СИСТЕМА ТЕСТОВ ДЛЯ ОПРЕДЕЛЕНИЯ ХАРАКТЕРИСТИК ПРОИЗВОДИТЕЛЬНОСТИ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

А.В. Адинец, П.А. Швец

Введение

В последнее время часто для решения вычислительных задач используются программируемые ускорители, в число которых входят, например, графические процессорные устройства (ГПУ), процессоры CELL BE [1] и процессоры ClearSpeed [2]. Одномашинные и кластерные системы на их основе в совокупности составляют класс неоднородных (гетерогенных) вычислительных систем. Наиболее перспективными в настоящее время представляются гетерогенные системы на основе ГПУ, что определяет актуальность задачи исследования и анализа производительности программ для таких систем, выявлению в них узких мест, определению потенциала для оптимизации и тонкой настройки.

Достаточно типичной является ситуация, когда пользователь написал программу для ГПУ, запустил её на гетерогенном кластере, а обещанного 100-кратного ускорения не получил. Такое происходит по целому ряду причин. Данные могли быть неудачно расположены в памяти, в результате обращения к данным соседних потоков не объединялись в одну транзакцию, и скорость чтения падала на порядок. Или же могло быть запущено слишком мало потоков, и их было недостаточно для сокрытия задержки доступа. Или соседние потоки могли расходиться на условных операторах. Или на ГПУ всё хорошо, но передача данных с хоста и обратно съедает все ресурсы. Разумеется, список можно продолжать ещё достаточно долго. Соответственно, обо всех подобных узких местах и причинах низкой производительности хотелось бы знать, чтобы иметь возможность их обходить. Кроме того, хотелось бы иметь стандартные примеры, которые показывали бы возможности по устранению таких узких мест, если они встречаются в конкретных программах.

Архитектура графических ускорителей сильно отличается от традиционных процессоров, и для написания высокоэффективных приложений необходимо не только понимать архитектуру, но и уметь предсказывать эффективность разных программных конструкций, учитывая особенности работы графического ускорителя. Некоторые характеристики и особенности прямо указываются производителями графических ускорителей. Однако другие, значения которых нельзя получить через стандартные интерфейсы, и которые приходится определять эмпирически, можно узнать, лишь столкнувшись с ними в какой-то конкретной задаче. Для решения таких проблем нужна система тестов, которая позволит определять неявные характеристики конкретного ГПУ, а также позволит определить производительность ГПУ на базовых и оптимизированных версиях некоторых стандартных вычислительных и/или алгоритмических ядер, и выявить возможные пути для оптимизации других ядер такого типа.

Задача данной работы — создание системы тестов, позволяющих измерить различные явные и неявные характеристики графических ускорителей. Создаваемые тесты должны позволять измерять такие характеристики ГПУ, как задержка и пропускная способность различных типов памяти, эффективность атомарных операций и размер кэш-линии.

Тесты требуется апробировать системах с ГПУ различных производителей. В качестве тестовых ГПУ в работе используются Tesla C1060, Tesla C2050, Radeon HD5830, а также ГПУ NVidia Tesla M2050, установленные в кластере «ГрафИТ!» НИВЦ МГУ[3].

В качестве языка для реализации системы тестов был выбран Nemerle и набор расширений NUDA(Nemerle Unified Device Architecture [4]), позволяющий писать программы для ГПУ с минимальными отличиями от обычных без ущерба для производительности. Система расширений NUDA в качестве нижнего уровня использует язык и среду исполнения OpenCL[5]. Таким образом, использование NUDA позволяет упростить техническую часть работы и обеспечить возможность работы системы тестов на графических ускорителях и от основных производителей: NVidia и от AMD.

Уже существуют некоторые похожие системы, но они не в полной мере реализуют требуемый функционал. Например, в работе [6] описывается набор тестов, измеряющих разные неявные параметры ГПУ архитектуры GT200 от NVidia. Имеется ряд тестов, специфичных только для ГПУ от NVidia. Но этот набор тестов не поддерживает архитектуру AMD. Набор тестов [7], наоборот, измеряет разные неявные параметры ГПУ архитектуры RV670, RV770, RV870 от AMD. Имеет ряд тестов, специфичных только для этих архитектур, и работает только на архитектурах AMD. Набор тестов SHOC [8] является переносимым набором бенчмарков; кроме того, для него имеются результаты измерений на большом числе карточек AMD и NVidia. В этой системе достаточный набор практических задач и бенчмарков, но недостаточно внимания уделено низкоуровневым тестам.

Ни одна из этих систем тестов не исследует все интересующие аспекты; например, ни один из них не исследует эффективность атомарных операций. Эти аспекты реализованы в созданной нами системе тестов. Кроме того, разрабатываемый набор будет совмещать в себе и тесты, и примеры оптимизаций, а также будет

переносим между карточками различных производителей. Наконец, при помощи переносимого средства программирования могут быть созданы в том числе и тесты специфичных для конкретных архитектур параметров.

Тест эффективности атомарных операций

В тесте сравнивается эффективность атомарных операций разного типа — локальных и глобальных, с обычными неатомарными операциями. Локальные атомарные операции работают с локальной памятью и в связи с этим гарантируют атомарность только в пределах одного блока нитей. Глобальные атомарные операции работают с глобальной памятью и гарантируют атомарность выполнения среди всех нитей. Исследуется также влияние конфликтности шаблона доступа к памяти на эффективность.

