NYU
New York University

CSCI-UA.0480-003

Parallel Computing

Lecture 17: GPUs - Intro

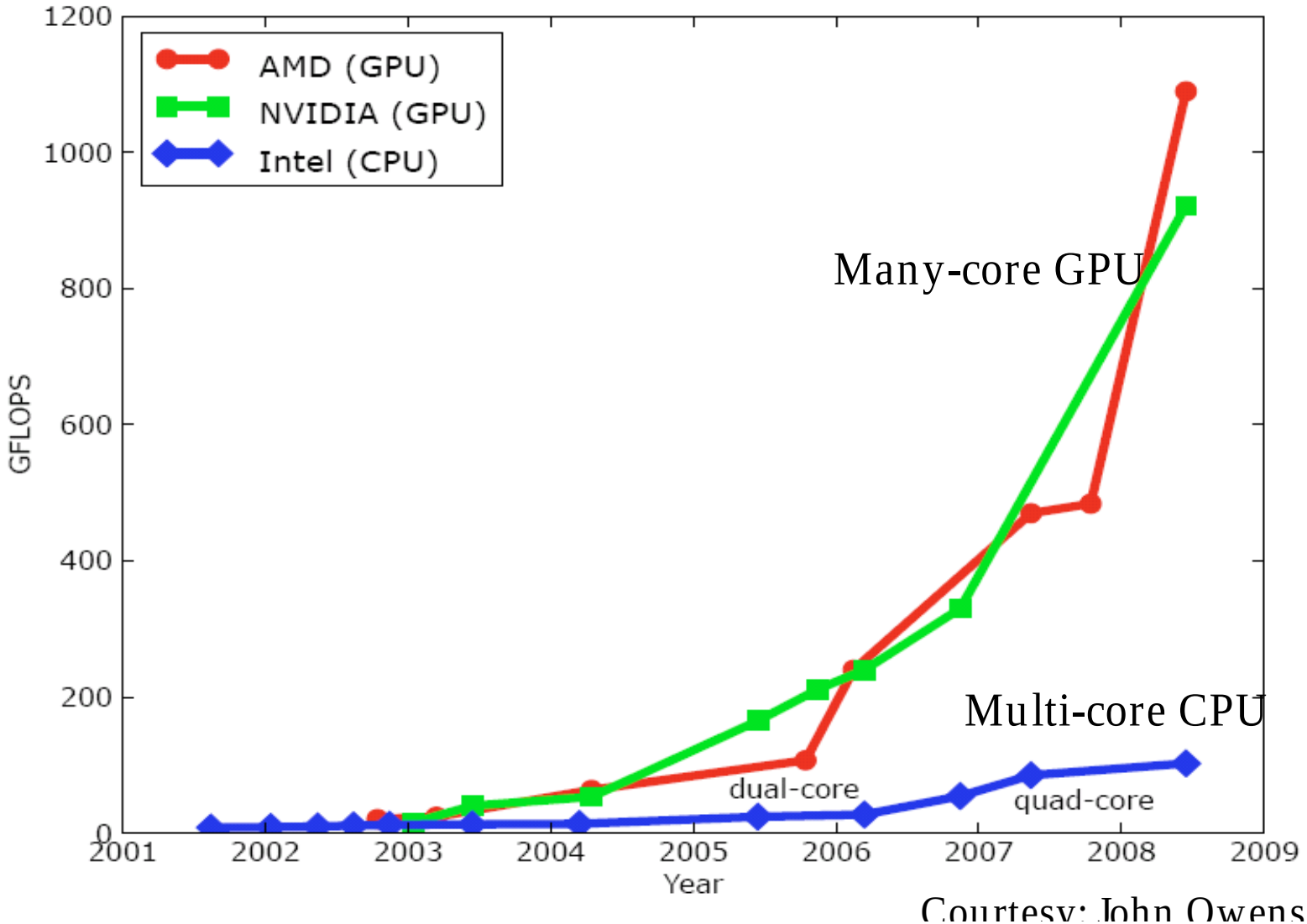
Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

Two Main Goals for Current Architectures

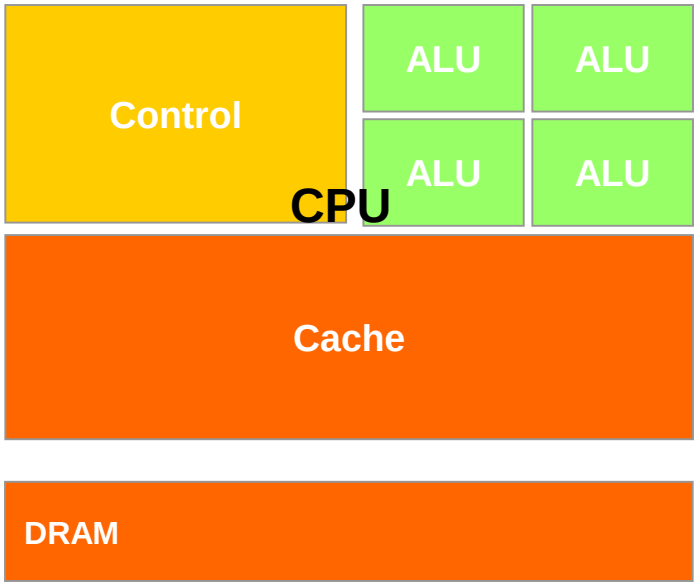
- Maintain execution speed of old sequential programs
- Increase throughput of parallel programs

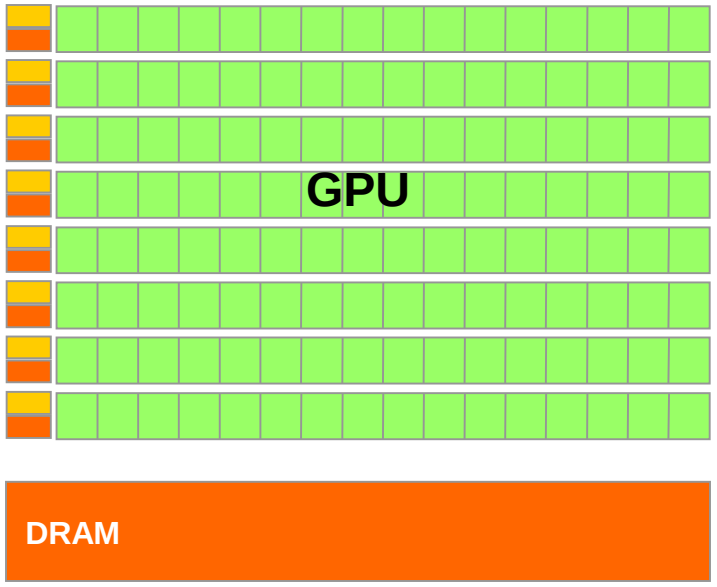
Two Main Goals for Current Architectures

- Maintain execution speed of old sequential programs → **CPU**
- Increase throughput of parallel programs → **GPU**

