

Введение в CUDA

История развития граф процессоров



S3 ViRGE

64-битный интегрированный 2D/3D
акселератор с наличием ТВ — выхода,
исполнение в AGP и PCI вариантах, 2-4 Mb
SGRAM (83 MHz) или EDO (66 MHz), 170 MHz
RAMDAC, поддержка Direct3D, BRender,
RenderWare, OpenGL и собственного API S3D,
возможно использование телевизионного
экрана вместо монитора

Первое поколение

Специализированный процессор для 3D графики

- буфер глубины
- аппаратное распараллеливание
- текстурирование
- задание цвета
- интерполяционная закраска

Второе поколение

nVidia GeForce 256



- Transform & Lighting

(преобразование координат вершин в плоские координаты, отображаемые на мониторе, и вычисление их освещенности)

Третье поколение

nVidia GeForce 2, 3, 4; Radeon 8500 – 9200

Возможность программирования (шейдеры)

- ассемблер для граф. процессора
- ограничены по длине
- Формат с фиксированной запятой

Четвертое поколение

nVidia GeForce 5, 6, 7; ATI Radeon 9500 – X800

- 32-х разрядная точность
- Direct3D, OpenGL, HLSL, GLSL, Cg
- начало GPGPU (General Purpose GPU)

Пятое поколение

nVidia GeForce 8 – GTX 200;

ATI Radeon X 1K – HD 5K

- расширенные возможности программирования (унифицированные шейдеры, CUDA, AMD FireStream)

