Using GPUs for wave-packet decomposition

Nikitin, Viktor

Published in:
[Publication information missing]

2011

Министерство образования и науки РФ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Белгородский государственный национальный исследовательский университет»
Управление заочного, очно-заочного обучения и электронных образовательных технологий НИУ БелГУ

ВСЕРОССИЙСКИЙ КОНКУРС НАУЧНО-ИССЛЕДОВАТЕЛЬСКИХ РАБОТ СТУДЕНТОВ И АСПИРАНТОВ В ОБЛАСТИ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В РАМКАХ ВСЕРОССИЙСКОГО ФЕСТИВАЛЯ НАУКИ

7 сентября – 9 сентября 2011 г.

Сборник научных работ

Том 1

Белгород 2011

Министерство образования и науки РФ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«Белгородский государственный национальный исследовательский университет»
Управление заочного, очно-заочного обучения и электронных образовательных технологий НИУ БелГУ

ВСЕРОССИЙСКИЙ КОНКУРС НАУЧНО-ИССЛЕДОВАТЕЛЬСКИХ РАБОТ СТУДЕНТОВ И АСПИРАНТОВ В ОБЛАСТИ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В РАМКАХ ВСЕРОССИЙСКОГО ФЕСТИВАЛЯ НАУКИ

7 сентября – 9 сентября 2011 г.

Сборник научных работ
Том 1

Белгород 2011

ПРИМЕНЕНИЕ ГРАФИЧЕСКИХ УСКОРИТЕЛЕЙ ДЛЯ РЕШЕНИЯ ЗАДАЧИ РАЗЛОЖЕНИЯ СЕЙСМИЧЕСКИХ ДАННЫХ ПО ТРЁМЕРНЫМ ВОЛНОВЫМ ПАКЕТАМ

В.В. Никитин
Новосибирский государственный университет, г. Новосибирск.

Аннотация

Сейсмические данные, получаемые при проведении сейсмических работ в настоящее время, характеризуются многомерностью, большим объемом, а также своей нерегулярностью.

В последние годы в сейсмике начали применяться методы, разработанные для анализа и оптимального представления изображений – разложение по базисным функциям. Волновые пакеты – базис, который очень хорошо подходит под структуру сейсмических данных. Однако разложение по такому базису является очень сложным и ресурсоёмким.

В данной работе проводится анализ алгоритма разложения по трехмерным волновым пакетам. На основании этого анализа предлагается алгоритм с использованием высокопроизводительных вычислительных устройств nVidia, позволяющих проводить параллельные вычисления в многопоточном режиме, используя технологию CUDA. Была проведена программная и аппаратная оптимизация полученной программы. Тестирование проводилось на трёхмерных данных размером 256^3, было получено ускорение ~40 раз по сравнению с последовательной программой. Такое значительное ускорение демонстрирует высокую эффективность применения графических ускорителей для решения задачи прямого и обратного преобразования сейсмических данных по трёхмерным волновым пакетам.

Введение

В настоящее время сейсморазведка является самым информативным методом изучения земных недр для целей разведки запасов нефти и газа. В ходе проведения сейсмических работ на земной поверхности проводится измерение волн, отраженных от геологических границ внутри земли. Записанное волновое поле используется для построения сейсмического разреза земной коры, который затем используется при поиске полезных ископаемых.

Основным сейсморазведочным методом в настоящее время является метод отраженных волн (МОВ) [1]. Технология работ состоит в использовании контролируемого источника упругих колебаний (взрыв или вибратор при наземных работах, воздушная пушка при морских работах). Волны, созданные источником, распространяются внутрь среды и отражаются от геологических границ раздела, т.к. эти границы характеризуются разрывом упругих свойств (например, граница разделяет разные горные породы). Схематически процесс проведения работ изображен на рисунке 1.

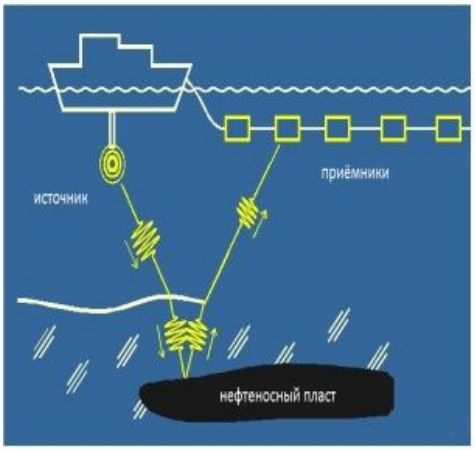

Рис. 1. Схема морской сейсмической разведки. Поиск нефти

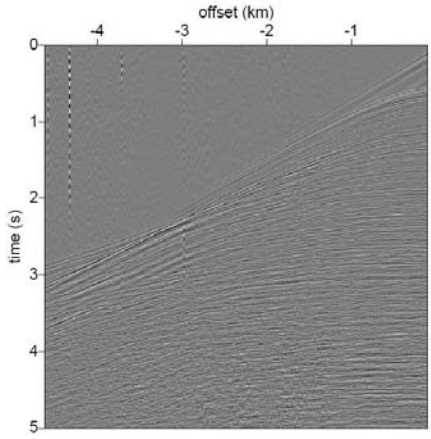

Рисунок 2: Пример сейсмограммы. Зависимость амплитуды полученного сигнала от времени для приёмников расположенных в координатах offset (km). источник сигнала расположен в начале

Отраженные волны распространяются вверх и регистрируются на поверхности косой приемников (каждый приемник записывает зависимость амплитуды колебаний грунта или давления в воде от времени, см. рисунок 2). После этого эксперимент повторяется для нового положения источника.

