационарных расчетов состояния кубитов квантового компьютера на основе совместного решения трехмерного уравнения Пуассона и нестационарного уравнения Шредингера для двух электронов в квантовом приборе на основе кремниевой квантовой проволоки с учетом спина этих электронов.

Последовательная версия программы занимает 683 строк.

В параллельной программе указаны директивы языка Фортран-DVMH для распределения данных и вычислений (23 распределенных массива, 5 регионов, 61 параллельный цикл), организации доступа к удаленным данным (8 мест), актуализации (5 мест). Текст программы занимает 1011 строк.

В таблице 5 приведены времена выполнения 192 итераций программы «Состояния кубитов» на сетке 121x121x241 на суперкомпьютере «К-100» на разном числе процессорных ядер и ГПУ (в секундах).

Таблица 5

Время выполнения программы «Состояния кубитов» на сетке 121x121x241 на разном числе процессорных ядер и ГПУ

12 ядер	**24 ядра**	**36 ядер**	**48 ядер**	**60 ядер**	**1 ГПУ**	**2 ГПУ**	**4 ГПУ**	**6 ГПУ**
190,38	104,20	74,57	56,71	39,96	102,06	59,66	32,97	22,04

При использовании 60 ядер программа «Состояния кубитов» ускорилась в 4,76 раз по сравнению с выполнением на 12 ядрах. При использовании 1 ускорителя программа ускоряется в 1,87 раз по сравнению с 12 ядрами.

Задача «Спекание 3D»

Программа «Спекание 3D» связана с задачей, в которой лазер плавит порошок, состоящий из твердой легкоплавкой компоненты. В порошке имеются поры, заполненные газом. В процессе плавления легкоплавкой компоненты порошка образуется жидкость.

Последовательная версия программы занимает 677 строк.

В параллельной программе указаны директивы языка Фортран-DVMH для распределения данных и вычислений (19 распределенных массивов, 4 региона, 51 параллельный цикл), организации доступа к удаленным данным (3 места), актуализации (22 места). Текст программы занимает 1236 строк.

В таблице 6 приведены времена выполнения 1 000 итераций программы «Спекание 3D» на сетке 103x103x103 на суперкомпьютере «К-100» на разном числе процессорных ядер и ГПУ (в секундах).

Таблица 6

Время выполнения программы «Спекание 3D» на сетке 103x103x103 на разном числе процессорных ядер и ГПУ

1 ядро	12 ядер	64 ядра	200 ядер	1 ГПУ	6 ГПУ	12 ГПУ	24 ГПУ
751	192,9	40,23	30,46	50,88	18,47	14,85	12,64

При использовании 200 ядер программа «Спекание 3D» ускорилась в 24,66 раз по сравнению с 1 ядром. При использовании 1 ускорителя программа ускоряется в 14,76 раз по сравнению с выполнением программы на 1 ядре.

Тесты на производительность из набора NASA NPB

Для сравнения программ на языке Фортран-DVMH с написанными вручную с использованием низкоуровневых технологий программами по достигаемой эффективности приводятся ускорения тестов EP, BT, SP, LU из набора NASA NPB на 1 ГПУ в сравнении со временем работы на 1 ядре ЦПУ.

Сравнение производилось с программами, написанными исследователями из Сеульского национального университета. Их работа «Performance Characterization of the NAS Parallel Benchmarks in OpenCL» опубликована в сборнике трудов международной конференции «2011 IEEE International Symposium on Workload Characterization», а исходные тексты программ находятся в свободном доступе.

В таблице 7 приведены достигнутые ускорения для тестов EP, BT, SP, LU различных классов на 1 ГПУ по отношению ко времени работы на 1 ядре ЦПУ для программ на языке Фортран-DVMH и программ на языке Си+OpenCL. Запуски проводились на кластере «К-100».

Таблица 7

Ускорения тестов NASA NPB для программ на языке Фортран-DVMH и на языке Си+OpenCL на 1 ГПУ по отношению ко времени на 1 ядре ЦПУ

Тест	**EP**	**BT**		**SP**			**LU**		
Класс	**C**	**A**	**B**	**A**	**B**	**C**	**A**	**B**	**C**
DVMH	46,24	2,4	2,22	5,88	6,37	7,72	6,47	9,06	11,1
OpenCL	44,63	0,89	1,04	3,05	3,36	2,87	5,43	6,72	6,4

DVMH-версии тестов по эффективности не уступают OpenCL-версиям, а на тестах, имеющих циклы с зависимостями, превосходят их благодаря имеющимся в компиляторе с языка Фортран-DVMH оптимизациям.

В **заключении** сформулированы основные результаты работы.

Основные результаты работы:

1. Разработаны принципы отображения DVMH-программ на кластеры с ускорителями, обеспечивающие динамическое распределение вычислений между универсальными многоядерными процессорами (ЦПУ) и несколькими графическими процессорами (ГПУ).
2. Разработаны и реализованы в системе поддержки выполнения DVMH-программ следующие алгоритмы:

- распределения независимых подзадач между узлами кластера, обеспечивающие балансировку загрузки узлов;
- распределения витков параллельных циклов внутри узлов – между ядрами ЦПУ и несколькими ГПУ;
- автоматического перемещения требуемых актуальных данных между памятью ЦПУ и памятями нескольких ГПУ.

