

# Вычисления на видеокартах

Курсовая работа  
студента 351 группы А. А. Григорьева

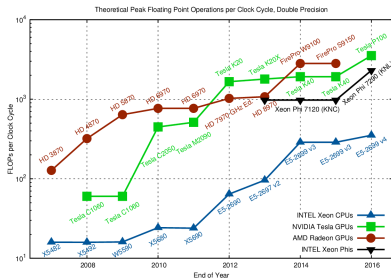
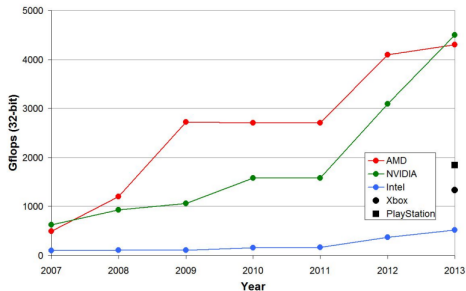
Саратовский государственный университет  
им. Н. Г. Чернышевского

Кафедра математической кибернетики  
и компьютерных наук

Научный руководитель: доцент Семенов М. С.

2019г.

GPU and CPU Peak Performance Trends



- 1 Понять, какие алгоритмы эффективнее исполнять на видеокартах;
- 2 Изучить типовую архитектуру видеокарты, научиться оптимизировать алгоритмы;
- 3 Получить практический опыт разработки программ на видеокартах с помощью OpenCL;
- 4 Провести исследование производительности параллельных программ на различных видеокартах.

Центральный процессор — небольшое количество «умных» ядер

Видеокарта — большое количество «простых» ядер

Выделяют следующие особенности вычислительных устройств в видеокарте:

- 1 массовый параллелизм;
- 2 единый указатель на инструкции;
- 3 быстрое обращение к локальной и регистровой памяти;
- 4 медленное обращение к «глобальной» памяти;

При разработке алгоритмов стоит учитывать:

- ❶ разбиение на рабочие группы, минимальная синхронизация между ними;
- ❷ соккрытие задержки при обращении к памяти в рабочих группах;
- ❸ правильное ветвление;
- ❹ использование локальной памяти и регистров для быстрого доступа к памяти;

Создана хостовая часть программы и kernel для выполнения следующих алгоритмов:

- 1 сумма двух векторов;
- 2 нахождение максимальной суммы на префиксе массива;
- 3 транспонирование матрицы\*;
- 4 умножение матриц\*.

\* — реализованы обычные и эффективные версии kernel.

Дано: массивы  $A$ ,  $B$  из  $N$  чисел

Задача: заполнить результирующий массив  $N$  поэлементными суммами векторов  $A$ ,  $B$

Решение демонстрирует базовые возможности программирования с помощью OpenCL



Дано: массив  $A$  из  $N$  целых чисел

Задача: найти максимальную сумму на префиксах массива  $A$

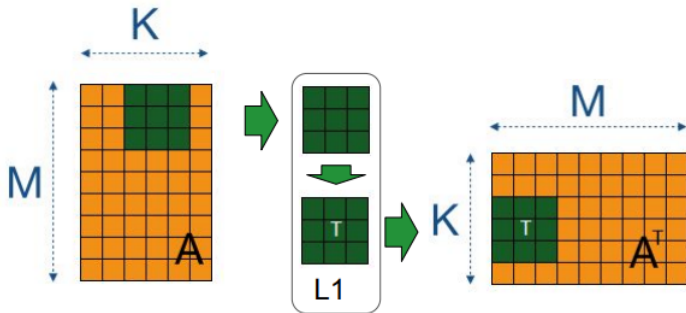
Для решения используется локальная память, барьеры, рекурсивное выполнение kernel

Дано: матрица  $A$  размерности  $N \times M$

Задача: найти транспонированную матрицу  $A^T$

Решена двумя способами:

- 1 Простой —  $N$  доступов к глобальной памяти для вычисления  $N$  элементов;
- 2 Эффективный —  $2 \times N$  доступов к «глобальной» памяти для вычисления  $N^2$  элементов за счет использования локальной памяти



**Рис.:** Транспонирование матрицы с использованием «плиток» в локальной памяти

Дано: матрица  $A$  размерности  $N \times K$ , матрица  $B$  размерности  $K \times M$