GPGPU

General Purpose computing on Graphical
Processing Units

- assembler
- Шейдеры
- OpenGL, DirectX
- CUDA, ATI FireStream, OpenCL

Частоты CPU

2004 г. - Pentium 4, 3.46 GHz

2005 г. - Pentium 4, 3.8 GHz

2006 г. - Core Duo T2700, 2333 MHz

2007 г. - Core 2 Duo E6700, 2.66 GHz

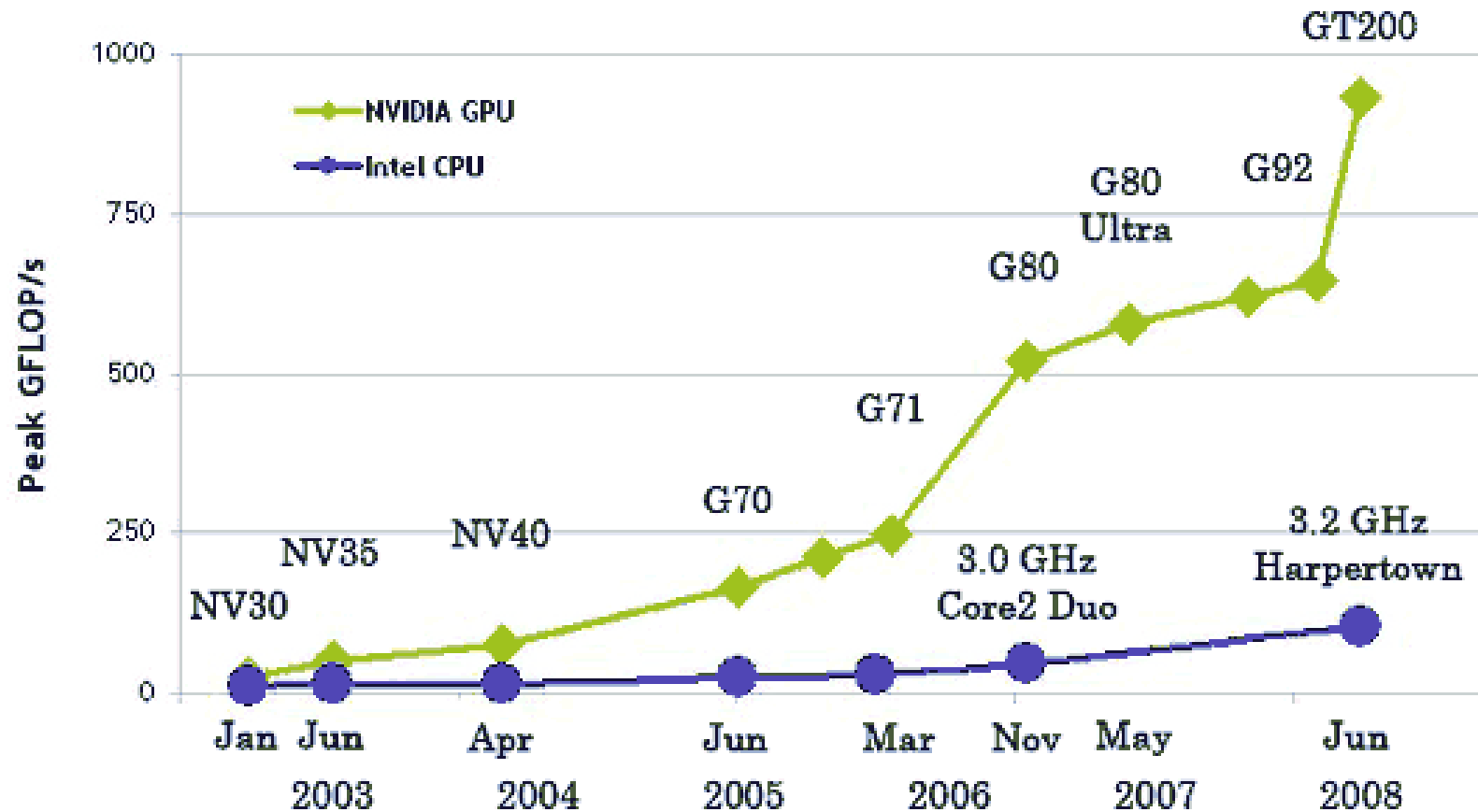
2007 г. - Core 2 Duo E6800, 3 GHz

2008 г. - Core 2 Duo E8600, 3.33 Ghz

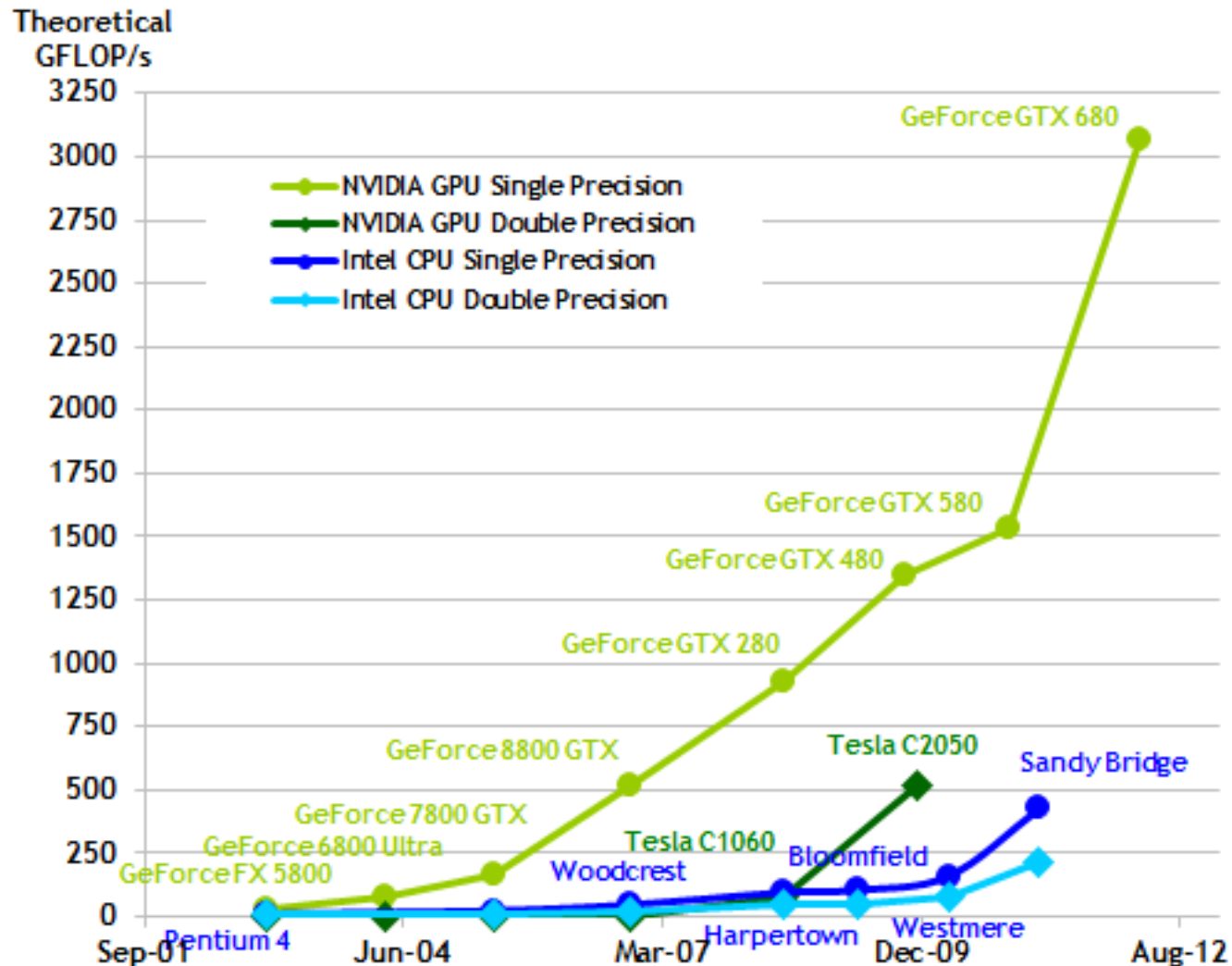
2009 г. - Core i7 950, 3.06 GHz

2013 г. – Core i7 3970x, 3.5 GHz

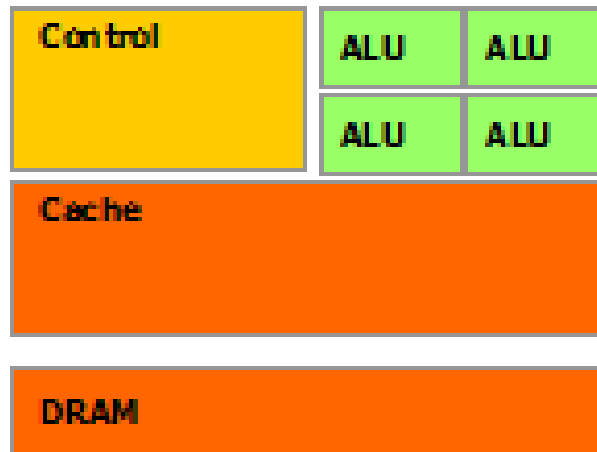
Floating point operations per second



