Применение современных технологий для высокопроизводительных вычислительных систем для решения задач локальной и глобальной сейсмики *

Н.И. Хохлов, И.Б. Петров

Московский физико-технический институт (государственный университет)

В данной работе рассмотрены вопросы распараллеливания программного комплекса, предназначенного для моделирования задач распространения динамических волновых возмущений в твердых телах, с применением различных современных технологий для высокопроизводительных вычислительных систем. Программный комплекс поддерживает двумерные и трехмерные структурные блочные сетки, явное задание неоднородностей и явное выделение контактных границ. Для численного интегрирования реализованы сеточно-характеристические и конечно-объемные методы повышенного порядка точности. Алгоритм распараллелен используя технологии MPI, CUDA, OpenMP и OpenCL. Описаны некоторые аспекты оптимизации кода с использованием потоковых SIMD инструкций центральных процессоров SSE и AVX. Приведены сравнительные тесты ускорения для рассмотренных технологий. В качестве примера работы программного комплекса приводятся результаты серии тестовых расчетов.

1. Введение

Численное моделирование распространения динамических волновых возмущений в твердых телах применяется при решении широкого круга задач. К таким задачам относятся задачи сейсморазведки, сейсмики, сейсмостойкости и прочностные задачи. Роль численного моделирования в каждой из данных областей очень важна. Численное моделирования распространения сейсмических волн представляет существенную часть работ при проведении геологоразведки в нефтяной отрасли. Математическое моделирование проводится в различных геологических средах, в том числе в слоистых средах и в средах с наличием неоднородностей (например, трещины или каверны). Задачи такого рода представляются очень ресурсоёмкими с точки зрения вычислительных ресурсов. Область вычисления, как правило, представляет собой сейсмический куб с длинной ребра от 1 км до 10 км. В тоже время, неоднородности могут быть размером в несколько метров. При моделировании задач сейсмостойкости также приходится сталкиваться с множеством неоднородностей. Параметры зданий, такие как толщина стен, размеры проемов намного меньше размеров расчетных областей, включающих в себя порой большие массивы породы, размерами более 10 км вдоль одного направления. Таким образом, расчетная сетка должна быть достаточно подробной, чтобы иметь возможность правильно выделить все неоднородности. Для получения достаточной точности расчета и учета большого числа неоднородностей требуется использование больших вычислительных сеток, в реальных расчетах используются сетки размерами до нескольких десятков миллиардов узлов.

В данной работе рассматривается созданный программный комплекс для моделирования задач распространения динамических волновых возмущений в твердых телах. Комплекс работает на двумерных и трехмерных структурных блочных сетках с наличием неоднородностей. Для численного интегрирования применяются сеточно-характеристические [1] и конечно-объемные методы 2-4 порядка точности.

*Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 15-37-20673 мол_а_вед и гранта Президента РФ МК-3383.2014.9.

Код распараллелен используя различные современные технологии для высокопроизводительных вычислительных систем. В настоящее время достигнута эффективность распараллеливания до 70 % используя технологию MPI при масштабировании до 16 тысяч вычислительных ядер. В системах с общей памятью алгоритм распараллелен используя технологию OpenMP. Также код распараллелен используя технологию CUDA, что дает ускорение до 50 раз по сравнению с одним ядром CPU. Программа может использовать несколько карточек в рамках одного узла. Для графических процессоров отличных от семейства карточек NVidia, код распараллелен используя технологию OpenCL. Что дает ускорение до 50 раз на графических ускорителях от AMD.

В данной работе рассмотрены результаты одного и того же алгоритма используя различные технологии. Приведены тесты распараллеливания до 16 тысяч ядер CPU и четырех устройств CUDA. Приведены результаты тестовых расчетов.

2. Математическая модель

2.1. Определяющие уравнения

Сформулируем основные уравнения линейной динамической теории упругости, которым подчиняется состояние бесконечно малого объема линейно-упругой среды. Рассмотрим нестационарные уравнения теории упругости для случая трех переменных, в некоторой ортонормированной системе координат (x_1, x_2, x_3):

$$\begin{aligned} \rho \dot{\upsilon}_i &= \nabla_j \sigma_{ij}, \\ \dot{\sigma}_{ij} &= q_{ijkl} \dot{\epsilon}_{kl} + F_{ij}, \end{aligned} \tag{1}$$

здесь ρ – плотность среды, υ_i – компоненты вектора скорости смещения, σ_{ij} и ϵ_{ij} – компоненты тензоров напряжений Коши и деформации, ∇_j – ковариантная производная по j-й координате, F_{ij} – добавочная правая часть. Вид компонент тензора четвертого порядка q_{ijkl} определяется реологией среды. Для линейно-упругого случая они имеют вид:

$$q_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}).$$

В этом соотношении, которое обобщает закон Гука, λ и μ – параметры Ламе, а δ_{ij} – символ Кронекера.

