ISSN 1992-6502 (Print)
2016. Т. 20, № 1 (71). С. 151–160
Вестник УГАТУ
ISSN 2225-2789 (Online)
http://journal.ugatu.ac.ru

УДК 004.051, 550.34.06

Эффективная реализации метода когерентного суммирования на ускорителях GPU NVIDIA*

М. А. Городничев[1,2,3], А. А. Дучков[2,4], В. Г. Сарычев[1,4]

[1] maxim@ssd.sscc.ru, [2] DuchkovAA@ipgg.sbras.ru, [3] sarychevmail@gmail.com

[1] ФГБОУ ВПО «Новосибирский государственный технический университет» (НГТУ)
[2] ФГАОУ ВПО «Новосибирский национальный исследовательский государственный университет» (НГУ)
[3] ФГБУН «Институт вычислительной математики и математической геофизики СО РАН» (ИВМиМГ СО РАН)
[4] ФГБУН «Институт нефтегазовой геологии и геофизики им. А. А. Трофимука СО РАН» (ИНГГ СО РАН)

Поступила в редакцию 05.03.2016

Аннотация. Процедура когерентного суммирования является основой для класса методов обработки сейсмических данных. Рассматривается проблема эффективной реализации когерентного суммирования на GPU, поддерживающих программную модель NVIDIA CUDA. Исследованы различные варианты отображения алгоритма на архитектуру вычислителя. Полученная реализация позволяет достигать до 70 % от пиковой производительности подсистемы памяти и арифметического устройства. В ходе тестирования выявлены предельные размеры задач, которые позволяют реализовать потоковую обработку данных в режиме реального времени для различных аппаратных архитектур графических процессоров NVIDIA: Fermi, Kepler, Maxwell, а также определены оптимальные параметры порций данных, одновременно отправляемых на обработку в GPU.

Ключевые слова: графические карты, параллельное программирование, CUDA, GPU, геофизика, оптимизация, архитектура графических процессоров, когерентное суммирование, разработка алгоритма, большие объемы данных, обработка данных, сейсмика.

ВВЕДЕНИЕ

В обработке сейсмических данных востребовано использование высокопроизводительных вычислительных платформ. Это связано с большими объемами данных и необходимостью их оперативной обработки [1–5]. Одним из примеров является задача микросейсмического мониторинга в процессе гидравлического разрыва пласта (ГРП) [6], который применяется для повышения проницаемости пород за счет создания в них системы трещин. Образование трещин при ГРП сопровождается генерацией сейсмических волн, которые регистрируются системой сейсмических датчиков. Микросейсмический мониторинг ГРП проводится в течение часов и суток, в результате чего получается большой объем данных (терабайты).

Одним из методов обработки данных микросейсмического мониторинга является метод эмиссионной томографии, основанный на когерентном суммировании [7]. Когерентное суммирование применяется для определения положения источников микросейсмических событий (локализации гипоцентров), что позволяет определить истинную геометрию образовавшейся трещины гидроразрыва. Для своевременного обнаружения источников сейсмических волн и поиска гипоцентров сейсмической активности необходимо осуществлять обработку получаемой в ходе мониторинга информации в режиме реального времени. Так как объемы получаемых данных исчисляются терабайтами, требуется использовать параллельные вычислительные системы для обеспечения высокой скорости обработки этих данных.

Цель работы состоит в эффективной реализации метода эмиссионной томографии (когерентного суммирования) для исполнения на графических ускорителях (Graphics Processing Unit, GPU), поддерживающих программно-аппаратную модель CUDA [8], которая позволяет производить вычисления с использованием графических процессоров компании NVIDIA.

Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии 2016»

CUDA предоставляет доступ к набору инструкций графического ускорителя и позволяет управлять его памятью.

Рассматриваются оптимизации по работе с различными типами памяти GPU и приводится анализ степени загруженности GPU при различных конфигурациях используемой памяти и наборах входных данных, описаны оптимизации по работе с жестким диском и операциями ввода/вывода, направленными на «маскировку» операций чтения/записи на фоне вычислений, а также приводится сравнение по скорости работы реализованного метода на различных поколениях архитектур GPU NVIDIA.

МЕТОД КОГЕРЕНТНОГО СУММИРОВАНИЯ

Технология наземного микросейсмического мониторинга состоит в установке сети сейсмоприемников на земной поверхности для регистрации микросейсмических событий, вызванных процессом гидроразрыва, разработкой месторождений или другими процессами. Проводится непрерывная запись колебаний каждым приемником в течение всего периода мониторинга.

Таким образом, в ходе мониторинга для каждого *r*-го приемника получаем запись сигнала, дискретизированного с шагом *h* по времени, или сейсмотрассу $d_r(t)$ для отрезка времени длиной *K*. Совокупность записей представляет собой матрицу $d_r(t_k)$, где $r=1,\ldots,R$ – индекс приемника с координатами $(\alpha_r,\beta_r,\gamma_r)$, t_k – отсчеты по времени, $k=1,\ldots,K$. Заметим, что число приемников *R* обычно соответствует сотням или тысячам, а количество отсчетов по времени *K* является очень большой величиной (порядка 150 млн). По этой причине все данные разбиты по времени на множество файлов; каждый файл содержит записи всех сейсмоприемников в течение определенного отрезка времени. В каждом файле расположены все данные с приемника $r = 1$, затем с приемника $r = 2$ и так далее (см. рис. 1). Координаты приемников $(\alpha_r, \beta_r, \gamma_r)$ располагаются в отдельном файле.