Ниже показаны результаты теста для бесконфликтного доступа(рис. 1) и доступа с шаблоном «все-в-один» (рис. 2). Как видно, они отличаются по скорости на порядок, за исключением локальных атомарных операций на карточках AMD, которые показывают себя довольно хорошо.

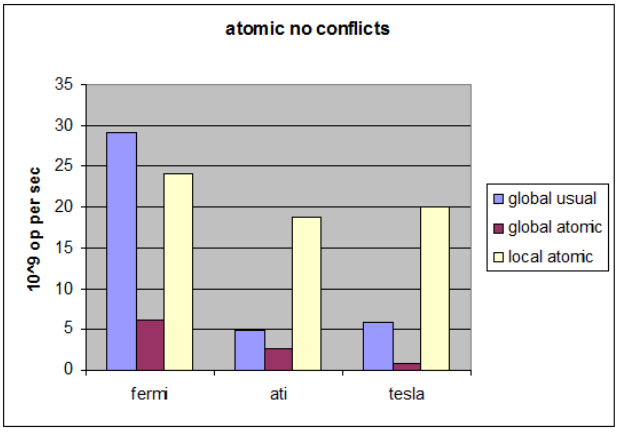

Рис. 1 Бесконфликтный доступ

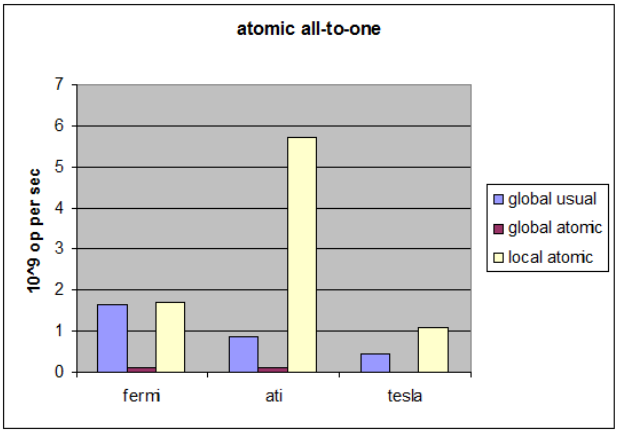

Рис. 2 Доступ «все-в-один»

Тест для измерения задержки глобальной, локальной и текстурной памяти

Глобальная память на ГПУ обладает очень большой латентностью. которая обычно покрывается за счёт большого числа одновременно работающих нитей. Локальная память работает быстрее, но меньше по размеру. Текстурная память физически расположена там же, где глобальная память, но для неё используется другое расположение элементов массива и другой механизм кэширования. Созданный тест выводит результат и в секундах, и в пересчёте на такты графического процессора, для удобства оценки.

Сравнить латентность разных типов памяти можно на рис. 3 (локальная память), рис. 4 (текстурная память) и рис. 5 (глобальная память). Локальная память, как и ожидалось, показывает гораздо лучшие результаты, чем глобальная. Текстурная память на карточках «Ферми» и «Тесла» показывает себя хуже, чем для карточки AMD, зато «Ферми» выигрывает при сравнении результатов глобальной памяти.

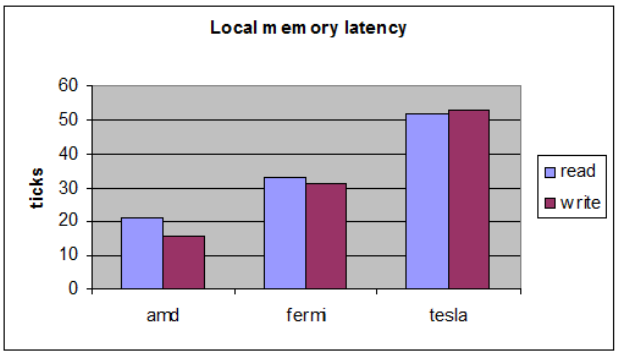

Рис. 3 Латентность локальной памяти

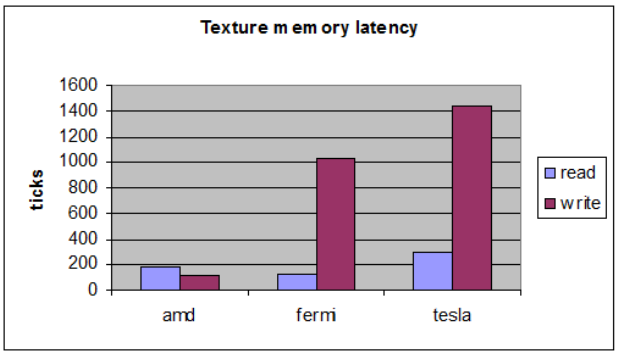

Рис. 4 Латентность текстурной памяти

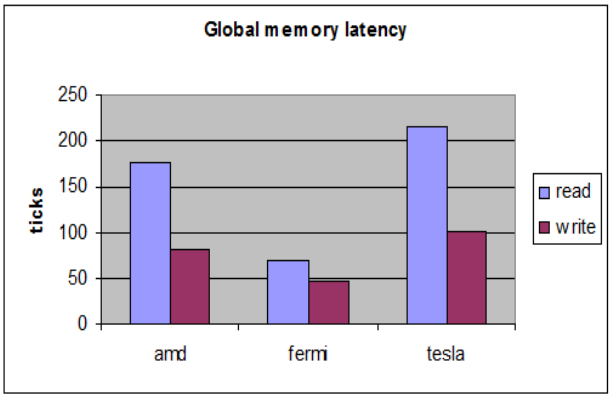

Рис. 5 Латентность локальной памяти

Тест пропускной способности глобальной, локальной и текстурной памяти

У ГПУ есть несколько важных особенностей работы с глобальной памятью:

1. Невозможно использовать всю пропускную способность памяти при малом количестве нитей.
2. Объединений запросов к памяти(coalescing). Если все нити в пределах полуварпа обращаются к одному участку памяти – происходит объединений запросов в одну транзакцию и это даёт выигрыш в производительности.
3. Наличие кэш-памяти. Полноценная кэш-память появилась в серии ГПУ "Ферми" от компании NVidia, и в ряде случаев она оказывает положительное влияние на производительность.

Тесты учитывает эти особенности и исследует зависимость пропускной способности от числа нитей для разных стандартных шаблонов обращения к памяти. В тестах присутствует специальная категория тестов «со сдвигом» — эти тесты учитывают объединение запросов к памяти, но по мере своих возможностей пытается нивелировать воздействие кэш-памяти. Набор тестов текстурной памяти аналогичен тестам глобальной памяти, за исключением технических тонкостей работы с двухмерными текстурами.

Локальная память расположена на самом чипе ГПУ, поэтому она гораздо меньше по объёму, гораздо быстрее глобальной и не кэшируется. Локальная память организована в виде банков, при доступе к которым могут возникать конфликты, снижает пропускную способность. Тесты исследует зависимость скорости доступа к локальной памяти от конфликтности шаблона доступа и числа нитей.

Обращения в память в исследуемом шаблоне могут быть либо только чтениями, либо только записями, либо или и чтениями, и записями. Адреса, по которым выполняется обращение, могут идти случайно, последовательно, или последовательно со сдвигом, т. е. разные нити обращается заведомо так, что обращения не попадают в одну кэш-линию. Таким образом, получаем 9 различных шаблонов доступа в память. Псевдокод теста чтения представлен в листинге 1.

Ниже приведены результаты работы теста с использованием глобальной памяти для последовательного доступа (рис. 6), доступа со сдвигом (рис. 7) и случайного доступа (рис. 8). Как видно из первого рисунка, «Ферми» получает очень большое преимущество над остальными за счёт наличия полноценного кэша, но оно исчезает в независимом от кэша тесте.