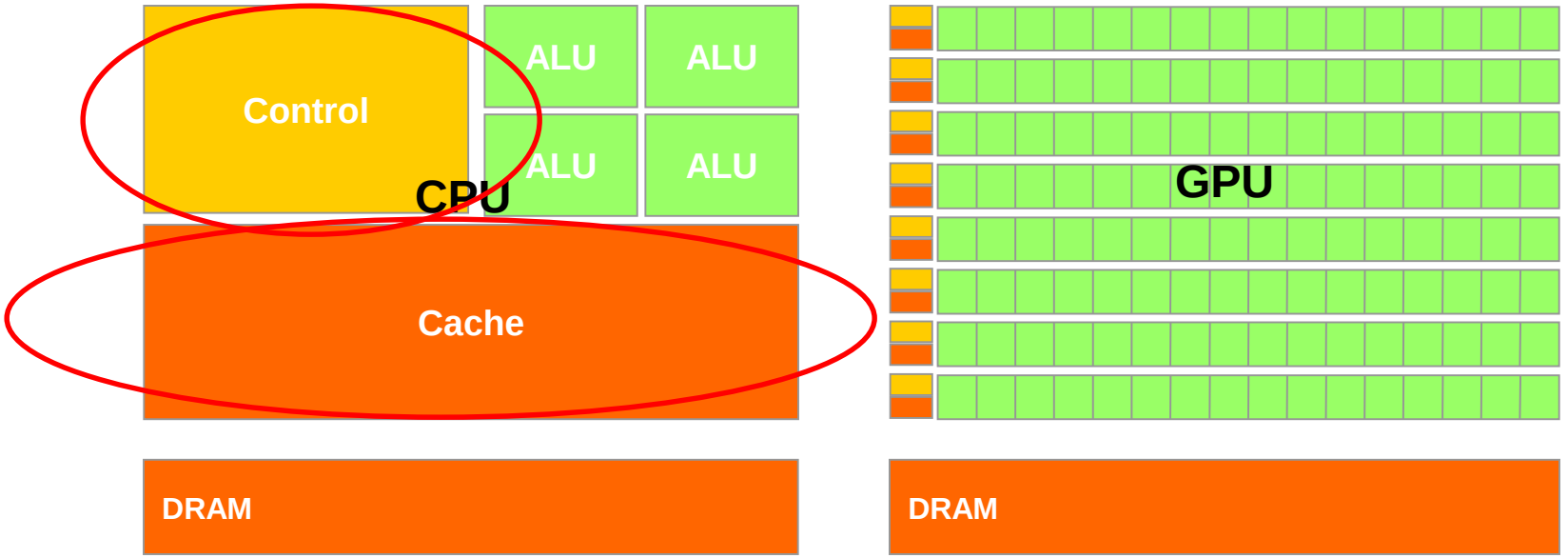


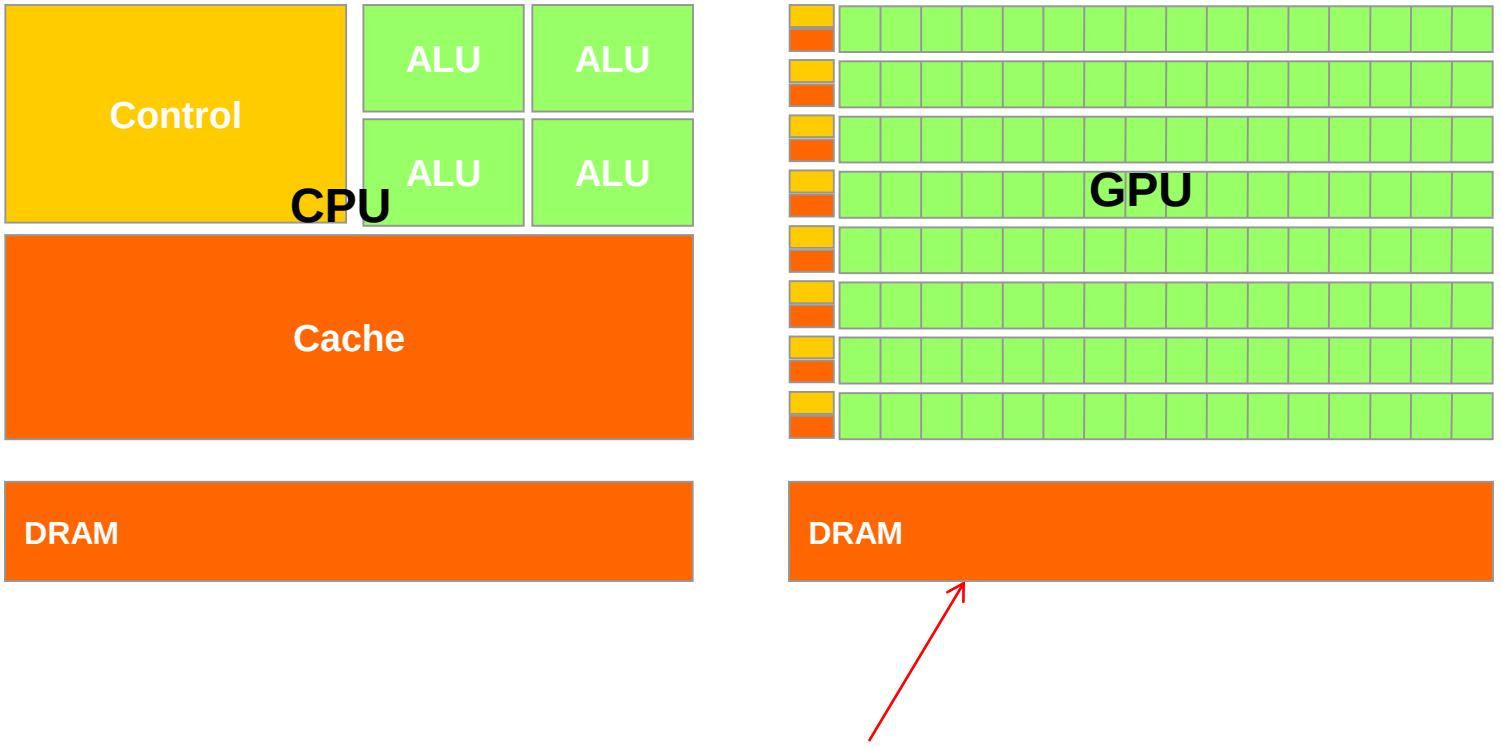
Courtesy: John Owens


Control
ALU
ALU
ALU
ALU
CPU
Cache
DRAM

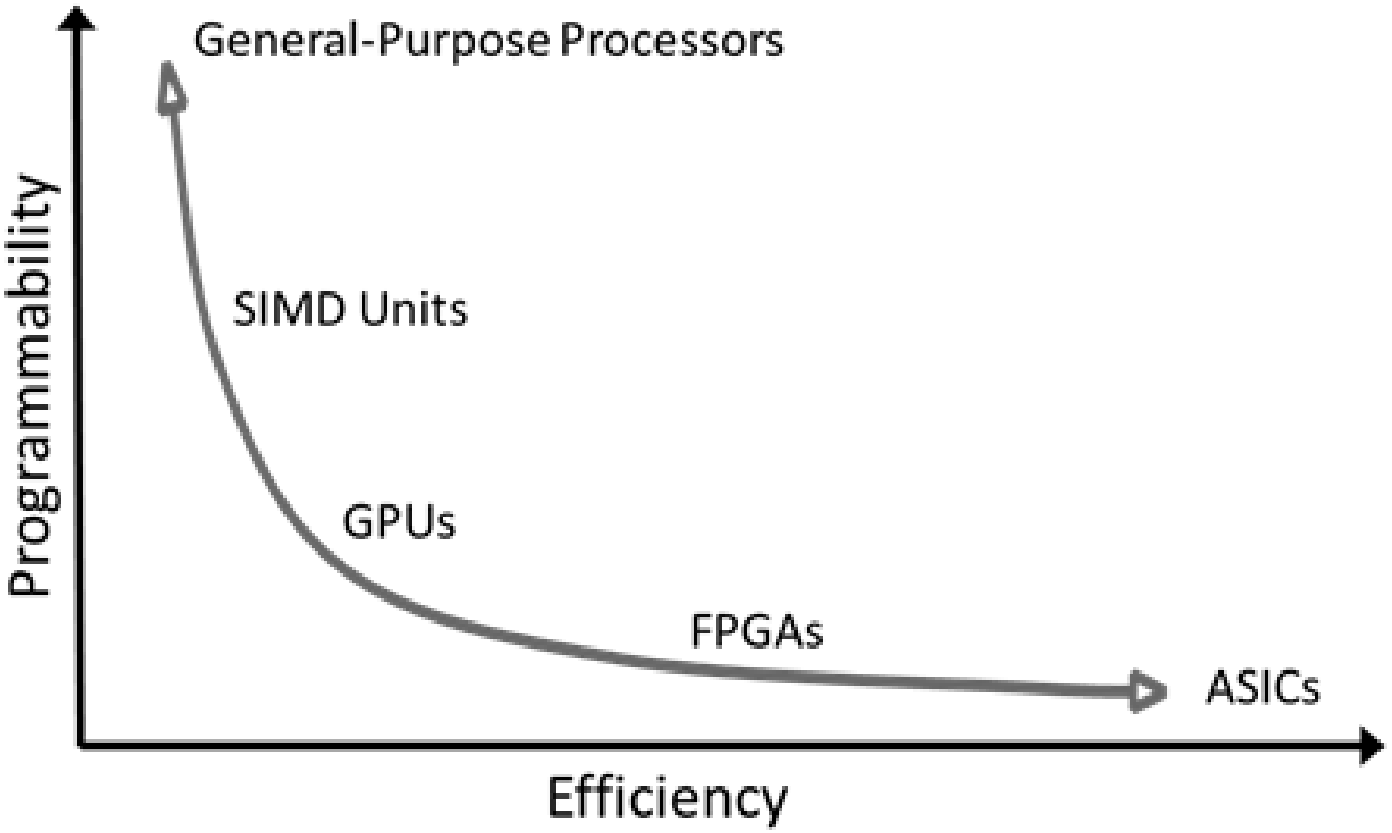

GPU
DRAM

CPU is optimized for sequential code performance




Control
ALU
ALU
ALU
ALU
CPU
Cache
DRAM
GPU
DRAM
Almost 10x the bandwidth of multicore
(relaxed memory model)

Where do GPU stand among other chips?



What are GPUs good for?

Regularity + Massive Parallelism







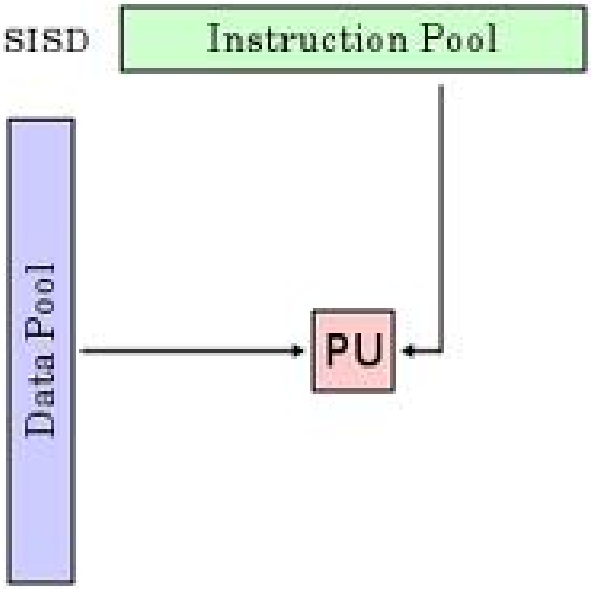

Is Any Application Suitable for GPU?