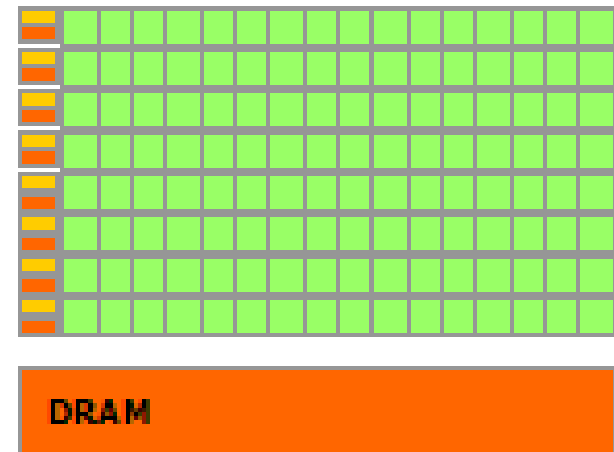
Floating point operations per second



Разница между CPU и GPU



CPU



GPU

CUDA = Compute Unified Device Architecture

- Многократное ускорение в каждой конкретной задаче достигается в результате усилий программиста
- Существовали ранние попытки использования графических карт для научных расчётов
- Проблема – алгоритмы должны были быть реализованы на специальном **“шейдерном” языке** (shade language)
- Для облегчения работы с алгоритмами общего назначения
- компания NVIDIA выдвинула инициативу создания **аппаратно/программной архитектуры общего назначения**