Для обработки данных наземного микросейсмического мониторинга используется метод эмиссионной томографии, основанный на принципе когерентного суммирования [7]. Для этого строится сетка так называемых пробных источников с координатами (x_j,y_j,z_j), где $j=1,\ldots,J$ – индекс пробного источника, которая покрывает исследуемое пространство. Далее происходит перебор узлов сетки и данные мониторинга для каждого момента времени τ_k суммируются по годографу прямой волны из пробного источника в этом узле. Так, для каждого узла вычисляется суммотрасса

$$s_j(x_j,y_j,z_j,\tau_k)=\sum_{r=1}^{R} d_r\left(t_r^h(x_j,y_j,z_j,\tau_k)\right), \quad (1)$$

где времена прихода прямых волн $t_r^h(x_j,y_j,z_j,\tau_k)$ из точки (x_j,y_j,z_j) в приемник $(\alpha_r,\beta_r,\gamma_r)$ рассчитываются для известной скоростной модели [9], в частности, в однородной среде со скоростью *V* они вычисляются по формуле

$$t_r^h(x_j,y_j,z_j,\tau_k)=\tau_k+\frac{1}{V}\sqrt{(x_j-\alpha_r)^2+(y_j-\beta_r)^2+(z-\gamma_r)^2}, \quad (2)$$

где τ_k – время возникновения события, *k* – номер отсчета по времени.

Положение источника определяется как максимум так называемого куба когерентности:

$$\tilde{s}(x_j,y_j,z_j)=max_k\left(s(x_j,y_j,z_j,\tau_k)\right),$$

где время τ_k является временем возникновения сейсмического события. В общем случае источников может быть несколько.

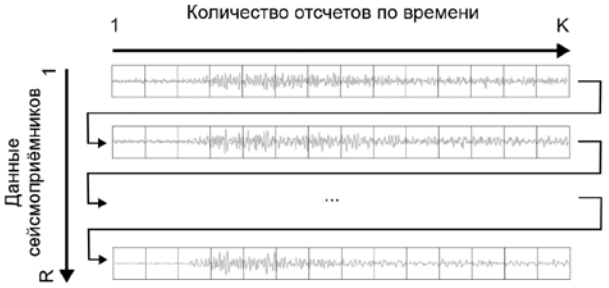

Рис. 1. Формат данных с сейсмоприемников

На рис. 2 схематично изображен алгоритм когерентного суммирования для случая двухмерного пространства.

Слева треугольниками обозначены сейсмоприемники, расположенные в линейку вдоль верхней кромки, а точками – сетка пробных источников. Справа изображены сейсмотрассы, записанные приемниками, пунктирной дугой

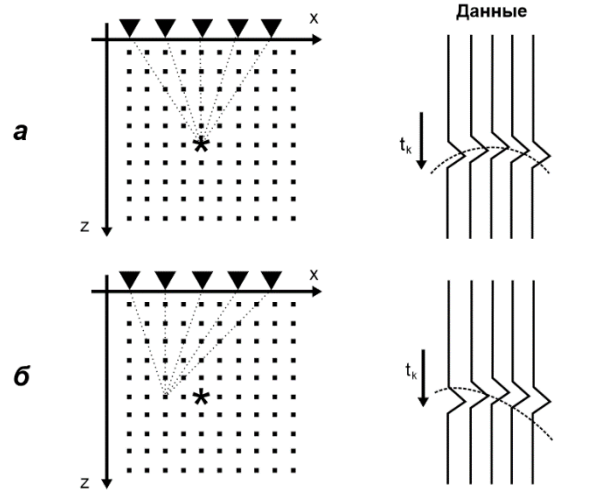

Рис. 2. Алгоритм когерентного суммирования

изображен годограф прямой волны, посчитанный в предположении, что событие возникло в момент времени τ_k. На рис. 2, *а* изображен случай, когда в предполагаемом пробном источнике произошло сейсмическое событие и рассчитанный для него годограф совпадает с волной, регистрируемой сейсмоприемниками. В результате при суммировании получается максимальное значение среди прочих пробных источников. На рис. 2, *б* изображен пример, когда пробный источник попадает мимо реального источника сейсмических волн.

РЕАЛИЗАЦИЯ АЛГОРИТМА НА GPU

Программно-аппаратная модель CUDA (см. подробно [8]) предполагает, что все вычисления на устройстве, реализующем эту модель, осуществляются множеством потоков, порождаемых одним вызовом процедуры, которая называется ядром (англ. kernel). Параметры запуска ядра определяют разбиение множества потоков на блоки. Вычислительное устройство состоит из нескольких мультипроцессоров. Количество мультипроцессоров различно в конкретных моделях устройств. Потоки выполняются ядрами (англ. cores) мультипроцессоров. Все потоки одного блока назначаются на исполнение в один из мультипроцессоров. Потоки планируются к исполнению на мультипроцессор группами фиксированного размера. Такая группа называется варпом. Варпы одного блока исполняются на мультипроцессоре в некотором порядке, возможно, попеременно. Количество блоков может превышать количество мультипроцессоров. Модель CUDA определяет несколько типов памяти, доступной потокам. В настоящей работе рассматривается работа с *константной, разделяемой, глобальной и кэш-памятью*. Константная память заполняется данными до запуска ядра и доступна всем потокам ядра только для чтения. Память, которая доступна для чтения и записи потокам только в рамках одного блока, называется разделяемой памятью. Глобальная память доступна для чтения и записи всем потокам ядра. Каждый мультипроцессор имеет кэш-память для кэширования доступа к глобальной памяти.

Вычисление суммотрасс является наиболее ресурсоемкой процедурой и на её выполнение уходит большая часть времени. Общий алгоритм формирования суммотрасс выглядит так, как показано на рис. 3.

