# Программная реализация метода когерентного суммирования на GPU с использованием программной модели NVIDIA CUDA

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

Новосибирский государственный технический университет[1], Новосибирский государственный университет[2], Институт вычислительной математики и математической геофизики СО РАН[3], Институт нефтегазовой геологии и геофизики им. А.А.Трофимука СО РАН[4]

Рассматривается подход к обработке больших объемов данных сейсмического мониторинга методом когерентного суммирования с использованием современных графических платформ поддерживающих программную модель NVIDIA CUDA. Приводятся оптимизации, направленные на повышение эффективности использования GPU позволяющие добиться до 70% от пиковой производительности подсистемы памяти и арифметического устройства. Приводятся результаты тестирования реализованного метода, по которым выявлены линейные зависимости по времени обработки данных от размеров сеток исследуемого пространства и объема обрабатываемых данных соответственно. Выявлены оптимальные параметры, для реализации потоковой обработки данных в режиме реального времени для различных аппаратных архитектур графических процессоров NVIDIA: Fermi, Kepler, Maxwell.

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

## 1. 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. Образования 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 GPU, поддерживающих программную модель NVIDIA CUDA [8]. Рассматриваются оптимизации по работе с различными типами памяти GPU и приводится анализ степени загруженности GPU при различных конфигурациях исполь-

зуемой 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.

## 2. 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 матрицы $d_r(t_k)$, где $r = 1,...,R$ индекс приемника с координатами $(\alpha_r, \beta_r, \gamma_r)$, $t_k$ – отсчеты по времени, $k = 1,...,K$. 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 последовательных интервалов времени, расположенные последовательно друг за другом (см. рис. 1). Координаты приемников $(\alpha_r, \beta_r, \gamma_r)$ располагаются в отдельном файле.

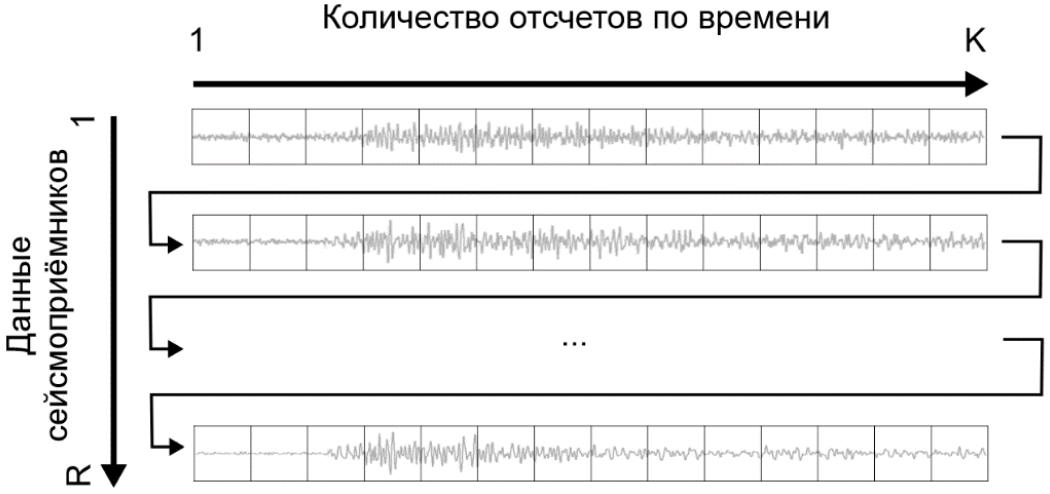


**Рис. 1.** Формат данных с сейсмоприемников

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 $j = 1,..., J$ – индекс пробного источника, которая покрывает исследуемое пространство. Далее происходит перебор узлов сетки и данные мониторинга для каждого момента времени $\tau_k$ суммируются по годографу прямой волны из пробного источника в этом узле. Так, для каждого узла вычисляется сумматрасса:

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

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:

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

где $\tau_k$ – время возникновения события, $k$ – номер отсчета по времени.

Положение источника определяется, как максимум так называемого куба когерентности:

$$\tilde{s}\left(x_j, y_j, z_j\right) = \max_k\left(s\left(x_j, y_j, z_j, \tau_k\right)\right), \quad (3)$$

где время $\tau_k$ является временем возникновения сейсмического события. В общем случае источников может быть несколько.