Таким образом, при проведении работ вдоль профиля (линии на поверхности земли) регистрируется трехмерный куб данных - амплитуда волн, как функция от координаты источника, координаты приемника и времени.

Перечислим основные характеристики сейсмических данных, получаемых при проведении сейсмических работ в настоящее время:

- многомерность (в самом простом случае данные являются трехмерной функцией, а в самом общем случае могут быть пятимерными);
- большой объем данных (данные для одного профиля будут занимать несколько Гб, а данные площадной съемки требуют несколько Тб памяти);
- нерегулярность данных (в полевых условиях возникают пропущенные и забракованные трассы, искривление профилей наблюдений и т.д.). В последние годы в сейсмике начали применяться методы, разработанные для анализа и оптимального представления изображений. Подход состоит в разложении сейсмических данных по специфическим базисным функциям – локализованным плоским волнам или волновым пакетам [2]. Оптимальность разложения заключается в том, что функция может быть представлена в виде линейной комбинации небольшого количества базисных функций, т.к. их форма хорошо подходит под структуру данных (Рис.3-4).

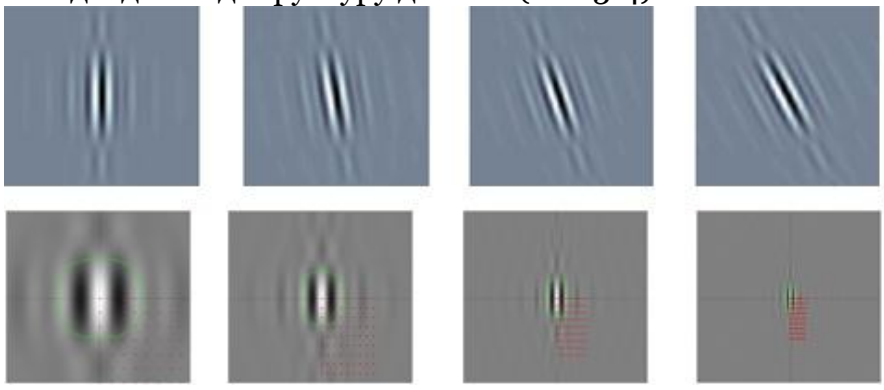

Рис. 3.: Примеры двумерных волновых пакетов различных ориентаций и размеров

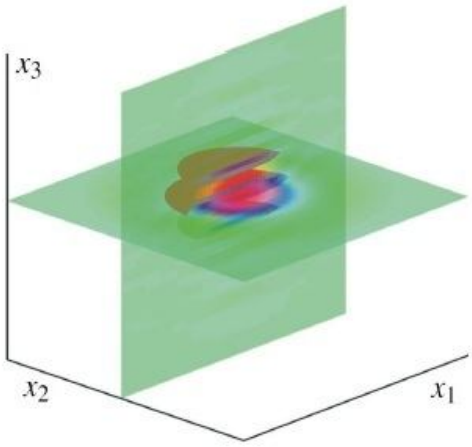

Рис. 4. Пример трёхмерного волнового пакета

Метод разложения по волновым пакетам позволяет решать целый ряд задач обработки сейсмических данных: их сжатие, подавление шума, интерполяция, регуляризация (пересчет с нерегулярной сетки на регулярную). Для решения всех перечисленных задач необходима эффективная программная реализация процедуры прямого и обратного преобразования по трехмерным волновым пакетам, позволяющая обрабатывать большие объемы данных.

Алгоритм разложения по волновым пакетам

Прямой оператор преобразования по волновым пакетам (ПВП) должен преобразовывать дискретно заданную трехмерную функцию $f(x)$ в набор коэффициентов $\{c_\gamma\}$, параметризованных индексом γ:

$$C: f(x) \rightarrow \{c_\gamma\}, \text{таких что}$$

$$f(x) = \sum_\gamma c_\gamma \varphi_\gamma, \text{где } \varphi_\gamma - \text{волновые пакеты.}$$

Каждый волновой пакет имеет конечный спектр, сосредоточенный в коробочке заданного размера, положения и ориентации. Коробочки, соответствующие разным волновым пакетам, распределены в трехмерной частотной области с перекрытием, как показано на рисунке 5. Объединение амплитудных спектров всех волновых пакетов задает разбиение единицы в частотной области. Таким образом, волновые пакеты создают (переопределенный) базис для разложения трехмерных функций.

