

CUDA ВВЕДЕНИЕ

Романенко А.А.
arom@ccfit.nsu.ru

Мощность вычислительных систем

Производительность

«K computer» **2011**
68 544 x 8-core SPARC64 VIIIfx.
8,162 Pflops



280 Tflops
212,992 CPUs

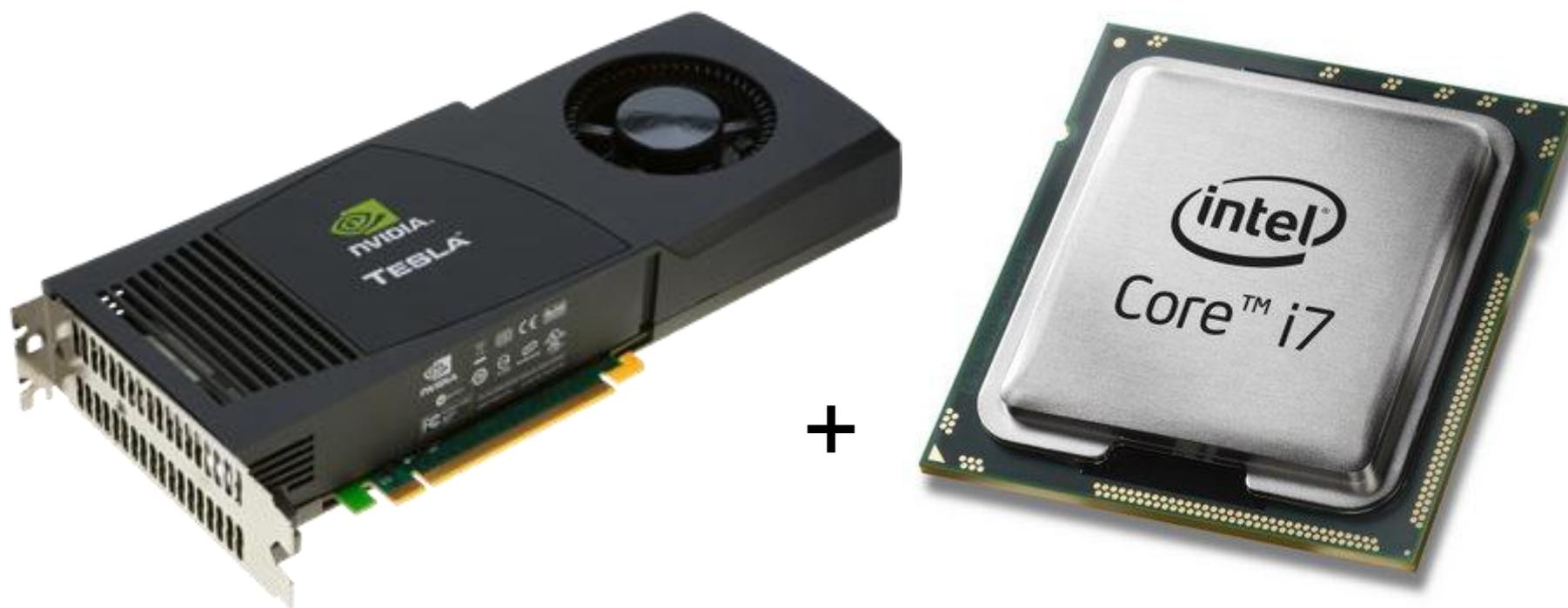


Время

Рост производительности

- За счет увеличения частоты процессоров
- За счет увеличения количества ядер/процессоров
- За счет усложнения архитектуры самих процессоров
 - Увеличение количества регистров
 - Изменение длины конвейера
 - Увеличение разрядности
 - пр.

Гетерогенные системы



3 из 5 в лидерах списка top500 !

Гетерогенные vs. традиционные ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Tianhe-1A (2 место)

Xeon X5670 6C 2.93 GHz, NVIDIA 2050



4,7 Pflops (peak)

4040 kW

Jaguar (3 место)

Opteron 6-core 2.6 GHz



2,3 Pflops (peak)