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

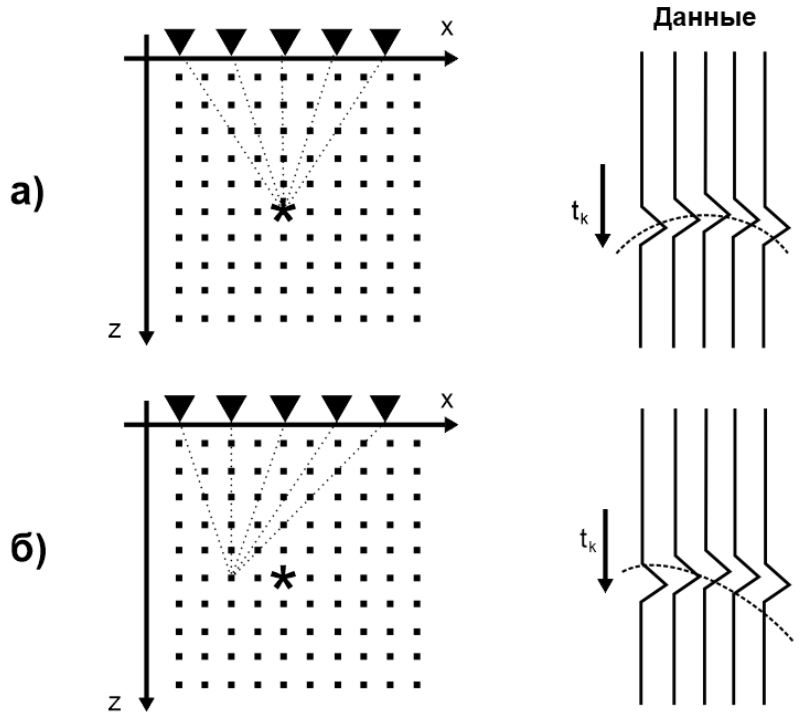


**Рис. 2.** Алгоритм когерентного суммирования

## 3. Реализация алгоритма на GPU

Расчет сумматрасс является наиболее ресурсоемкой процедурой и на ее выполнение уходит большая часть времени. На рис. 3 показан общий алгоритм формирования сумматрасс.

```
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 сумматрасс

Так как количество отсчетов по времени $K$ очень велико, невозможно поместить всю длину сумматрассы в память вычислительного устройства целиком. Для решения данной проблемы предлагается разбить длину $K$ на $N$ равных отрезков длиной $\Delta K$ ($K = N \cdot \Delta K$)., которые бы целиком помещались в памяти устройства. 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 $\Delta K$ будет зависеть количество используемых ресурсов вычислительного устройства и скорость формирования сумматрасс. Таким образом, алгоритм формирования сумматрасс 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 сумматрасс

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 $R$. Каждая нить GPU осуществляет расчет (2) времени прихода прямой волны для одного сейсмоприемника. Полученное значение записывается в глобальную память устройства и используются в дальнейшем для получения сумматрасс.

Сумматраса (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.


**Рис. 5.** Распределение вычислений по потокам

Каждый поток GPU (нить) 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.

# 4. Оптимизация

В данной части работы приведены следующие оптимизации:

4.1. Оптимизация выбора типов памяти для хранения различных величин;

4.2. выбор длины сейсмотрасс $\Delta K$;

4.3. 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).

## 4.1 Использование различных типов памяти GPU

На устройствах, поддерживающих программную модель CUDA, присутствуют различные типы памяти. Грамотное использование каждого из них, в большей степени, определяет скорость работы программы [10].