Так как количество отсчетов по времени K очень велико, невозможно поместить всю длину суммотрассы в память вычислительного устройства целиком. Для решения данной проблемы предлагается разбить длину K на N равных отрезков длиной ΔK ($K=N\Delta K$), которые бы целиком помещались в памяти устройства.

```
for 1..J do
  for 1..R do
      Расчет годографа (2)
  end for
  for 1..K do
      for 1..R do
          Суммирование вдоль годографа (1)
      end for
  end for
end for
```

Рис. 3. Алгоритм формирования суммотрасс

Также, по сравнению с общим алгоритмом, принято решение рассчитать сначала все годографы для пробных источников, и уже после этого проводить суммирование. Такой подход позволит не пересчитывать годографы для каждого из N отрезков. Также стоит отметить, что от выбора длины ΔK будет зависеть количество используемых ресурсов вычислительного устройства и скорость формирования суммотрасс. Таким образом, алгоритм формирования суммотрасс будет выглядеть следующим образом (см. рис. 4).

```
for 1..J do
  for 1..R do
    Расчет годографа (2)
  end for
end for
for 1..N do
  for 1..J do
    for 1.. ΔK do
        for 1..R do
            Суммирование вдоль годографа (1)
        end for
    end for
  end for
end for
```

Рис. 4. Алгоритм формирования суммотрасс

Годограф (2) представляет собой массив целых чисел (возможна интерполяция, но считается, что шаг дискретизации достаточно мал и обеспечивает корректность получаемых данных). Каждое такое число обозначает количество шагов по времени, за которое волна доходит от пробного источника до конкретного сейсмоприемника. Длина массива соответствует количеству сейсмоприемников R. Каждый поток GPU осуществляет расчет (2) времени прихода прямой волны для одного сейсмоприемника. Полученное значение записывается в глобальную память устройства и используется в дальнейшем для получения суммотрасс.

Суммотраса (1) представляет собой массив вещественных чисел с одинарной точностью. Суммирование производится вдоль годографа прямой волны (см. рис. 2) для каждого пробного источника. На рис. 5 продемонстрировано, как распределяется работа между потоками.

Каждый поток GPU осуществляет суммирование (1) для определенного шага по времени. Полученные суммы помещаются в глобальную память устройства, после чего выгружаются в оперативную память хоста для последующей обработки.

Рис. 5. Распределение вычислений по потокам

Такое распределение работы между потоками GPU выбрано в связи с тем, что чтение данных осуществляется из непрерывных блоков памяти, что необходимо для эффективного обращения к памяти и обеспечения кэширования данных. Альтернативные способы распределения вычислений между потоками GPU либо нарушают последовательный порядок обращения в память при чтении данных, либо же требуют одновременной записи данных несколькими потоками в одну и ту же область памяти. Эти факторы негативным образом сказываются на скорости обработки.

Так как каждый поток внутри блока осуществляет запись в область глобальной памяти, не изменяемую другими потоками, не возникает накладных расходов на поддержание когерентности кэшей.

ОПТИМИЗАЦИЯ

В данной части работы приведены следующие оптимизации:

1) оптимизация выбора типов памяти для хранения различных величин;
2) выбор длины сейсмотрасс ΔK;
3) оптимизация работы с жестким диском.

Параметры производительности программ в данной части работы проводятся для GPU с архитектурой Fermi с Compute Capability версии 2.0 и 2.1 для карт Tesla C2050 и GTX 460SE соответственно.

Использование различных типов памяти GPU

Программно-аппаратная модель CUDA определяет несколько типов памяти, доступной вычислительным потокам. Организация хранения данных по видам памяти может быть выполнена различными способами, и выбор способа существенно влияет на скорость работы программы.

Константная память. Константная память используется для хранения координат сейсмоприемников и параметров исследуемого пространства, так как они не изменяются в процессе работы программы. В процессе расчета годографов прямой волны используется только константная память и регистры, что обеспечивает высокую скорость работы программы.

Разделяемая память. Разделяемую память можно использовать для записи рассчитанных годографов перед суммированием трасс, но так как работа с годографами сводится к однократному считыванию, то использование разделяемой памяти становится нецелесообразным. Вместо нее предлагается использовать глобальную память. Даже с учетом более длинного пути через кэш L2 и L1, осуществляется группировка запросов в память (в англоязычной литературе coalescing), вследствие чего за один такт из памяти считывается блок данных размером 128 бит. Также, не используя разделяемую память, выгодно для работы с большим объемом данных осуществлять переключение приоритета для кэш-памяти первого уровня (16 Кбайт разделяемой памяти и 48 Кбайт кэша L1 для архитектуры Fermi).

Для двух версий программы – с использованием разделяемой памяти и с использованием только глобальной памяти – было получено количество посчитанных элементов суммотрасс за одну секунду работы алгоритма. Версия с использованием разделяемой памяти – 10,68 млн сумм/сек, версия с использованием глобальной памяти – 10,77 млн сумм/сек. Таким образом, использование только глобальной памяти позволяет получать хотя и незначительное, но преимущество по количеству посчитанных элементов суммотрасс в секунду (для 1920 сейсмоприемников).

Глобальная память и кэш L1. В реализации метода основное время работы затрачивается на обращение в глобальную память.

Для достижения наилучших результатов требуется считывать данные из непрерывных

блоков памяти, выровненных по адресам, при этом осуществляется группировка запросов в память и за один такт считывается по несколько значений, которые попадают в быстрый кэш L1 (коалесинг).