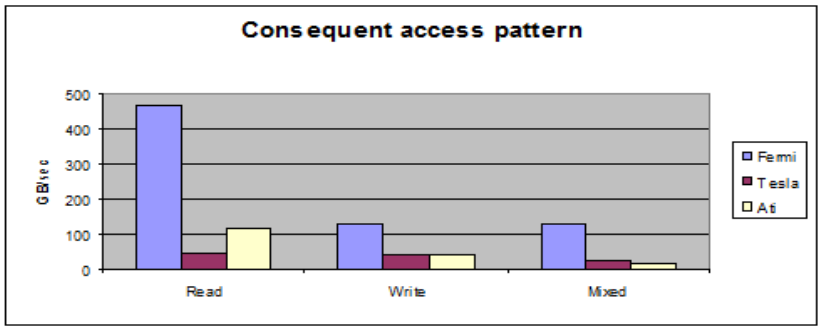

Рис. 6 Глобальная память, последовательный доступ

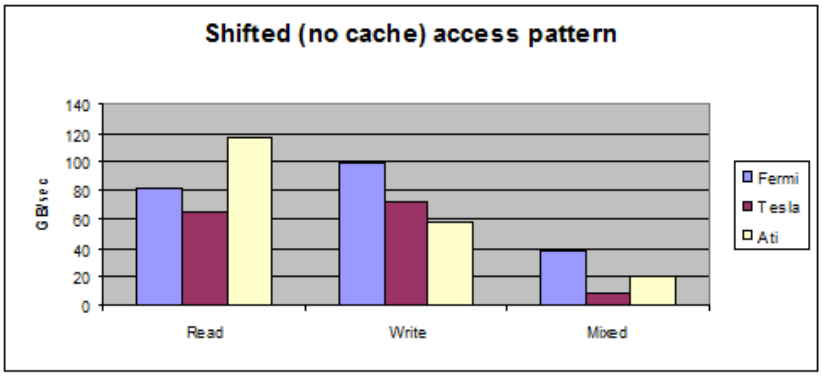

Рис. 7 Глобальная память, последовательный доступ с нивелированием кэша

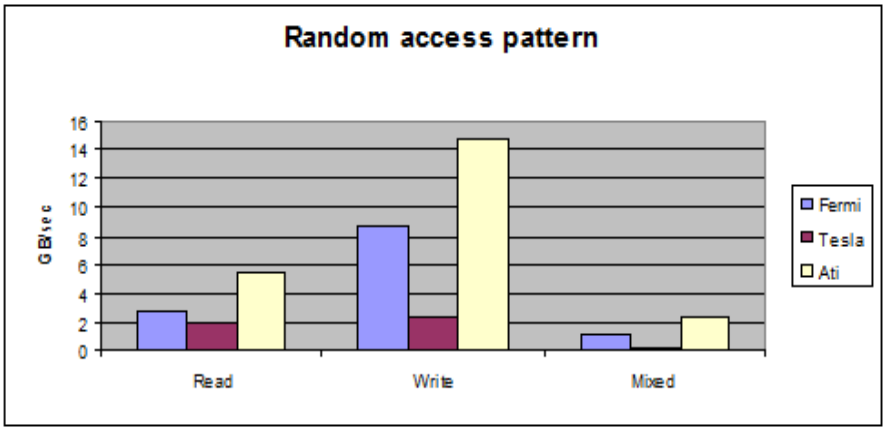

Рис. 8 Глобальная память, случайный доступ

```
int index = thread_id;
for(int i = 0; i < N; i++){
   index = data[index];
}
CountTime();
```

Листинг 1: Псевдокод теста на чтение

Тест эффективности последовательного чтения со случайной записью и случайного чтения с последовательной записью.

Некоторые вычислительные алгоритмы, при переносе на архитектуру ГПУ можно реализовать в двух вариантах – последовательное чтение со случайной записью или случайные чтения с последовательной

записью. Был реализован имитирующий это тест и результаты показали что большой разницы в целом нет (рис. 9)

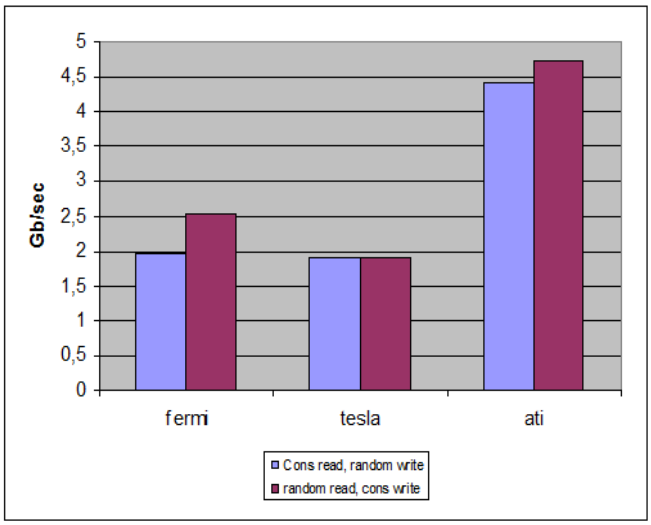

Рис. 9 Результаты теста на сравнение случайной записи с последовательным чтением и случайного чтения с последовательной записью

Тест для определения размера кэш-линии

Данный тест производит определённое количество чтений из глобальной памяти с разным шагом, по результатам замеров которых можно сделать предположение о размере кэш-линии на ГПУ. Псевдокод теста представлен в листинге 2.

Результат работы теста на ГПУ архитектуры "Ферми" приведены на рис. 10. При увеличение шага чтения до какого-то момента увеличивается время, это связано с тем, что всё меньше и меньше запросов попадают в одну кэш-линию. Точность не позволят точно указать размер кэш-линии, но его размер однозначно меньше 32 слов — график после 32 линейный.

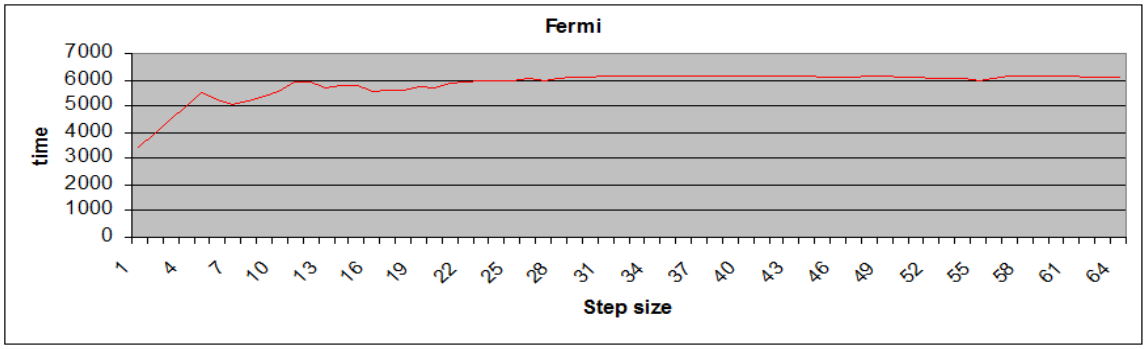

Рис. 10 Тест размера кэш-линии на ГПУ "Ферми"

Результат работы теста на карточке архитектуры "Тесла" 10-й серии приведены на рис. 11. Случайная структура и небольшой разброс позволяют предположить отсутствие кэша.

