ISSN 1992-6502 (Print)
2017. Т. 21, № 3 (77). С. 87–96
Вестник УГАТУ
ISSN 2225-2789 (Online)
http://journal.ugatu.ac.ru

УДК 004.65

АВТОМАТИЗИРОВАННОЕ РАСПАРАЛЛЕЛИВАНИЕ ЗАДАЧИ МОДЕЛИРОВАНИЯ РАСПРОСТРАНЕНИЯ УПРУГИХ ВОЛН В СРЕДАХ СО СЛОЖНОЙ 3D-ГЕОМЕТРИЕЙ ПОВЕРХНОСТИ НА КЛАСТЕРЫ РАЗНОЙ АРХИТЕКТУРЫ*

Н. А. Катаев[1], А. С. Колганов[1,2], П. А. Титов[3]

[1] kaniandr@gmail.com, [2] alexander.k.s@mail.ru, [3] tapawel@gmail.com

[1] ФИЦ «Институт прикладной математики им. М. В. Келдыша» (ИПМ)
[2] «Факультет ВМК МГУ им. М. В. Ломоносова» (МГУ)
[3] «Институт вычислительной математики и математической геофизики СО» (ИВМиМГ СО)

Поступила в редакцию 01.06.2017

Аннотация. В работе рассмотрено применение систем DVM и САПФОР для автоматизации распараллеливания задачи моделирования трехмерных упругих волн на высокопроизводительные кластеры различной архитектуры. Отличительной особенностью данной задачи является использование криволинейной трехмерной сетки, которая хорошо согласуется с геометрией строения свободной поверхности. Но использование криволинейных сеток значительно усложняет как ручное, так и автоматизированное распараллеливание. Для решения данной проблемы был предложен метод отображения криволинейной поверхности на структурированную сетку. Последовательная программа, использующая метод конечных разностей на структурированной сетке, была отображена в параллельную программу на языке Fortran-DVMH с использованием инструментов анализа системы САПФОР. Рассмотрены особенности автоматизированного распараллеливания. Представлены результаты оценки эффективности и ускорения параллельной Fortran-DVMH программы, а также сравнение производительности полученной FDVMH-программы с программой, написанной вручную с использованием технологии MPI.

Ключевые слова: Автоматизация распараллеливания; гетерогенный вычислительный кластер; инкрементальное распараллеливание; теория упругости; 3D-моделирование; криволинейная сетка; ГПУ.

ВВЕДЕНИЕ

Моделирование трехмерных упругих волн в средах различного строения является важным аспектом создания геофизических трехмерных моделей и изучения особенностей волновых полей. Решить обратную задачу геофизики (восстановление строения и параметров среды по экспериментально полученным записям сигналов) зачастую очень сложно, и одним из методов является решение набора прямых задач (моделирование сейсмополей в среде с заданными параметрами и строением) с варьированием значений параметров и геометрии среды при сравнении реальных данных с результатами моделирования.

Широко используемый метод для решения прямой задачи – метод конечных разностей [1–4]. Отметим, что исследуемая область может иметь сложную геометрию трехмерной поверхности, поэтому важным отличительным моментом рассматриваемой

Работа поддержана грантами РФФИ 16-07-01067, 16-01-00455, 16-07-00434, 17-01-00820, а также программы фундаментальных исследований РАН 4.9 «Модельные и экспериментальные исследования вулканических структур методами активной и пассивной сейсмологии».

задачи является построение криволинейной трехмерной сетки. Например, объектом исследования может быть магматический вулкан. Изучение строения среды и мониторинг подобного объекта является важной практической задачей, требующей больших вычислительных мощностей для достаточно быстрого получения результата. Теория построения и применения криволинейных сеток для решения реальных задач хорошо описана в работах [5–7].

Подобный подход к численному моделированию упругих волн подразумевает работу с большим количеством 3D-данных. Учитывая масштабы области при решении реальных задач (сотни километров в каждом координатном направлении с шагом дискредитации до 1 км), задача численного моделирования становится не выполнима на персональной однопроцессорной рабочей станции даже с установленным в ней графическим процессором.

Поэтому возникает повышенный интерес к использованию многоядерных и гибридных многопроцессорных вычислительных систем для достижения максимальной производительности при расчете вычислительно-емких задач подобного класса. При этом умение писать параллельные программы, способные эффективно выполняться на таких системах, граничит с искусством и требует от программиста знания не только используемых программных моделей, но и особенностей аппаратуры, на которой будет выполняться разрабатываемая параллельная программа. Ситуация особенно усложнилась с появлением широко используемых архитектур, таких как графические ускорители NVidia или сопроцессоры Intel Xeon Phi. С целью задействовать все уровни параллелизма при разработке эффективных параллельных программ одновременно должны быть использованы различные технологии параллельного программирования (MPI, OpenMP, CUDA, OpenACC, OpenCL и др.).