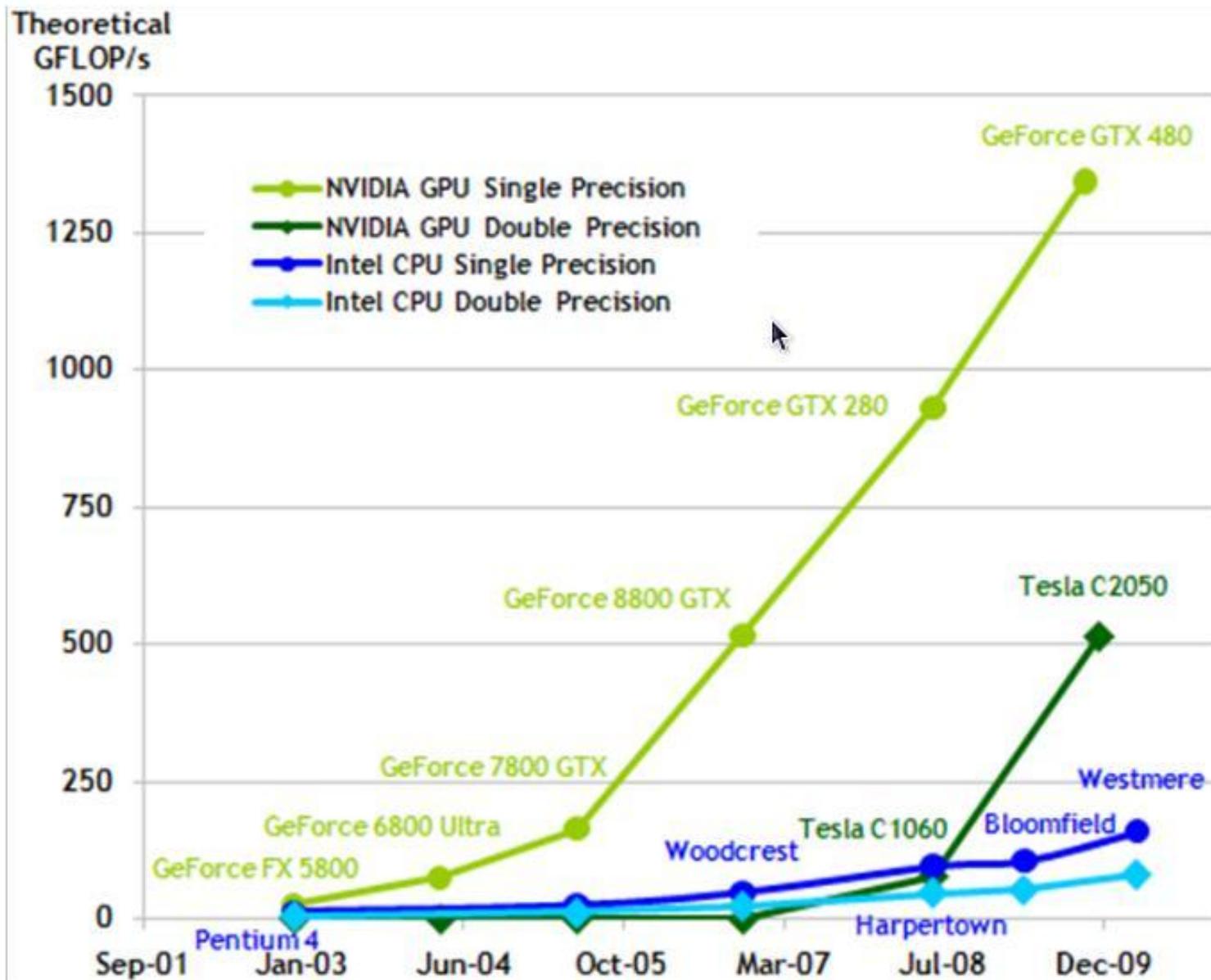
6950 kW

Обработка графики

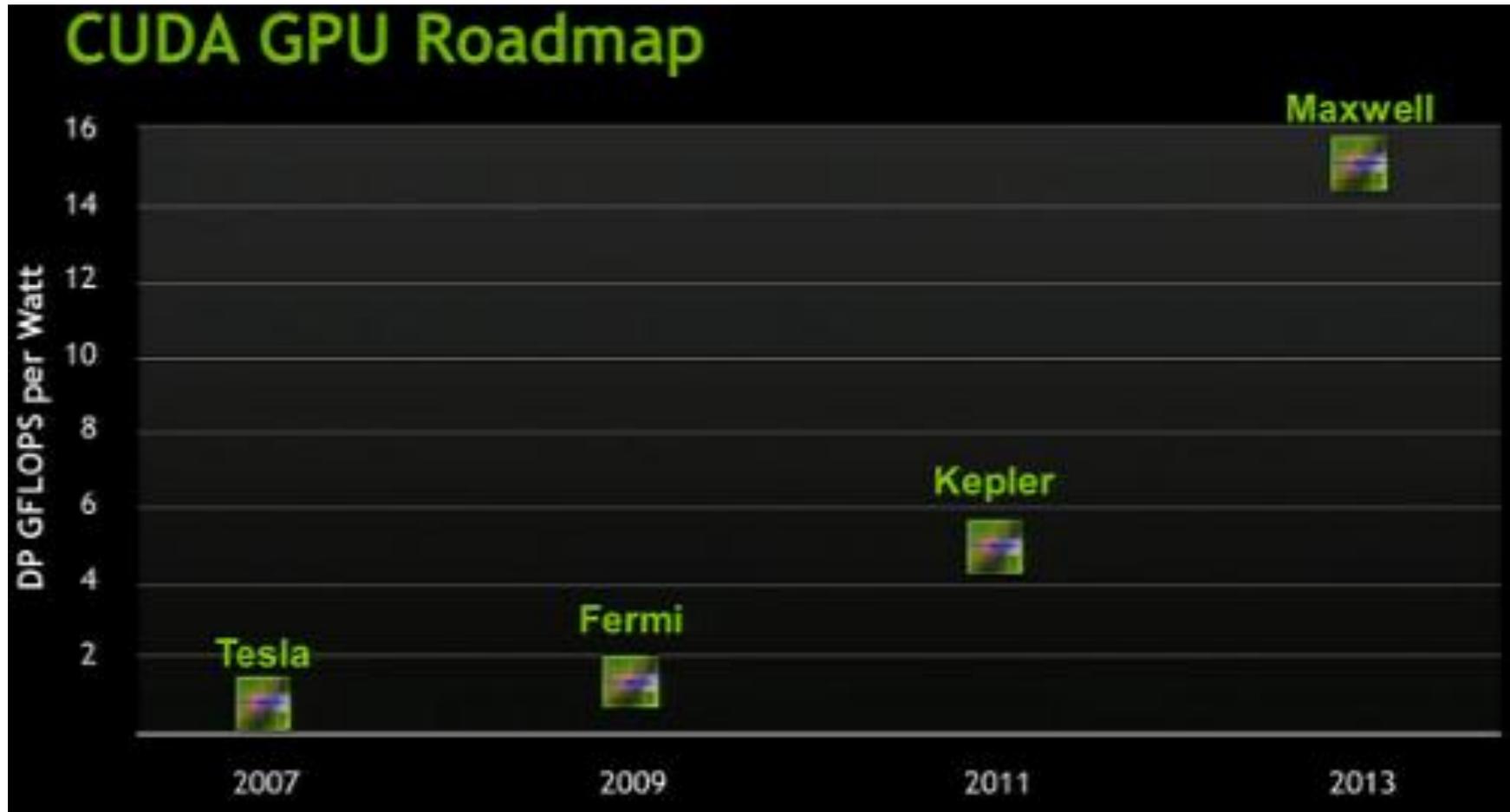


- Работа с векторами
 - Работа с маленькими матрицами
 - Фильтры/post-processing
 - Вычисление проекций
 - пр.
-
- Однотипные операции над большим количеством данных