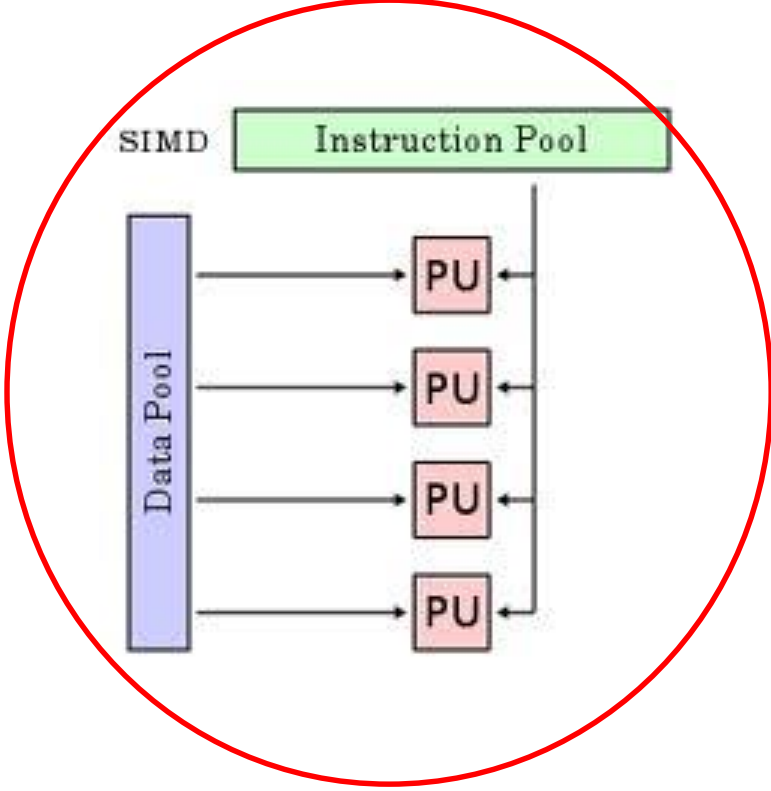
- Heck no!
- You will get the best performance from GPU if your application is:
  - Computation intensive
  - Many independent computations
  - Many similar computations

Let's Remember Flynn Classification

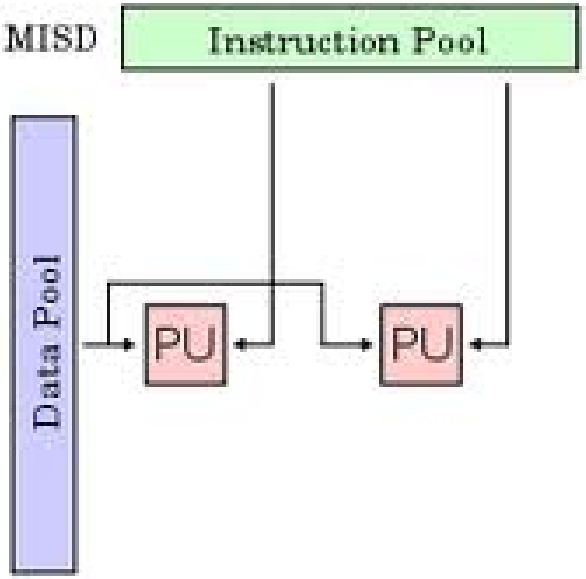
- A taxonomy of computer architecture
- Proposed by Micheal Flynn in 1966
- It is based two things:
  - Instructions
  - Data

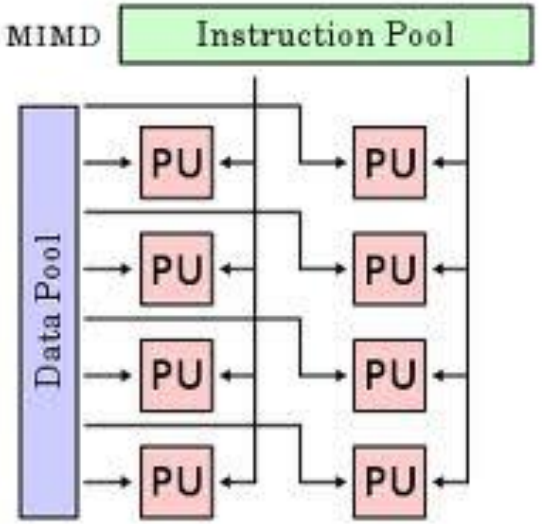
	Single instruction	Multiple instruction
Single data	SISD	MISD
Multiple data	SIMD	MIMD