Первая строка в системе уравнений (1) представляет три уравнения движения, вторая - шесть реологических соотношений. Вектор искомых функций, состоящий из 9-ти компонент имеет вид:

$$\mathbf{u} = \{\upsilon_1, \upsilon_2, \upsilon_3, \sigma_{11}, \sigma_{12}, \sigma_{13}, \sigma_{22}, \sigma_{23}, \sigma_{33}\}^T.$$

Тогда перечисленные модели твердого тела допускают запись системы уравнений (1) динамики деформируемого твердого тела в матричном виде [2]:

$$\frac{\partial \mathbf{u}}{\partial t} = \sum_{j=1}^{3} \mathbf{A_j} \frac{\partial \mathbf{u}}{\partial x_j}, \tag{2}$$

где $\mathbf{A}_j$ - матрицы размера 9×9.

2.2. Численные методы

Для численного моделирования задач динамики деформируемого твердого тела широко применяется сеточно-характеристический метод [1]. Вначале применяется метод расщепления по пространственным координатам, в результате чего имеем три одномерных системы:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{A_j} \frac{\partial \mathbf{u}}{\partial x_j}, \; j = 1, 2, 3. \tag{3}$$

Каждая из этих систем является гиперболической и обладает полным набором собственных векторов с действительными собственными значениями, поэтому каждую из систем можно переписать в виде:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{\Omega_j^{-1}} \mathbf{\Lambda_j} \mathbf{\Omega_j} \frac{\partial \mathbf{u}}{\partial x_j},$$

где матрица $\mathbf{\Omega_j}$ - матрица составленная из собственных векторов, $\mathbf{\Lambda_j}$ - диагональная матрица, элементами которой являются собственные значения. Для всех координат матрица $\mathbf{\Lambda}$ имеет вид:

$$\mathbf{\Lambda} = diag\{c_p, -c_p, c_s, -c_s, c_s, -c_s, 0, 0, 0\},$$

где $c_p = \sqrt{(\lambda + 2\mu)/\rho}$ - продольная скорость звука в среде, $c_s = \sqrt{\mu/\rho}$ - поперечная скорость звука.

После замены переменных $\nu = \mathbf{\Omega u}$ каждая из систем (3) распадается на девять независимых скалярных уравнений переноса (индекс j далее опускается, где это возможно):

$$\frac{\partial \nu}{\partial t} + \mathbf{\Lambda} \frac{\partial \nu}{\partial x} = 0.$$

Одномерные уравнения переноса решаются с помощью метода характеристик, либо обычными конечно-разностными схемами.

После того, как все компоненты ν перенесены, восстанавливается само решение:

$$\mathbf{u^{n+1}} = \mathbf{\Omega^{-1}} \nu^{\mathbf{n+1}}.$$

В данной работе использовались TVD-разностные схемы [3] 2-го порядка точности. В программе реализовано 15 различных лимитеров [4], в расчетах в основном использовался ограничитель MC [3].

Также использовались сеточно-характеристические монотонные разностные схемы, принцип построения которых описан в [1]. В программе реализованы схемы от второго до четвертого порядка точности, большинство расчетов проводилось используя схему 4-го порядка точности. Приведем ее для численного решения одномерного линейного уравнения упругости $u_t + au_x = 0, \sigma = a\tau/h$, τ – шаг по времени, h – шаг по координате:

$$u_m^{n+1} = u_m^n - \sigma(\Delta_1 - \sigma(\Delta_2 - \sigma(\Delta_3 - \sigma\Delta_4))),$$

$$\Delta_1 = \tfrac{1}{24}(-2u_{m+2}^n + 16u_{m+1}^n - 16u_{m-1}^n + 2u_{m-2}^n),$$

$$\Delta_2 = \tfrac{1}{24}(-u_{m+2}^n + 16u_{m+1}^n - 30u_m^n + 16u_{m-1}^n - u_{m-2}^n),$$

$$\Delta_3 = \tfrac{1}{24}(2u_{m+2}^n - 4u_{m+1}^n + 4u_m^n - 2u_{m-2}^n),$$

$$\Delta_4 = \tfrac{1}{24}(u_{m+2}^n - 4u_{m+1}^n + 6u_m^n - 4u_{m-1}^n + u_{m-2}^n).$$

Кроме того, используется сеточно-характеристический критерий монотонности [1].

3. Программный комплекс

Для моделирования задач распространения волновых возмущений в гетерогенных средах был создан программный комплекс, позволяющий численно решать поставленные задачи, используя вышеописанные методы.

При создании программного комплекса одним из основных требований являлась возможность проведения расчетов на достаточно больших расчетных сетках (более 10 млрд узлов). В этом случае возникают значительные потребности, как в ресурсах процессорного времени, так и оперативной памяти ЭВМ. Далее рассмотрим ряд технологий, позволяющих существенно ускорить время работы вычислительного модуля.