Производительность видеокарт

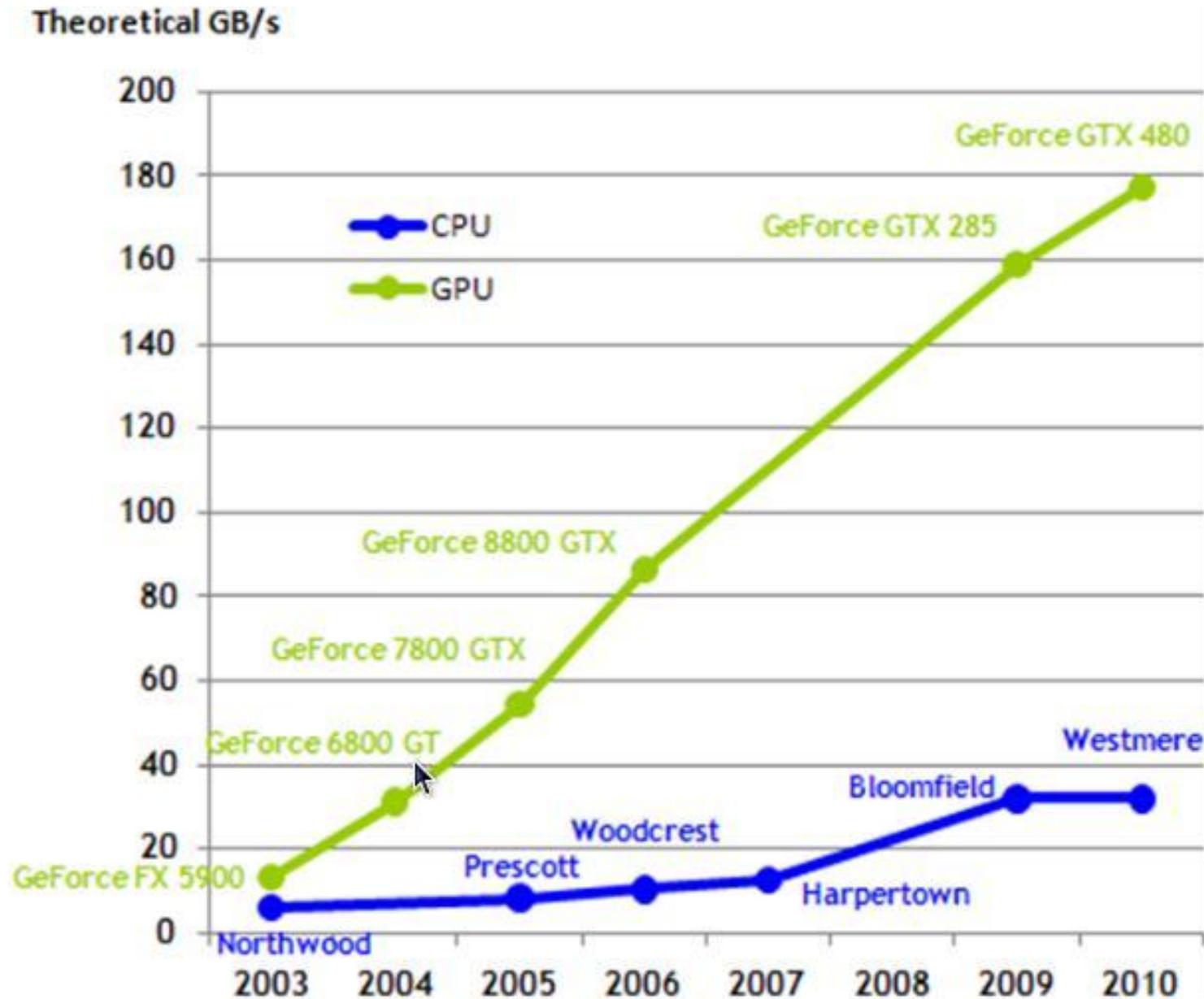


Данные на сентябрь 2010



- Kepler: 28nm, 1.4 Tflops DP
- Maxwell: 22nm, 4 Tflops DP

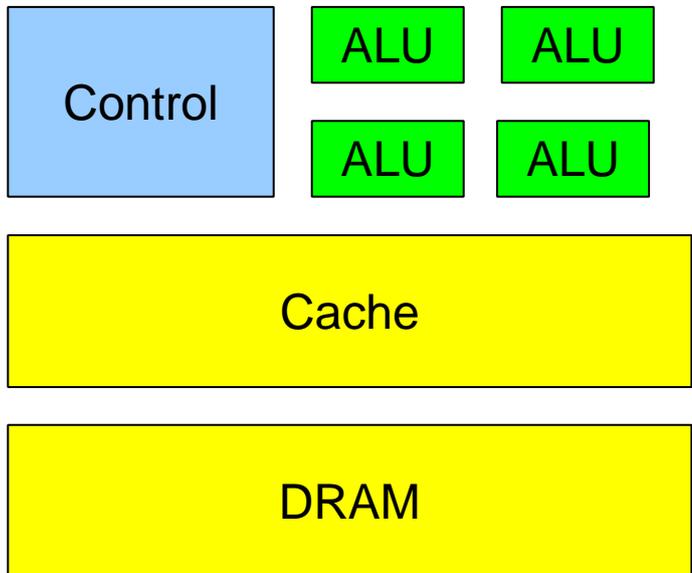
Пропускная способность шины



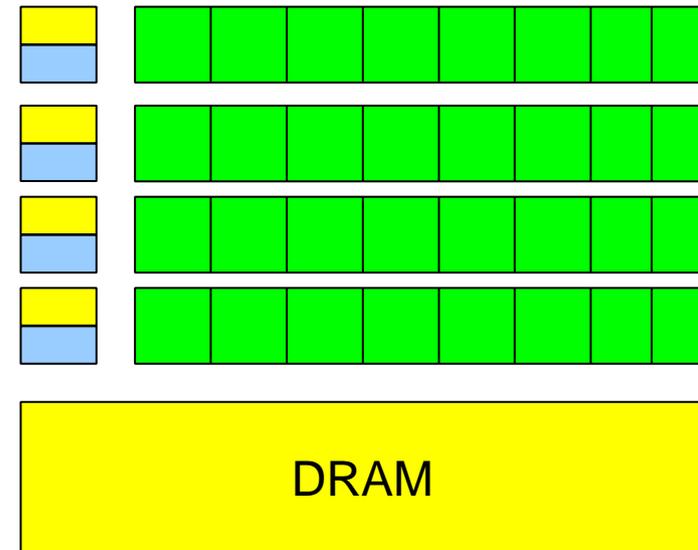
GPU — Graphical Processing Unit

- GPU — процессор на видеокарте. Имеет свою архитектуру