3. Созданы средства сравнительной отладки DVMH-программ, базирующиеся на сопоставлении результатов одновременного выполнения фрагментов программы на ЦПУ и на ГПУ.

Проведенные эксперименты с тестами и реальными приложениями показали, что для класса задач, при решении которых используются разностные методы на статических структурных сетках, можно писать программы на языке Фортран-DVMH, которые эффективно выполняются на кластерах с ГПУ.

Публикации по теме диссертации

1. Кривов М.А., Притула М.Н. Конвейерная модель представления параллельных программ. // Материалы Девятой международной конференции-семинара “Высокопроизводительные параллельные вычисления на кластерных системах”, ноябрь 2009 г., г. Владимир. – Владимир: Издательство ВлГУ, 2009, стр. 255-258.
2. Притула М.Н. Эффективный алгоритм планирования задач, допускающих параллельный счет на разном числе процессоров. Его использование в системе DVM. // Труды Международной научной конференции “Научный сервис в сети Интернет: суперкомпьютерные центры и задачи”, сентябрь 2010 г., г. Новороссийск. – М.: Изд-во МГУ, 2010, с. 679-681.
3. В.А. Бахтин, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, Ю.Л. Сазанов. Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами. // Супервычисления и математическое моделирование. Труды XIII Международного семинара / Под ред. Р.М. Шагалиева. – Саров: ФГУП “РФЯЦ-ВНИИЭФ”, 2012, с. 84-91.
4. Bakhtin V.A., Krukov V.A., Pritula M.N. Extension of DVM parallel programming model for clusters with heterogeneous nodes. // Research conference on information technology. Honoring volume on Pollack Mihaly faculty of engineering and information technology. Seventh international PhD & DLA symposium, октябрь 2011 г., г. Pecs, Hungary. – Komlo: Rotari Press, 2011, с. C17.
5. В.А. Бахтин, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, Ю.Л. Сазанов. Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами. Вестник Южно-Уральского государственного университета, серия "Математическое моделирование и программирование", №18 (277), выпуск 12 – Челябинск: Издательский центр ЮУрГУ, 2012, с. 82-92.

6. В.А. Бахтин, Н.А. Катаев, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула. Автоматическое распараллеливание Фортран-программ на кластер с графическими ускорителями. // Параллельные вычислительные технологии (ПаВТ'2012): труды международной научной конференции (Новосибирск, 26 марта - 30 марта 2012 г.). – Челябинск: Издательский центр ЮУрГУ, 2012. с. 373-379.
7. М.А. Кривов, М.Н. Притула, С.А. Гризан, П.С. Иванов. Оптимизация приложений для гетерогенных архитектур. Проблемы и варианты решения. Информационные технологии и вычислительные системы, № 2012/03. ISSN 2071-8632. http://www.jitcs.ru/
8. В.А. Бахтин, И.Г. Бородич, Н.А. Катаев, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, Ю.Л. Сазанов. Распараллеливание с помощью DVM-системы некоторых приложений гидродинамики для кластеров с графическими процессорами. // Труды Международной суперкомпьютерной конференции “Научный сервис в сети Интернет: поиск новых решений”, сентябрь 2012 г., г. Новороссийск. – М.: Изд-во МГУ, 2012, с. 444-450.
9. В.А. Бахтин, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, Ю.Л. Сазанов. Использование языка Fortran DVMH для решения задач гидродинамики на высокопроизводительных гибридных вычислительных системах. // Параллельные вычислительные технологии (ПаВТ'2013): труды международной научной конференции (г. Челябинск, 1-5 апреля 2013 г.). – Челябинск: Издательский центр ЮУрГУ, 2013, с. 58-67.
10. М.Н. Притула. Отображение DVMH-программ на кластеры с ускорителями. // Параллельные вычислительные технологии (ПаВТ'2013): труды международной научной конференции (г. Челябинск, 1-5 апреля 2013 г.). – Челябинск: Издательский центр ЮУрГУ, 2013, с. 515-520.
11. В.А. Бахтин, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, А.А. Смирнов. Использование языка Fortran DVMH для решения задач гидродинамики на высокопроизводительных гибридных вычислительных системах. Вестник Южно-Уральского государственного университета, серия "Вычислительная математика и информатика", том №2, выпуск №3 – Челябинск: Издательский центр ЮУрГУ, 2013, с 106-120
12. В.А. Бахтин, А.С. Колганов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула. Отображение на кластеры с графическими процессорами циклов с зависимостями по данным в DVMH-программах. // Труды Международной суперкомпьютерной конференции “Научный сервис в сети Интернет: все грани параллелизма”, сентябрь 2013 г., г. Новороссийск. – М.: Изд-во МГУ, 2013, с. 250-257.