**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 не целесообразным. 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 сумматрасс 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 сумматрасс 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, считывается напрямую из непрерывного блока глобальной памяти, однако, попадание в кэш L1, и группировка запросов в память не гарантируется 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 $\Delta K$, 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 влияния оптимизаций по работе с памятью на скорость обработки данных. Влияние кэша L1, а точнее, попадание в L1 кэш можно оценить по количеству посчитанных элементов сумматрасс 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 ну уровне 55-65% является наиболее эффективным вариантом реализации данного метода на используемых GPU (архитектура Fermi).

Для данного метода 100% попадания в кэш L1 при расчете сумматрасс невозможно обеспечить без дополнительных накладных расходов по реструктуризации данных в памяти. Эти накладные расходы, реализованные на программном уровне, будут превышать время работы программы с 55-65% попаданием в кэш L1, что делает их нецелесообразными.

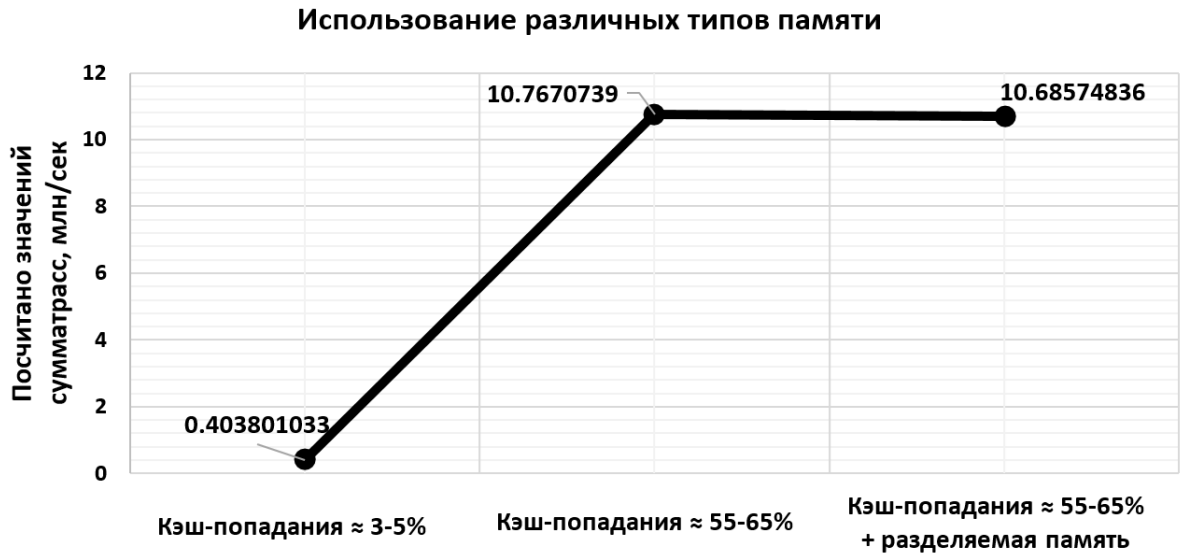


**Рис. 6.** Использование различных типов памяти. GM_CM (Global Memory Cache Miss) – использование глобальной памяти устройства со значительными кэш промахами; GM_CH+SM (Global Memory Cache Hit + Shared Memory) – использование глобальной памяти устройства с кэш-попаданиями, а также использование разделяемой памяти

### 4.2 Выбор длины сейсмотрасс $\Delta K$

От количества сейсмоприемников $R$ и длины трасс $\Delta K$ зависит количество производимых операций по расчету сумматрасс. 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 $\Delta K$ ситуация несколько отличается. При малой длине трассы не получается загрузить устройство «полезной работой», так как количество используемых потоков напрямую зависит от выбранной длины $\Delta K$, но 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 $\Delta K$ 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 сумматрасс 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 $\Delta K$, порядка 100 тысяч, для количества сейсмоприемников в диапазоне 200-300 штук, при увеличении числа сейсмоприемников начинает сказываться влияние кэш-промахов при доступе к глобальной памяти GPU и необходимо уменьшать длину рассчитываемых сейсмотрасс $\Delta K$ 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.