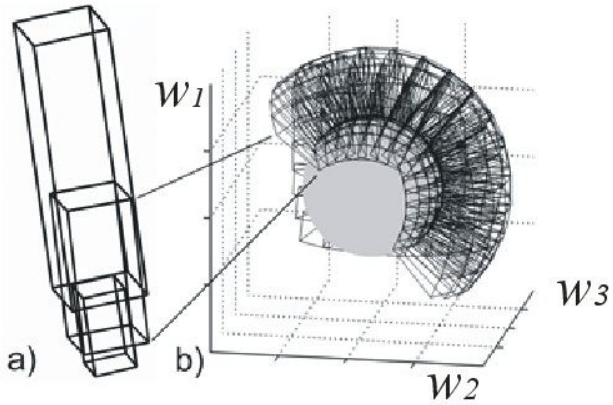

Рис. 5. Покрытие частотной области коробочками (каждая из коробочек задает волновой пакет); (а) коробочки покрывают частотную область с перекрытием; (б) коробочки одного слоя имеют разную ориентацию.

Заметим, что коробочки имеют прямоугольную форму, что важно для выполнения третьей операции – локального обратного преобразования Фурье на каждой коробочке (в этом случае на каждой коробочке можно задать локальную регулярную сетку). Однако они имеют разную ориентацию. Это повлияет на схему вычислений для дискретной реализации прямого ПВП:

-прямое быстрое преобразование Фурье (БПФ) на глобальной сетке;

- интерполяция спектра на локальные (повернутые) сетки, заданные в каждой коробочке;

-умножение спектра в коробочке на весовую функцию;

-обратное БПФ на каждой из коробочек (малое обратное преобразование Фурье).

Объединение локальных сеток на всех коробочках можно рассматривать, как нерегулярную сетку, покрывающую спектральную область. Спектральные функции обычно бывают осциллирующими, так что интерполяцию спектра лучше проводить с гарантированной точностью. Для эффективного выполнения этой операции принято использовать алгоритмы быстрого преобразования Фурье на нерегулярные сетки (USFFT) [4]. Вычислительная скорость таких алгоритмов близка к скорости стандартного БПФ.

Можно переписать процедуру разложения. Рассмотрим входную трехмерную функцию $f(\mathbf{x})$ размера N^3, заданную на регулярной сетке. Сначала выполняется процедура USFFT [8] (1.1-1.4), а затем две последних операции ПВП:

1.1 умножение начального изображения на весовую функцию на сетке N^3;

1.2 увеличение сетки в 2 раза (дополнение нулями) до размера $(2N)^3$;

1.3 прямое БПФ на регулярную сетку размера $(2N)^3$;

1.4 «интерполяция», для каждой точки нерегулярной сетки нужно произвести суммирование (в некоторой окрестности) с весом значений регулярной сетки.

2.умножение спектра в коробочке на весовую функцию;

3.обратное БПФ на каждой из коробочек (малое обратное преобразование Фурье).

Именно процесс интерполяции с глобальной (регулярной) сетки на локальную (нерегулярную) создает наибольшую вычислительную трудность. Рассмотрим его поподробнее.

Собирающее усреднение (прямое преобразование)

Для задания базисных функций спектральная область покрывается коробками (подобластями), на каждой из которых задана локальная сетка. На рисунке 6 это разбиение проиллюстрировано для двумерного случая; сетки разного цвета соответствуют разным коробкам, а их объединение можно рассматривать, как одну нерегулярную сетку.

Процесс интерполяции проводится по алгоритму USFFT, предложенному в [8]. Обозначим Фурье спектр входной функции как массив $F(\omega)$, заданный на глобальной регулярной сетке.

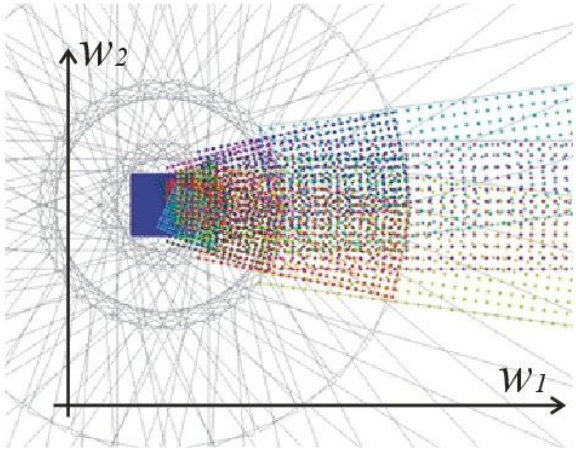

Рисунок 6: Нерегулярная сетка в спектральной области (объединение повернутых сеток на всех коробочках)

