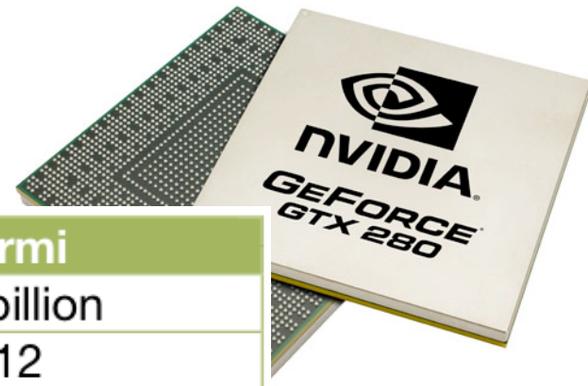
CUDA introduction

Felipe A. Cruz

Nagasaki Advanced Computing Center
Nagasaki University, Japan

NVIDIA GPU Architecture



GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Special Function Units (SFUs) / SM	2	2	4
Warp schedulers (per SM)	1	1	2
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit

Comparison of NVIDIA GPU generations. Current generation: GT200. Table from NVIDIA Fermi white-paper.

Strong points of CUDA

- **Abstracting from the hardware**

  Abstraction by the **CUDA API**. You don't see every little aspect of the machine.

  Gives **flexibility to the vendor**. Change hardware but keep legacy code.

  - **Forward compatible**.

- **Automatic Thread management** (can handle **+100k threads**)

  Multithreading: hides latency and helps maximize the GPU utilization.

  Transparent for the programmer (you don't worry about this.)

  Limited **synchronization between threads** is provided.

  Difficult to dead-lock. (No message passing!)

Programmer effort

- Analyze algorithm for **exposing parallelism**:

  Block size

  Number of threads

  - **Tool: pen and paper**

Challenge: **Keep machine busy** (with limited resources)

Global data set (Have efficient data transfers)

Local data set (Limited on-chip memory)

Register space (Limited on-chip memory)

- **Tool: Occupancy calculator**

Outline

- Thread hierarchy.
- Memory hierarchy.
- Basic C extensions.
- GPU example.

Thread hierarchy

- Kernels are executed by **thread**.

  A kernel is a **simple C** program.

  Each thread has it **own ID**.

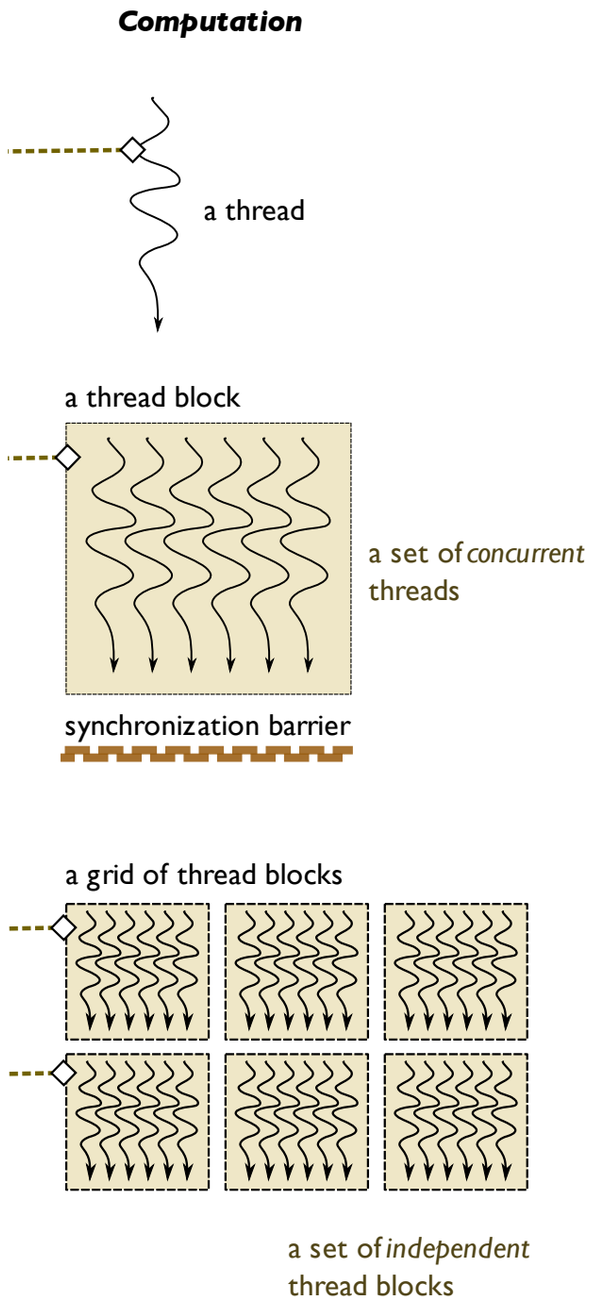
  Thousands of threads execute same kernel.