Общие положения CUDA

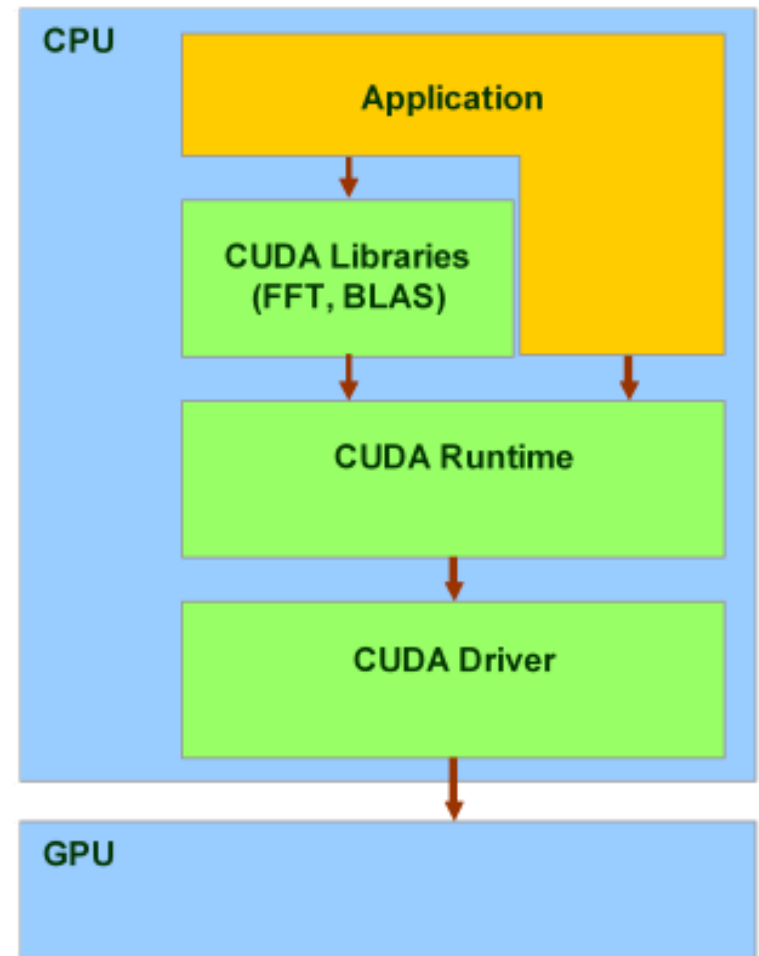
- GPU – **сопроцессор** для CPU (**хоста**)
- У GPU есть собственная память (**device memory**)
- GPU способен одновременно обрабатывать множество процессов (**threads**) данных одним и тем же алгоритмом
- Для осуществления расчётов при помощи GPU хост должен осуществить запуск вычислительного ядра (**kernel**), который определяет конфигурацию GPU в вычислениях и способ получения результатов (алгоритм)
- Процессы GPU (в отличие от CPU) очень просты и многочисленны