- Программа на GPU не может общаться с хостом
- Программа на GPU не может писать в память хоста
- Загрузка и выгрузка данных на видеокарту производятся через шину PCI Express 1 (2). Передача данных инициируется на стороне хоста
- Видеокарта - сопроцессор

CPU vs. GPU



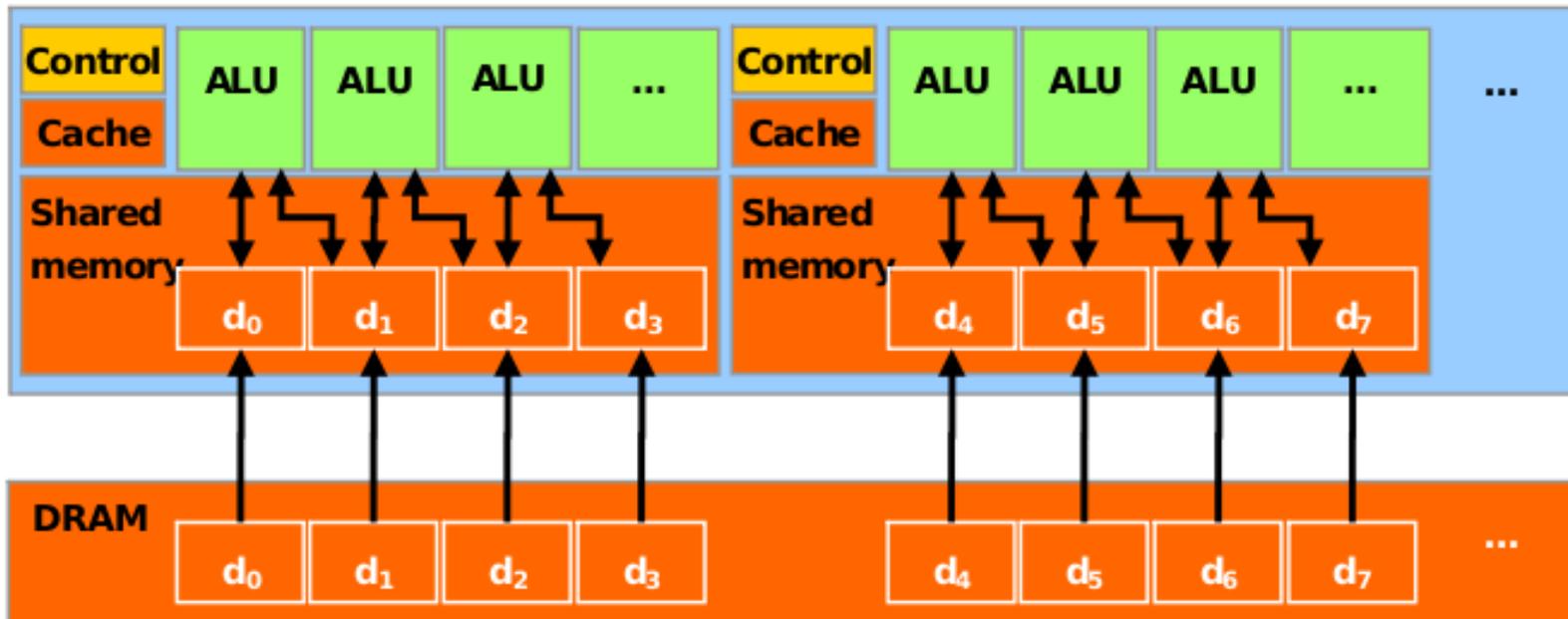
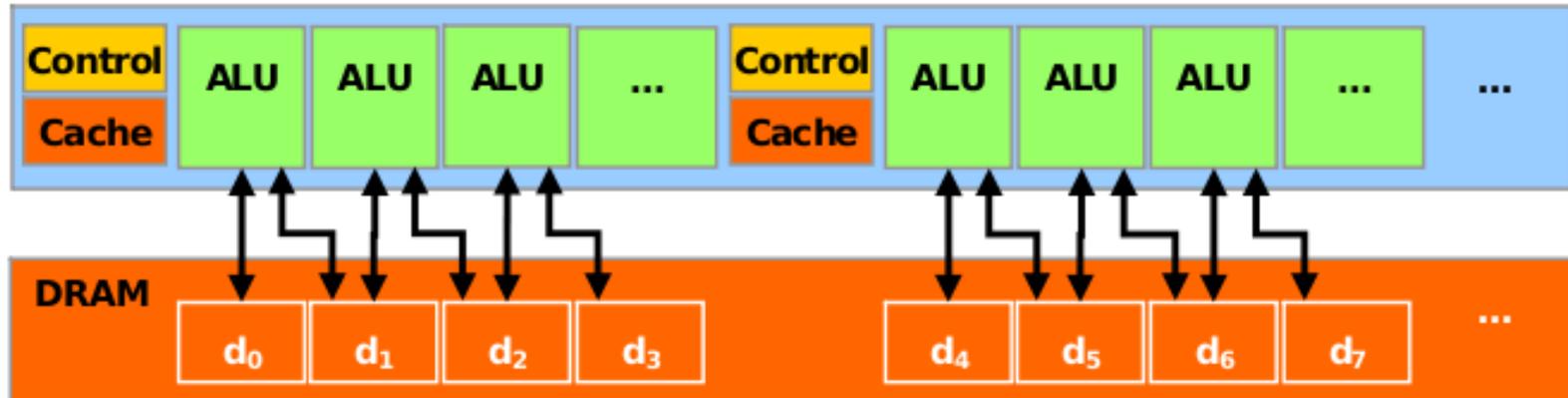
CPU