- Threads are grouped into **blocks**.

  Threads in a block can **synchronize** execution.

- Blocks are grouped in a **grid**.

  Blocks are **independent** (Must be able to be executed in any order.)



Memory hierarchy

- Three **types** of memory in the graphic card:

  Global memory: 4GB

  Shared memory: 16 KB

  Registers: 16 KB

Latency:

  Global memory: 400-600 cycles

  Shared memory: Fast

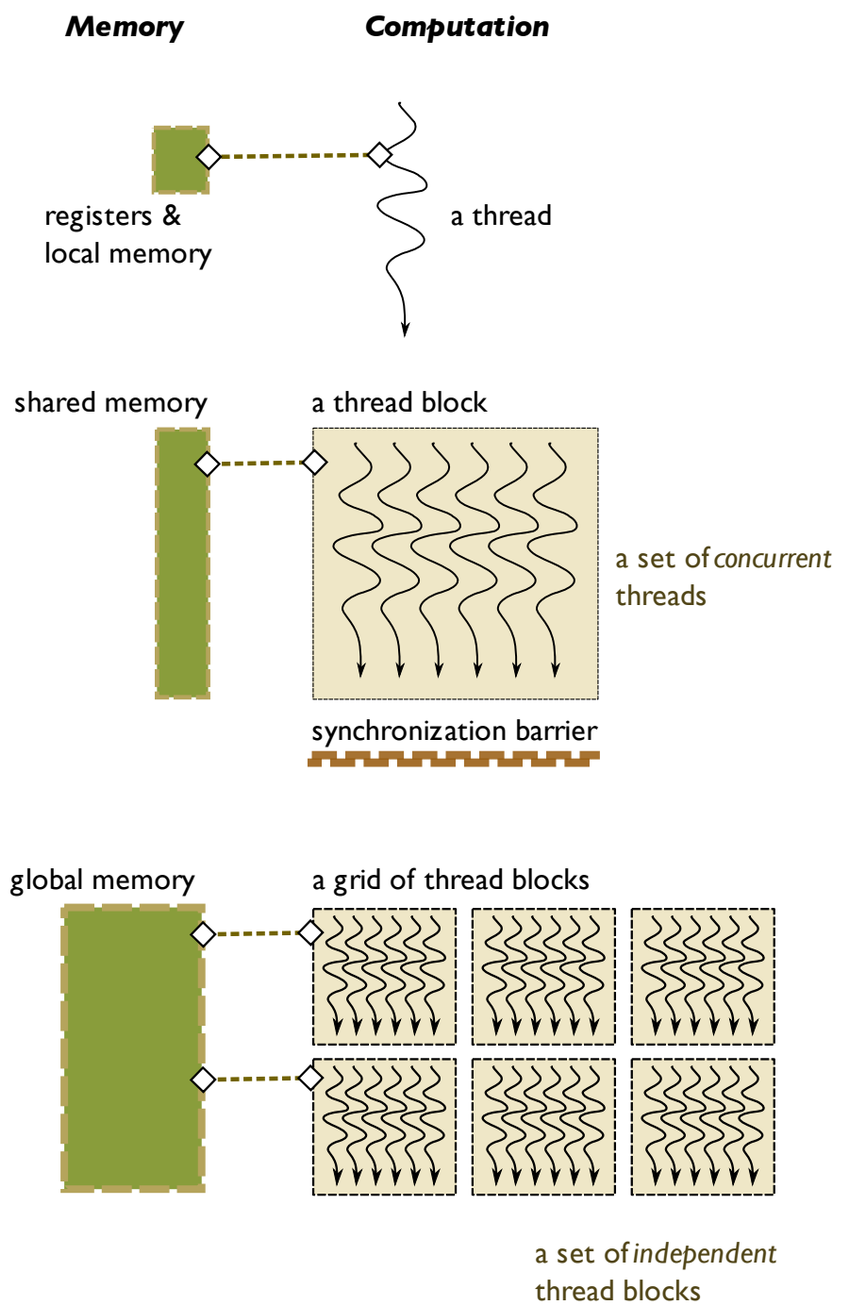
  Register: Fast

Purpose:

  Global memory: IO for grid

  Shared memory: thread collaboration

  Registers: thread space



Basic C extensions

Function modifiers

- __global__ : to be called by the host but executed by the GPU.
- __host__ : to be called and executed by the host.

Kernel launch parameters

- Block size: (x, y, z). x*y*z = Maximum of 768 threads total. (Hw dependent)
- Grid size: (x, y). Maximum of thousands of threads. (Hw dependent)

Variable modifiers

- __shared__ : variable in shared memory.
- __syncthreads() : sync of threads within a block.

Check CUDA programming guide for all the features!