Еще более непростой ситуация становится при попытке отобразить ранее написанный последовательный код на параллельную архитектуру. Зачастую требуется значительное изменение подлежащих распараллеливанию участков, которые в противном случае не могут быть выполнены параллельно. Особенно важным оказывается вопрос поддержки двух версий программы: последовательной и параллельной. Изменение в последовательной версии программы (например, оптимизация существующего метода или внедрение нового метода) может повлечь за собой серьезные изменения в параллельной программе.

Разработка инструментов, способных оказывать помощь программисту при решении этих задач, а тем более разработка инструментов, способных решать указанные проблемы в автоматизированном режиме, может внести значительный вклад в развитие отрасли суперкомпьютерных вычислений и способна значительно снизить трудозатраты на распараллеливание существующих и разработку новых программ.

Автоматизированное распараллеливание полагается на возможность выявления участков программного кода, которые могут быть выполнены параллельно, и, следовательно, требует выполнения точного статического и, возможно, динамического анализов исходного кода программы, определения зависимостей по данным и по управлению, присутствующих в ней.

Применение интерактивных систем вносит значительный вклад в автоматизацию параллельного программирования [11–16]. Некоторые системы включают в свой состав автоматически распараллеливающие компиляторы [14, 15], другие ориентированы на исследование программы статическими и/или динамическими методами и предполагают, что распараллеливание программы (преобразование исходного кода, расстановка директив параллельного программирования и др.) выполняется пользователем вручную. В системе САПФОР используются средства статического и динамического анализа для отображения последовательных программ в первую очередь на кластеры. Данная система помогает перевести последовательную пользовательскую программу в параллельную программу в модели DVMH [17, 18].

МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ

Решение прямой задачи геофизики связано с решением системы уравнений теории упругости в трехмерной области. В данной работе рассматриваются упругие волны. Модель среды задается тремя параметрами: коэффициентами Ламэ λ, μ и плотностью ρ. Например, в работе [1] уравнения представлены в терминах скоростей смещений. В такой постановке необходимо работать с системой из 9 уравнений и, следовательно, с 9 параметрами на стадии численной реализации. Использование подобного подхода было бы достаточно ресурсоемким. Поэтому было принято решение использовать систему, которая описывается с помощью трех компонент вектора смещений $(U,V,W)^T$, так как такой способ является более оптимальным с точки зрения использования оперативной памяти и времени счета.

Уравнения теории упругости в смещениях

Для решения прямой задачи необходимо построить трехмерную модель рассматриваемой среды: определить размеры и форму области, а также задать параметры λ, μ, ρ для каждой из составляющих частей среды (среда может быть неоднородной).

Преобразование криволинейной сетки в структурированную сетку

В данной работе используется построение криволинейной трехмерной сетки для расчетной области. Важнейшим моментом является ортогональность ребер ячеек возле свободной поверхности: все пересекающиеся ребра каждой криволинейной ячейки возле поверхности локально-ортогональны. Это означает, что в каждой точке поверхности вертикальные ребра ячеек перпендикулярны плоскости касательной к поверхности в этой точке. В этих же точках ортогональны и ребра, соответствующие горизонтальным направлениям.

Такой подход для решения подобных динамических задач геофизики позволяет увеличить точность аппроксимации граничных условий для численного решения задачи (для случая произвольной формы поверхности с первого порядка при использовании регулярной гексаэдральной сетки до второго порядка при использовании криволинейных ячеек). Наиболее предпочтительным в данном случае оказался метод трансфинитной интерполяции [5]. После построения сетки нужно преобразовать уравнения для новой системы координат (q^1,q^2,q^3), где сетка является регулярной гексаэдральной.

Таким образом, мы получаем новый вид уравнений, на основе которых далее строится алгоритм решения задачи. Численное решение задачи производится на основе метода конечных разностей. Этот метод зарекомендовал себя хорошо для создания на его основе эффективного параллельного алгоритма [9, 10]. За основу берутся формулы из работы [8], адаптированные для трехмерного случая. Схема имеет второй порядок аппроксимации по пространству и по времени.

ОСОБЕННОСТИ РЕАЛИЗАЦИИ ПАРАЛЛЕЛЬНОЙ ВЕРСИИ ПРОГРАММЫ С ИСПОЛЬЗОВАНИЕМ MPI

Численный алгоритм построен на основе широко известного метода конечных разностей, как это было отмечено выше. Для проведения параллельных расчетов для каждого блока необходимо разместить и хранить в оперативной памяти девять трехмерных массивов со значениями для компонент смещений U,V,W, параметры среды λ, μ, ρ, а также координат узлов криволинейной сетки X,Y,Z. Были разработаны две последовательные программы для построения сетки и реализации расчетов по разностной схеме для рассматриваемой задачи с использованием языка Fortran 95, а также соответствующие им две параллельные программы с использованием технологии MPI. Обе последовательные (а также параллельные) программы были объединены в одну последовательную (параллельную) программу. Данный подход позволил строить сетку «на лету», а не считывать ее из файла, так как операции с диском выполняются достаточно долго.

Для массивов U,V и W нужно организовать обмен данными между соседними бло-

ками, так называемыми теневыми гранями. Каждый блок имеет теневые грани, в которые он принимает данные от своих соседей. Соответственно свои данные он шлет в аналогичные теневые грани своих соседей. Коммуникации осуществляются через созданную 3D-топологию с помощью средств MPI. Каждому вычислительному процессу присваивается тройка номеров – декартовы координаты внутри трехмерной топологии.