3.1. Работа с потоковыми инструкциями SSE и AVX

Рассмотрим базовую организацию работы расчетного алгоритма. Используются явные сеточные методы, переход от одного шага по времени к следующему происходит путем обхода всей расчетной сетки и вычисления значения в новом узле исходя из значения в исходном узле и некоторой его окрестности, называемой шаблоном разностной схемы. Обход сетки организуется путем двух вложенных циклов, причем при правильной организации обхода можно хранить одновременно в памяти только один временной слой. Обновление происходит последовательно в данном слое. Исследование такой реализации показало, при достаточно большом размере расчетной сетки возникает определенное число промахов кеша процессора, что ведет к увеличению времени работы программы. Также применение потоковых инструкций SSE и AVX требует эффективного использования кеш-памяти процессоров. Простейшим решением является изменение организации обхода цикла и хранение двух временных слоев, однако это в два раза увеличивает затраты по памяти. В данной работе применен алгоритм, который позволяет дублировать только небольшие объемы памяти. Если N – число узлов в сетке, то для двумерного случая объем дублированных данных равен $\sqrt{N}$, для трехмерного – $\sqrt[3]{N}$. Путем такого изменения алгоритма обхода расчетной сетки удалось оптимизировать работу с памятью, что позволило в ряде случаев ускорить работу алгоритма на больших расчетных сетках.

Рассмотрим ускорение за счет использования потоковых SIMD-расширений центрального процессора (SSE и AVX). SSE включает в архитектуру несколько расширенных регистров (8 на современных процессорах) размером 128 бит, либо 256 бит в случае технологии AVX. Преимущество в производительности достигается в том случае, если необходимо произвести одну и ту же последовательность действий над разными данными. В таком случае блоком SIMD осуществляется распараллеливание вычислительного процесса по данным.

Для возможности оперировать данными в SSE регистрах потребовалось изменение формата хранения расчетной сетки. Чтобы произвести какие-либо операции используя SSE регистры требуется поместить в регистр необходимые расчетные данные, в случае когда эта операция занимает много времени все преимущество потоковых инструкций теряется. В связи с эти возникает необходимость быстрой возможности загрузки и выгрузки данных в SSE регистры. Изначально данные хранились в виде массива структур, или AOS (Array of structures), при таком подходе необходимо по одному числу типа float или double копировать данные в SSE регистры, затем производить вычисление и копировать данные обратно. При этом переменные, предназначенные для использования в одном SSE регистре будут расположены в памяти не последовательно.

Для ускорение данного процесса необходимо расположить все компоненты структуры в памяти последовательно, и затем подгружать их по необходимости в SSE регистры. Был произведен переход к хранению структуры массивов, или SOA (Structure of arrays), что позволило быстрее осуществлять копирование в SSE регистры.

Тестирование проводилось на процессорах Intel и AMD, компиляторы icc и gcc:

- Intel Xeon E5-2697, gcc-4.4.7, icc-15.0.0;
- AMD Opteron 6272, gcc-4.4.7.

На рис. 1 показано время работы различных версий алгоритма на одних и тех же данных, для каждого из процессоров время нормировано на базовую реализацию.

В конечной реализации векторизация реализована при помощи расширения OpenMP 4.0 [7] (#pragma omp simd), поэтому результат представлен только для компилятора icc.

Дополнительно исследована производительность работы алгоритма в FLOPS от пиковой производительности процессора. На рис. 2 приведен результат такого исследования.

Запуск производился на одном ядре процессора, число FLOPS работы алгоритма было рассчитано теоретически. Как видно на процессоре Intel производительность достигает 22%

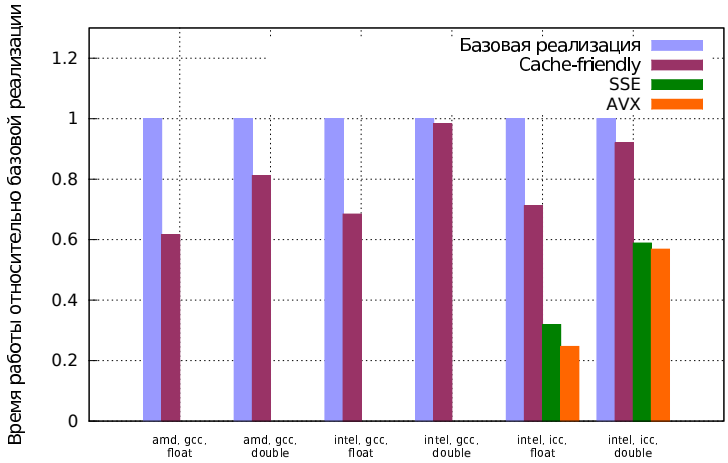

Рис. 1. Работа различных вариантов оптимизации, время относительно базовой реализации