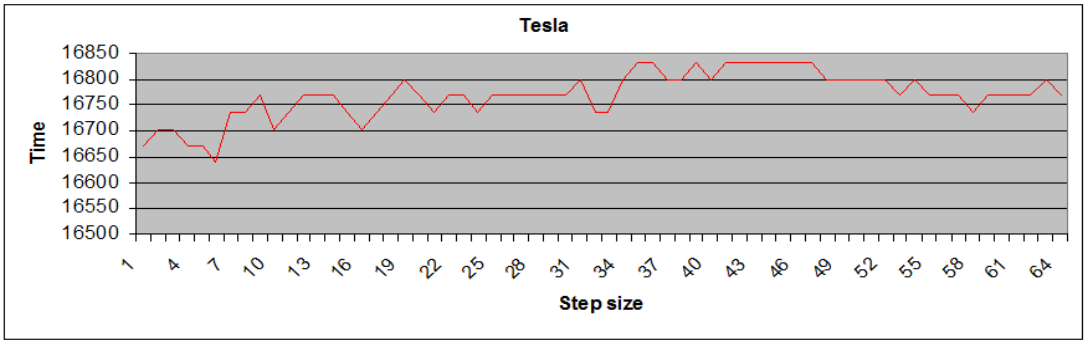

Рис. 11 Тест размера кэш-линии на ГПУ "Тесла" 10-й серии

```
for(int step = 1; i < STEP_MAX; step++){
```

```
            int t = 0;
            for(int i = 0; i < N; i++){
                t += data[i*step];
            }
            CountTime();
        }
```

Листинг 2: Псевдокод теста, определяющего размер кэша

Тест когерентного доступа в память

Для традиционных систем существует тест APEX-MAP [9] , позволяющий оценить эффективность работы с памятью некоторых классов задача. Была сделана попытка адаптировать этот тест под графические ускорители в последовательном и параллельном вариантах, но этот тест не показал показательных результатов и даже не смог достигнуть 20% максимальной пропускной способности (рис. 12). Это является ещё одним доказательством того, что графические процессоры требуют другого подхода эффективного программирования.

Недостатком традиционного теста APEX-MAP теста является то, что он не учитывает влияния когерентности обращений к памяти от разных нитей. В рамках системы тестов реализуется специальный тест памяти, учитывающий эту особенность. В новом тесте по памяти перемещаются группы потоков и варьируется разрозненность чтений внутри каждой группы.

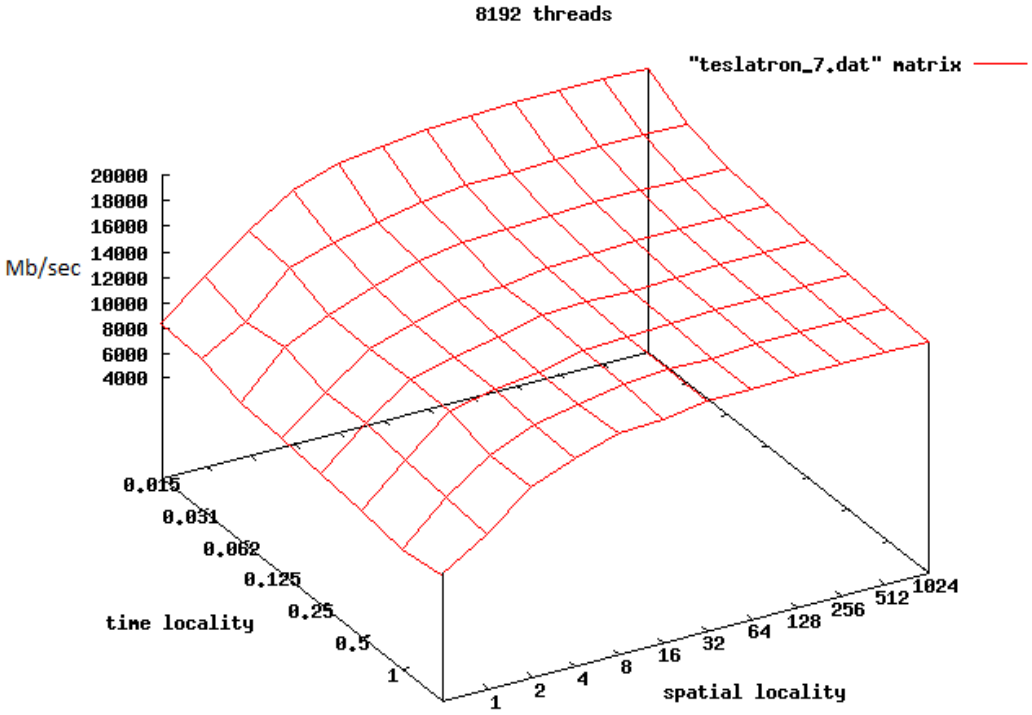

Рис. 12 Результат запуска классического теста APEX-MAP на ГПУ

Разработанный специальный тест памяти позволяет выбрать всю пропускную способность памяти и демонстрирует гораздо большую зависимость от когерентности обращений групп потоков к памяти. Результаты работы созданного теста на различных ГПУ изображены на рис. 13,14,15.

На оси «group size» откладывается размер группы совместных потоков.На оси «threads per element in group»(TPEIG) показывается сколько потоков из группы читают одинаковые ячейки памяти. Например, при TPEIG = 32 и размере группы = 64, потоки с 64N по 64N+31 читают один элемент данных, а потоки с номерами с 64N+32 по 64N+63 — ещё один, при этом читаемые элементы отстоят на расстояние, равное TPEIG, т. е. в данном случае 32. Адреса, по которым читают элементы потоки из разных групп, вообще говоря, никак не связаны. Псевдокод теста представлен в листинге 3.