Важно отметить, что пересылки осуществляются не только между процессами, соседними по одному из координатных направлений. Обмен данными ведется также между процессами, соседствующими по всем диагональным направлениям. Общее количество пересылок в случае трехмерной процессорной решетки равно 26.

Использование метода конечных разностей позволяет осуществлять обмен данными между процессами при помощи неблокирующих пересылок MPI_Isend и MPI_Irecv. При этом используется буферный массив, куда копируются значения u, v и w, что позволяет для двух соседних блоков все данные переслать в одном сообщении вместо трех. Вычисление компонент U,V и W для каждого блока разбито на 2 независимые части: внутренняя часть блока и вычисление граней блока, которые являются теневыми гранями для соседних процессов. Данное разбиение сделано для того, чтобы быстрее запустить асинхронные обмены.

РАСПАРАЛЛЕЛИВАНИЕ ПРОГРАММЫ С ПОМОШЬЮ СИСТЕМ САПФОР И DVM

Использование статического анализатора САПФОР

Система автоматизированного распараллеливания САПФОР [14] включает набор инструментов, ориентированных на то, чтобы снизить трудозатраты прикладного программиста на разработку параллельной версии его последовательной программы. Система САПФОР позволяет выполнять распараллеливание для гетерогенных вычисли тельных кластеров в модели DVMH, в данный момент поддерживается язык Fortran-DVMH. Данная система применялась для статического исследования свойств программной реализации предложенного последовательного алгоритма, характеризирующих его информационную структуру.

Были исследованы зависимости по данным, присутствующие в программе, и выполнена их классификация по рекомендуемому способу их устранения. Полученные результаты были исследованы с помощью интерактивной диалоговой оболочки. Рассмотрим разновидности зависимостей по данным, выявление которых поддерживается в системе САПФОР.

Одним из видов зависимостей являются обратные зависимости (ANTI) и зависимости по выходу (OUTPUT), которые могут быть устранены с помощью приватизации переменных. В DVMH-языках для устранения этих зависимостей используются спецификации PRIVATE с указанием списка переменных. При выполнении программы на системах с распределенной памятью данные спецификации не требуются, так как каждый узел обладает своей отдельной памятью, но при распараллеливании внутри узла, где память является общей, или при выполнении расчетов на ускорителях отсутствие данных спецификаций приведет к гонкам данных (RACE CONDITION).

Источником приватных переменных в основном являются вычисления, локализованные в рамках одной итерации параллельного цикла, в этом случае такие переменные используются как место хранения промежуточных результатов вычислений. При большом объеме вычислений в рамках одного цикла количество промежуточных данных может оказаться значительным и затруднит ручное задание спецификации PRIVATE. Например, в параллельной DVMH-версии разработанного алгоритма количество объявленных приватных переменных в рамках одного цикла доходило до 53. Автоматическая генерация таких спецификаций, а также проверка корректности заданных программистом указаний позволяют снизить затраты на разработку и отладку параллельной программы [19]. Так, объявление в параллельном цикле переменной, которая используется только на чтение, в списке приватных переменных приведет к

ошибочным вычислениям в программе, которые обнаружить очень трудно.

Еще одним источником устранимых зависимостей являются операции редукции в циклах, например, определение максимального элемента массива или вычисление суммы всех элементов массива. Данные операции порождают прямую зависимость (FLOW) за счет того, что накапливают результат вычислений в некоторой переменной: для вычисления нового значения данной переменной необходимо ее значение, посчитанное на предыдущей итерации цикла. Особенностью редукционных операций является их ассоциативность, например, вычисление выражения (((A+B)+C)+D) можно выполнять в другом порядке (A+B)+(C+D) и делать это параллельно. Система САПФОР помогла выявить операции редукции в распараллеливаемой программе, применяемые для нахождения минимума среди множества выражений, зависящих от элементов нескольких массивов. В DVMH-языках операция редукции задается с помощью спецификации REDUCTION.

Другой интересной возможностью системы САПФОР является диагностирование регулярных зависимостей в циклах программы. К таким зависимостям относятся прямые зависимости (FLOW), расстояние которых ограничено некоторой константой, определяющей количество предшествующих итераций цикла, необходимых для выполнения данной итерации. Параллельное выполнение программ с такого вида зависимостями поддерживается на уровне DVMH-языков с помощью задания спецификации ACROSS, указывающей компилятору на необходимость организации конвейерного выполнения цикла. Статический анализ в системе САПФОР не выявил зависимостей данного вида в последовательной реализации разработанного алгоритма.