Возможности NVIDIA CUDA

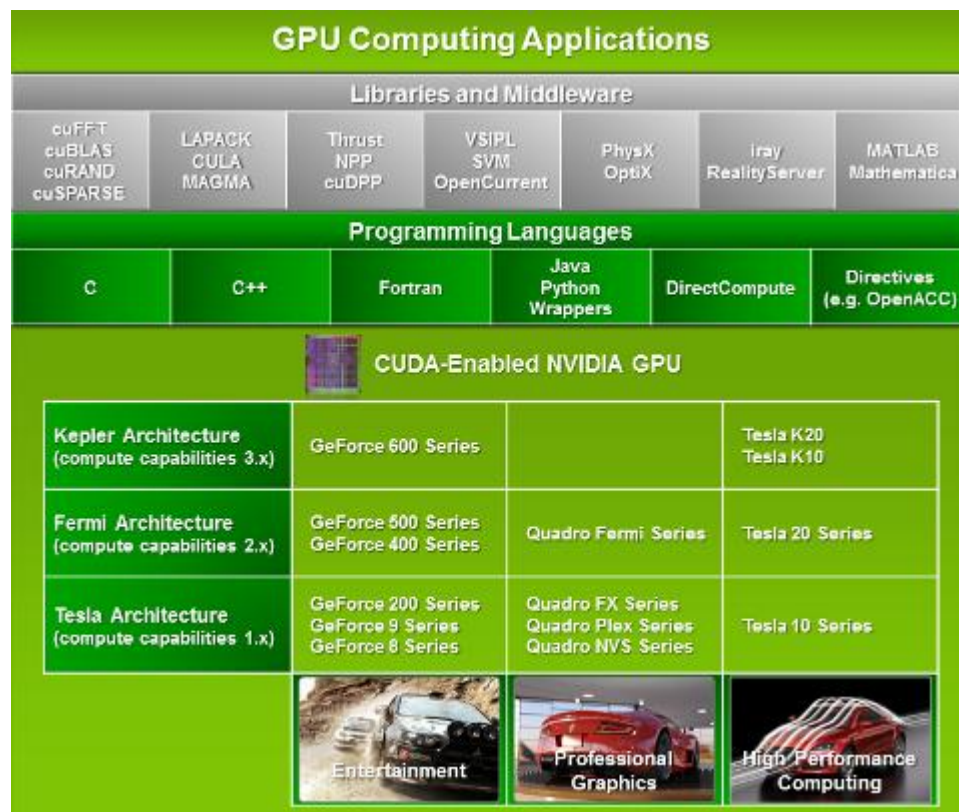
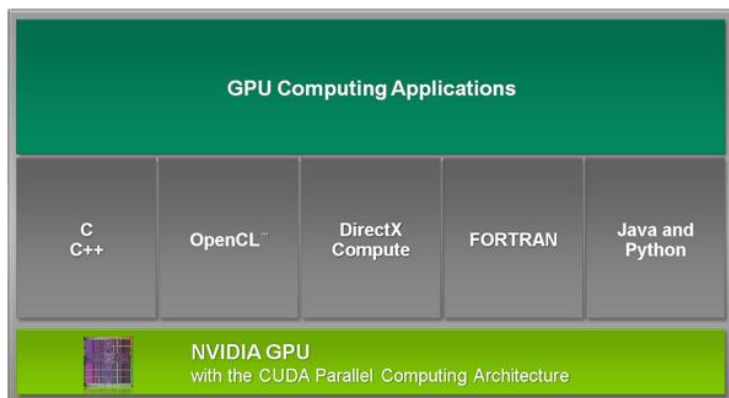
- унифицированное программно-аппаратное решение для параллельных вычислений на видеочипах NVIDIA
- большой набор поддерживаемых решений, от мобильных до мультичиповых
- стандартный язык программирования Си
- дополнительные библиотеки работающие с GPU
- оптимизированный обмен данными между CPU и GPU
- взаимодействие с графическими API OpenGL и DirectX
- поддержка 32- и 64-битных операционных систем: Windows XP, Windows Vista, Linux и MacOS X
- возможность разработки на низком уровне

Состав CUDA

- CUDA включает два API: высокого уровня (CUDA Runtime API) и низкого (CUDA Driver API)
- **CUBLAS** — CUDA вариант BLAS (Basic Linear Algebra Subprograms), предназначенный для вычислений задач линейной алгебры и использующий прямой доступ к ресурсам GPU;
- **CUFFT** — CUDA вариант библиотеки Fast Fourier Transform для расчёта быстрого преобразования Фурье, широко используемого при обработке сигналов.