Данные, соответствующие обрабатываемой трассе, считываются напрямую из непрерывного блока глобальной памяти, однако попадание в кэш L1 и группировка запросов в память не гарантируются в связи с тем, что считываемые данные, из-за смещения вдоль годографа, не выровнены по адресам. Таким образом, попадание в кэш первого уровня будет происходить не постоянно. В зависимости от количества сейсмоприемников R и длины трассы ΔK, будет изменяться количество кэш-попаданий и, соответственно, кэш-промахов. В результате тестирования программы с различными конфигурациями параметров было установлено, что кэш-попадания происходят в пределах 55–65 % случаев. Так, чем меньше количество сейсмоприемников и короче длина трассы, тем выше процент попадания в кэш L1.

На рис. 6 (платформа Tesla C2050, архитектура Fermi, Compute Capability 2.0) представлен график, демонстрирующий влияние оптимизаций по работе с памятью на скорость обработки данных.

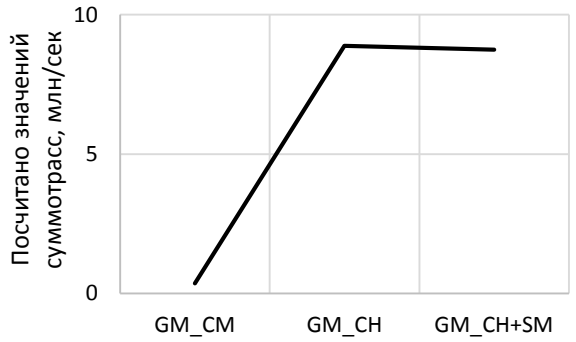

Рис. 6. Использование различных типов памяти

Влияние кэша L1, а точнее, попадание в L1 кэш можно оценить по количеству посчитанных элементов суммотрасс за одну секунду работы алгоритма. С целью продемонстрировать влияние количества кэш-попаданий, при работе с памятью была реализована версия программы, в которой обход памяти производился случайным образом.

Итого: версия со значительными кэш-промахами GM_CM (3–5 % кэш-попаданий) – 0,43 млн сумм/сек, версия с использованием разделяемой памяти GM_CH+SM – 10,68 млн сумм/сек, а версия с использованием лишь глобальной памяти GM_CH и кэш-попаданиями на уровне 55–65 % – 10,77 млн сумм/сек. Таким образом, можно заключить, что использование лишь глобальной памяти устройства с кэш-попаданиями на уровне 55–65 % является наиболее эффективным вариантом реализации данного метода на используемых GPU (архитектура Fermi). Для данного метода невозможно обеспечить 100 % попаданий в кэш L1 при расчете суммотрасс без дополнительных накладных расходов по реструктуризации данных в памяти между фазами работы программы. Эти накладные расходы по обеспечению выровненного обращения в память будут превышать время работы программы с 55–65 % попаданием в кэш L1, что делает их нецелесообразными.

Выбор длины сейсмотрасс ΔK

От количества сейсмоприемников R и длины трасс ΔK зависит количество производимых операций по расчету суммотрасс. Чем меньше число сейсмоприемников, тем больше оказываются загружены варпы мультипроцессора и количество исполняемых операций за такт возрастает. Это связано с меньшим количеством ветвлений встречающихся по ходу вычислений, а также меньшим обращением в глобальную память. Также чем меньше сейсмоприемников, тем больше кэш-попаданий в глобальную память. В случае с длиной сейсмотрасс ΔK ситуация несколько отличается. При малой длине трассы не получается загрузить устройство «полезной работой» (т.е. создать достаточное количество потоков), так как количество используемых потоков напрямую зависит от выбранной длины ΔK. Однако при выборе большой длины трассы, порядка нескольких десятков тысяч, заметно возрастают промахи по L1 кэшу, что негативным образом сказывается на скорости обработки данных. Таким образом, необходимо найти компромисс между длиной трасс ΔK и процентом кэш-попаданий при заданном числе сейсмоприемников. На рис. 7 и 8 (платформа Tesla C2050, архитектура Fermi, Compute Capability 2.0) продемонстрирована динамика изменения числа подсчитанных элементов суммотрасс в зависимости от количества сейсмоприемников, длины трасс и процента кэш-попаданий в L1.

Исходя из полученных результатов, можно сделать вывод, что необходимо загружать как можно более длинные отрезки сейсмотрасс ΔK, порядка 100 тысяч, для количества сейсмоприемников в диапазоне 200–300 штук, при увеличении числа сейсмоприемников начинает сказываться влияние кэш-промахов при доступе

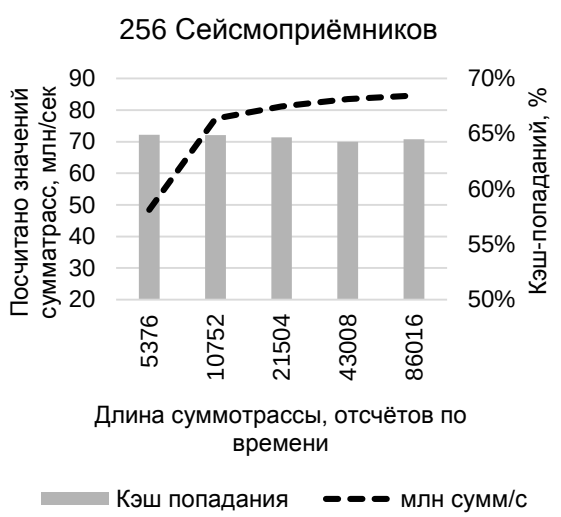

Рис. 7. Зависимость времени формирования суммотрасс от количества сейсмоприемников, длины трасс и процента кэш-попаданий в L1 кэш. 256 сейсмоприемников