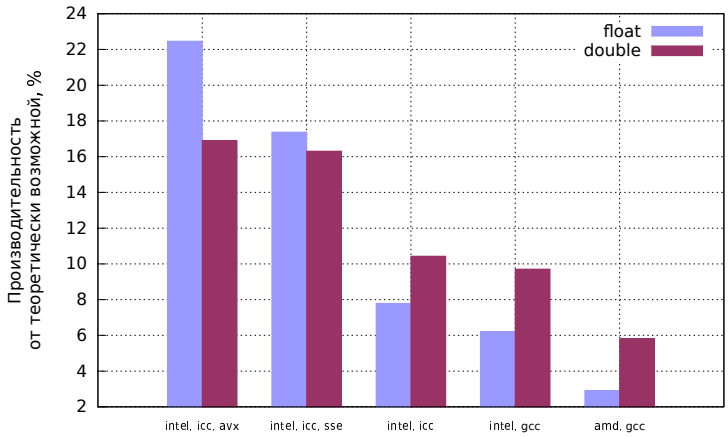

Рис. 2. Работа различных вариантов оптимизации, производительность от пиковой производительности одного ядра процессора

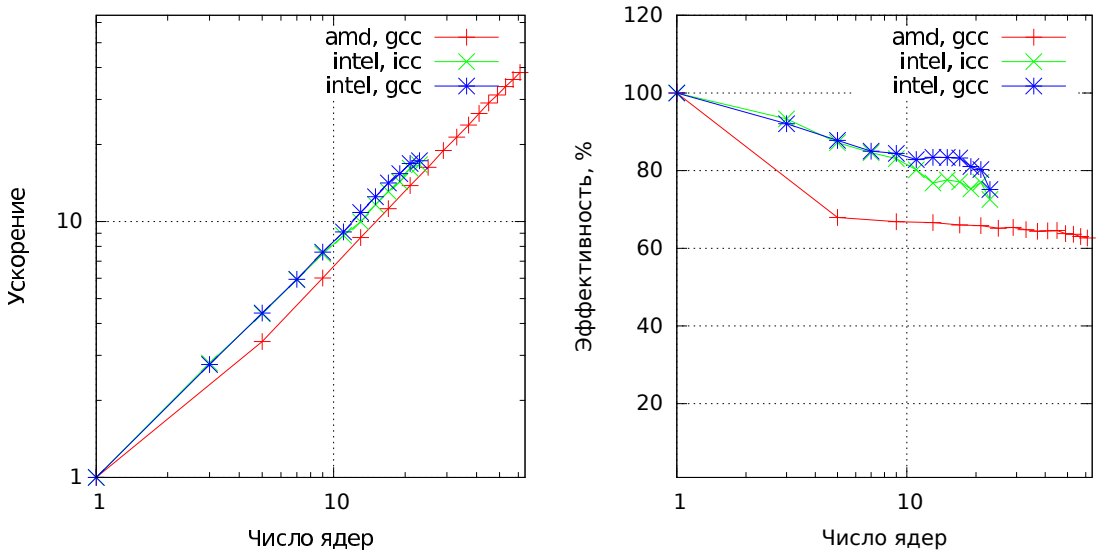

Рис. 3. Графики ускорение и эффективности от числа ядер для OpenMP реализации

от пиковой, что говорит о достаточно хорошей реализации алгоритма.

3.2. Распараллеливание OpenMP

Алгоритм также распараллелен в системах с общей памятью используя технологию OpenMP. Исследовались несколько вариантов распараллеливания. В первой реализации алгоритм распараллелен используя директиву #pragma omp for с различными наборами опций. Данная реализация показала достаточно хорошее ускорение. Дальнейшие исследования позволили ускорить первоначальную реализацию.

Согласно стандарту OpenMP при первом обращении потока к аллоцированной памяти, OpenMP будет стараться выделить ее в памяти того ядра, на котором исполняется поток. Если затем потоки, работающие на других ядрах, будут к ней обращаться, то время обращения может быть больше, чем обращение к своей памяти. Для увеличения эффективности распараллеливания перед началом работы основного вычислительного цикла программы, потоки выделяют у себя блоки, равные размеру их части расчетной сетки. При этом требуются более сложные операции копирования узлов между потоками, но благодаря тому, что их количество мало по сравнению с размерами сетки, эти накладные расходы оказываются несущественными по сравнению с другими операциями. Более быстрое обращение потоками в свою память увеличивает эффективность распараллеливания. Такая реализация показала самое большое ускорение.

При расчете на большом числе ядер важен также такой параметр, как закрепление потока за физическим ядром процессора (CPU affinity). Тесты показали, что в ряде случаев этот параметр существенно влияет на эффективность работы на большом числе ядер.

Результат распараллеливания финальной реализации приведен на рис. 3.

Конечный результат для процессора Intel – 16.7 раза при компиляторе icc и 17.3 раза при компиляторе gcc на 24 ядрах, для процессора AMD – 38 раз на 64 ядрах.