Модель CUDA



Поддерживаемые устройства

[Log In](#)

[HOME/ABOUT](#) | [TECHNOLOGIES](#) | [TOOLS](#) | [RESOURCES](#) | [CONTACT](#)

CUDA ZONE

[Get the Newsletter](#)

[Home](#) | [Education](#) | [Tech & Innovation](#)

CUDA GPUs

100% GPU-accelerated software solutions, ready-made and open-source, around the world, accelerating (1) big data, (2) machine learning, (3) scientific, (4) financial, and (5) creative work.

Find Out All About CUDA and GPU Computing by watching our [GPU Computing Video Series](#) and joining our free [GPU Computing Developer Program](#).

• Learn about how for technical and scientific computing

• Learn about GPU for professional visualization


CUDA-Enabled Tesla GPU Computing Products

	Tesla Workstation Products			Tesla Data Center Products	
	GPU	Compute Capability		GPU	Compute Capability
	Tesla K10	3.0		Tesla K10	3.0
	Tesla C20	3.0		Tesla C20	3.0
	Tesla D2000	2.0		Tesla M100	2.0
	Tesla D1000	2.0		Tesla M100	2.0
	Tesla D100	3.0		Tesla D100	3.0
	Tesla D100	3.0		Tesla D100	3.0
	Tesla S100	3.0		Tesla S100	3.0
	Tesla S100	3.0		Tesla S100	3.0

CUDA-Enabled Quadro Products


	Quadro Workstation Products			Quadro Mobile Products	
	GPU	Compute Capability		GPU	Compute Capability
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro 6000	3.0		Quadro 6000	3.0
	Quadro				

CUDA-Enabled NVS Products

	Desktop Products		Mobile Products	
	GPU	Compute Capability	GPU	Compute Capability
	Quadro NVQ 400	3.1	NVS 1400M	3.1
	Quadro NVQ 420	3.1	NVS 2100M	3.1
	1910SL NVL 320	3.2	NVS 400M	3.2
	Quadro NVQ 270	3.1	NVS 3100M	3.2
			NVS 1300M	3.2
			NVS 1200M	3.2