PU = Processing Unit

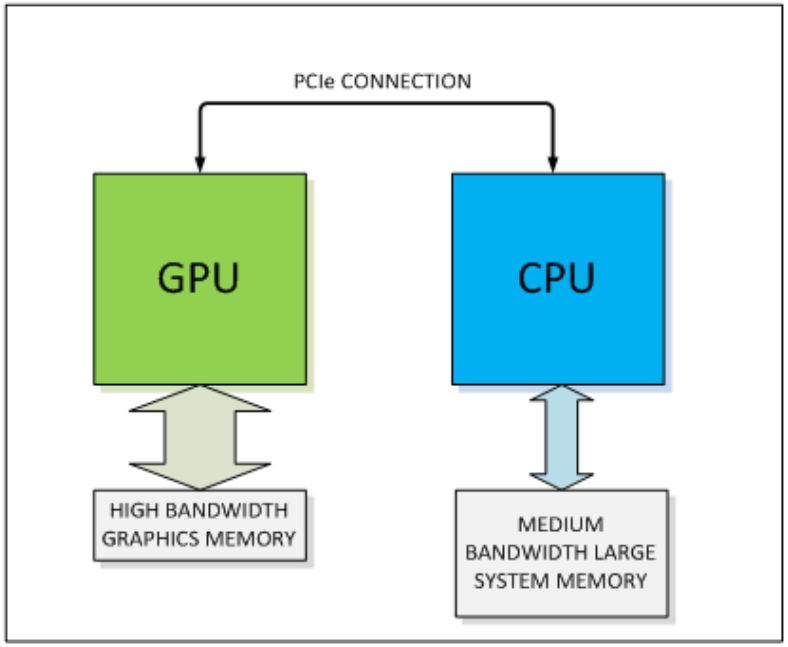


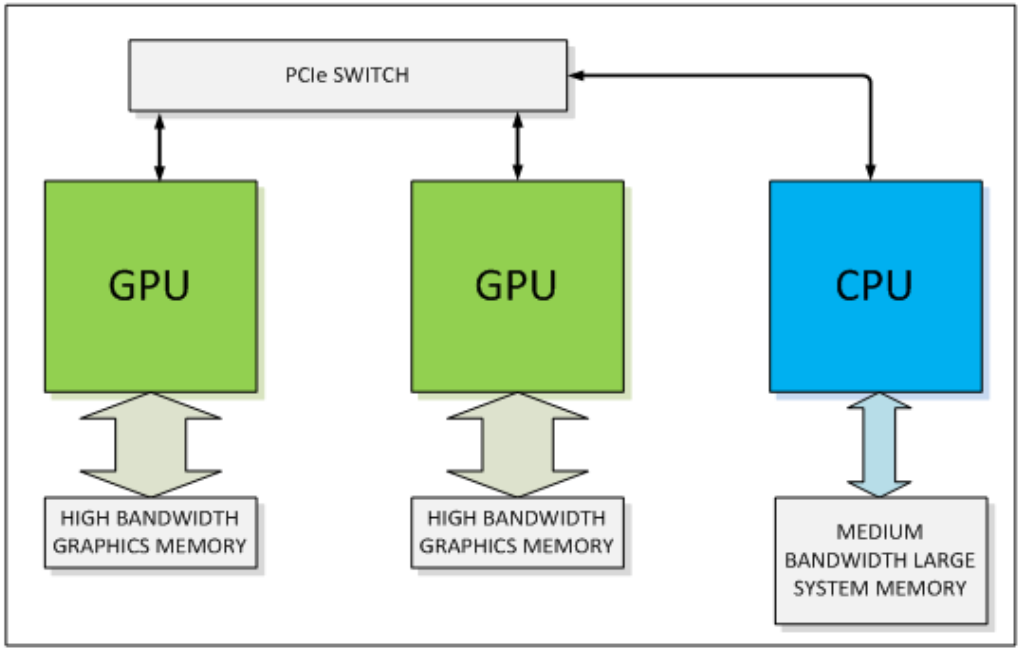


Problems Faced by GPUs

- Need enough parallelism
- Under-utilization
- Bandwidth to CPU

Let's Take A Closer Look: The Hardware

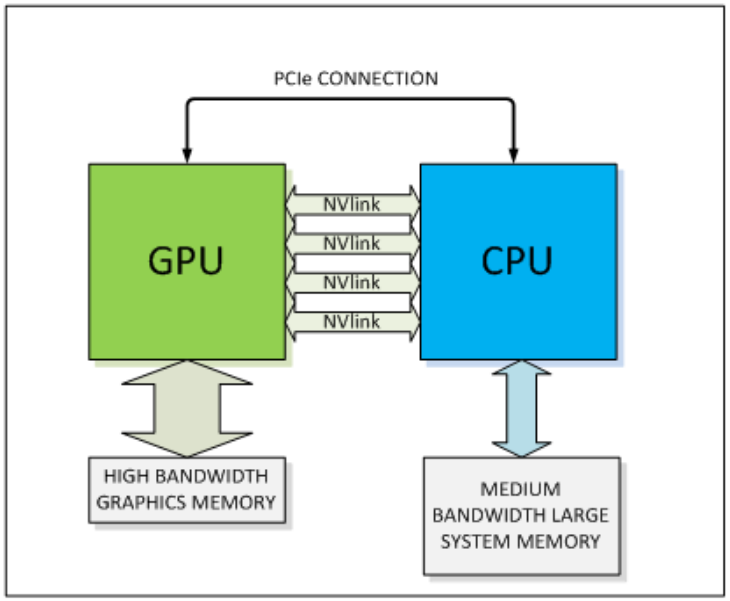


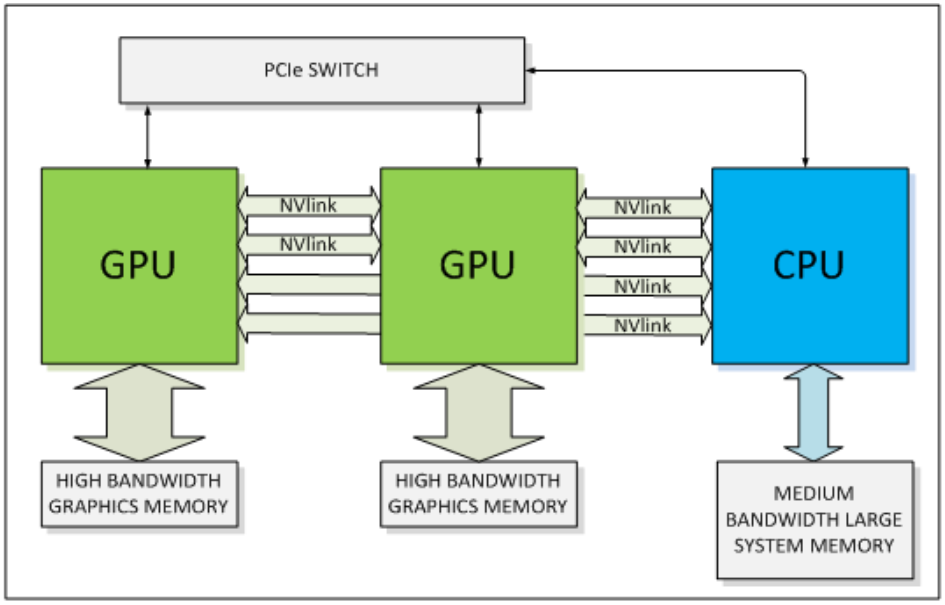


- PCIe 3.0 speeds ~32 GB-transfers per second per lane
- PCIe 4.0 is about the double of version 3.0
- widest supported links = 16 lanes
- Recently: NVLINK

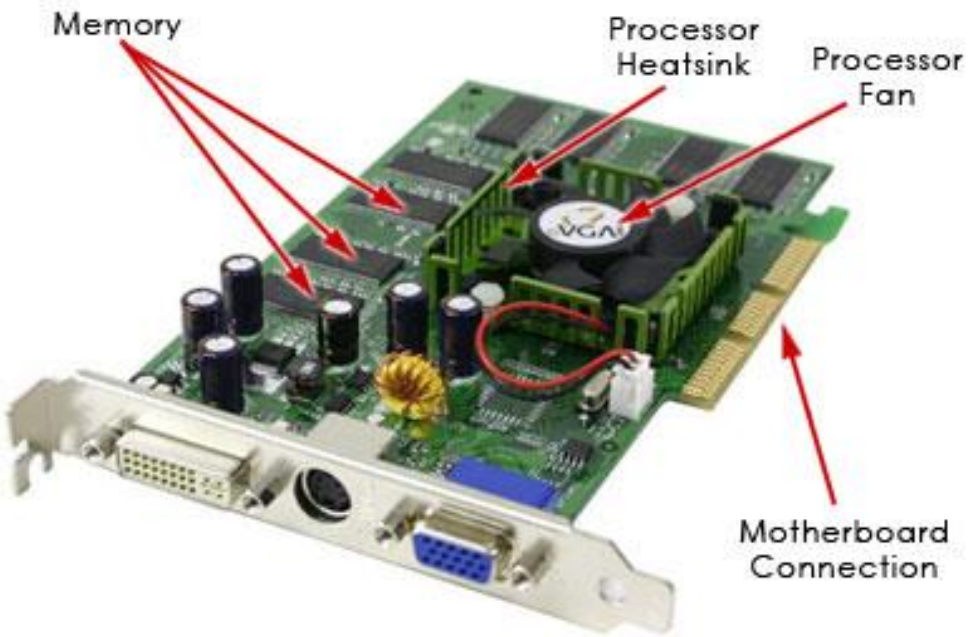
With the new NVLink

Bandwidth of ~80GB/s per link





Source: NVIDIA

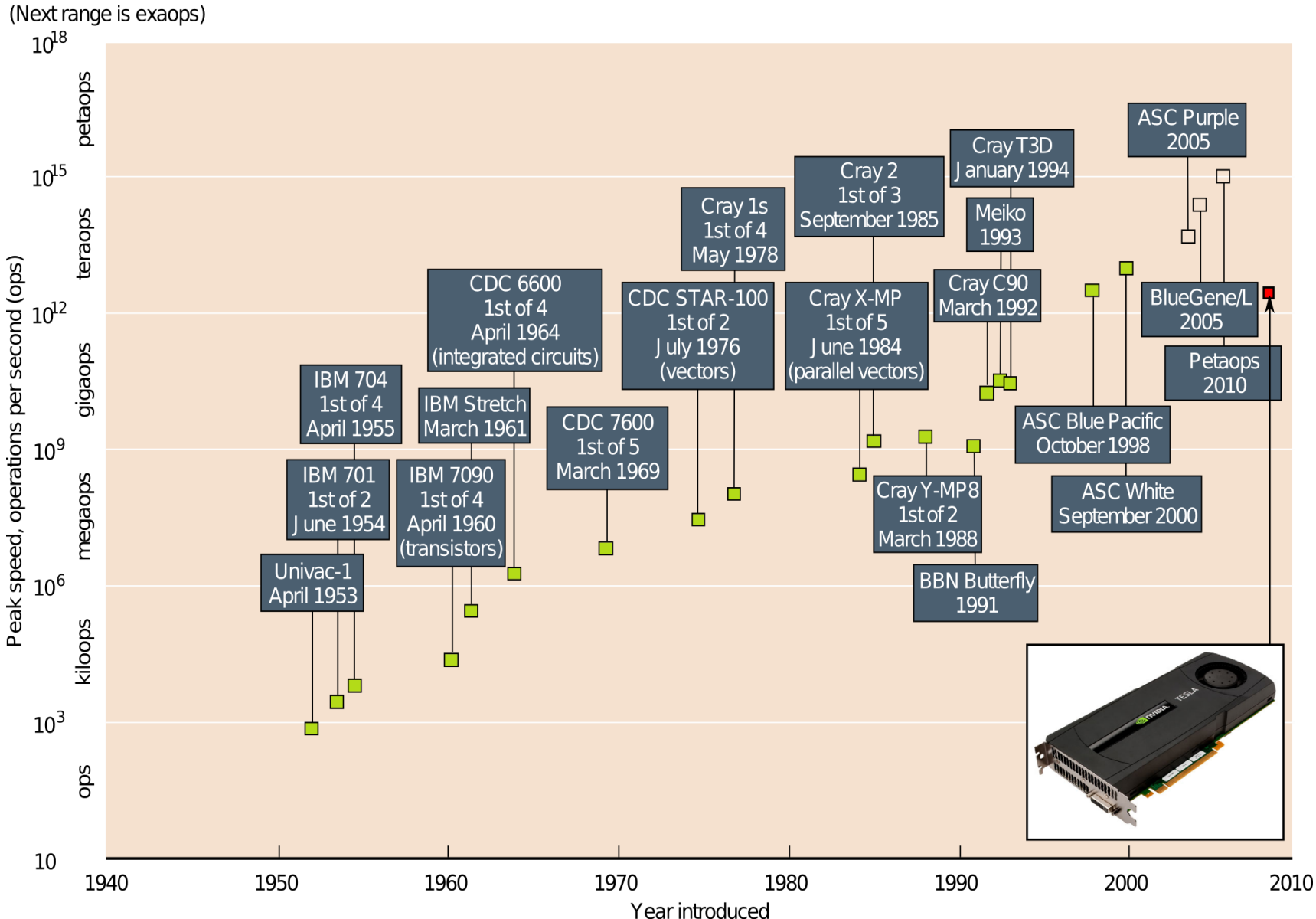


source: http://static.ddmcdn.com/gif/graphics-card-5.jpg

Modern GPU Hardware

- GPUs have many parallel execution units and higher transistor counts, while CPUs have few execution units and higher clock speeds
- GPUs have much deeper pipelines.
- GPUs have significantly faster and more advanced memory interfaces as they need to shift around a lot more data than CPUs