GPU

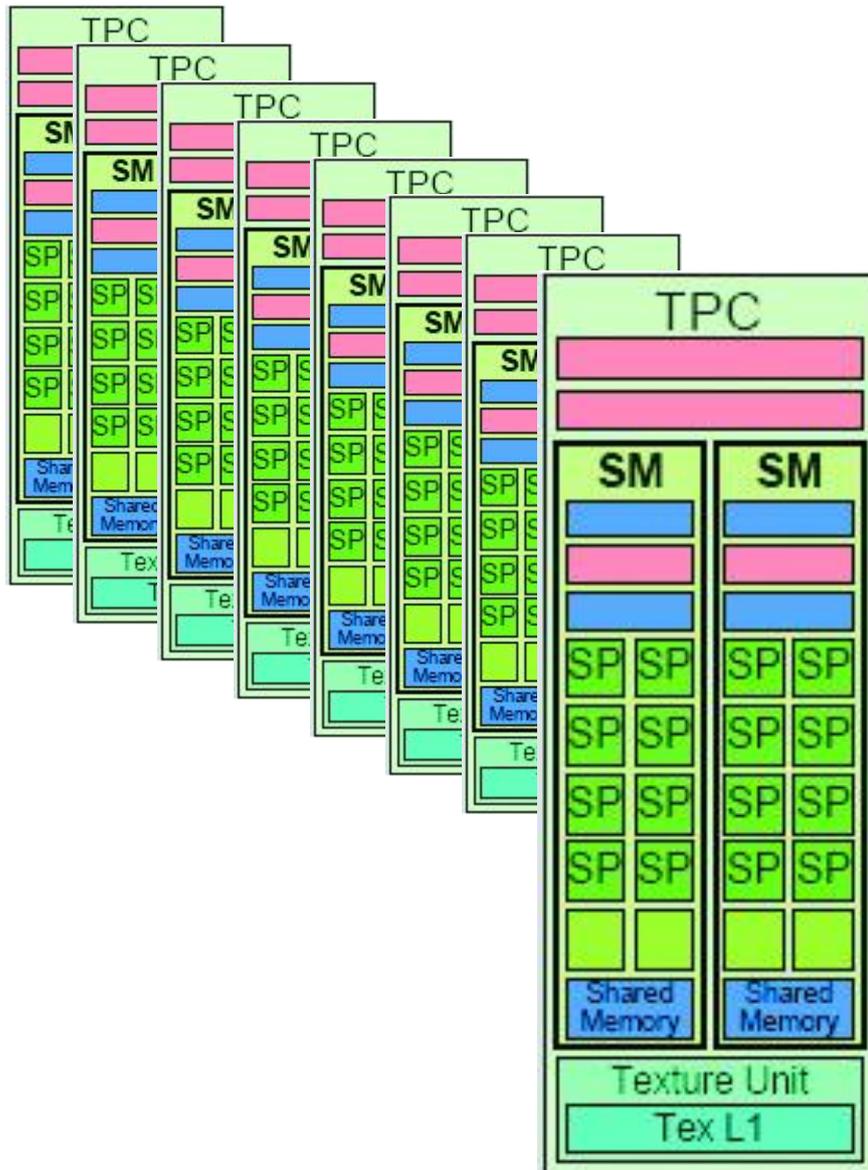
- Менше транзисторов на управление и кэш
- Больше на АЛУ

Доступ к памяти



Аппаратная архитектура GPU (GT200)

Streaming Processor Array



- TPC - Texture Processor Cluster
 - Multi-threaded processor core
 - Fundamental processing unit for CUDA thread block
- SM — Streaming Multiprocessor
 - Multi-threaded processor core
 - Fundamental processing unit for CUDA thread block
- SP — Streaming Processor
 - Scalar ALU for a single CUDA thread

	Количество SM
GeForce 8800 GTX	16
GeForce 8800 GTS	12
Tesla D870	2x16
Tesla S870	4x16
Tesla C1060, GT200, Tesla T10	30
Tesla S1070	4x30