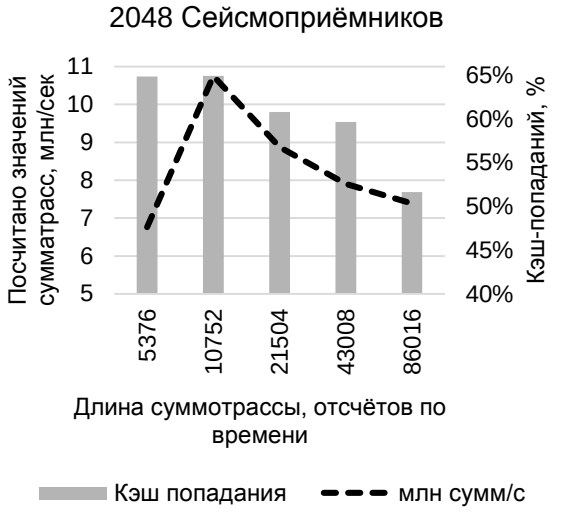

Рис. 8. Зависимость времени формирования суммотрасс от количества сейсмоприемников, длины трасс и процента кэш-попаданий в L1 кэш. 2048 сейсмоприемников

к глобальной памяти GPU и необходимо уменьшать длину рассчитываемых сейсмотрасс ΔK вплоть до нескольких десятков тысяч отсчетов по времени. Так, при количестве сейсмоприемников равном 2048 максимальной эффективности по скорости обработки данных можно добиться при длине сейсмотрассы в диапазоне 10–12 тысяч отсчетов по времени. Как видно из обоих графиков на рис. 7 и 8, при задании слишком малой длины трассы происходит резкое падение скорости обработки, вызванное малой загруженностью мультипроцессоров. Таким образом, можно сделать вывод, что при выборе длины сейсмотрассы, необходимо находить компромисс между процентом кэш-попаданий и длиной трасс опытным путем, в зависимости от архитектуры устройства и количества доступных ресурсов.

Оптимизация работы с жестким диском

При обработке данных, в зависимости от длины обрабатываемой трассы ΔK, возникают ситуации, в которых необходимо считывать данные из разных файлов. При этом необходимо учитывать, что длина суммотрассы в процессе обработки складывается из суммы самой длины суммотрассы и максимального значения отклонения годографа прямой волны. На рис. 9 схематично изображена работа по обработке данных. Снизу на оси изображены файлы длиной D и их количество, а сверху жирной линией показана длина трассы ΔK и максимальная длина отклонения годографа, которую требуется копировать в начало следующей обрабатываемой трассы. Такой подход к работе с файлами позволяет корректно обрабатывать данные, которые содержатся как в «мелких», так и в «больших» файлах, а также осуществлять механизм потоковой обработки данных по мере их поступления.

Так как данные разбиты на множество файлов, их необходимо подгружать в процессе вычислений. Как правило, считывание данных между вычислениями вызывает простой вычис-

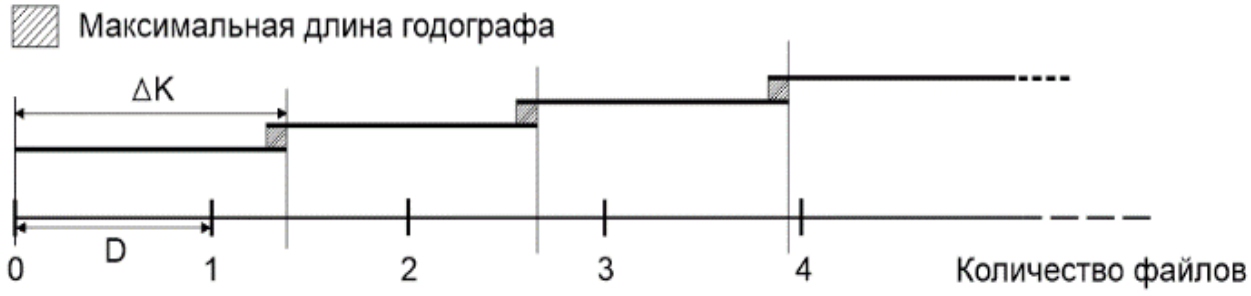

Рис. 9. Размещение данных с сейсмоприемников в файлах и их обработка

лительного оборудования. Чтобы скрыть задержки считывания данных из файлов, применяется двойная буферизация. Два буфера данных формируются в глобальной памяти GPU и поочередно заполняются данными для обработки. Таким образом, задержек ожидания новой порции данных не происходит, и GPU на протяжении всего времени работы программы занято вычислениями.

Pinned-память. Для уменьшения времени обмена данными между устройством и хостом используется pinned-память (память, которую запрещено выгружать из ОЗУ). В ходе тестирования была замерена скорость копирования данных из выгружаемой памяти (обычной) хоста и из pinned-памяти в глобальную память GPU: 3,29 и 6,36 GB/s соответственно, а также копирования из глобальной памяти GPU в выгружаемую память и pinned-память хоста: 3,26 и 6,34 GB/s соответственно, для карты Tesla C2050. Таким образом, использование pinned-памяти позволяет сократить время обмена данными практически в два раза (шина PCI 2.0).

Оценка эффективности использования GPU

С использованием всех вышеупомянутых оптимизаций были получены следующие значения по эффективности использования ресурсов (платформа GTX 460SE, архитектура Fermi, Compute Capability 2.1): достигнутая скорость обмена данными 74,78 GB/s; достигнутая скорость обработки данных 16,53 GFLOPS + 255,73 GIOPS, что соответствует 72 % и 71 % от пиковой пропускной способности подсистемы памяти и пиковой производительности арифметического устройства соответственно для задачи с использованием 1920 сейсмоприемников и длиной формируемой суммотрассы, равной 15000 отсчетам по времени.