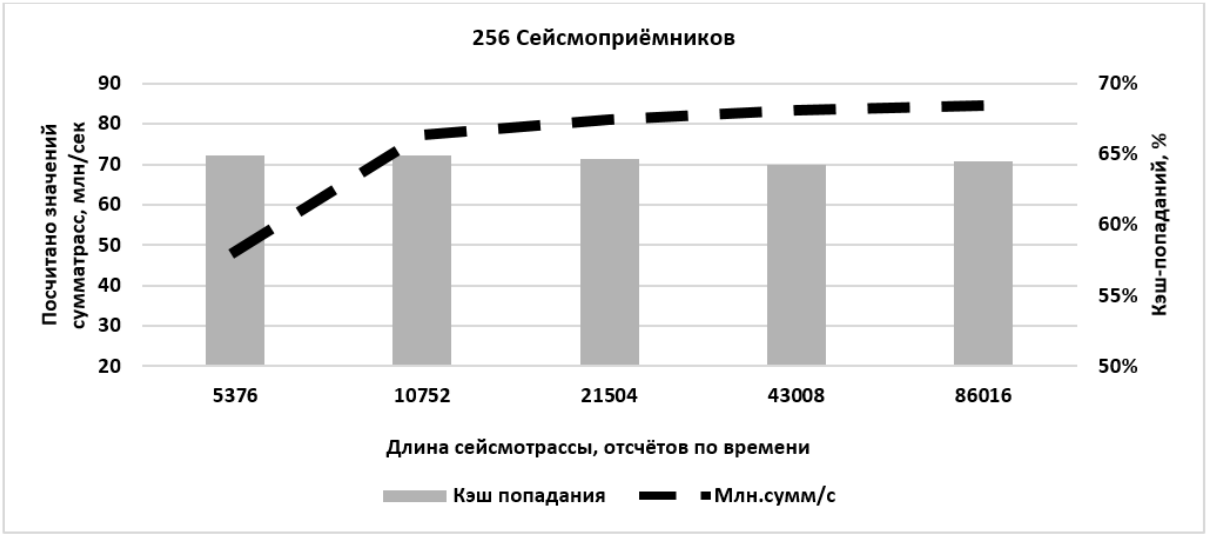


**Рис. 7.** Зависимость времени формирования сумматрасс от количества сейсмоприемников, длины трасс, и процента кэш-попаданий в L1 кэш. 256 сейсмоприемников

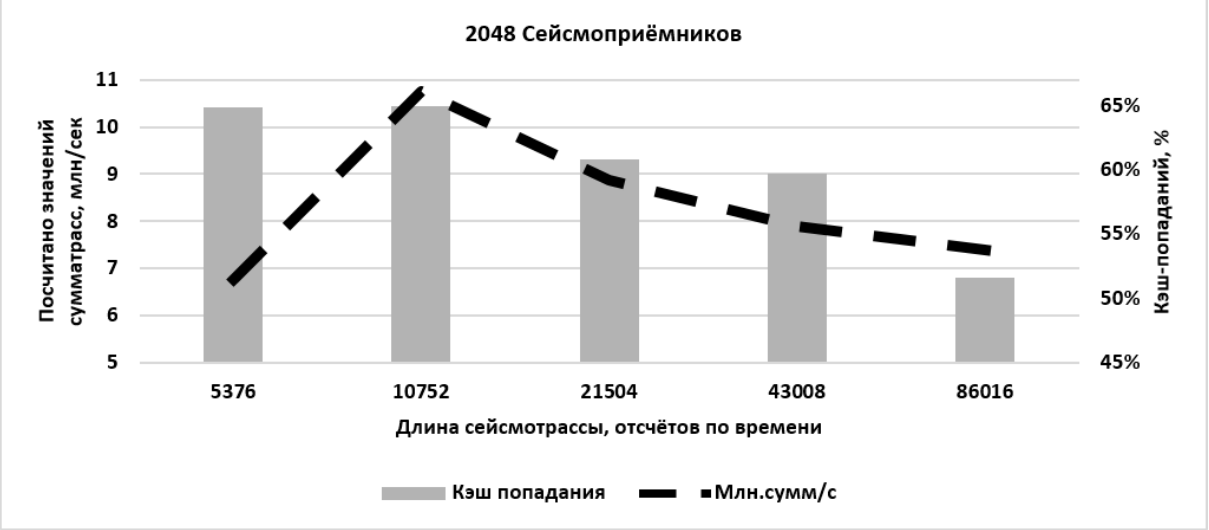


**Рис. 8.** Зависимость времени формирования сумматрасс от количества сейсмоприемников, длины трасс, и процента кэш-попаданий в L1 кэш. 2048 сейсмоприемников

## 4.3 Оптимизация работы с жестким диском

При обработке данных, в зависимости от длины обрабатываемой трассы $\Delta K$, возникают ситуации, в которых необходимо считывать данные из разных файлов. При этом необходимо учитывать, что длина сумматрассы, в процессе обработки, складывается из суммы самой длины сумматрассы 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 $\Delta K$ 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.

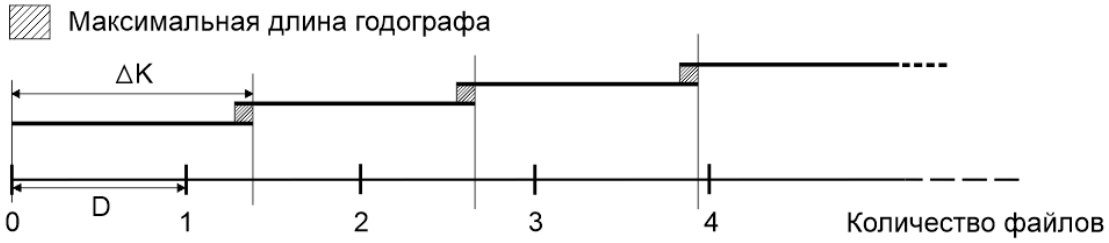


**Рис. 9.** Размещение данных с сейсмоприемников в файлах и их обработка

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

### 4.4 Оценка эффективности использования GPU

Используя все вышеупомянутые оптимизации, 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 40.53 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 длинной формируемой сумматрассы равной 15000 отсчетам по времени.

## 5. 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 (структура данных описана в п.2, рис. 1) представляют собой файл, имитирующий непрерывный мониторинг для 1920 сейсмоприемников в течении определенного отрезка времени. Размер файла варьируется в зависимости от проводимого теста, например, непрерывный мониторинг для указанного количества сейсмоприемников в течении одного часа с шагом по времени $h = 0.002$ с, составляет 12.8 Гб.

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 Гб, что соответствует от 1 до 64 часов мониторинга. На рис. 10. представлен график по времени обработки в зависимости от объема данных.