3.3. Распараллеливание MPI

Для работы в системах с распределенной памятью программный комплекс распараллелен используя технологию MPI. При распараллеливании применялся классический алгоритм для явных сеточных методов, основанный на принципе геометрического параллелиз-

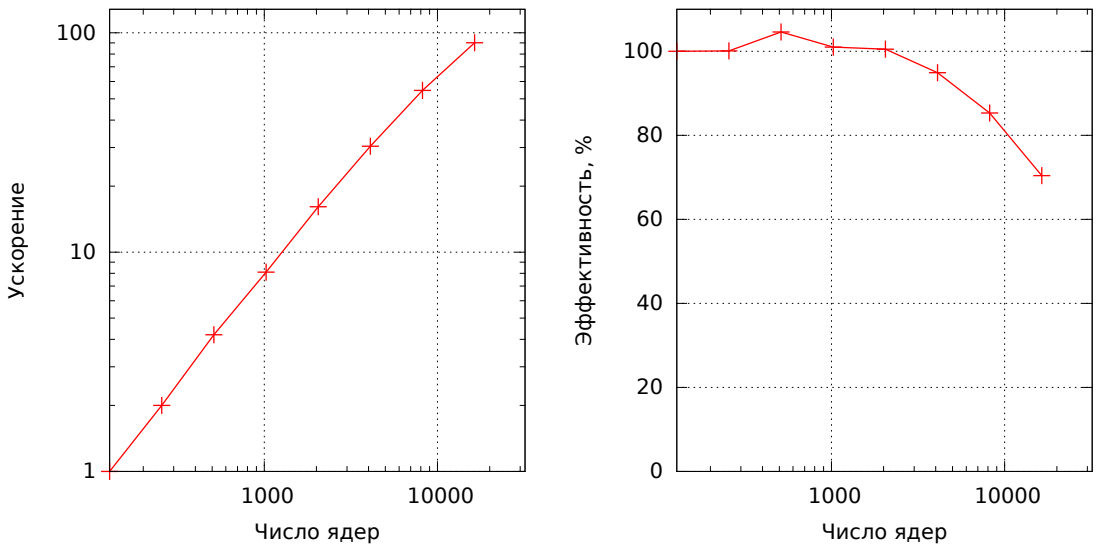

Рис. 4. Графики ускорения и эффективности MPI реализации

ма [8]. Расчетная сетка разделялась между процессами на возможно более равные части с перекрытием равным половине ширины шаблона разностной схемы. На каждом шаге происходит обмен данными из приграничных ячеек. Обмер организован используя функции MPI Isend/Irecv.

Одно из основных требований к алгоритму – работа на большом числе вычислительных ядер (тысячи) для обеспечения приемлемого времени расчета на больших задачах. На рис. 4 приведены результаты тестирования ускорения и эффективности алгоритма при увеличении числа расчетных ядер от 128 до 16384.

В тесте использовалась расчетная сетка размером $1000 \times 1000 \times 1000$ узлов. Тестирование проводилось на кластере HECToR. Данный суперкомпьютер состоит из 2816 вычислительных узлов. Каждый из узлов оснащен двумя процессорами 16-core AMD Opteron 2.3GHz Interlagos. Оперативная память составляет 32 ГБ на узел. Ускорение составляет 90 раз при увеличении числа вычислительных ядер в 128 раз.

Тестирование показало, что эффективность составляет около 70 %, что является хорошим результатом для такого количества вычислительных ядер.

Еще один тест – увеличение размеров задачи при одновременном увеличении числа ядер. В данном тесте на каждое вычислительное ядро приходится одинаковое число узлов расчетной сетки. На рис. 5 приведены результаты такого тестирования.

Тестирование проводилось при изменении числа вычислительных ядер от 1 до 4096. Максимальный размер сетки в данном тесте – около 33 млрд узлов. Как можно видеть, алгоритм показывает хорошее ускорение и позволяет проводить расчеты на больших расчетных сетках.

3.4. Распараллеливание на графических GPU процессорах

Алгоритм также был распараллелен на графических GPGPU процессорах NVidia используя технологию CUDA [5]. Потребовалось полное переписывание части расчетного модуля под данную архитектуру. Однако это дало существенное ускорение работы алгоритма, по сравнению с CPU версией. Так применение данной технологии позволило получить ускорение до 44 раз по сравнению с одним ядром CPU Intel Xeon E5-2697. Использование нескольких устройств GPU позволяет получить дополнительное ускорение.