ТЕСТИРОВАНИЕ

Тестирование производилось на следующих GPU NVIDIA:

1. Tesla C2050 (Fermi, Compute Capability 2.0);
2. GeForce GTX 460SE (Fermi, Compute Capability 2.1);
3. Tesla K20 (Kepler, Compute Capability 3.5);
4. GeForce GTX 950 (Maxwell, Compute Capability 5.2).

Исходные данные для тестирования (рис. 1) представляют собой файл, имитирующий результаты непрерывного мониторинга для 1920 сейсмоприемников в течение определенного отрезка времени. Размер файла варьируется в зависимости от проводимого теста, например, непрерывный мониторинг для указанного количества сейсмоприемников в течение одного часа с шагом по времени $h = 0.002$ с составляет 12,8 GB.

Для всех обозначенных GPU было проведено тестирование различного объема данных при одинаковом размере сетки пробных источников равным 10^3. Размер тестовых данных составлял от 12,8 до 819,2 GB, что соответствует от 1 до

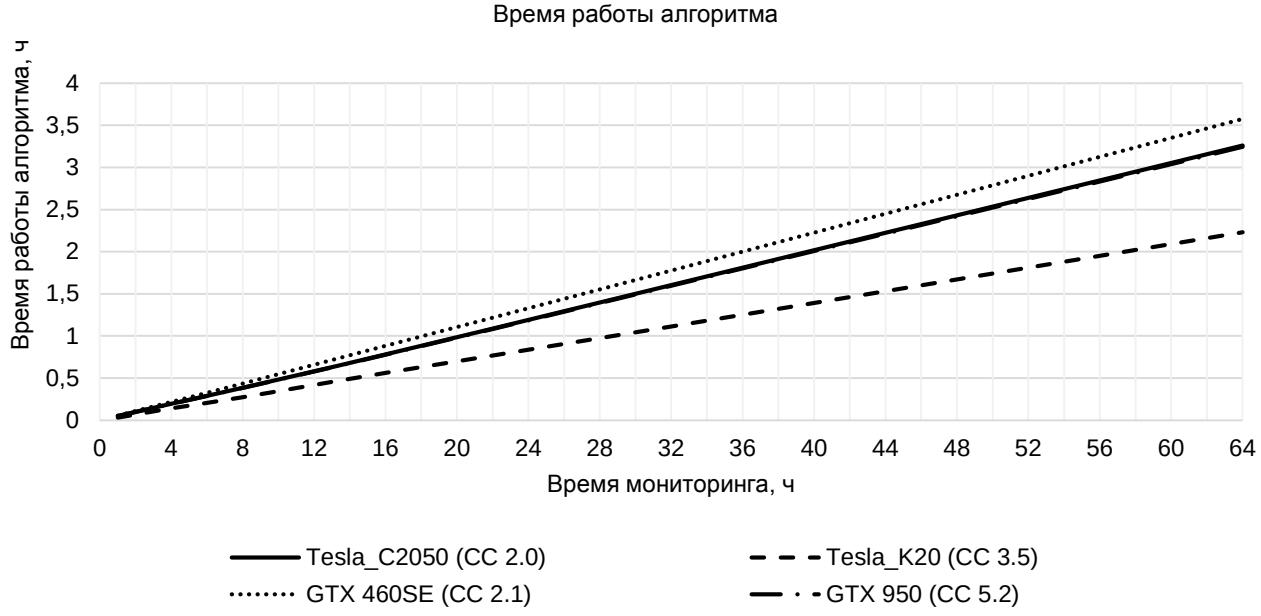

Рис. 10. Время работы алгоритма на различных платформах

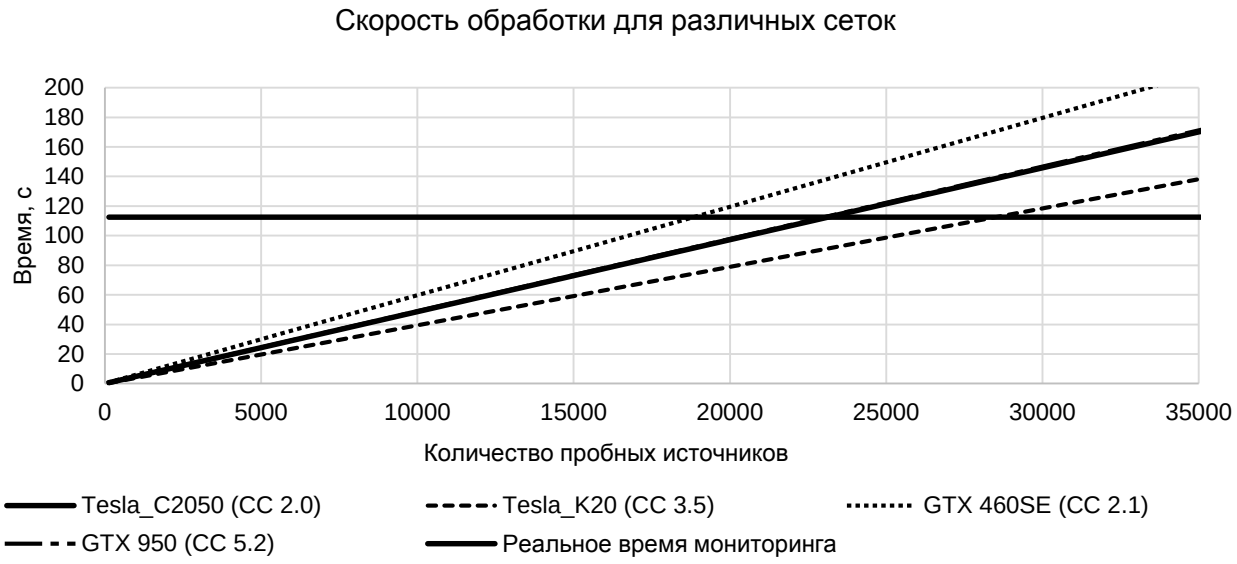

Рис. 11. Скорость обработки данных для различных сеток пробных источников

64 часов мониторинга. На рис. 10 представлен график зависимости времени обработки от объема данных.