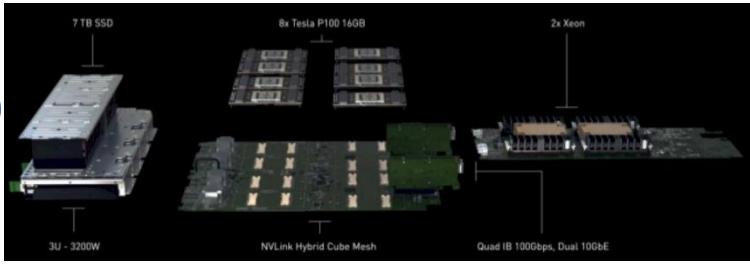
Single-Chip GPU vs Supercomputers



One of the latest GPUs

PASCAL GP100 GPU

DGX-1

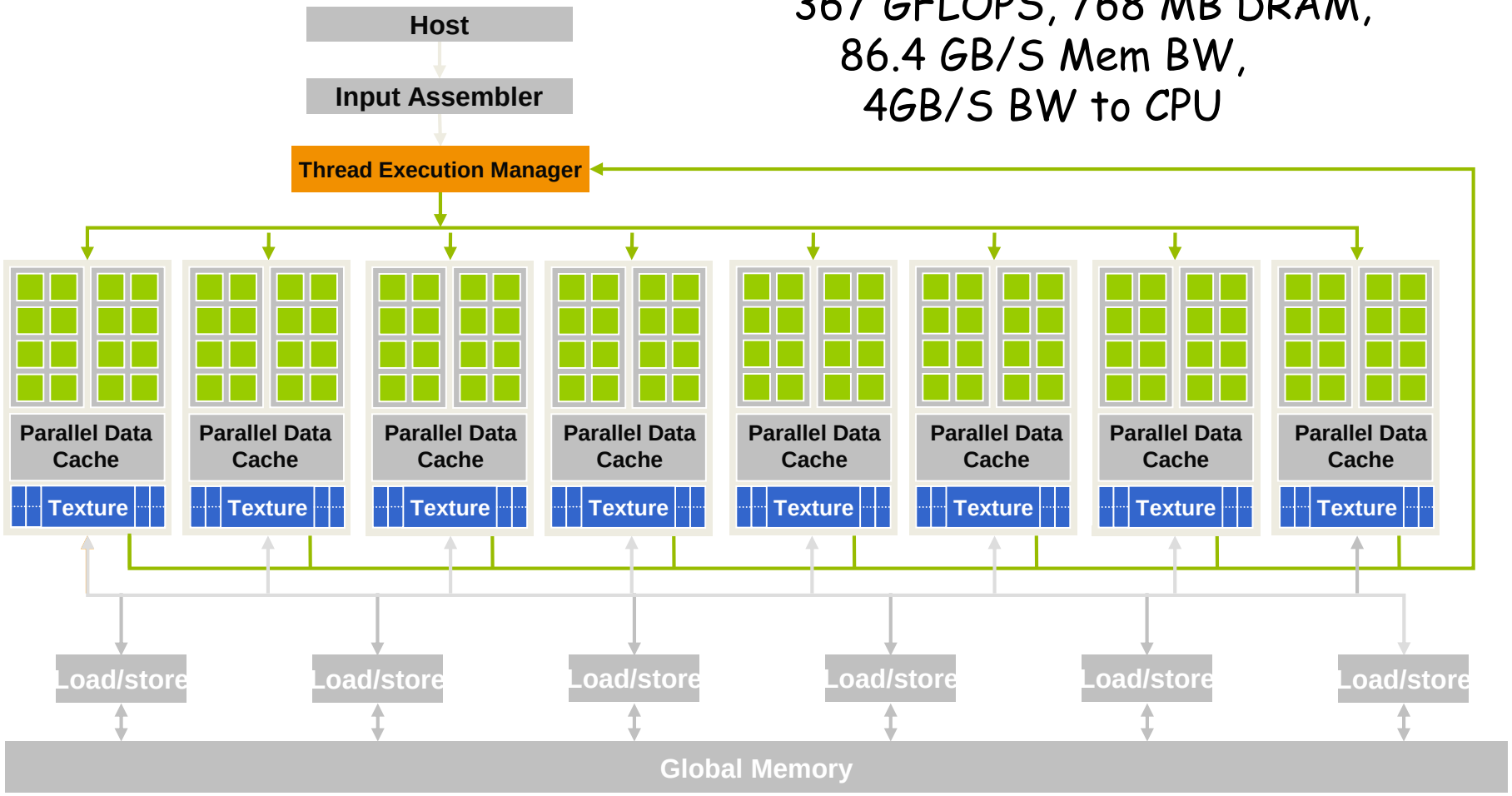


x124

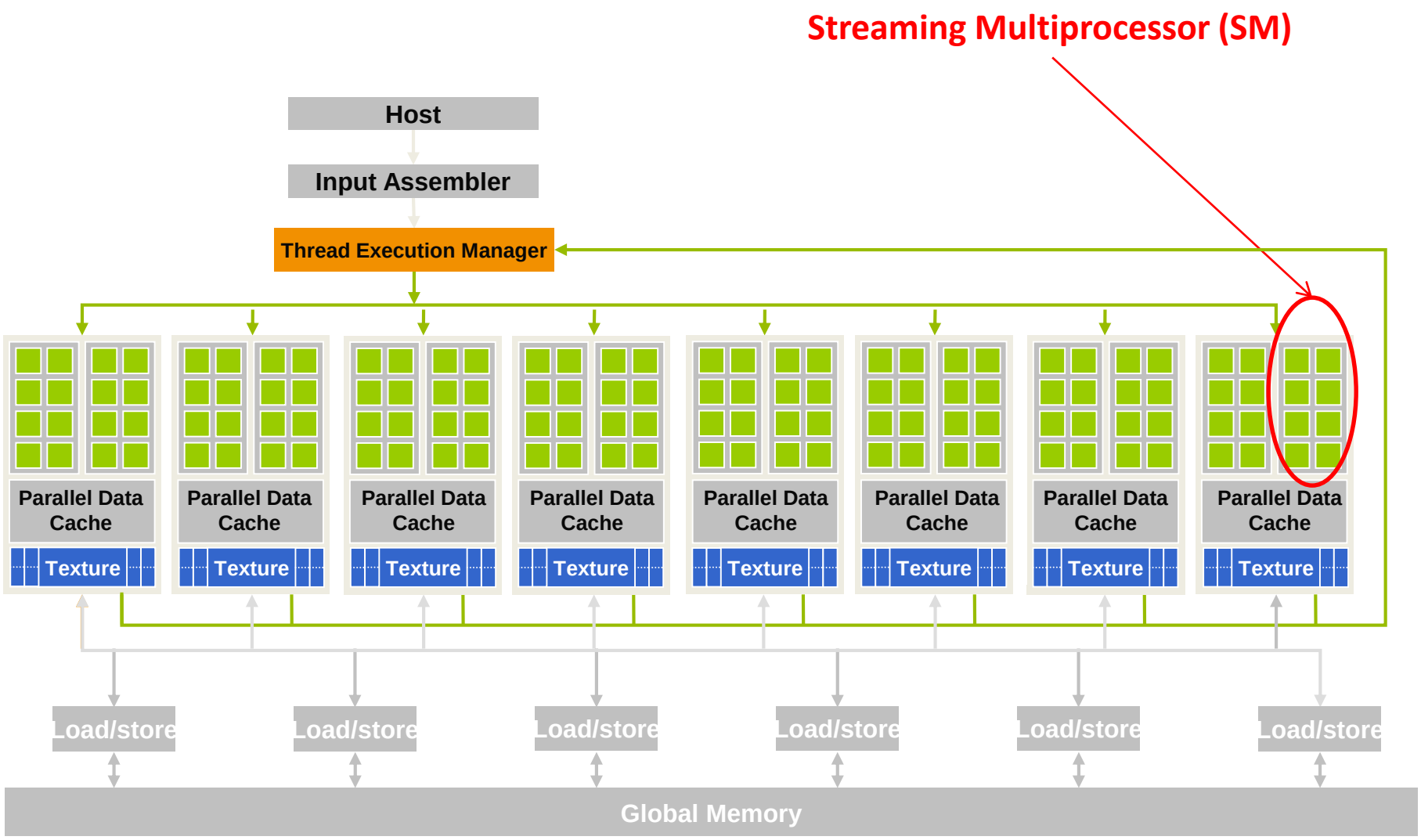
SATURN V
(Top 500 list, Nov 2016)

A Glimpse at At A GPGPU: GeForce 8800 (2007)