Статический анализ как этап автоматического распараллеливания обладает существенным недостатком: необходимостью принятия консервативных решений. Это означает, что если невозможно достоверно гарантировать отсутствие зависимости, с целью обеспечения корректного выполнения программы принимается решение о ее наличии. Применение статического анализа как средства диагностирования зависимостей в совокупности с интерактивной оболочкой системы САПФОР и последующим ручным распараллеливанием с помощью директив языка Fortran-DVMH позволило обойти указанную трудность. Стоит отметить, что использование ко-дизайна при разработке предложенного алгоритма обеспечило минимальное количество ложных срабатываний статического анализа при классификации зависимостей в циклах (в 6% циклов). Появление ложных зависимостей в основном было вызвано наличием операторов ветвления в программе, для которых анализ не смог определить условие перехода. В более сложных ситуациях система САПФОР допускает применение динамического анализа, который определяет отсутствие зависимостей при запуске программы на определенных наборах данных. Интерактивная оболочка системы позволяет исследовать результаты динамического анализа.

Создание параллельной программы в модели DVMH

Для распараллеливания программы в модели DVMH необходимо расставить DVMH-директивы распределения (DISTRIBUTE) и выравнивания (ALIGN) данных, а также для каждого цикла, в котором используются распределенные данные, расставить директиву PARALLEL. Директива PARALLEL позволяет отобразить итерационное пространство цикла на индексное пространство распределенного массива (выполнять виток цикла необходимо на том процессоре, где находятся локальная часть распределенного массива). При необходимости нужно добавить к директиве PARALLEL клаузы: REDUCTION для указания редукционных зависимостей, PRIVATE для указания приватных переменных, ACROSS для указания регулярных зависимостей по данным, SHADOW_RENEW для выполнения перед данным циклом теневых обменов. Более подробно пример распараллеливания простой программы рассмотрен на сайте DVM-системы [20].

Для того чтобы понять, как связаны между собой массивы, необходимо проана-

лизировать все циклы программы. DVMH-модель требует выполнения правила собственных вычислений при выполнении параллельного цикла. Правило собственных вычислений говорит о том, что процессор должен производить изменения только тех данных, которые у него имеются. Таким образом, достаточно проанализировать связь параметров цикла с индексными выражениями массивов, в которые осуществляется запись в этом цикле. По полученной информации со всех циклов можно определить, как необходимо связать массивы между собой, чтобы минимизировать обмены между процессами.

Всего в программе используются более 100 массивов. Но не все массивы можно сделать распределенными. Перевод криволинейной поверхности в структурированную сетку требует размножения некоторых вычислений (под размноженными вычислениями понимается дублирование одни и тех же вычислений разными процессорами) и данных, так как данный алгоритм использует косвенную адресацию, которая не поддерживается моделью DVMH. Массивы, для которых не заданы правила распределения или выравнивания, считаются размноженными (присутствуют на каждом процессоре в виде полной копии).

Количество распределенных массивов в данной программе составило 51. Так как программист заранее знает конфигурацию массивов, используемых в программе, можно однозначно определить выравнивание всех массивов друг на друга. Например, следующие директивы

!DVM$ DISTRIBUTE DQ1_DX (**BLOCK**, **BLOCK**, **BLOCK**)
!DVM$ ALIGN (i, j, k) **WITH** DQ1_DX(i, j, k) :: DQ2_DX, DQ3_DX
!DVM$ ALIGN (i, j, k) **WITH** DQ1_DX(2 * i, 2 * j, 2 * k) :: alambda

позволяют распределить трехмерный массив DQ1_DX равными блоками по всем трем измерениям и выровнять массивы DQ2_DX и DQ3_DX на массив DQ1_DX «один к одному» или так, чтобы элементы (i,j,k) массивов DQ2_DX и DQ3_DX располагались там же, где и элементы (i,j,k) массива DQ1_DX.

Более интересное выравнивание массива alambda – элементы (i,j,k) массива alambda должны располагаться там же, где и элементы (2 * i, 2 * j, 2 * k) массива DQ1_DX. Данное распределение позволяет выровнять массив alambda, который по каждому измерению имеет в два раза меньшие размеры, чем массив DQ1_DX.

После определения правил распределения и выравнивания, необходимо расставить директивы распределения вычислений PARALLEL во всех циклах, где используются распределенные массивы. С помощью полученной информации от САПФОР, можно расставить необходимые клаузы PRIVATE, REDUCTION и ACROSS в директиве PARALLEL. Для расстановки клаузы SHADOW_RENEW требуется проанализировать все операции чтения из распределенных массивов и определить, присутствуют ли обращения в нелокальные элементы распределенных массивов (удаленные обращения). Способ определения удаленных данных для параллельного цикла описан в примере на сайте DVM-системы [20].

После расстановки всех необходимых DVMH-директив, необходимо проверить их корректность с помощью динамического DVMH-отладчика. Для этого необходимо скомпилировать программу как параллельную в специальном режиме с отладочной информацией. После этого можно запустить полученную программу в отладочном режиме и получить диагностики в случае некорректной расстановки директивы PARALLEL с привязкой к циклу исходной программы. Данный инструмент позволяет определить корректность расстановки клауз к директиве PARALLEL. Всего в программе были распараллелены 63 цикла из 91.