Помимо технологии CUDA алгоритм также был распараллелен используя технологию

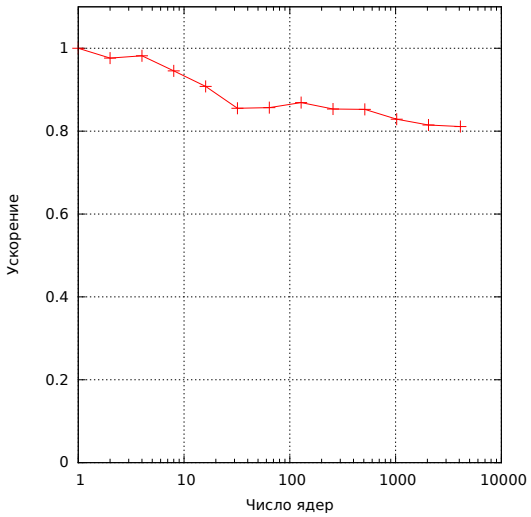

Рис. 5. Ускорение параллельного алгоритма при увеличении размера задачи на MPI

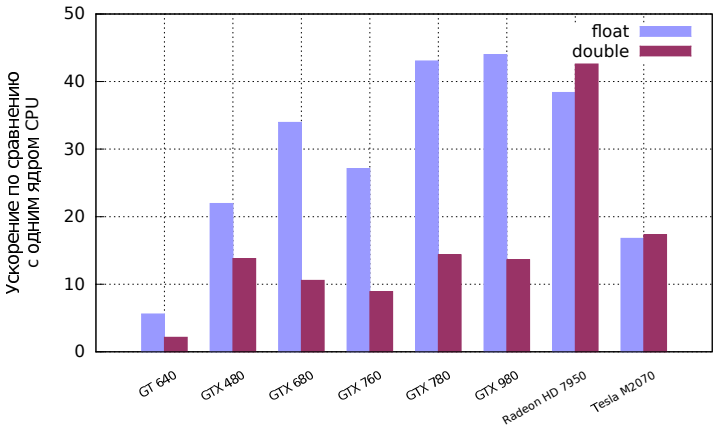

Рис. 6. Ускорение на графических устройствах

OpenCL [6], что позволило использовать для ускорения расчетов графические ускорители от AMD.

Результат ускорения по сравнению с одним ядром CPU приведен на рис. 6.

При этом для устройства от AMD (Radeon HD 7950) реализация была на OpenCL, для других графических ускорителей на CUDA. Разница скорости работы на устройствах от NVidia реализация на CUDA и OpenCL незначительна. На устройствах для настольных компьютеров от NVidia ускорение значительно падает при переходе от float к double, это связано с особенностью архитектуры. На карте Tesla M2070 такого резкого падения в скорости работы не наблюдается. Следует заметить, что графический ускоритель для настольных компьютеров от AMD Radeon HD 7950 лишен этого недостатка и дает примерно одинаковое ускорение на числах с двойной и одинарной точностью.

Результат ускорения работы на нескольких GPU устройствах приведен на рис. 7. Для обмена данными между устройствами используется технология CUDA 6 обмена данными минуя память хоста. Это позволило получить существенное ускорение на нескольких устройствах, по сравнению с работой на одном. Кроме того, работа на нескольких устройствах позволяет считать большие задачи, за счет распределения расчетной сетки между

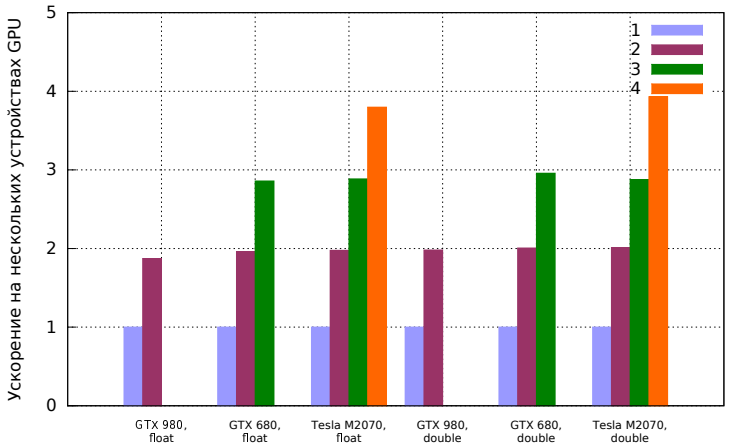

Рис. 7. Ускорение на нескольких графических устройствах

устройствами.

4. Результаты расчетов

Приведем некоторые результаты тестовых расчетов, выполненных на созданном программном комплексе.

4.1. Моделирование распространения возмущений от гипоцентра землетрясения

В данной работе исследуется процесс распространения упругих волн, возникающих в процессе землетрясения в гетерогенных средах.

Для моделирования очага землетрясения была выбрана сдвиговая модель возмущения в гипоцентре. В этой модели задавалась прямоугольная область шириной 40 м и длиной 500 м с некоторой ненулевой скоростью. Одна часть области двигается в одну сторону, другая в - противоположную. Такая модель физически соответствует ситуации, когда в земной коре существует разлом, по которому происходит подвижка при землетрясении. В данной работе проводилось сравнение распространения возмущений от двух типов землетрясений с горизонтальным и вертикальным сдвигом.

Выбор величины модуля скорости был произведён путём сопоставления данных моделирования и реальных экспериментальных данных. Данные были взяты из землетрясения произошедшего вблизи Guadalupe Victoria 6 июля 2010 года. По оценкам автоматической системы, очаг землетрясения располагался на глубине 1,5 км. Из карты максимальных скоростей известно, что максимальные скорости грунта на поверхности земли составили порядка 1 см/с. Путём задания различных по величине скоростей в очаге землетрясения и численного моделирования было установлено, что для получения амплитуд скоростей на поверхности того же порядка, необходимо задавать скорость начального возмущения равной 10 см/с.