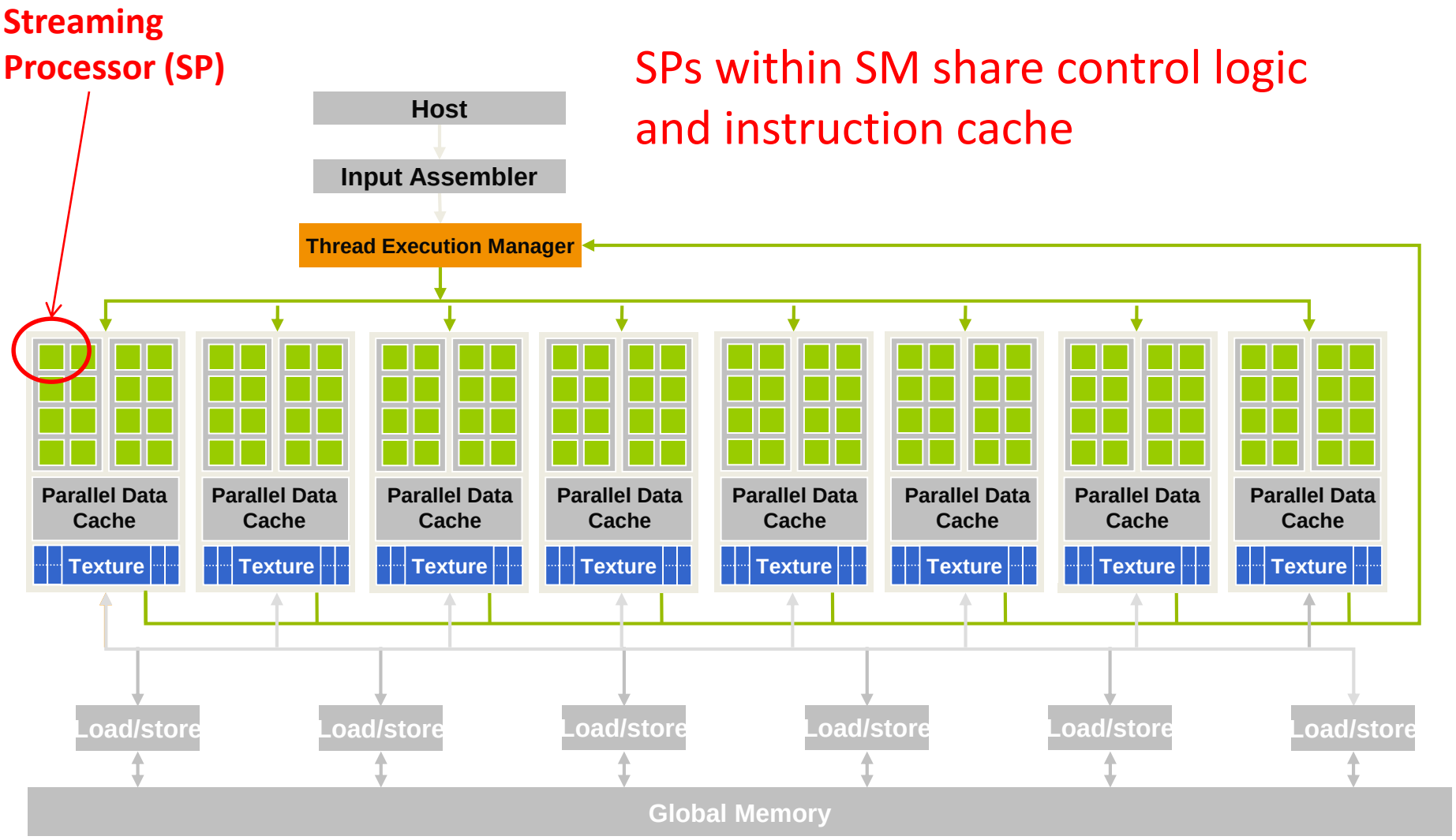
16 highly threaded SM's, >128 FPU's,
367 GFLOPS, 768 MB DRAM,
86.4 GB/S Mem BW,
4GB/S BW to CPU