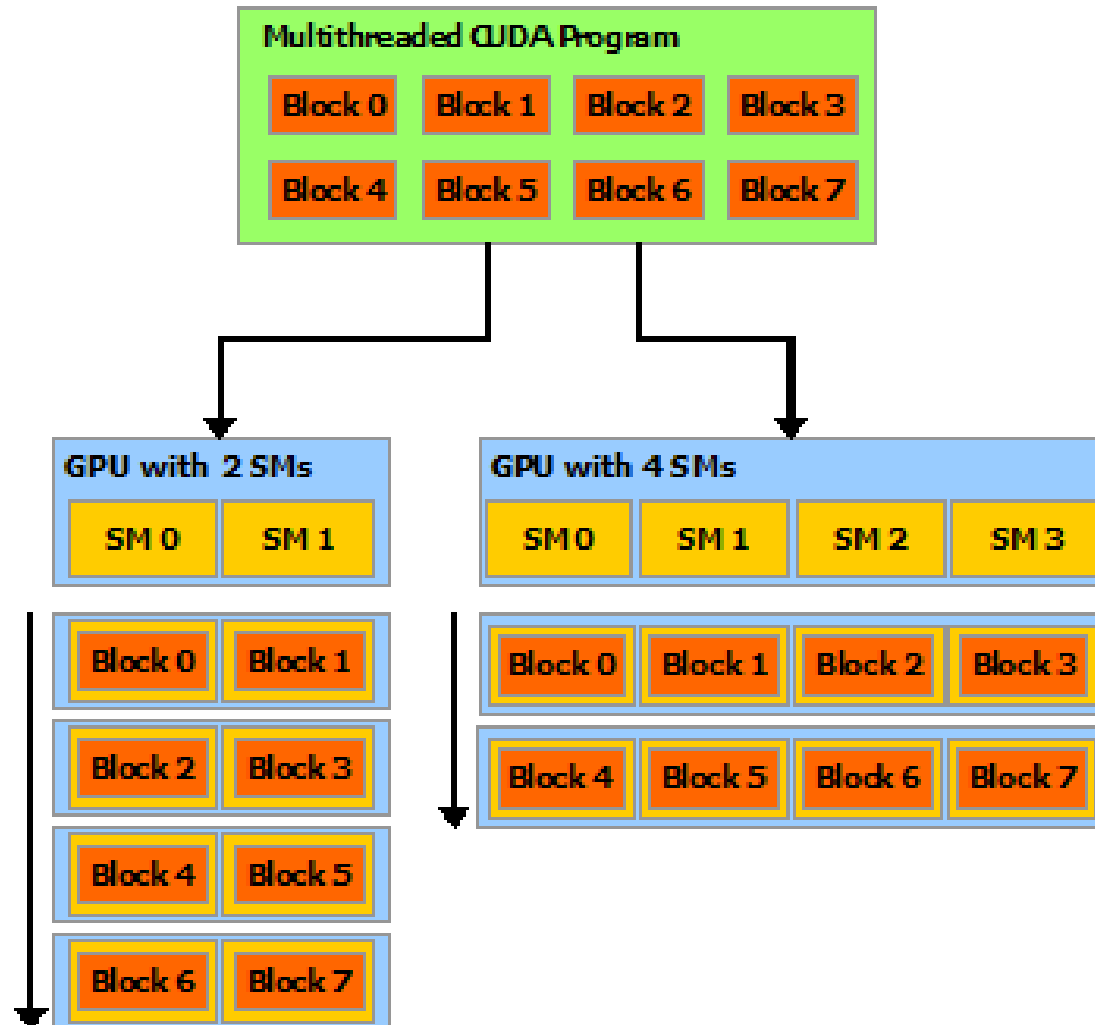
CUDA-Enabled GeForce Products

(GeForce 3, 4, 5, 680, 700, 400-series, 500-series, and 600-series GPUs with a minimum of 128MB of local graphics memory)

	GeForce Desktop Products		GeForce Notebook Products	
	GPU	Compute Capability	GPU	Compute Capability
	GeForce 570 Ti MAX	3.5	GeForce 570c M0M	3.0
	GeForce 570c T80	3.5	GeForce 570c F70M	3.0
	GeForce 570c S70	3.0	GeForce 570c N0M	3.0
	GeForce 570c T60	3.0	GeForce 570c N0M	3.0
	GeForce 570c A90	3.0	GeForce 570c B0M0	3.0
	GeForce 570c G00	3.0	GeForce 570c B0M0	3.0

[illegible][illegible]

Распараллеливание CUDA



SIMD подход

Single Instruction Multiple Data

группа параллельно работающих процессоров осуществляют действия над разными данными, но при этом все они в произвольный момент времени должны выполнять одинаковую команду

Классификация по Флинну

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

Классы систем

- CPU (одноядерный) — SISD (одновременно выполняется только одна инструкция над одним набором операндов);
- CPU (многоядерный) — MIMD (Одновременно несколько ядер могут работать совершенно независимо, каждое как SISD);
- GPU (NVIDIA *ComputeCapability* версии < 2.0) — SIMD (одновременно на графическом адаптере может выполняться только один поток вычислений, который работает с большим набором данных);
- GPU (NVIDIA *ComputeCapability* версии ≥ 2.0) — MIMD (одновременно на графическом адаптере может выполняться несколько потоков вычислений, каждый из которых работает с большим набором данных).