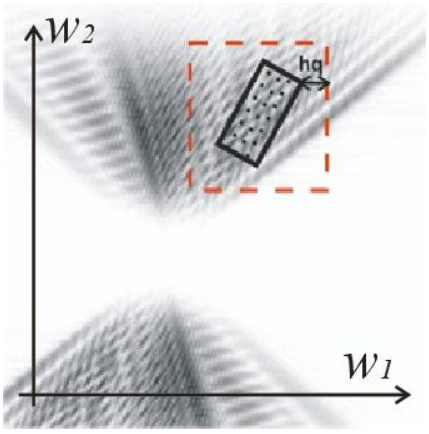

Рис. 7. Выделение области спектра (пунктирный квадрат), содержащей коробку

Сам процесс называют "собирающей" интерполяцией и он проиллюстрирован на рисунке 8. Серыми квадратиками обозначен фрагмент регулярной сетки $F(\omega)$, подгруженный в память GPU. Там же показана коробочка и крестиками показана локальная сетка на ней. После интерполяции будет получен массив значений спектра $F(\varpi)$ на локальной сетке ϖ :

$$F(\varpi_i) = \oint_{\Omega} w(\mathbf{r}) F(\omega_i + \mathbf{r}) d\mathbf{r} \qquad (1)$$

где ϖ_i - точка локальной сетки (крестики), ω_i - ближайшая к ней точка глобальной сетки, $\mathbf{r}$ пробегает точки глобальной сетки (серые квадратики) в шаре Ω радиуса hq; $w(\mathbf{r})$ - весовая функция.

Рассеивающее усреднение (обратное преобразование).

При построении обратного преобразования необходимо выполнить описанные операции в обратном порядке. Вначале для коэффициентов, относящихся к одной коробке, проводится малое прямое преобразование Фурье.

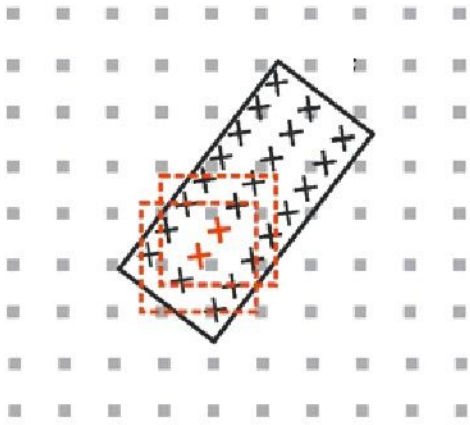

Рис. 8. Интерполяция при прямом преобразовании. Глобальная сетка - серые квадратики, локальная сетка - крестики.

Интерполяцию теперь необходимо провести в обратном направлении (рис. 8), т.е. с узлов нерегулярной сетки (крестиков) на узлы глобальной сетки (серые квадратики). Ее называют "рассеивающей" интерполяцией. В этом случае значение $F(\varpi)$ в каждой точке ϖ_i коробочки дает вклад в глобальный спектр $F(\omega)$, в точках ω_i, находящихся в шаре радиуса шаре Ω радиуса hq с весом $w(\mathbf{r})$.

То есть значение из точки ϖ_i (крестик) нужно с разными весами записать в элементы глобальной сетки (серые квадратики), находящиеся в красной области.

Перенос программы на GPU

Процесс интерполяции может проводиться независимо для каждой коробки, если обеспечить доступ к спектру, заданному на глобальной сетке. В случае, когда весь спектр не помещается в память графического процессора, необходимо подгружать в память часть спектра, содержащую интересующую коробку или группу коробок; на рисунке 7 эта область показана красным пунктиром. Если загрузить область, содержащую несколько коробочек, то можно проводить интерполяцию для них в рамках запуска одного ядра. Для выполнения интерполяции на графическом процессоре запускается число потоков, равное количеству узлов в коробочке.

После интерполяции, проведенной для всех точек коробки, необходимо выполнить малое обратное преобразование Фурье для массивов $F(\varpi)$. Оно также выполняется с помощью вызовов из библиотеки CUFFT [5]. При этом никаких предварительных манипуляций с данными не производится, т.к. данные уже находятся в глобальной памяти видеокарты.

Рассмотрим теперь обратное преобразование. Необходимо провести рассеивающее усреднение. Как видно из рисунка 8, при таком подходе возникает вероятность одновременного обращения к одной ячейке памяти для записи. Рассмотрим два крестика, находящихся рядом и для каждого выделим область влияния (красные квадратики); видно, что они пересекаются. Следовательно, возможны случаи, когда два или более потоков будут пытаться одновременно писать в один и тот же участок памяти, что неминуемо приведет к неправильному результату (race condition — ошибки соревнования).

Есть два решения проблемы: воспользоваться атомарными операциями, либо искать новый способ вычисления.

Атомарная операция происходит в два этапа: блокировка ресурса и выполнение самой операции. Здесь необходимо выполнить атомарное сложение. Возможны случаи, когда к одной ячейки памяти будут одновременно обращаться примерно 700-800 потоков. Причем таких случаев довольно много. Очевидно, что использование атомарных операций в данном случае пагубно скажется на производительности программы. Впрочем, сами производители NVidia не рекомендуют пользоваться атомарными операциями, потому что они настолько уменьшают скорость работы, что программа начинает работать медленнее последовательной. Остается вариант пересмотреть алгоритм интерполяции с нерегулярной сетки на регулярную.