И наконец, необходимо каждый параллельный цикл или группу подряд идущих параллельных циклов заключить в регион – область с одним входом и выходом, чтобы данные циклы DVMH-компилятор смог отобразить на графический процессор и на ядра центрального процессора. Для ГПУ также необходимо указать DVMH-директивы актуальности данных (ACTUAL, GET_ACTUAL), которые позволяют опре-

делить те данные, которые необходимо загрузить на ГПУ в случае выполнения на нем региона и выгрузить с ГПУ в случае необходимой их модификации на ЦПУ.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ. СРАВНЕНИЕ С РУЧНЫМ РАСПАРАЛЛЕЛИВАНИЕМ

Оценка эффективности полученной DVMH-программы и параллельной программы с использованием технологии MPI выполнялась на суперкомпьютере К100 [21], имеющем процессоры Intel Xeon X5670 и ГПУ NVIDIA Tesla C2050 с архитектурой Fermi, и на сервере с процессором Intel Xeon E5 1660 v2 и ГПУ NVIDIA GeForce GTX Titan с архитектурой Kepler. Была произведена оценка слабой и сильной масштабируемостей. Использование графических ускорителей разных поколений при тестировании DVMH-программы позволяет оценить зависимость от ограничений архитектуры Fermi (например, уменьшенное количество регистров, более быстрые операции с общей памятью и т.д.).

Каждый узел суперкомпьютера К100 содержит по два 6-ти ядерных процессора и три графических ускорителя. Два процессора связаны общей памятью (архитектура NUMA). В результате опытов было установлено, что на каждый узел для запуска DVMH-программы наиболее оптимально отображать два MPI процесса (по одному на каждый физический процессор), каждый из которых будет использовать 6 ядер ЦПУ и 1 ГПУ. Для запуска программы, написанной на MPI, на каждый узел отображалось 12 процессов. Для сравнения с программой, написанной только с использованием MPI, DVMH-программа была также запущена в режиме использования только MPI-процессов.

Как DVMH-программа, так и MPI-программа запускались с одинаковым количеством используемых ядер, только DVMH-программа запускалась и в режиме использования только MPI-процессов, и в гибридных режимах MPI+OpenMP и MPI+OpenMP+CUDA. Для последнего режима применялись средства автоматической балансировки производительности между всеми ядрами ЦПУ и ГПУ внутри каждого процесса. В силу нестабильной работы узлов кластера соотношение весов между ЦПУ и ГПУ могут отличаться от узла к узлу. Так как количество итераций, которое необходимо выполнить для окончания расчета, зависит от размера входных данных, для корректного сравнения времени работы программы была выбрана метрика количества итераций в секунду. Общее время параллельного счета на одном узле рассматриваемого ниже размера задачи примерно 3000 с.

Для замера слабой масштабируемости был выбран такой размер данных, при котором на каждое ядро приходится примерно по 2 ГБ данных. Всего на один узел использовал примерно 24 ГБ данных. Результаты слабой масштабируемости представлены в табл. 1. На одном узле было задействовано 2 ГПУ и 12 ядер ЦПУ. Всего было задействовано 480 ядер ЦПУ и 80 ГПУ (40 узлов кластера К100), размер задачи при этом составил примерно 1000 ГБ. Из табл. 1 видно, что DVMH-программа не уступает по скорости выполнения программе с ручным распараллеливанием.

Таблица 1

Слабая масштабируемость (количество итераций в секунду)

Количество узлов	1	2	4	8	20	40
DVMH (MPI)	0,86	0,85	0,85	0,82	0,82	0,82
DVMH (MPI/ OpenMP)	0,86	0,85	0,89	0,88	0,85	0,84
DVMH (MPI/ OpenMP/ CUDA)	1,56	1,53	1,50	1,51	1,48	1,49
MPI-программа	0,85	0,85	0,85	0,85	0,82	0,82

Разница между ручным распараллеливанием и DVMH-программой при использовании только ядер центрального процессора составляет не более 5% в пользу DVMH несмотря на то, что в MPI-программе используются асинхронные пересылки, а в DVMH-программе – синхронные. Использование асинхронных пересылок не дает преимущества в данной программе, так как обмены между соседними процессами занимают долю времени не более 10% по сравнению со временем основного расчета.

Добавление графических процессоров в каждом процессе позволило ускорить DVMH-программу почти в 2 раза. На данной задаче основной вычислительный цикл имеет очень много вычислений (около 4000 операций умножения, сложения и вычитания двойной точности). С помощью средств DVMH можно получить подробную статистику по использованию ресурсов ГПУ от компилятора NVidia с привязкой к циклам. Данная статистика показывает, что компилятор пытается использовать все ресурсы по максимуму. При использовании более новой архитектуры Kepler, в которой одна нить может использовать не 64 регистра, а 256 регистров, можно получить существенное ускорение DVMH-программы.

Результаты гибридных запусков с использованием разных ГПУ приведены в табл. 2. Из табл. 2 видно, что при использовании центральных процессоров разных поколений производительность только процессорных ядер практически не отличается (20% в пользу более современного процессора).