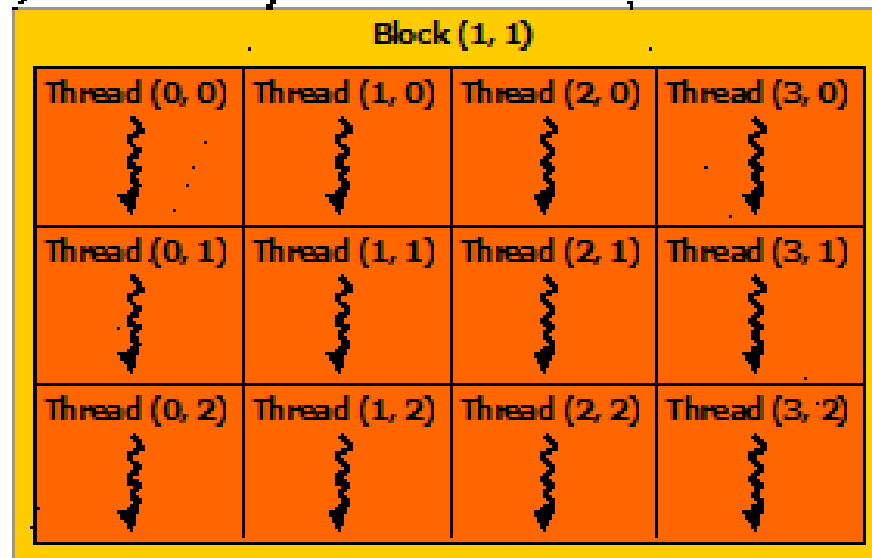
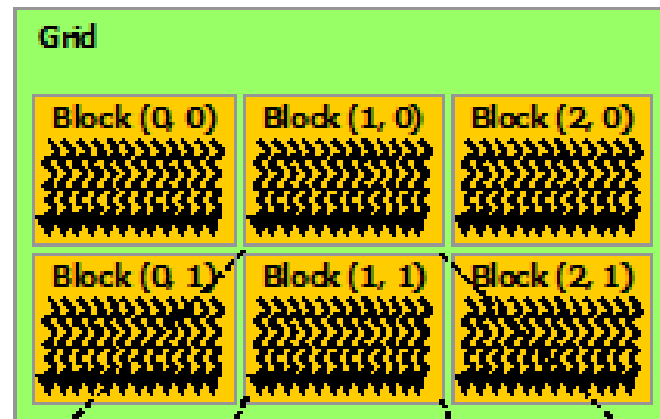
Графические процессоры изначально предназначены для параллельного решения одной массивно-параллельной задачи

Ограничения CUDA

- Отсутствие поддержки рекурсии для выполняемых функций на первых видеокартах
- Закрытая архитектура
- Поддержка только устройств NVIDIA

Программная модель

- Хост (Host) — центральный процессор, управляющий выполнением программы.
- Устройство (Device) — видеоадаптер, выступающий в роли сопроцессора центрального процессора.
- Тред (Thread, поток) — единица выполнения программы. Имеет свой уникальный идентификатор внутри блока.
- Блок (Block) — объединение тредов, которое выполняется целиком на одном SM. Имеет свой уникальный идентификатор внутри грида.
- Грид (Grid) — объединение блоков, которые выполняются на одном устройстве.
- Ядро (Kernel) — параллельная часть алгоритма, выполняется на гриде.
- Варп (Warp) — 32 последовательно идущих тредов, выполняется физически одновременно.



C Program Sequential Execution

Serial code

Parallel kernel
Kernel0<<<>>>()

Serial code

Parallel kernel
Kernel1<<<>>>()

Host



Device

Grid 0



Host



Device

Grid 1