По результатам тестирования сделан вывод, что скорость обработки не зависит от обрабатываемого объема данных и время работы алгоритма увеличивается линейно относительно объема исходных данных.

При масштабировании трехмерной сетки пробных источников не наблюдается каких-либо существенных изменений в скорости обработки. На рис. 11 показана динамика по скорости обработки данных для различных размеров сеток пробных источников, горизонтальная линия обозначает реальное время мониторинга.

Время, затрачиваемое на работу алгоритма, так же, как и в случае с объемом исходных данных, растет линейно с увеличением сетки. Исходя из результатов тестирования, можно сделать вывод, что для потоковой обработки данных в режиме реального времени возможно использовать объемные сетки пробных источников от 25^3 до 30^3 для платформ от GTX 460SE до Tesla K20 соответственно.

Операции копирования данных между жестким диском, оперативной памятью хоста и глобальной памятью устройства, работающие в отдельном потоке CPU, не превышают времени работы ядра GPU и полностью «маскируются» за вычислениями в случае, если не производится сохранение всех суммотрасс на диск. В противном случае, копирование успешно маскируется, если объем суммотрасс не превышает исходного объема исследуемых данных мониторинга (количество пробных источников меньше количества приемников).

ЗАКЛЮЧЕНИЕ

Разработан алгоритм реализации метода когерентного суммирования на GPU, позволяющий эффективно использовать ресурсы вычислительного устройства и обрабатывать большие объемы данных. Разработанный алгоритм подходит для использования как на специализированных вычислителях, так и обычных домашних графических картах, поддерживающих программную модель NVIDIA CUDA. После проведенных оптимизаций с использованием различных типов памяти хоста и устройства удалось добиться загруженности устройства на уровне 70 % от пиковой производительности как подсистемы памяти, так и арифметического устройства. Выявлена зависимость скорости обработки данных от пропускной способности памяти устройства. Также из-за нерегулярного обращения в память нет возможности эффективно использовать кэш-память устройства без дополнительных накладных расходов по реструктуризации данных, которые значительно увеличивают общее время работы программы. Выявлен эффект от реализации двойной буферизации: обработка данных производится на фоне считывания очередной порции данных, что позволило практически полностью избавиться от задержек, связанных со считыванием. Проведен анализ и тестирование разработанной программы на специализированных платформах NVIDIA: Tesla C2050 (архитектура Fermi), Tesla K20 (архитектура Kepler) и на обычных видеокартах, не предназначенных для высокопроизводительных вычислений: GTX 460 SE (архи-

тектура Fermi), GTX 950 (архитектура Maxwell). Показана линейная зависимость времени обработки от объема исходных данных и размеров сеток пробных источников. По результатам тестирования выявлены оптимальные данных в режиме реального времени для различных платформ. Таким образом, для потоковой обработки данных в режиме реального времени возможно использовать объемные сетки пробных источников от 25^3 до 30^3 для платформ от GTX 460SE до Tesla K20 соответственно, при одинарной точности вычислений.

СПИСОК ЛИТЕРАТУРЫ

1. **Лесной Г. Д., Мерщий В. В., Опанасенко А. С**. Обработка сейсморазведочных данных на основе параллельных вычислений с использованием графических процессоров // Міжнародна конференція "Високопродуктивні обчислення" (Україна, Київ, 8-10.10.2012). С. 235–242. [G. D. Lesnoy, V. V. Mershchiy, A. S. Opanasenko, "*Processing of seismic data based on parallel computing using GPUs*" (in Russian), in *Proc. 2nd Int. Conf. On High Performance Computing* (HPC-UA 2012), Kiev, 8-10.10.2012, pp. 235-242]

2. **Курин Е. А**. Сейсморазведка и суперкомпьютеры // Выч. методы и программирование 12.1 (2011). С. 38–43. [E. A. Kurin, "*Seismic processing and supercomputers*" (in Russian), in *Vychislitel'nye metody i programmirovanie* 12.1 (2011), pp. 38-43]

3. **Morton S.** Industrial Seismic Imaging on a GPU Cluster. [Электронный ресурс]. URL: http://www.nvidia.com/content/PDF/sc_2010/CUDA_Tutorial/SC10_Industrial_Seismic_Imaging_on_a_GPU_Cluster.pdf (дата обращения: 30.11.2015). [S. Morton (2015, Nov. 30). "*Industrial Seismic Imaging on a GPU Cluster*" [Online]. Available: http://www.nvidia.com/content/PDF/sc_2010/CUDA_Tutorial/SC10_Industrial_Seismic_Imaging_on_a_GPU_Cluster.pdf]

4. **Fast** seismic modeling and reverse time migration on a GPU cluster / Rached A. [и др.] // In Proc. Int. Conf. on High Performance Computing & Simulation, IEEE, 2009 (HPCS'09), Leipzig, P. 36–43. [A. Rached, et. al., "*Fast seismic modeling and reverse time migration on a GPU cluster*", in *Proc. Int. Conf. on High Performance Computing & Simulation, IEEE, 2009* (HPCS'09), Leipzig, pp. 36-43]