Было проведено моделирование распространения волн при землетрясении, происходящем в слоистой среде. Плотность всех слоёв считалась постоянной и равной 2500 кг/м3. Толщины слоёв, а также скорости продольной и поперечной волн приведены в таблице 1.

На рис. 8 представлены распределения модуля скорости при распространении волн от гипоцентра до дневной поверхности для землетрясения с горизонтальным и вертикальным сдвигами.

Результаты представлены для одних и тех же моментов времени. Хорошо заметен силь-

H, м	V_p, км/с	V_s, км/с
300	4,19	2,79
400	4,65	3,1
500	5,85	3,5
2800	6,13	3,9

Таблица 1. Параметры слоистой среды

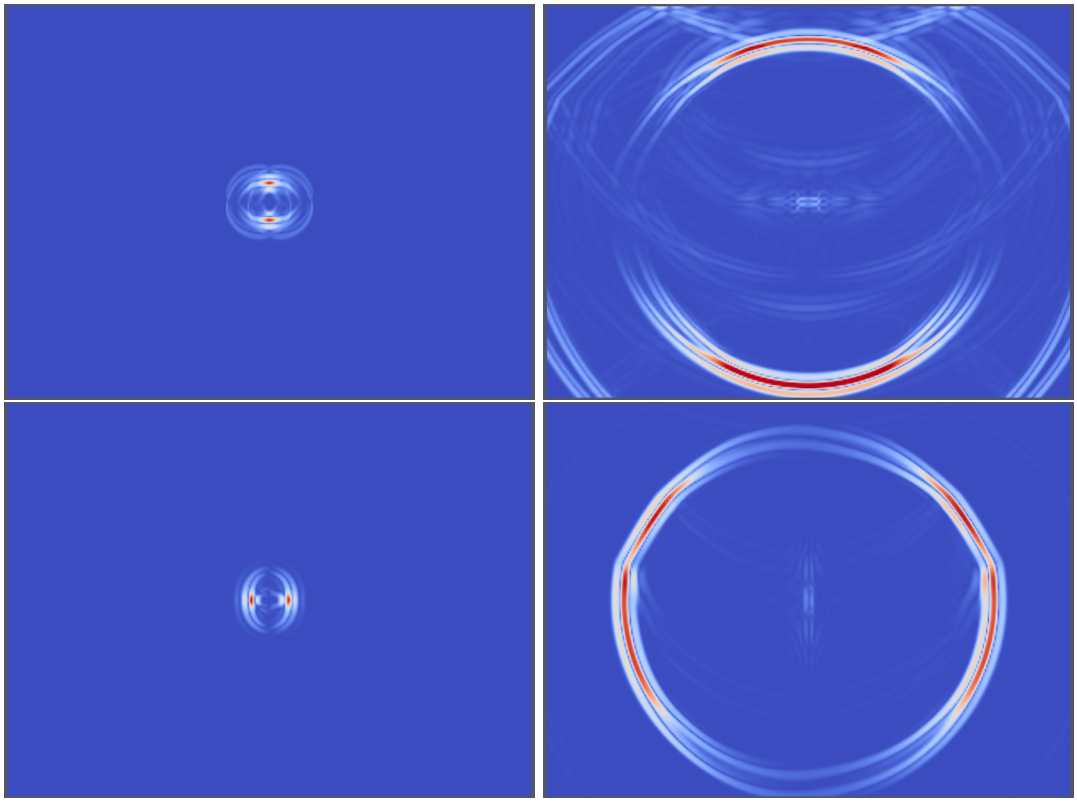

Рис. 8. Результаты моделирования землетрясения в случае горизонтальной (сверху) и вертикальной подвижки (снизу)