На карточках «Тесла» 10-й серии и карточке AMD производительность при обращении в один элемент большого числа потоков (правый верхний угол) резко падает, что скорее всего связано с банками памяти. На карточах «Ферми» этот эффект отсутствует, предположительно из-за наличия кэш-памяти.

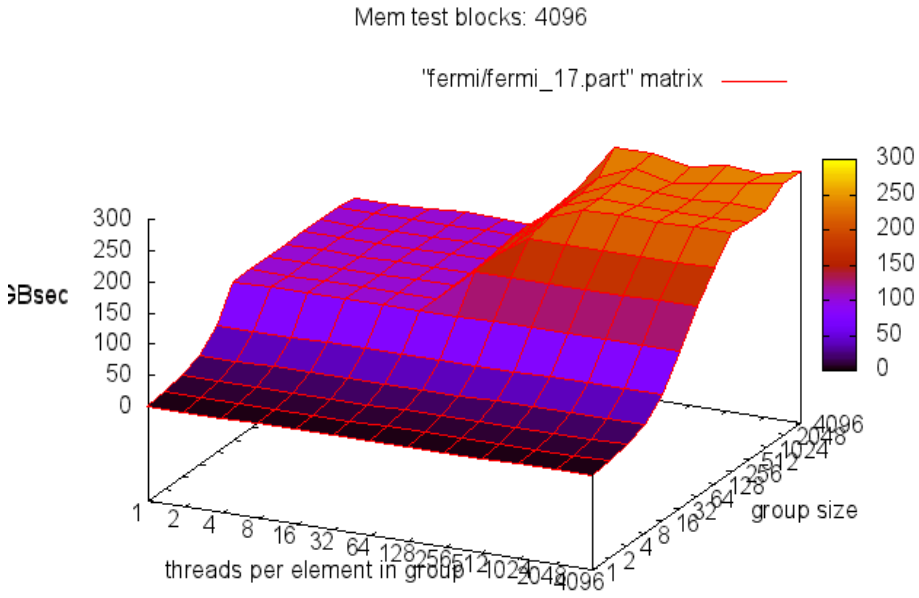
Mem test blocks: 4096
"fermi/fermi_17.part" matrix
GBsec
300
250
200
150
100
50
0
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
threads per element in group
group size

Рис 13 Специальный тест памяти, "Ферми"