Стандартный способ действия в таких ситуациях – это изменение обхода данных таким образом, чтобы каждый поток отвечал за запись одного элемента данных. Эта проблема решается, если переформулировать "рассеивающую" интерполяцию в форме "собирающей", как показано на рисунке 9. В этом случае цикл пробегает не по сетке $F(\varpi)$ (крестики), а по узлам глобальной сетки (серые квадратики). Тогда запись в каждую точку сетки (серые квадратики) производится один раз, а вычисления проводятся по формуле "собирающей" интерполяции (1).

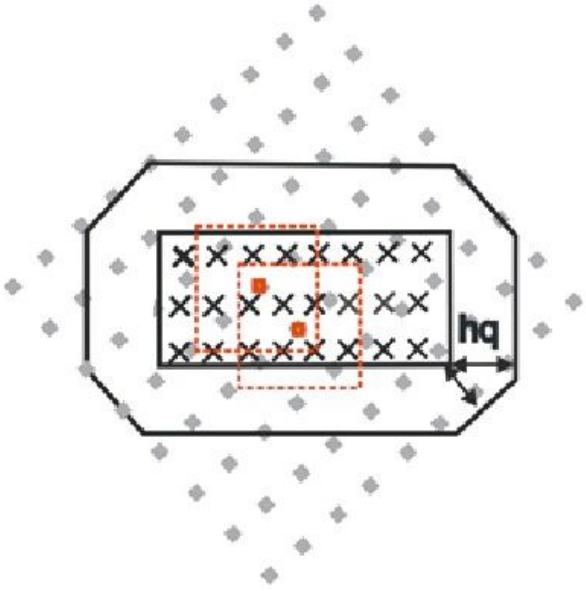

Рис.9. Модификация "рассеивающей" интерполяции в "собирающую".

Оптимизация

Оптимизация производительности, как правило, сводится к следующим шагам: оптимизация доступа в память, оптимизация математики, максимальное использование параллелизма задачи [6-7]. На основании этого и будем проводить оптимизацию.

Рассмотрим общую структуру графического процессора (рис.10).

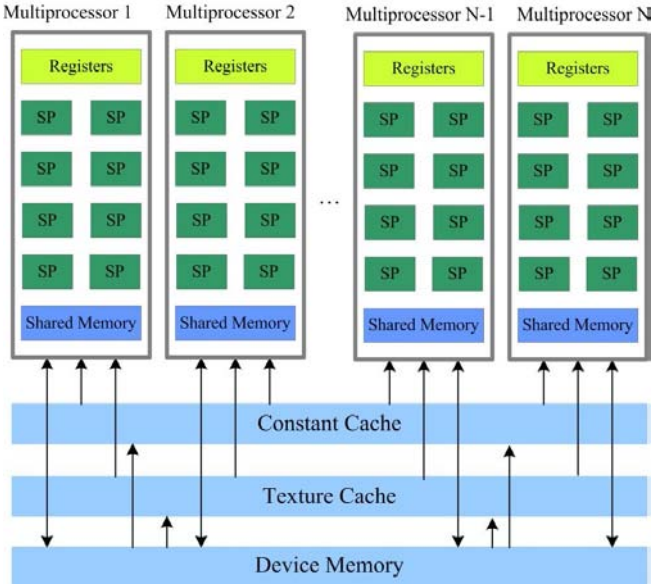

Рис. 10. Структура графического процессора

У графического процессора есть память, доступ к которой осуществляется намного быстрее, чем к глобальной памяти: регистры, разделяемая память, константная память, текстурная память. Объём регистровой и разделяемой памяти очень мал, и обращение к ним занимает 4 такта. Также 4 такта занимает и обращение к текстурной и константной памяти (если нет промахов по кэшу), размеры текстурного и константного кэшей существенно больше. Использование таких видов памяти значительно ускоряет программу, так как доступ к глобальной памяти в свою очередь занимает 400-600 тактов.

В нашей программе происходит большое количество чтений из глобальной памяти видеокарты. Чтение происходит из трёхмерных массивов, каждый из которых имеет большую размерность. Данный процесс можно

ускорить использованием текстур. Текстурная ссылка “привязывается” (bind) к соответствующей области выделенной памяти, и обращение через эту ссылку к данным становится намного быстрее посредством текстурного кэша. Каждый поток осуществляет около одной тысячи чтений из глобальной памяти, поэтому использование текстур даст существенное ускорение.

В функции, выполняющейся на видеокарте, используются константы, не изменяющиеся за всё время выполнения ядра. Это матрица поворота, размеры коробки, расстояние от коробки до центра регулярной сетки и другие. Большое количество параметров нельзя передавать ядру, поэтому для них необходимо было бы завести участок глобальной памяти. Но мы воспользуемся константной памятью. Именно в ней и поместим структуру, содержащую все эти параметры.

Далее произведём профилирование программы. Compute Visual Profiler [8] – это графический инструмент, который обеспечивает профилирование C-приложений, работающих на GPU (рис.11).