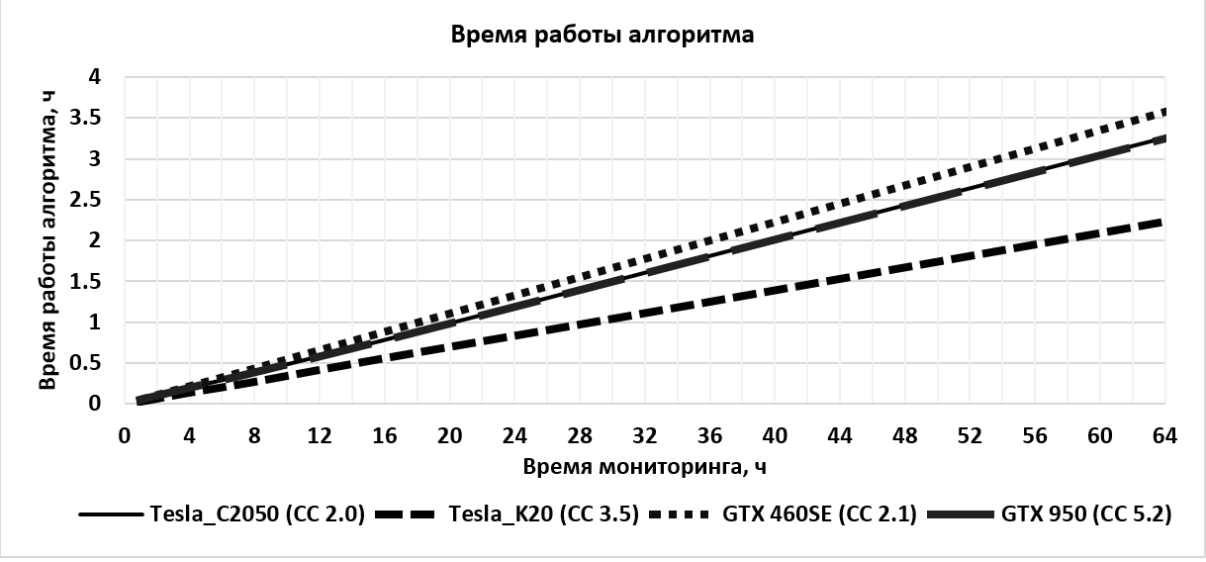


**Рис. 10.** Время работы алгоритма на различных платформах

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 об-

работки данных для различных размеров сеток пробных источников, горизонтальная линия обозначает реальное время мониторинга.

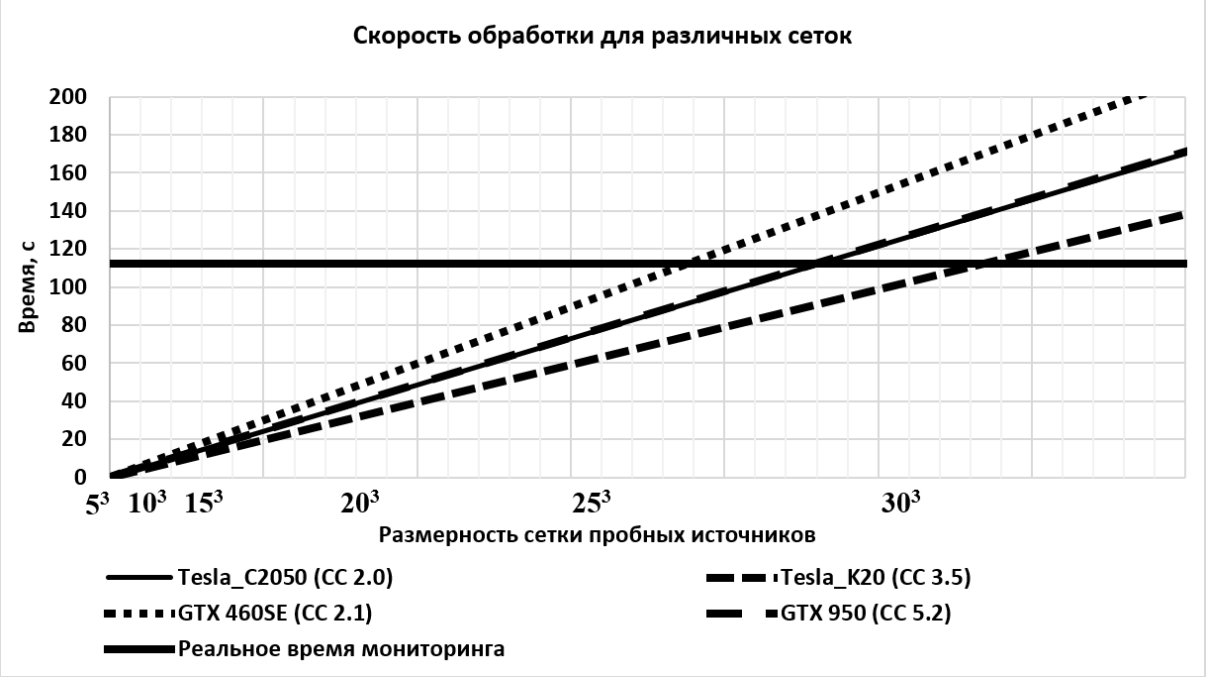


**Рис. 11.** Скорость обработки данных для различных сеток пробных источников

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 чтения/записи 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 объем получаемых данных (сумматрасс) не превышает исходного объема исследуемых данных мониторинга и есть необходимость в их сохранении на жесткий диск.

## 6. 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. 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 (архитектура Fermi), GTX 950 (архитектура Maxwell). Показана линейная зависимость времени обработки от объема исходных данных и размеров сеток пробных источников. По результатам тестирования выявлены оптимальные параметры для реализации потоковой обработки данных в режиме реального времени для различных платформ. Таким образом для потоковой обработки данных в режиме реального времени возможно ис-

пользовать объемные сетки пробных источников от $25^3$ до $30^3$ для платформ от GTX 460SE до Tesla K20 соответственно, при одинарной точности вычислений.

## Литература

1. Лесной, Г. Д., В. В. Мерщий, А. С. Опанасенко. Обработка сейсморазведочных данных на основе параллельных вычислений с использованием графических процессоров // Міжнародна конференція "Високопродуктивні обчислення" HPC-UA'2012 (Україна, Київ, 8-10 жовтня 2012 року). – С. 235-242
2. Курин Е. А. Сейсморазведка и суперкомпьютеры // Вычислительные методы и программирование 12.1 (2011): С. 38-43.
3. Morton S. Industrial Seismic Imaging on a GPU Cluster Geophysical Technology HessCorporation. 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).
4. Rached A., Calandra H., Coulaud O., Roman J., and Latu G. "Fast seismic modeling and reverse time migration on a GPU cluster." In High Performance Computing & Simulation, 2009. HPCS'09. International Conference on, pp. IEEE, 2009. P. 36-43.