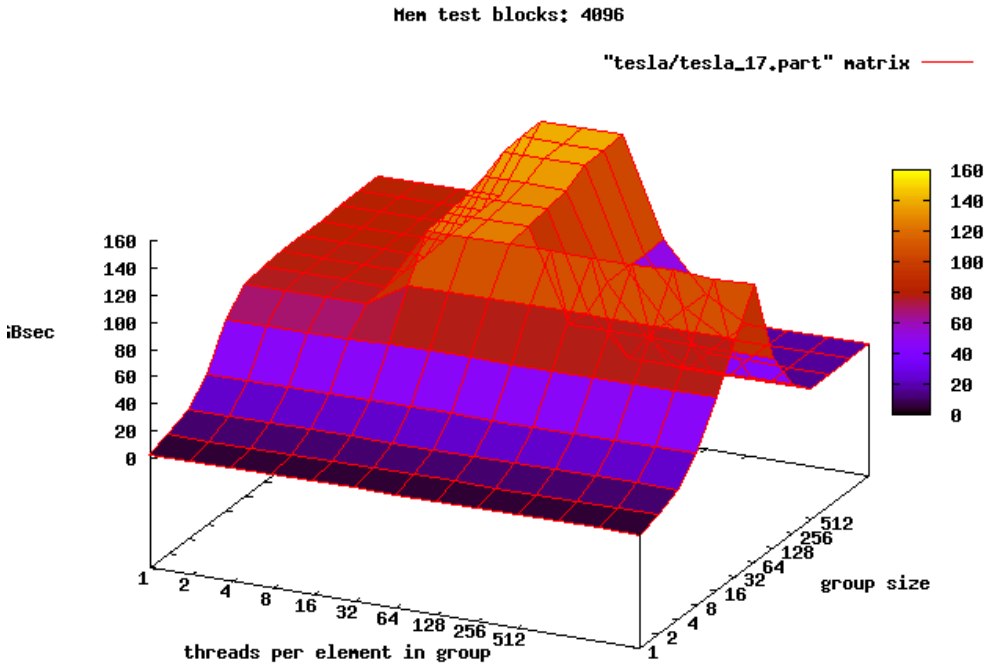

Рис. 14 Специальный тест памяти, "Тесла" 10-й серии

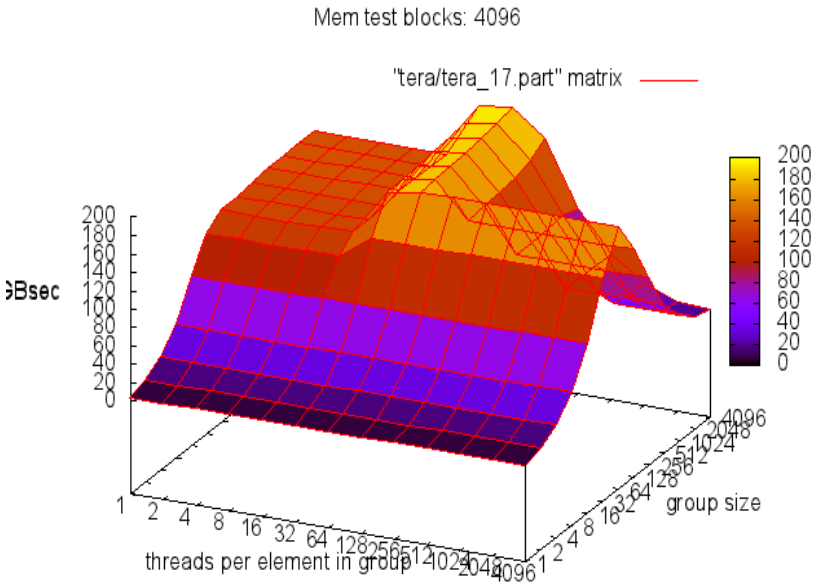

Рис. 15 Специальный тест памяти, ГПУ AMD

```
int start  =  thread_n ^ (thread_n                & (group_size-1));
int offset = (thread_n ^ (thread_n & (reads_count-1))) & (group_size-1);
```

```
for(int i = 0; i < ITERATIONS_COUNT; i++){
        start = data[start+offset];
}

CountTime();
```

Листинг 3: Псевдокод теста памяти

В качестве дополнительного исследования к данному тесту была добавлена зависимость от временной и пространственной локальности обращений, по аналогии с оригинальным тестом APEX-MAP. Для построения трёхмерного графика были взяты диагональные значения из специального теста и построен двумерный график зависимости диагональной «срезки» от временной и пространственной локальности обращений к памяти.

Из рисунков 16 и 17 видно, что временная и пространственная локальность (на графиках они изменяются по оси, направленной от нас) влияют на форму диагональной срезки не так сильно как изменение когерентности, но всё-таки вносят свой вклад в общую пропускную способность. Можно сделать вывод, что при оптимизации работы с памятью стоит больше внимания отдавать когерентности доступа, а после этого — проверять локальность обращений.

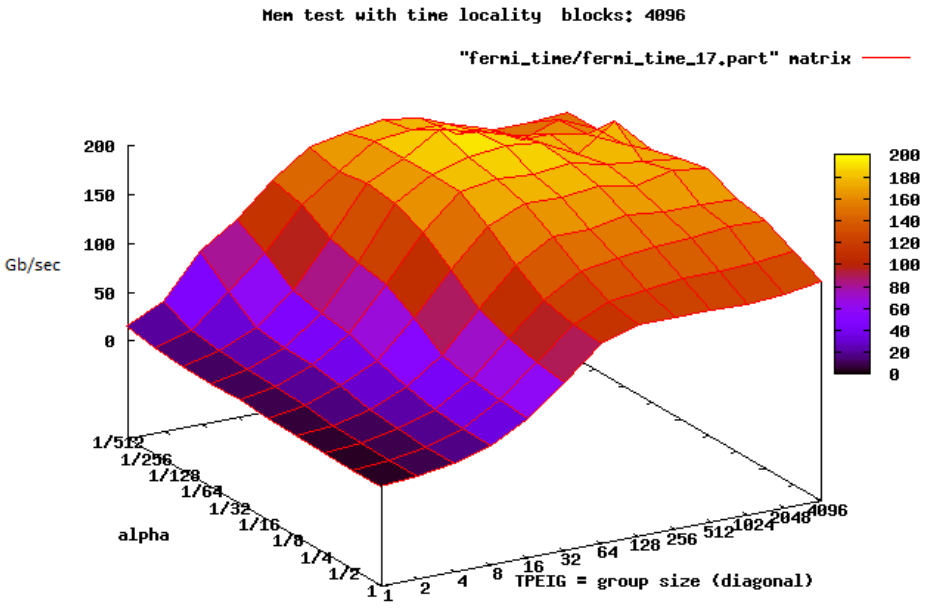

Рис. 16 Специальный тест с временной локальностью, ГПУ "Ферми"