A Glimpse at A Modern GPU



A Glimpse at A Modern GPU

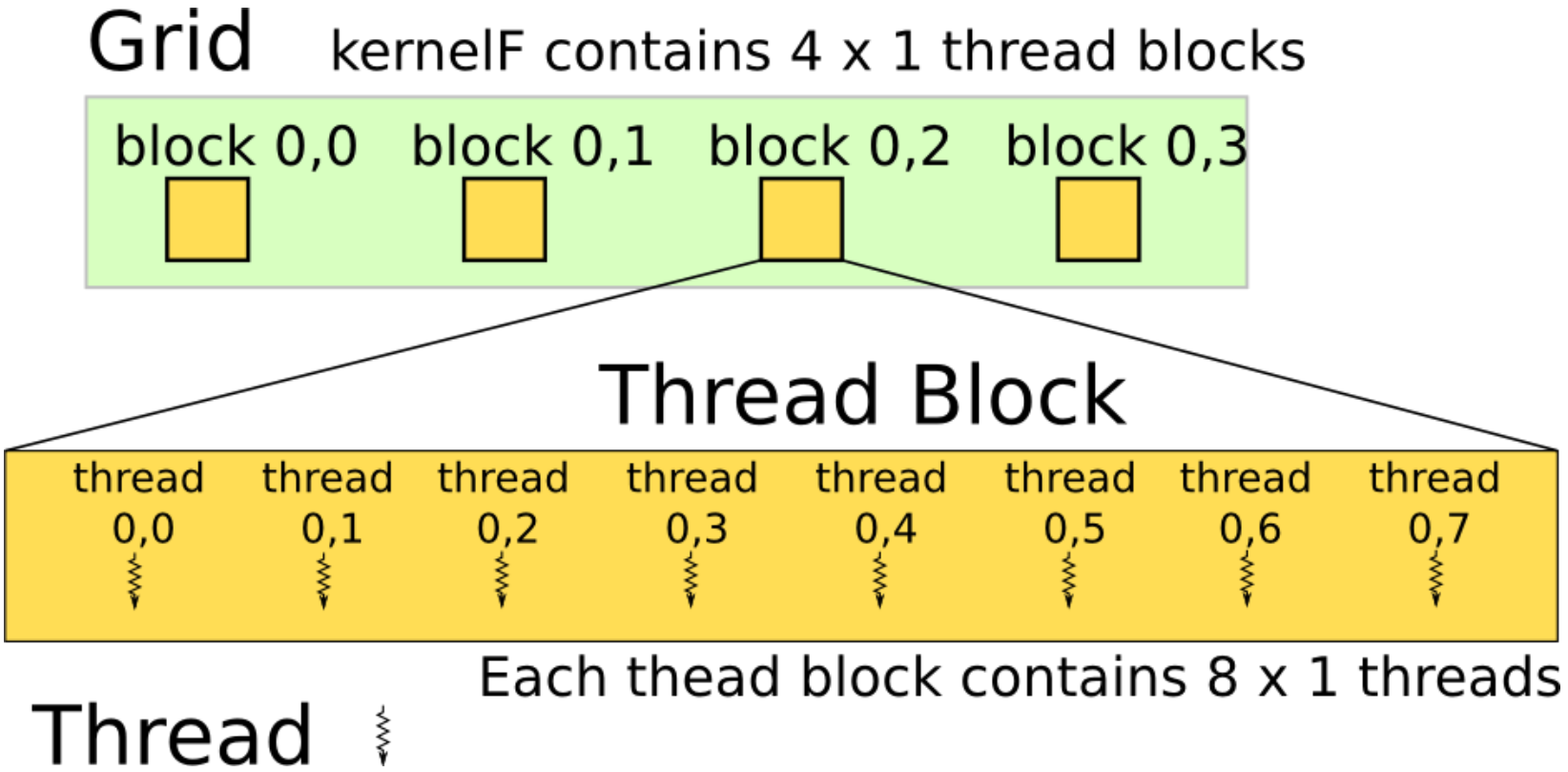


Scalar vs Threaded

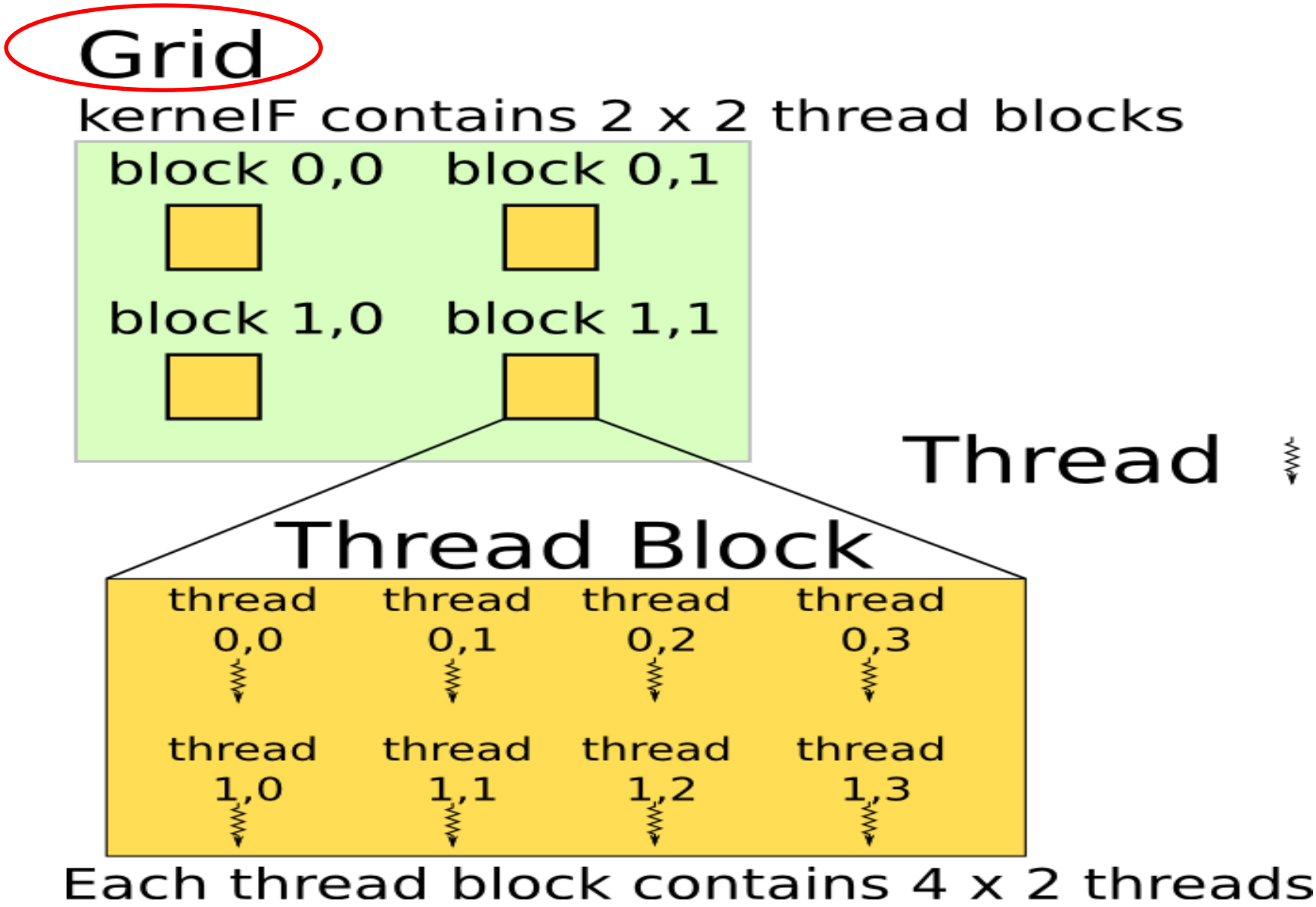
Scalar program (i.e. sequential)

```
float A[4][8];

for(int i=0;i<4;i++){

    for(int j=0;j<8;j++){
        A[i][j]++;
    }
}
```

Multithreaded: (4x1)blocks – (8x1) threads



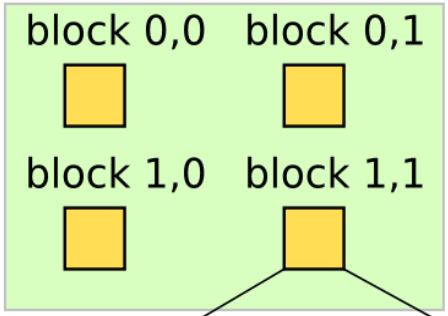
Multithreaded: (2x2)blocks – (4x2) threads