5. Taro O., Takenaka H., Nakamura T., and Aoki T. "Accelerating large-scale simulation of seismic wave propagation by multi-GPUs and three-dimensional domain decomposition." Earth, planets and space62, no. 12 (2010): P. 939-942.
6. Каневская, Р. Д. Математическое моделирование разработки месторождений нефти и газа с применением гидравлического разрыва пласта. М.: Недра, 1999. 212 с.
7. Николаев А.В., Троицкий П.А, Чеботарева И.Я. Изучение литосферы сейсмическими шумами // Доклады АН СССР. 1986. Т. 286, № 3. С. 586–591
8. CUDA C Programming Guide. URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#axzz3aOqAdBFv (дата обращения: 30.11.2015).
9. Степченков, Ю. А., Табаков, А. А., Решетников, А. В., Рыковская, Н. В., и Баранов, К. В. (2006). Оценка модели среды по полному векторному полю ВСП. Технологии сейсморазведки, (02), С. 19-23.
10. А.В. Боресков, А.А. Харламов, Н.Д. Марковский, Д.Н. Микушин, Е.В. Мортиков, А.А. Мильцев, Н.А.Сахарных, В.А. Фролов. Параллельные вычисления на GPU. Архитектура и программная модель CUDA. Москва: Издательство Московского университета, 2012. 333 с.

# Efficient GPU-Implementation of Coherent Stacking with CUDA

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

Novosibirsk State Technical University[1], Novosibirsk State University[2], Institute of Computational Mathematics and Mathematical Geophysics SB RAS[3], Trofimuk Institute of Petroleum Geology and geophysics SB RAS[4]

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 can not 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.

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

## References

1. Lesnoy, G. D., V. V. Mershchiy, A. S. Opanasenko. Obrabotka seysmorazvedochnykh dan-nykh na osnove parallel'nykh vychisleniy s ispol'zovaniem graficheskikh protsessorov [Processing of seismic data based on parallel computing using GPUs]. Mizhnarodna konferencija "Vysokoproduktyvni obchyslennja" HPC-UA'2012 (Ukrai'na, Kyi'v, 8-10 zhovtnja 2012 roku) [International Conference "HPC" HPC-UA'2012 (Kiev, October 8-10, 2012)]. – P. 235-242