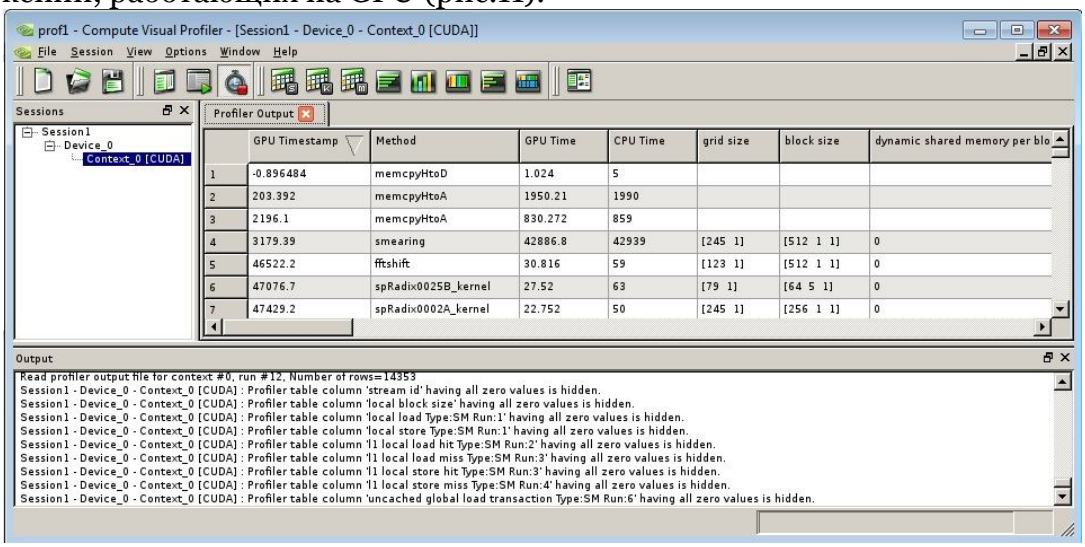

Рис. 11. Compute Visual Profiler.

Данный инструмент позволяет узнать время работы отдельных частей программы, количество непоследовательных обращений к памяти, размеры блоков потоков и сетки блоков, количество промахов по кэшу и многое другое. Всё это поможет в дальнейшей оптимизации.

Рассмотрим количество промахов по текстурному кэшу. При помощи данных профилировщика получаем, что около половины запросов к кэшу оказались неудачными. Чтобы предвыборка в кэш была оптимальной, необходимо сделать все обращения к памяти последовательными или, по крайней мере, часть из них. После проведения такой оптимизации количество промахов по кэшу удалось уменьшить до 15%.

Далее рассмотрим отчёт профилировщика о трёхмерных преобразованиях Фурье внутри процесса интерполяции. Преобразования производятся при помощи библиотеки CUFFT. Особенностью библиотеки является то, что на больших объёмах данных преобразование Фурье происходит по плоскостям. Например, для трёхмерных данных размером 38x38x76 производится 76 запусков ядра с размерами блока [38 38 1]. Изменим структуру данных так, чтобы их размерность была 76x38x38. Тогда получим 38 запусков ядра с размером блока [76 38 1], что очевидно будет работать быстрее.

Следующим этапом оптимизации является уменьшение количества тактов арифметических операций. Во-первых, рассмотрим деление на значение типа float. Оно занимает 36 тактов, однако если деление заменить умножением (4 такта) и взятием обратного (16 тактов), то получится всего 20 тактов. Во-вторых,

так как при вычислении используется тип float, то необходимо явно указывать тип float, например вместо 1.3 писать 1.3f, так как тип double на видеокарте занимает две ячейки памяти, в результате чего количество тактов на вычисление операций увеличивается. Также будем пользоваться функциями __expf(), __sinf(), __cosf() вместо exp(), sin(), cos(), так как их точность нас устраивает.

Тестирование

Программа разложения сейсмических данных тестировалась на различных типах платформ. Измерялось время работы последовательного алгоритма на процессоре Intel Core i7 и параллельного алгоритма на 4 Dual Core AMD Opteron 2218 (MPI). Также программа тестировалась на двух графических процессорах фирмы NVidia,основанных на технологии Fermi: Quadro FX 4000 и Tesla C2050.

В таблице 1 представлено полное времявыполнения программы для входных данных размерностью 256^3 (время записи, чтения из файла не учитывались).

Из таблицы видно, что использование GPU-платформ даёт рост производительности около 45 раз (прямое преобразование) и 35 раз (обратное преобразование) в сравнении с последовательной программой.

Таблица 1

Время выполнения программы на различных платформах.

Платформа	Прямое преобр-е (сек)	Обратное преобр-е (сек)	Ускорение (раз)
Tesla C2050	65,51	89,48	45 / 35
GeForce Quadro FX 4000	88,05	119,06	33 / 26
GeForce 9800 GX2	368,98	464,87	8 / 7
4 x Dual-Core AMD Opteron(tm) 2218 HE	1123,79	1150,13	2,7/ 2,6
Intel Core i7	2947,50	3137,04	1 / 1

Код тестировался на синтетических данных, представляющих собой 3D куб, размерностью 256^3. Рассмотрим сжатие сейсмических данных и подавление шумов.