Задача: найти матрицу  $C$  — произведение матриц  $A$  и  $B$

Решена двумя способами:

- 1 Простой —  $N^3$  доступов к глобальной памяти для вычисления  $N^2$  элементов;
- 2 Эффективный —  $3 \times N$  доступов к «глобальной» памяти для вычисления  $N^2$  элементов за счет использования локальной памяти

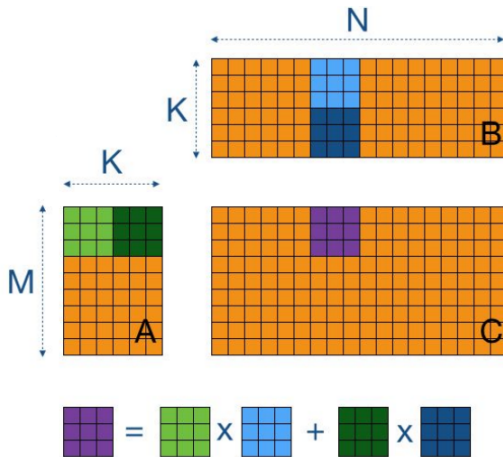






Рис.: Транспонирование матрицы с использованием «плиток» в локальной памяти

Устройство	Алгоритм	Размер р.группы/данных	Время работы, мкс
NVIDIA Geforce 1050 Ti	Сумма векторов	1/2097152	25026
NVIDIA Geforce 560 Ti	Сумма векторов	1/2097152	44150
NVIDIA Geforce 1050 Ti	Сумма векторов	128/2097152	14345
NVIDIA Geforce 560 Ti	Сумма векторов	128/2097152	29097
NVIDIA Geforce 1050 Ti	Макс. префикс	128/2097152	17345
NVIDIA Geforce 560 Ti	Макс. префикс	128/2097152	32097
NVIDIA Geforce 1050 Ti	О. транспонирование	$32 \times 32 / 4096 \times 2048$	21648
NVIDIA Geforce 560 Ti	О. транспонирование	$32 \times 32 / 4096 \times 2048$	36519
NVIDIA Geforce 1050 Ti	Умножение матриц	$1 \times 1$	577230
NVIDIA Geforce 560 Ti	Умножение матриц	$1 \times 1$	1274058
NVIDIA Geforce 1050 Ti	О. Умножение матриц	$K \times 16$	27290
NVIDIA Geforce 560 Ti	О. Умножение матриц	$K \times 16$	49821

В результате курсовой работы:

- 1 изучены особенности программирования на видеокартах с использованием OpenCL;
- 2 рассмотрены возможности оптимизации алгоритмов на видеокартах;
- 3 проведены вычисления для разных конфигураций рабочих групп, возможных алгоритмов и видеокарт.

-  [https://www.nvidia.com/content/PDF/fermi-white-papers/NVIDIA's Next Generation CUDA Compute Architecture: Fermi](https://www.nvidia.com/content/PDF/fermi-white-papers/NVIDIA's%20Next%20Generation%20CUDA%20Compute%20Architecture%20Fermi.pdf)
-  A. Munshi, B. R. Gaster, T. G. Mattson, J. Fung, D. Ginsburg  
OpenCL Programming Guide  
Ann Arbor, Michigan: 2012.
-  <http://opencl.ru/node/8>  
Введение | OpenCL
-  <https://www.khronos.org/registry/OpenCL/specs/>  
The OpenCL Specification





<https://compscicenter.ru/courses/video-cards-computation/>

Введение в OpenCL. Архитектура видеокарты



<https://docs.nvidia.com/gameworks/content/developertools/>

Achieved Occupancy



<https://software.intel.com/ru-ru/articles/>

Неоднородные рабочие группы OpenCL 2.0



[https://www.khronos.org/registry/OpenCL/sdk/1.0/docs/  
clEnqueueNDRangeKernel](https://www.khronos.org/registry/OpenCL/sdk/1.0/docs/clEnqueueNDRangeKernel)



<https://compscicenter.ru/courses/video-cards-computation/>

Умножение матриц | Вычисления на видеокартах



<https://www.karlrupp.net/2016/08/flops-per-cycle-for>  
FLOPs per Cycle for CPUs, GPUs and Xeon Phi

СПАСИБО ЗА ВНИМАНИЕ!