2. Kurin E. A.. Seysmorazvedka i superkomp'yutery [Seismic processing and supercomputers]. Vychislitel'nye metody i programmirovanie 12.1 (2011). [Numerical Methods and Programming 12.1 (2011)]. P. 38-43.
3. Morton S. Industrial Seismic Imaging on a GPU Cluster Geophysical Technology HessCorporation. URL:http://www.nvidia.com/content/PDF/sc_2010/CUDA_Tutorial/SC10_Industrial_Seismic_Imaging_on_a_GPU_Cluster.pdf (accessed: 30.11.2015).
4. Rached A., Calandra H., Coulaud O., Roman J., and Latu G. "Fast seismic modeling and reverse time migration on a GPU cluster." In High Performance Computing & Simulation, 2009. HPCS'09. International Conference on, pp. IEEE, 2009. P. 36-43.
5. Taro O., Takenaka H., Nakamura T., and Aoki T. "Accelerating large-scale simulation of seismic wave propagation by multi-GPUs and three-dimensional domain decomposition." Earth, planets and space62, no. 12 (2010): P. 939-942.
6. Kanevskaya, R. D. Matematicheskoe modelirovanie razrabotki mestorozhdeniy nefti i gaza s primeneniem gidravlicheskogo razryva plasta. [Kanev, RD Mathematical modeling of development of oil and gas using hydraulic fracturing]. M.: Nedra, 1999. 212 p.
7. Nikolaev A.V., Troitskiy P.A., Chebotareva I.Ya. Izuchenie litosfery seysmicheskimi shumami [The study of the lithosphere seismic noise]. Doklady AN SSSR [Reports of the USSR Academy of Sciences]. 1986. T. 286, № 3. P. 586–591
8. CUDA C Programming Guide. URL: http://docs.nvidia.com/cuda/cuda-c-programming-

guide/index.html#axzz3aOqAdBFv (accessed: 30.11.2015).

9. Stepchenkov, Yu. A., Tabakov, A. A., Reshetnikov, A. V., Rykovskaya, N. V., & Baranov, K. V. (2006). Otsenka modeli sredy po polnomu vektornomu polyu VSP [Evaluation model environment for the full vector field of the GSP]. Tekhnologii sey-smorazvedki, (02) [Seismic technology, (02)]. P. 19-23.
10. A.V. Boreskov, A.A. Kharlamov, N.D. Markovskiy, D.N. Mikushin, E.V. Mortikov, A.A. Mil'tsev, N.A.Sakharnykh, V.A. Frolov. Parallel'nye vychisleniya na GPU. Arkhitektura i programmnaya model' CUDA. [Parallel computing on the GPU. Architecture and CUDA programming model]. Moskva: Izdatel'stvo Moskovskogo universiteta, 2012 [Moscow: Publishing House of Moscow University, 2012]. – 333 p.