Tesla C1060

1 TFlops



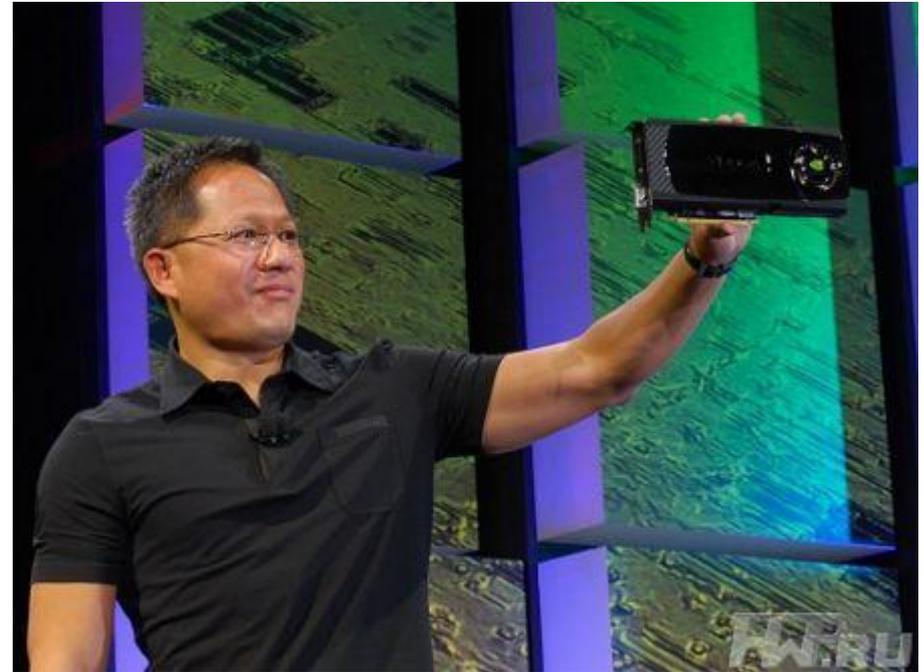
Tesla S1070

4 TFlops



Fermi

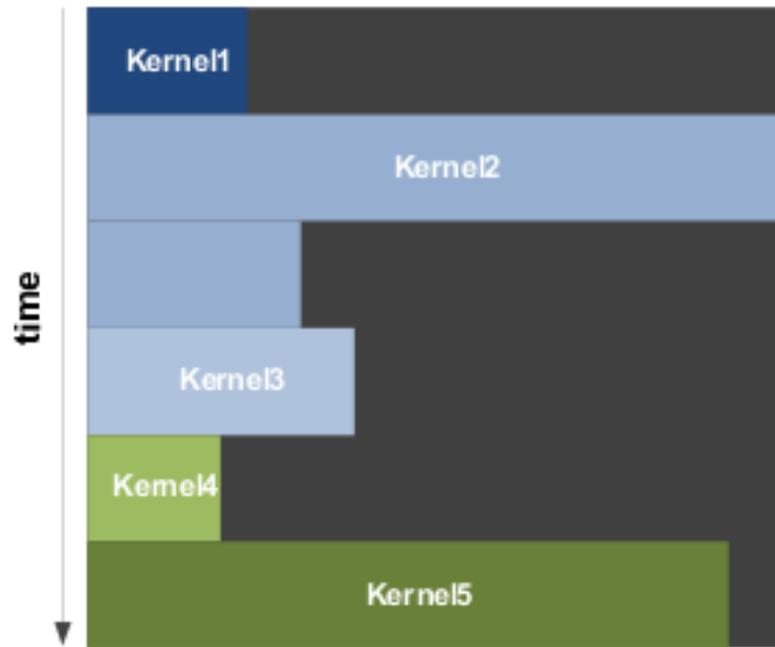
- Анонс в сентябре 2009г.
- В мае 2010 года начала продаж видеокарт серии GT300 с новым чипом с кодовым названием Fermi



Fermi. Характеристики

- 3 млрд. транзисторов, 40-нм техпроцесс TSMC
- 512 ядер CUDA с поддержкой IEEE 754-2008, объединенные в 16 потоковых мультипроцессоров
- Тактовая частота ядер CUDA - около 1,5 ГГц
- 128 блоков выборки текстур
- 384-битный контроллер памяти GDDR5 (6x64 бит)
- Пропускная способность шины памяти - около 192 Гбайт/с
- 1,5 Тфлопс SP, 750 Гфлопс DP
- Интерфейс - PCI Express x16 2.0
- C++, в дополнение к поддержке C, Fortran, Java, Python, OpenCL и DirectCompute.
- ECC

Fermi. Особенности



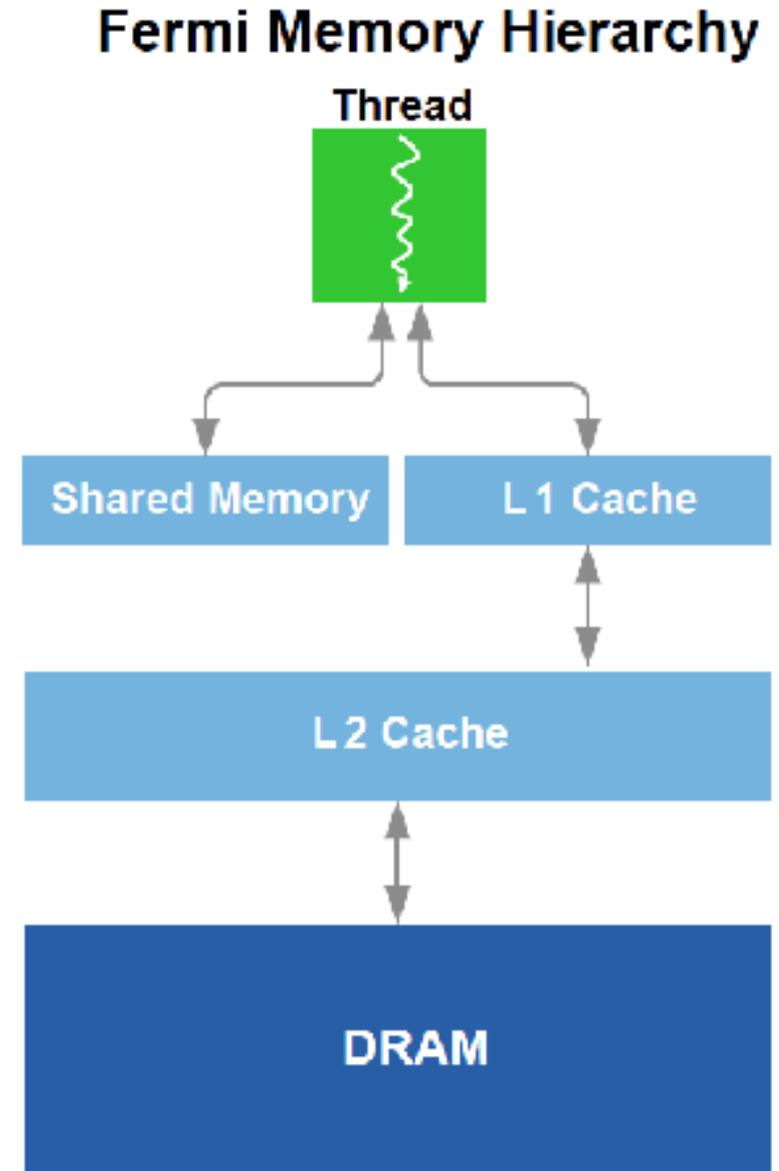
Serial Kernel Execution



Concurrent Kernel Execution

Fermi. Особенности

- NVIDIA Parallel DataCache™ - первый иерархический кэш на GPU



Ускорение для некоторых приложений

Example Applications	URL	Application Speedup
Seismic Database	http://www.headwave.com	66x to 100x
Mobile Phone Antenna Simulation	http://www.acceleware.com	45x
Molecular Dynamics	http://www.ks.uiuc.edu/Research/vmd	21x to 100x
Neuron Simulation	http://www.evolvedmachines.com	100x
MRI processing	http://bic-test.beckman.uiuc.edu	245x to 415x
Atmospheric Cloud Simulation	http://www.cs.clemson.edu/~jesteel/clouds.html	50x