Двумерный срез куба показан на рисунке 12,a). После применения прямого преобразования по волновым пакетам берётся только N_{wp} наибольших коэффициентов, остальные коэффициенты отбрасываются. Затем применяется обратное преобразование по волновым пакетам для получения восстановленного изображения.

Таким образом, получается сжатие сейсмических данных. Коэффициент сжатия определяется как $CR = N_{wp}/N_{in}$, где N_{in} – количество точек в оригинальном кубе данных. Двумерные срезы кубов восстановленных данных показаны на рисунках 12,b) и 12,c) для CR = 0,14 и CR = 0,02 соответственно.

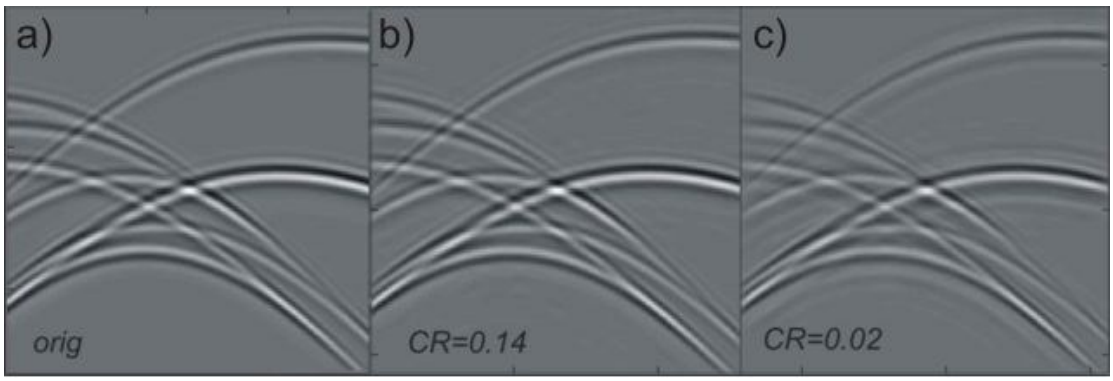

Рис. 12. Сжатие данных a) – срез оригинальных данных b,c) – восстановленные после сжатия данные.

На рисунке 12,b) заметны лишь малые ухудшения качества изображения. По рисунку 12,c) всё ещё можно узнать необходимую информацию о кинематике волн, несмотря на ослабевающие амплитуды.