Таблица 2

Сравнение двух поколений ГПУ (количество итераций в секунду)

Вариант запуска DVMH-программы	Производительность
DVMH (K100 OpenMP 6 ядер)	0,43
DVMH (K100 GPU Fermi)	0,34
DVMH (Xeon E5 OpenMP 6 ядер)	0,52
DVMH (Xeon E5 GPU Kepler)	3,4

Но при задействовании более современного ГПУ есть возможность получить 6,5-кратное ускорение по сравнению со всеми ядрами ЦПУ и 10-кратное ускорение по сравнению с ГПУ архитектуры Fermi.

Также можно заметить, что совокупная производительность двух ГПУ и 12 ядер ЦПУ узла К100 в два раза ниже, чем производительность одного ГПУ архитектуры Kepler (табл. 1). Но несмотря на низкую производительность ГПУ в узле суперкомпьютера К100, с помощью автоматической балансировки между доступными ядрами и графическими ускорителями внутри узла удается задействовать все доступные ресурсы.

Для оценки сильной масштабируемости была запущена задача на 1, 2, 4, 6 и 12 потоках 6-ти ядерного процессора Intel Xeon E5. Данные результаты представлены в табл. 3. Из табл. 3 видно, что при использовании 12 потоков получается 100% эффективность использования ресурсов, если сравнивать с параллельной программой, выполненной на 1 потоке. Если сравнивать с последовательной программой, то эффективность при использовании 6 ядер составляет 96%.

Таблица 3

Сильная масштабируемость (количество итераций в секунду)

Количество потоков	1	2	4	6	12
DVMH (Xeon E5 OpenMP)	0,08	0,15	0,31	0,45	0,52
Последовательная	0,09	N	N	N	N

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены возможности применения систем САПФОР [14] и DVM [17] для автоматизации разработки параллельных программ на примере программы численного моделирования 3D-сейсмических полей в упругих средах для областей со сложной геометрией свободной поверхности. Важной особенностью данной задачи является построения криволинейной 3D-сетки, локально-ортогональной возле свободной поверхности, а также учет особенностей различных параллельных архитектур при проектировании последовательной версии алгоритма.

Благодаря такому подходу к проектированию стало возможным применение средств автоматизированного анализа и распараллеливания программы. С помощью средств САПФОР были получены результаты анализа, которые в дальнейшем позволили без особых усилий расставить директивы языка Fortran-DVMH. Помимо средств анализа были использованы встроенные средства анализа эффективности программ с помощью DVMH-интервалов, использовались средства динамической отладки DVMH-указаний для определения корректности расстановки всех директив в программе. В силу того, что программа состоит более, чем из 4000 строк, ручная проверка DVMH-указаний очень затруднительна. Была использована сравнительная отладка между ЦПУ и ГПУ, чтобы убедиться в корректности сгенерированного DVMH-

компилятором кода для графического процессора.

Проведенные тесты показали хорошие, почти линейные результаты сильной и слабой масштабируемостей работы как DVMH-программы, так и MPI-программы. DVMH-программа при прочих равных условиях не уступает MPI-программе. Можно сделать вывод, что разработанная с помощью инструментов САПФОР и DVM параллельная программа не уступает по эффективности программе с использованием MPI, и при этом позволяет эффективно использовать все доступные ресурсы вне зависимости от используемого кластера путем автоматической балансировки нагрузки, реализованной в системе поддержки DVMH. При этом размер исходной программы увеличился всего на 20 DVMH-директив распределения данных и 50 DVMH-директив распределения вычислений. Также реализованную DVMH-программу можно одинаково эффективно отобразить на разные архитектуры, в том числе и на Intel Xeon Phi без каких-либо модификаций исходного кода программы.

Рассмотренные инструменты для автоматизации распараллеливания могут существенно снизить трудозатраты на получение эффективных программ, которые способны отображаться на различные архитектуры, а так же помочь в разработке и оптимизации масштабируемых алгоритмов для вычислительных систем экзафлопсного уровня.

Исходные коды разработанных программ доступны по ссылке [22].

СПИСОК ЛИТЕРАТУРЫ

1. **Численное** моделирование и экспериментальное исследование грязевого вулкана «Гора Карабетова» вибросейсмическими методами / Глинский Б. М. [и др.] // Вычислительные методы и программирование. 2010. Т. 11. С. 95–104. **[** B. M. Glinskij, et al., “Numerical modeling and experimental study of the mud volcano "Mountain Karabetova" by vibroseismic methods,” (in Russia), *in Vychislitelnye metody i programmirovanie*, vol. 11, pp. 95-104, 2010. **]**

2. **Graves R. W.** Simulating seismic wave propagation in 3D elastic media using staggered grid finite differences // Bulletin of the Seismological Society of America. 1996. Vol.86(4). P. 1091–1106. **[** R. W. Graves, “Simulating seismic wave propagation in 3D elastic media using staggered grid finite differences”, *in Bulletin of the Seismological Society of America*, vol. 86, no 4. pp. 1091-1106, 1996. **]**

3. **Virieux J.** P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method // Geophysics. 1986. Vol. 51. No. 4, P. 889–901. **[** J. Virieux. “P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method,” *in Geophysics*, vol. 51, no. 4, pp. 889-901, 1986. **]**