3 способа написать программу для GPU

Приложение

Оптимизированные библиотеки

Директивы компилятора

Языки программирования (C/C++/Фортран)

Ускорение до нескольких десятков раз

Максимум производительности

CUDA vs OpenCL

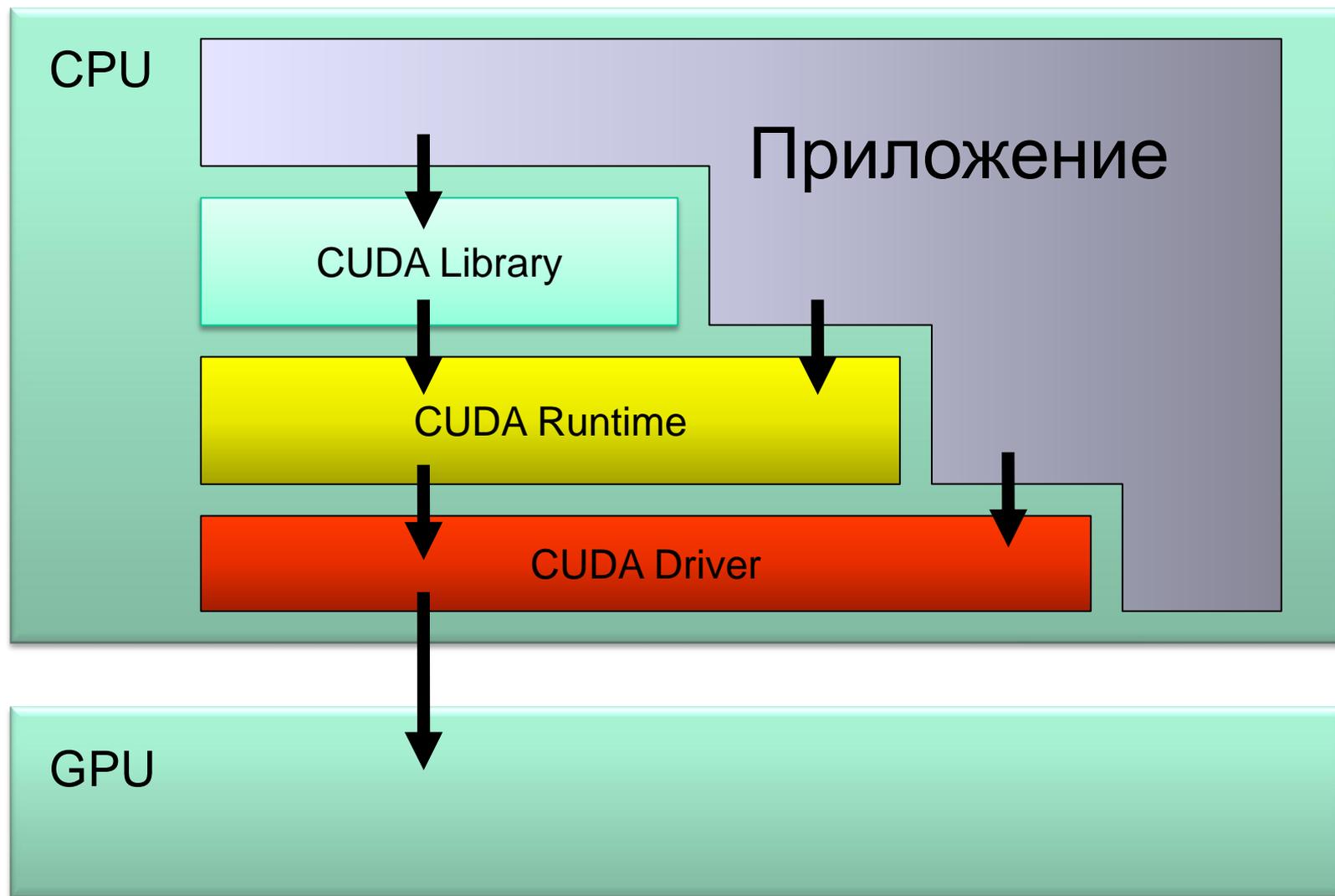
- **CUDA**

- Процессоры NVIDIA (решения Cray, HP, IBM, T-Platforms, NextIO...)
- Производительность
- Функциональность
- Удобство разработки
- Поддержка
- Учебные материалы и библиотеки

- **OpenCL**

- Архитектура не фиксирована, требуется универсальность
- Производительность не приоритетна

CUDA - Compute Unified Device Architecture



CUDA Roadmap

- CUDA 1.0 — 2007
- CUDA 2.0 — 2008
- CUDA 3.0 — 2009
- CUDA 4.0 — 2011



2012 – 5 лет!