5. **Accelerating** large-scale simulation of seismic wave propagation by multi-GPUs and three-dimensional domain decomposition / Taro O. [и др.] // Earth, planets and space, Vol. 62, Iss. 12, 2010: P. 939–942. [O. Taro, et al., "*Accelerating large-scale simulation of seismic wave propagation by multi-GPUs and three-dimensional domain decomposition*", in Earth, planets and space, Vol. 62, Iss. 12 , 2010, pp. 939-942]

6. **Каневская Р. Д.** Математическое моделирование разработки месторождений нефти и газа с применением гидравлического разрыва пласта. М.: Недра, 1999. 212 с. [R. D. Kanevskaya, "*Mathematical modeling of development of oil and gas fields using hydraulic fracturing*" (in Russian). M.: Nedra, 1999. 212 p]

7. **Николаев А. В., Троицкий П. А, Чеботарева И. Я**. Изучение литосферы сейсмическими шумами // Доклады АН СССР. 1986. Т. 286, № 3. С. 586–591. [A. V. Nikolaev, P. A. Troitskiy, and I. Ya. Chebotareva. "The study of the lithosphere seismic noise" (in Russian), in Doklady AN SSSR. 1986. T. 286, № 3, pp. 586-591]

8. **CUDA C Programming Guide** [Электронный ресурс]. URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#axzz3aOqAdBFv (дата обращения: 30.11.2015). [**CUDA C Programming Guide** (2015, Nov. 30) [Online]. Available:. URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#axzz3aOqAdBFv]

9. **Оценка** модели среды по полному векторному полю ВСП / Ю. А. Степченков [и др.] // Технологии сейсморазведки, №. 02. 2006. С. 19–23. [Yu. A. Stepchenkov, et. al., "Evaluation of environment model with the full vector field of VSP" (in Russian), in Tekhnologii seysmorazvedki, Iss. 02, 2006, pp. 19-23]

ОБ АВТОРАХ

ГОРОДНИЧЕВ Максим Александрович, мл. науч. сотр. ИВМиМГ СО РАН, асс. каф. параллельных вычислительных технологий НГТУ, асс. каф. параллельных вычислений НГУ. Магистр прикладной математики и информатики (НГТУ, 2004). Иссл. в обл. распределенных вычислительных систем, параллельного программирования, суперкомпьютерных вычислений.

ДУЧКОВ Антон Альбертович, зав. лаб. динамических проблем сейсмики, с.н.с. ИНГГ СО РАН. Доц. каф. геофизики ГГФ НГУ. Канд. физ.-мат. наук (ИГФ СО РАН, 2004). Иссл. в обл. эффективных алгоритмов обработки сейсмических данных, сейсмической миграции, регуляризации сейсмических данных, микросейсмического мониторинга.

САРЫЧЕВ Виктор Геннадьевич, магистрант каф. параллельных вычислительных технологий НГТУ. Бакалавр математики (НГТУ, 2014). Готовит дис. об оптимизации параллельных программ для решения задач сейсмики и визуальном конструировании параллельных программ с использованием теории структурного синтеза программ на вычислительных моделях.

METADATA

Title: Efficient GPU-Implementation of Coherent Stacking with CUDA.

Authors: M. A. Gorodnichev[1,2,3], A. A. Duchkov[2,4], V. G. Sarychev[1,4]

Affiliation:

[1] Novosibirsk State Technical University (NSTU), Russia.

[2] Novosibirsk State University (NSU), Russia.

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

[4] Trofimuk Institute of Petroleum Geology and Geophysics SB RAS (IPGG SB RAS), Russia.

Email: [1] maxim@ssd.sscc.ru, [2] DuchkovAA@ipgg.sbras.ru, [3] sarychevmail@gmail.com.

Language: Russian.

Source: Vestnik UGATU (scientific journal of Ufa State Aviation Technical University), vol. 20, no. 1 (71), pp. 151–160, 2016. ISSN 2225-2789 (Online), ISSN 1992-6502 (Print).

Abstract: Coherent stacking is a key procedure for a class of algorithms that are used to process seismic data. The paper presents an efficient implementation of coherent stacking algorithm on CUDA-based GPUs. We discuss a complex of optimizations that allowed the implementation to reach 70% of peak hardware performance. Tests reveal linear dependency between computing time and problem size. Terabytes of seismic data cannot be placed into the memory of GPU card at once and thus the processing must be organized in portions. Optimal portion sizes where found for the following generations of Nvidia GPUs: Fermi, Kepler, Maxwell.

Key words: graphics cards, parallel computing, CUDA, GPU, geophysics, optimization, GPU architecture, coherent stacking, algorithm development, big data, data processing, seismic.

About authors:

GORODNICHEV, Maxim Aleksandrovich, Jr. researcher at the Institute of Computational Mathematics and Mathematical Geophysics SB RAS, ass. prof. at Novosibirsk State University, Novosibirsk State Technical University. Interests: distributed computing systems, parallel programming technologies, supercomputing.

DUCHKOV, Anton Albertovich, Head of Laboratory of Dynamic Problems in Seismic, Trofimuk Institute of Petroleum Geology and Geophysics SB RAS, PhD in Geophysics, 2004. Interests: microseismic monitoring, seismic data processing, regularization, and imaging.

SARYCHEV, Victor Gennadjevich, M.Sc. student at the dept. of parallel computing technologies, Novosibirsk State Technical University. BSc in Mathematics, NSTU, 2014. Interests: parallel algorithms and efficient programs for seismic data processing, high-level visual parallel programming systems, parallel program synthesis.