4. **Levander A. R.** Fourth-order finite-difference P-SV seismograms // Geophysics. 1988. Vol. 53. Issue 11. P. 1425–1436. **[** A. R.Levander, “Fourth-order finite-difference P-SV seismograms,” in Geophysics, vol. 53, no 11, pp. 1425-1436, 1988. **]**

5. **Лисейкин В. Д.** Разностные сетки. Теория и приложения. Новосибирск: СО РАН, 2014, 254 с. **[** V. D. Lisejkin, *Difference grids. Theory and applications*, (in Russian). Novosibirsk: Institute of Computational Mathematics and Mathemati-cal Geophysics, 2014. **]**

6. **Хакимзянов Г. С., Шокин Ю. И.** Разностные схемы на адаптивных сетках. Новосибирск: Редакционно-издательский центр НГУ, 2005, 130 с. **[** G. S. Hakimzjanov, U. I. Shokin, Difference schemes on adaptive grids, (in Russia), Novosibirsk: Novosibirsk State University, 2005. **]**

7. **Шокин Ю. И.** Лекции по разностным схемам на подвижных сетках К.: Редакционно-издательский центр Каз-НУ им. аль-Фараби, 2006, 183 с. **[** U. I. Shohin, *Lectures on the difference scheme on moving grids*, (in Russian), Kazan: Editorial and Publishing Council Al-Farabi Kazakh National University, 2006. **]**

8. **Appelo D., Petersson N. A**. A Stable Finite Difference method for the Elastic wave equation on complex geometries with free surfaces // Communications in Computational Physics. 2009. Vol.5, No. 1, P. 84–107. **[** D. Appelo, N. A. Petersson, “A Stable Finite Difference method for the Elastic wave equation on complex geometries with free surfaces*,” in Communications in Computational Physics*, vol. 5, no. 1, pp. 84-107, 2009. **]**

9. **Komatitsch D., Erlebacher G., Goddeke D.** High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster // Journal of Computational Physics. 2010 Vol. 229, No 20. P. 7692–7714. **[** D. Komatitsch, G. Erlebacher, D. Goddeke, “High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster,” *in Journal of Computational Physics*, vol. 229, no 20, pp. 7692-7714, 2010. **]**

10. **Karavaev D. A., Glinsky B. M., Kovalevsky V.V.** A technology of 3D elastic wave propagation simulation using hybrid supercomputers // CEUR Workshop Proceedings 1st Russian Conference on Supercomputing Days. 2015. Vol. 1482. P. 26–33. **[** D. A. Karavaev D.A., B. M. Glinsky, V. V. Kovalevsky, “A technology of 3D elastic wave propagation simulation using hybrid supercomputers,“ (in Russian), *in CEUR Workshop Proceedings 1st Russian Conference on Supercomputing Days*, vol. 1482, pp. 26-33, 2015. **]**

11. **Intel** Parallel Studio [Электронный ресурс]. URL: http://software.intel.com/en-us/intel-parallel-studio-home (дата обращения 1.01.2017). **[***Intel Parallel Studio* [Online]. Available: http://software.intel.com/en-us/intel-parallel-studio-home**]**

12. **Liao S., Diwan A., Bosch R.** Suif Explorer: an Interactive and Interprocedural Parallelizer // Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 1999. P. 37–48. **[** S. Liao, A. Diwan., R. Bosch, “ Suif Explorer: an Interactive and Interprocedural Parallelizer,“ in Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 37-48, 1999. **]**

13. **Sudhakar S., Vinay G.** Vaidya Review of Parallelization Tools and Introduction to Easypar // International Journal of Computer Applications. Vol. 56, No. 12. 2012. P. 17–29. **[** S. Sudhakar, G. Vinay, "Vaidya Review of Parallelization Tools and Introduction to Easypar," International Journal of Computer Applications, vol. 56, no. 12, pp. 17-29, 2012. **]**

14. **Диалог** с программистом в системе автоматизации распараллеливания САПФОР / Бахтин В.А. [и др.] // Вестник Нижегородского университета им. Н.И. Лобачевского. №5 (2), 2012. С. 242–245. **[** V. A. Bakhtin. [et al.], " Dialogue with the programmer in the automation of parallelization SAPFOR," (in Russian), *in Vestnik NNGU*, no (2), pp. 242-245, 2012. **]**

15. **Диалоговый** высокоуровневый оптимизирующий распараллеливатель (ДВОР) / Юрушкин М. В. [и др.] // Труды Международной суперкомпьютерной конференции Научный сервис в сети Интернет. Новоросийск .2010. С. 71–75. [M. V. Jurushkin [at al.], " Interactive High-level Optimizing Parallelizer," (in Russian), *in Proc. Proceedings of the International Scientific Conference, Novorossijsk, Russia, 2010, pp. 71-75.* **]**

16. **ParaWise** – Widening Accessibility to Efficient and Scalable Parallel Code. // Parallel Software Products White Paper WP-2004-01, 2004. **[** ParaWise – Widening Accessibility to Efficient and Scalable Parallel Code, *in Parallel Software Products White Paper WP-2004-01*, 2004. **]**