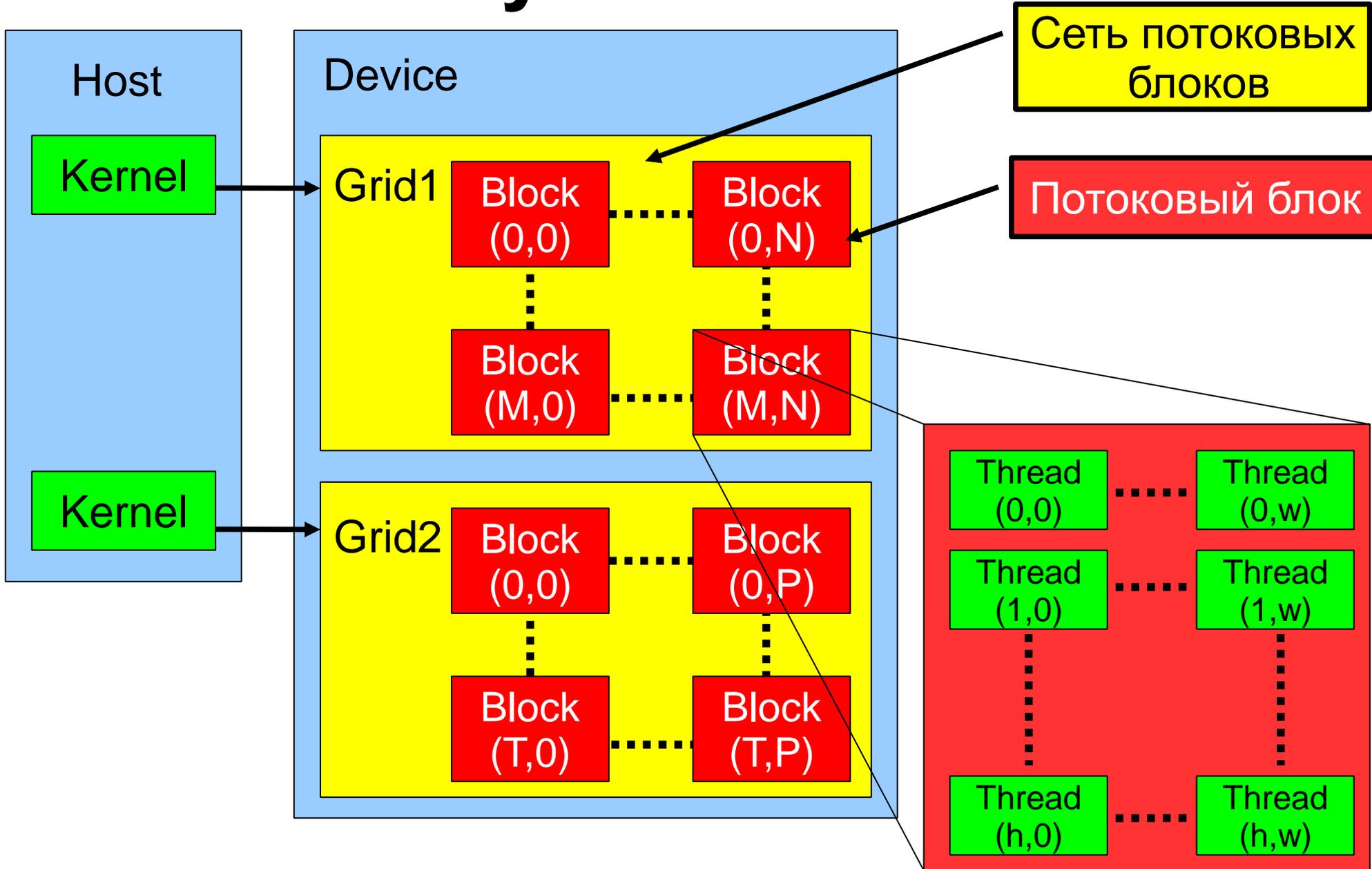
Термины

- **Поток** (Thread) — единица исполнения потока команд
- **Потоковый блок** (Thread blok) — группа связанных между собой потоков.
- **Варп** (Warp)— группа потоков внутри потокового блока, которая исполняется физически одновременно (32 потока)
- **Сеть** (Grid) — набор блоков, который должен быть обработан прежде чем исполнение программы пойдет дальше.

Программная модель

- GPU имеет свою память
- Программа в виде потоков выполняется на SP
- SP имеет доступ только к разделяемой памяти внутри своего SM и памяти GPU
- Синхронизация потоков возможна только внутри SM
- Исполнение организовано как сеть (GRID) потоковых блоков (thread block)
- Программа выполняемая потоком — ядро (kernel)

Запуск потоков



ПОТОКОВЫЙ БЛОК

- Каждый поток в блоке имеет свой идентификатор — `threadIdx`
- Для удобства потоки могут отражаться на одномерную, двумерную, трехмерную сетку. При этом координаты потока задаются через (x, y, z)
- Размеры области отображения задаются при запуске ядра
- Количество потоков в блоке ≤ 512

Сеть потоковых блоков

- Каждый блок в сети имеет свой идентификатор — BlockIdx
- Для удобства блоки могут отражаться на одномерную, двумерную сетку. При этом координаты блок задаются через (x, y, z)
- Размеры области отображения задаются при запуске ядра

Пример

- Пусть при запуске задана двумерная сеть из блоков размером $H \times W$ и каждый блок содержит $M \times K$ потоков
- Таким образом область моделирования разбивается на
 - $H * M$ потоков по вертикали
 - $W * K$ потоков по горизонтали
- координаты потока в пространстве
 - $(\text{blockID}.x * M + \text{threadID}.x, \text{blockID}.y * K + \text{threadID}.y)$

Модель выполнения

- Блоки выполняются на Stream Multiprocessor
 - Один блок только на одном SM
 - Последовательность исполнения блоков не определена
- Количество блоков на SM определяется количеством регистров, требуемых потоку и количеством разделяемой памяти на блок
- Исполняемый в текущий момент поток называется **активным блоком**

Модель выполнения

- Каждый активный блок разбивается на SIMD группы потоков — варпы (**warps**). Каждый ворп содержит одинаковое количество потоков. $WarpSize = 32$
- Планировщик потоков периодически передает управление от одного ворпа к другому
- Распределение потоков по ворпам всегда одинаковое

Литература

- http://www.nvidia.com/object/cuda_home.html
- <http://steps3d.narod.ru/tutorials/cuda-tutorial.html>
- <http://steps3d.narod.ru/tutorials/cuda-2-tutorial.html>