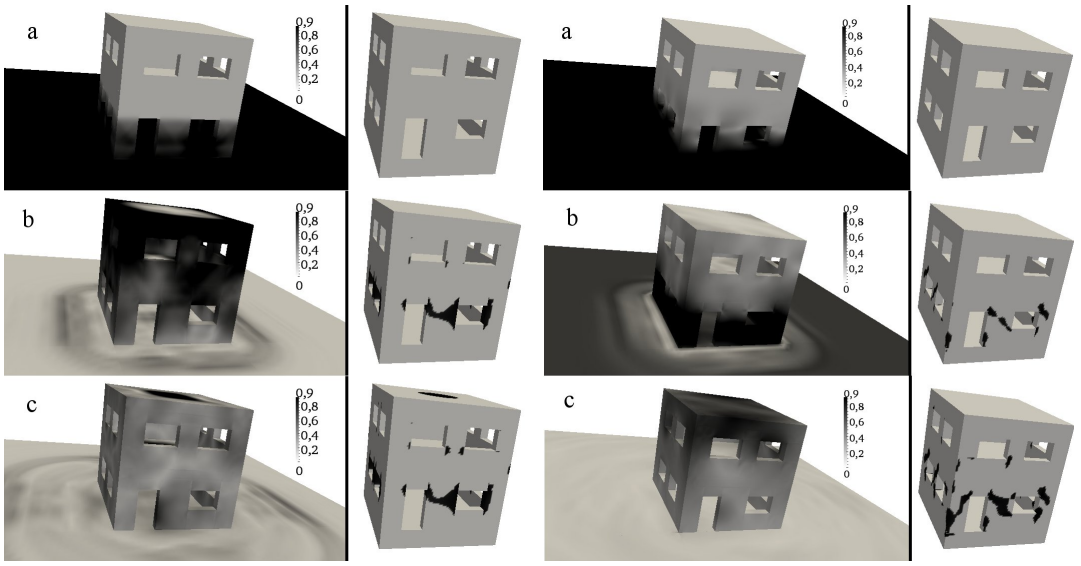

Рис. 9. Результаты расчета для продольной волны (слева) поперечной волны (справа) в последовательные моменты времени (сверху вниз)

ный второй фронт поперечной волны, он имеет несколько меньшую скорость и доходит до земной поверхности позднее. Также заметны отражения от различных геологических слоев. Картины от различных видов возмущений отличаются и имеют хорошее качественное сходство с экспериментальными данными.

4.2. Задачи сейсмостойкости

Рассмотрим процесс прохождения плоских продольной и поперечной волн через наземный объект, в нашем случае небольшое сооружение из бетона. Фронт волны параллелен свободной поверхности земли, распространение волны от центра земли к поверхности. В данном расчете такой импульс рассматривался как один пик от землетрясения.

На рис. 9 показано прохождение продольной и поперечной волн в различные моменты времени. Слева отображен модуль скорости среды, а справа цветом показаны участки где хотя бы раз за расчет выполнилось условие Мизеса, т. е. возможны разрушения.

Как видно из рисунков, при таком типе волны основные разрушения происходят на первом этаже и откол на потолке второго этажа здания. При землетрясениях основной урон происходит от поперечных волн, поскольку их амплитуда намного больше. Для поперечной волны картина похожая, но основные разрушения находятся в углах, дверных и оконных проемах. Нет повреждений потолков, как это хорошо было видно в случае продольной волны.

5. Заключение

В работе представлен результат применения широкого круга современных технологий написания параллельных приложений под различные архитектуры для задач моделирования процессов сейсмики с применением сеточно-характеристического и конечно-объемных методов. Полученные результаты говорят о эффективной реализации алгоритма под различные архитектуры. Распараллеливание на большое число ядер позволяет решать задачи, которые раньше решить было либо очень проблематично, либо невозможно из-за недостатка вычислительных ресурсов.

В дальнейшем планируется перенести рабочий алгоритм на архитектуру MIC от Intel

(Xeon Phi) и гибридный параллелизм GPU+MPI.

Литература

1. Холодов, А.С., Холодов, Я.А. О критериях монотонности разностных схем для уравнений гиперболического типа. // Журнал выч. математики и мат. физики. 2006. Т. 46, № 9, С. 1560–1588.

2. Петров, И.Б.,Холодов, А. С. Численное исследование некоторых динамических задач механики деформируемого твердого тела сеточно-характеристическим методом. // Журнал вычислительной математики и математической физики. 1984. Т. 5, № 24, С. 722–739.

3. Harten, Ami High Resolution Schemes for Hyperbolic Conservation Laws. // Journal of Computational Physics. 1997. V. 135,N. 2, P. 260–278.

4. Петров, И.Б., Хохлов, Н.И. Сравнение TVD лимитеров для численного решения уравнений динамики деформируемого твердого тела сеточно-характеристическим методом. // Математические модели и задачи управления. Сборник научных трудов. М.: МФТИ, 2011. С. 104–111.

5. Nickolls, J., Buck, I., Garland, M., Skadron, K. Scalable Parallel Programming with CUDA. Queue 6, March 2008.

6. Stone, J. E., Gohara, D., Shi, G. OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. // IEEE Des. Test. May 2010. V. 12, N. 3, P. 66–73.

7. OpenMP Application Program Interface. Version 4.0. July 2013. OpenMP Architecture Review Board.

8. Якобовский, М.В. Введение в параллельные методы решения задач: Учебное пособие М.: Издательство Московского университета, 2013. 328 с.

Application of modern high-performance techniques for solving local and global seismic problems

Nikolay Khokhlov and Igor Petrov

Keywords: MPI, OpenMP, CUDA, OpenCL, grid-characteristic method, finite-volume method, seismic

This paper discusses the parallelization of software, designed for dynamic modeling of the spread of wave disturbances in solids, using various advanced high-performance techniques. The software package supports two-dimensional and three-dimensional structural block meshes, explicit reference irregularities and the apparent isolation of contact boundaries. For numerical integration used grid-characteristic and finite volume methods of high order. The algorithm is parallelized using technology MPI, CUDA, OpenMP and OpenCL. Described some aspects of optimization of code using SIMD instructions of CPUs such as SSE and AVX. Comparative tests for the acceleration are given. As an example of the software system provides results of a series of test calculations.