17. **Fortran-DVM** – язык разработки мобильных параллельных программ / Коновалов Н. А. [и др.] // Программирование, № 1. 1995. С. 49–54. **[** N. A. Konovalov [et al.]**, "**Fortran-DVM – язык разработки мобильных параллельных программ", (in Russian), *in Programmirovanie*, no. 1, pp. 49-54, 1995. **]**

18. **Расширение** DVM-модели параллельного программирования для кластеров с гетерогенными узлами / Бахтин В. А. [и др.] // Вестник Южно-Уральского государственного университета, №18 (277), выпуск 12. 2012. С. 82–92. **[** V. A. Bahtin [et al.], "Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами," (in Russian), *in Vestinik SUSU*, vol. 8 (277), no. 12, pp. 82-92, 2012. **]**

19. **Катаев Н. А.** Статический анализ последовательных программ в системе автоматизированного распараллеливания САПФОР // Вестник Нижегородского университета им. Н.И. Лобачевского, N5(2). 2012. С. 359–366. **[** N. A. Kataev, " Static analysis of sequential programs in the automatic parallelization environment SAPFOR, ", (in Russian), *in Vestnik NNGU*, vol. 5(2), pp. 359-366, 2012. **]**

20. **Пример** распараллеливания программы в модели DVMH. [Электронный ресурс]. URL: http://dvm-system.org/ru/examples/ (дата обращения: 1.06.2017). **[** Exmaple of program parallelization using DVMH-model. [Online]. Available: http://dvm-system.org/ru/examples/ **]**

21. **Гибридный** вычислительный кластер К-100 [Электронный ресурс]. URL: http://www.kiam.ru/MVS/resourses/k100.html (дата обращения: 1.06.2017). **[** eterogeneous computational cluster K100 [Online]. Available: http://www.kiam.ru/MVS/resourses/k100.html **]**

22. **Исходные** коды. [Электронный ресурс]. URL: https://bitbucket.org/dvm-system/elastic-wave-3d (дата обращения: 1.06.2017). **[** Source code**.** [Online]. Available: https://bitbucket.org/dvm-system/elastic-wave-3d **]**

ОБ АВТОРАХ

КАТАЕВ Никита Андреевич, н.с. ИПМ им. М.В Келдыша. Дипл. прикладной математик-информатик (МГУ, 2009). Готовит дис. об анализе и преобразовании последовательных программ для их распараллеливания.

КОЛГАНОВ Александр Сергеевич, аспирант каф. Системного программирования МГУ. Дипл. прикладной математик-информатик (МГУ, 2014). Готовит дис. об экспертной системе автоматизированного распараллеливания последовательных программ.

ТИТОВ Павел Андреевич, м.н.с ИВМиМГ СО РАН. Готовит дис. об использовании криволинейных сеток для моделирования сейсмополей в областях, характерных для магматических вулканов.

METADATA

Title: Automated parallelization of a simulation method of elastic wave propagation in media with complex 3D geometry surface on high-performance heterogeneous clusters.

Authors: N. A. Kataev[1], A. S. Kolganov[1,2], P. A. Titov[3]

Affiliation:

[1] Keldysh Institute of Applied Mathematics (KIAM) RAS, Russia.

[2] Lomonosov Moscow State University (MSU), Russia.

[3] Institute of Computational Mathematics and Mathematical Geophysics SB (ICMMG) RAS, Russia.

Email: [1]kaniandr@gmail.com, [2]alexander.k.s@mail.ru, [3]tapawel@gmail.com.

Language: Russian.

Source: Vestnik UGATU (scientific journal of Ufa State Aviation Technical University), vol. 21, no. 3 (77), pp. 87-96, 2017. ISSN 2225-2789 (Online), ISSN 1992-6502 (Print).

Abstract: The paper considers application of DVM and SAPFOR in order to automate mapping of 3D-elastic waves simulation method on high-performance heterogeneous clusters. A distinctive feature of the proposed method is the use of a curved three-dimensional grid, which is consistent with the geometry of free surface. Usage of curved grids considerably complicates both manual and automated parallelization. Technique to map curved grid on a structured grid has been presented to solve this problem. The sequential program based on the finite difference method on a structured grid, has been parallelized using Fortran-DVMH language. Application of SAPFOR analysis tools simplified this parallelization process. Features of automated parallelization are described. Authors estimate efficiency and acceleration of the parallel program and compare performance of the DVMH based program with a program obtained after manual parallelization using MPI programming technology.

Key words: Automation of parallelization; heterogeneous computational cluster; 3D-modeling; curvilinear grid; GPU; Xeon Phi.

About authors:

KATAEV, Nikita Andreevich, Postgrad. (PhD) Student, Dept. of System Programming. Master of Informatics & Mathematics (MSU, 2009).

KOLGANOV, Alexander Sergeevich, Postgrad. (PhD) Student, Dept. of System Programming. Master of Informatics & Mathematics (MSU, 2014).

TITOV, Pavel Andreevich, Postgrad. (PhD) Student, Dept. of System Programming. Master of Informatics & Mathematics (NGU, 2013).