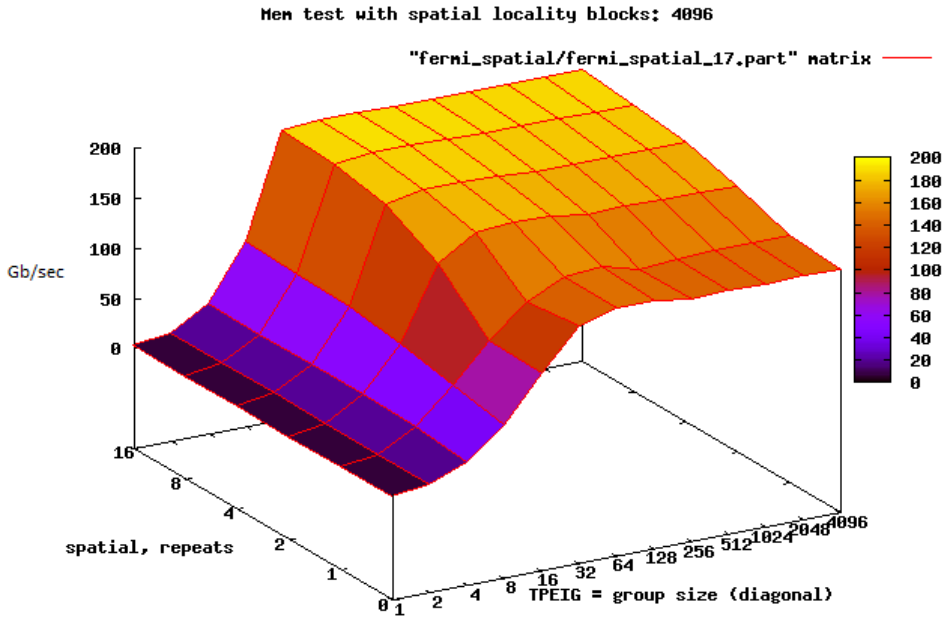

Рис. 17 Специальный тест с пространственной локальностью, ГПУ "Ферми"

Результаты по использованию некоторых из вышеописанных тестов для определения однородности кластера ГрафИТ!

Запуск некоторых тестов на нескольких карточках из кластера ГрафИТ! показал, что кластер обладает хорошей однородностью по памяти — разброс времени всех запусков для всех карточек приблизительно одинаков и составляет меньше 1%. Похожий тест для измерения однородности по вычислениям показал аналогичные результаты.

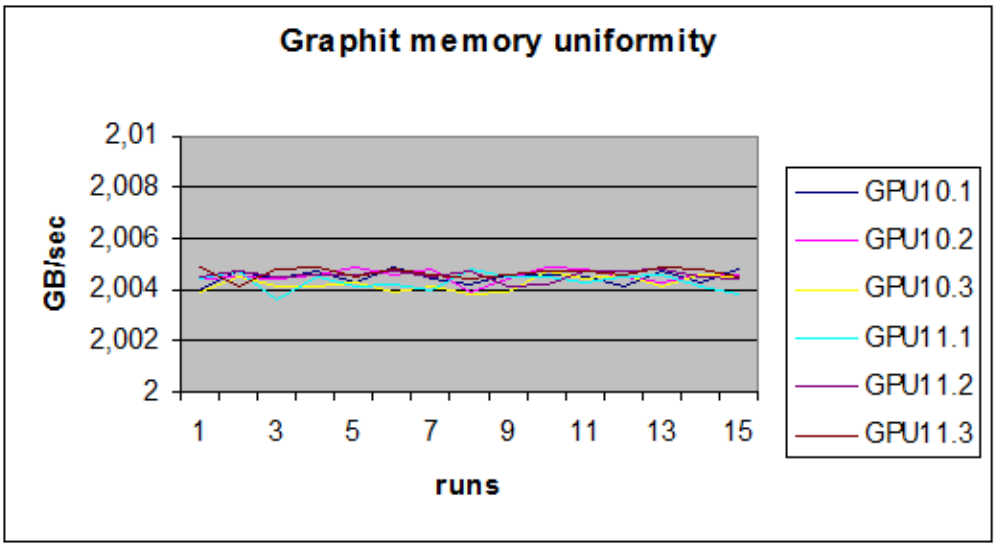

Рис. 18 Однородность по памяти — случайный доступ к памяти

Заключение

В результате работы был реализован переносимый набор тестов и проведён сбор данных с различных графических ускорителей. Был получен ряд весьма интересных результатов, таких как существенное влияние кэш-памяти на пропускную способность некоторых шаблонов доступа к памяти или оценка латентности доступа к разным типам памяти. Текущая версия системы тестов свободно доступна в исходных кодах в Интернете [11].

Планами дальнейшего развития проекта является написание набора тестов-примеров оптимизаций,

сравнивающих время работы неоптимизированных стандартных вычислительных ядер, таких как умножение матриц или преобразование Фурье, с версиями, оптимизированными под архитектуру графических ускорителей.

Работа выполняется при поддержке Российского фонда фундаментальных исследований (грант 11-07-93960-ЮАР_а).

ЛИТЕРАТУРА:

1. https://www-01.ibm.com/chips/techlib/techlib.nsf/products/Cell_Broadband_Engine
2. http://www.clearspeed.com/
3. http://parallel.ru/
4. http://www.khronos.org/opencl/
5. http://sourceforge.net/projects/nuda/
6. H. Wong, M. Papadopoulou, M. Sadooghi-Alvandi, A. Moshovos «Demystifying GPU microarchitecture through microbenchmarking»
7. R. Taylor, X. Li «A Micro-benchmark Suite for AMD GPUs»
8. A. Danalis, G. Marin, C. McCurdy, J.S. Meredith, P.C. Roth, K. Spafford, V. Tipparaju, J. Vetter «The Scalable Heterogeneous Computing (SHOC) benchmark suite”
9. http://www.prace-project.eu/documents/17_apexmap_vw.pdf
10. http://developer.amd.com/gpu/AMDAPPSDK/Pages/default.aspx
11. Сайт проекта. http://sourceforge.net/projects/gpuperfomance/