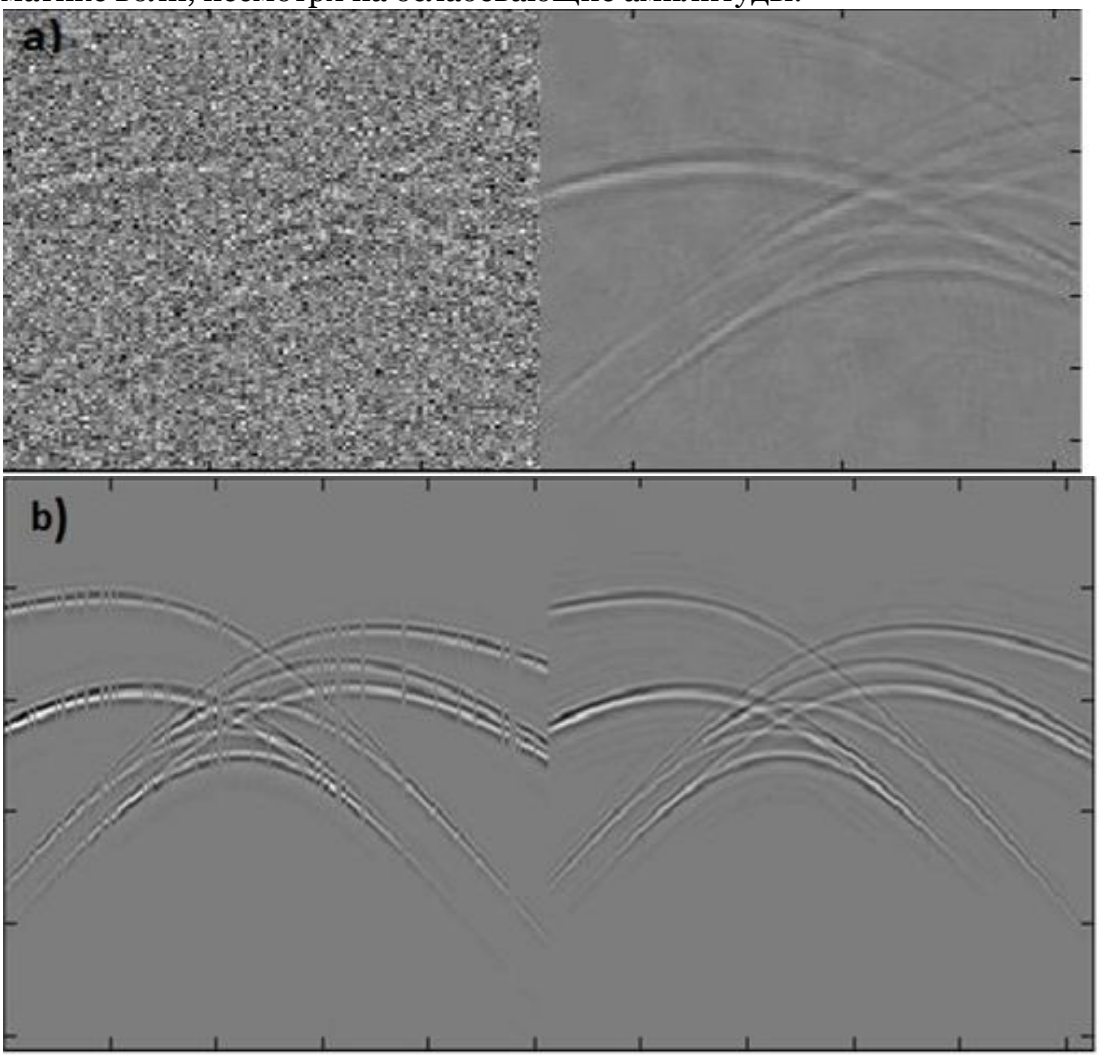

Рис.13. Восстановление при помощи коэффициентов a) - подавление шума, b) – интерполяция.

Подавление случайного шума и интерполяция данных производятся аналогично сжатию. Производим прямое преобразование по волновым пакетам, отсекаем коэффициенты меньшие заданного порога и выполняем обратное преобразование. На рисунке 13,a) представлено подавление гауссова шума с дисперсией 0,5 для данных, нормированных к единице.

Заключение

В процессе данной работы был выполнен перенос последовательного кода

программы разложения трехмерных функций по базису волновых пакетов на графический процессор при помощи технологии CUDA. Была проедена оптимизация при помощи инструментариев для работы с GPU: Compute Visual Profiler, CUDA Occupancy Calculator.

Получено 45-кратное ускорение для прямого преобразования по волновым пакетам и 35-кратное для обратного. В ходе тестирования выявилось, что результаты последовательной программы и программы, адаптированной на GPU, совпадают с точностью до 5 знака после запятой. Такая точность является допустимой для анализа сейсмических данных.

Программа разложения трехмерных функций по волновым пакетам будет представлять интерес для научно-исследовательских групп, занимающихся развитием методов сейсморазведки, и производственных организации, занимающиеся обработкой сейсмических данных.

С другой стороны, разложение по волновым пакетам является таким же универсальным преобразованием, как преобразование Фурье, вейвлет-преобразование и т.д. Так что круг возможных потребителей может оказаться достаточно широким. В перспективе разложение по волновым пакетам может оказаться полезным при анализе трехмерных изображений и выделении границ, например, в медицинской томографии и т. д.

Благодарности

Автор выражает благодарность за научное руководство и консультации Романенко А.А., Дучкову А.А. и Andersson F. Работа выполнена при частичной поддержке группы компаний «Т-Платформы» (приз в конкурсе “Эффективное использование GPU-ускорителей”) и Шведского фонда по международному сотрудничеству в науке и высшем образовании (the work was partly supported by Swedish Foundation for International Cooperation in Research and Higher Education).

Литература:

1. Гурвич, И.И., Боганик, Г.Н. Сейсмическая разведка. - М.: Недра, 1980. - 551с.
2. Andersson, F.A., Hoop, M.V., Smith, H.A. A multi-scale approach to hyperbolic evolution equations with limited smoothness: Communications in Partial Differential Equations, 2008. - pp. 988-1017.
3. Duchkov, A.A., Andersson, F.A., Hoop, M.V. Discrete almost-symmetric wave packets and multiscale geometrical representation of (seismic) waves // IEEE Transactions on Geoscience and Remote Sensing, Vol. 48, No. 9, 2010. - pp. 3408–3423.
4. Dutt, A.F., Rokhlin, V.I. Fast Fourier transforms for nonequispaced data // SIAM Journal on Scientific Computing, Vol. 14, 1993. - pp. 1368–1393.
5. NVIDIA Corporation, CUDA CUFFT library, // Электронный ресурс об основах работы с Compute Visual Profiler [Электронный ресурс, 2008] — Режим доступа: http://developer.download.nvidia.com/compute/cuda/1_1/CUFFT_Library_1.1.pdf, свободный.
6. Боресков, А.В., Харламов, А.А. Основы работы с технологией CUDA, М.:ДМК Пресс, 2010. - сс. 45 -162.
7. Kirk, D.B., Hwu, W.W. Programming Massively Parallel Processors. A Hands-on Approach // Elsevier, Vol. 5-6, 2010. - pp. 77-141.
8. NVIDIA Corporation, Compute Visual Profiler. User Guide. // Электронный ресурс об основах работы с Compute Visual Profiler [Электронный ресурс, 2010] — Режим доступа: http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/VisualProfiler/Compute_Visual_Profiler_User_Guide.pdf, свободный.