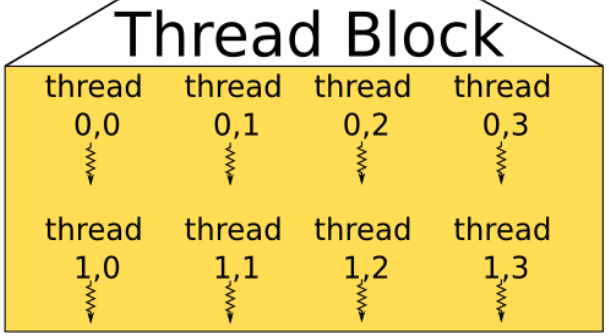


Scheduling Thread Blocks on SM

Grid

kernelF contains 2 x 2 thread blocks

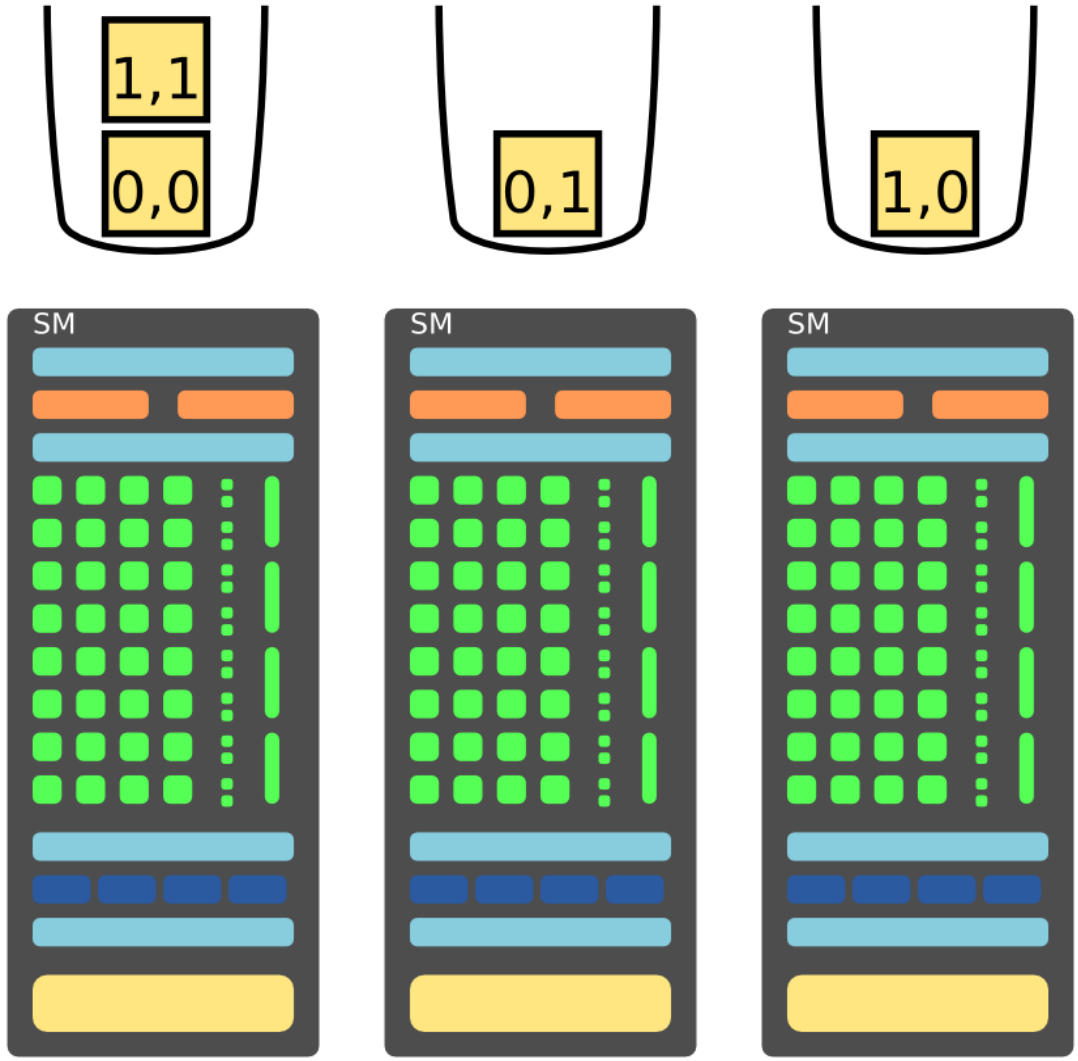




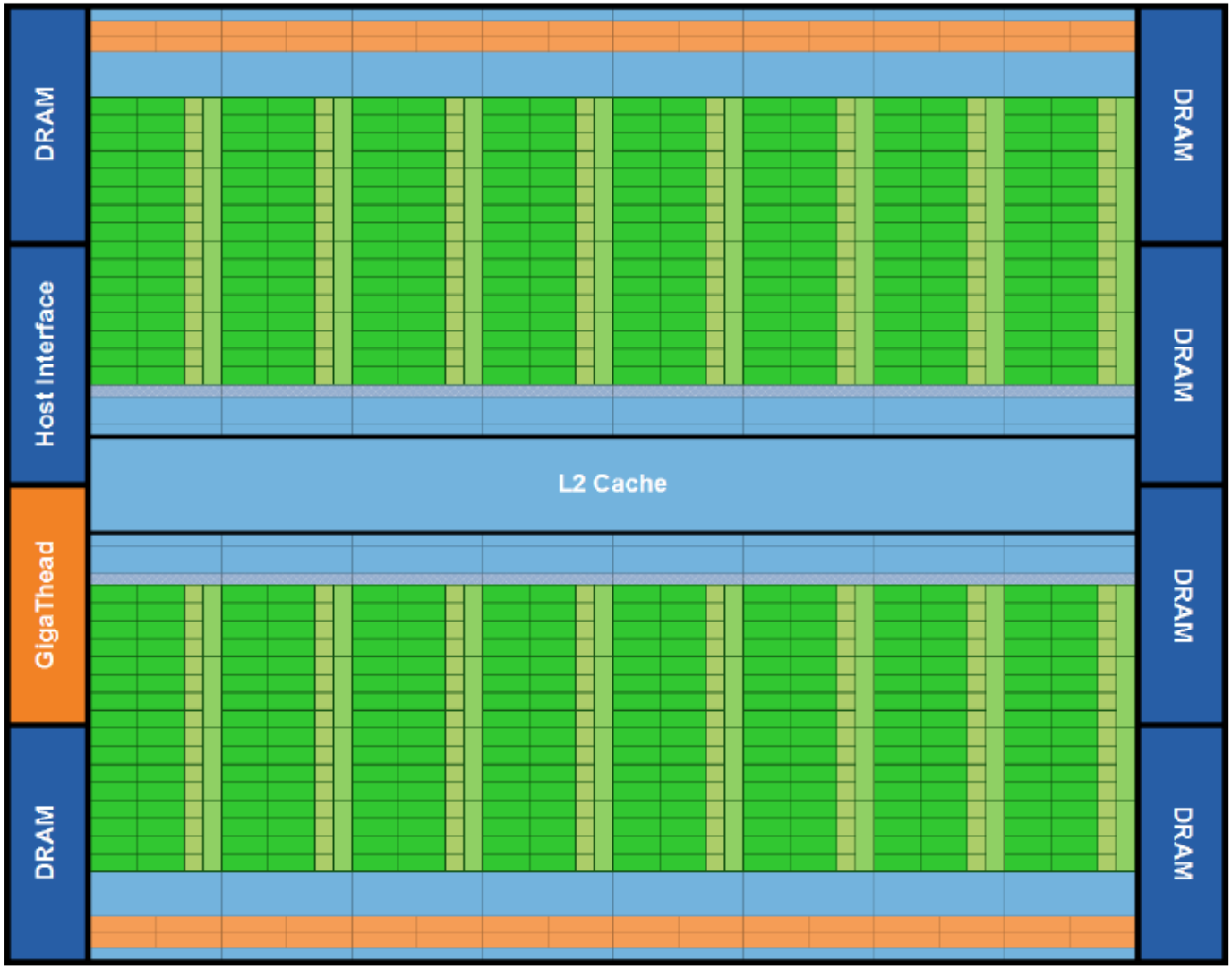
Each thread block contains 4 x 2 threads

Example:

Scheduling 4 thread blocks on 3 SMs.



Another NVIDIA GPU: FERMI



32 cores/SM

~3B Transistors

Another NVIDIA GPU: Kepler

~7.1B transistors
192 cores per SMX

Nvidia Chip GK110 Based on Kepler Architecture

- 7.1 billion transistors
- More then 1 TFlop of double precision throughput
  - 3x performance per watt of Fermi
- New capabilities:
  - Dynamic parallelism
  - Hyper-Q (several cores using the same GPU)
  - Nvidia GPUDirect

Another NVIDIA GPU: Maxwell

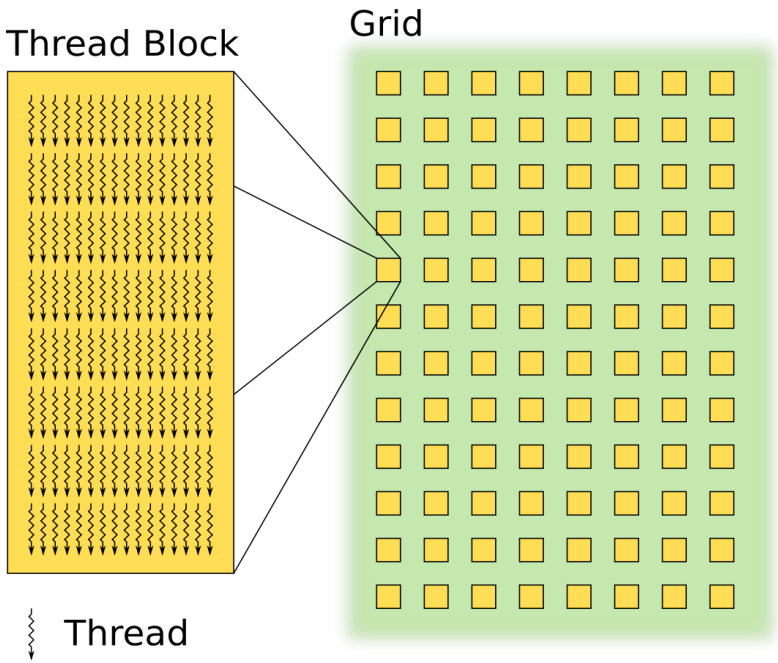
~8B transistors
128 cores per SMM
(Nvidia claims a **128** CUDA core SMM has **90**% of the performance of a **192** CUDA core SMX.)

Main Goals of Newer GPUs

- Increasing floating-point throughput
- Allowing software developers to focus on algorithm design rather than the details of how to map the algorithm to the hardware
- Power efficiency

Quick Glimpse At Programming Models

Application → **Kernels** → **Threads** → **Blocks**



↓

Grid

Quick Glimpse At Programming Models

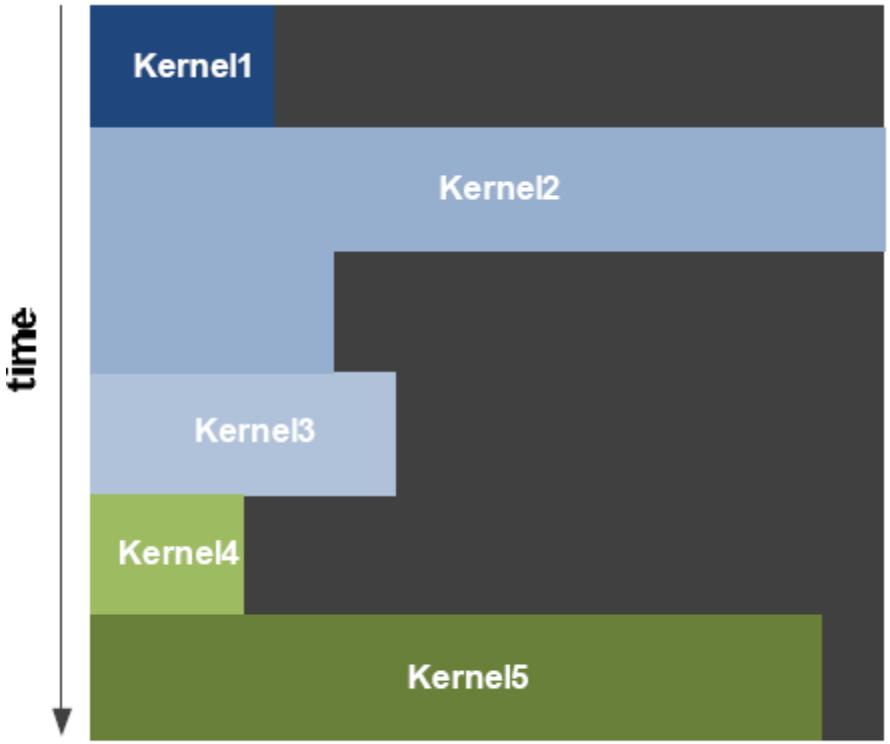
- Application can include multiple kernels
- Threads of the same block run on the same SM
  - So threads in SM can operate and share memory
  - Block in an SM is divided into warps of 32 threads each
  - A warp is the fundamental unit of dispatch in an SM
- Blocks in a grid can coordinate using global shared memory
- Each grid executes a kernel

Scheduling In NVIDIA GPUs

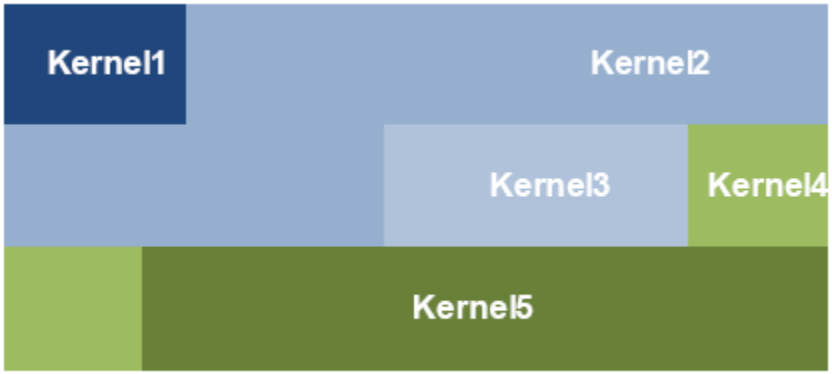
- At any point of time the entire device is dedicated to a single application
  - Switch from an application to another takes ~25 microseconds
- Modern GPUs can simultaneously execute multiple kernels of the same application
- Two warps from different blocks (or even different kernels) can be issued and executed simultaneously
- More advanced GPUs can do more than that but we will concentrate on the above only here.

Scheduling In NVIDIA GPUs

- Two-level, distributed thread scheduler
  - At the chip level: a global work distribution engine schedules thread blocks to various SMs
  - At the SM level, each warp scheduler distributes warps of 32 threads to its execution units.



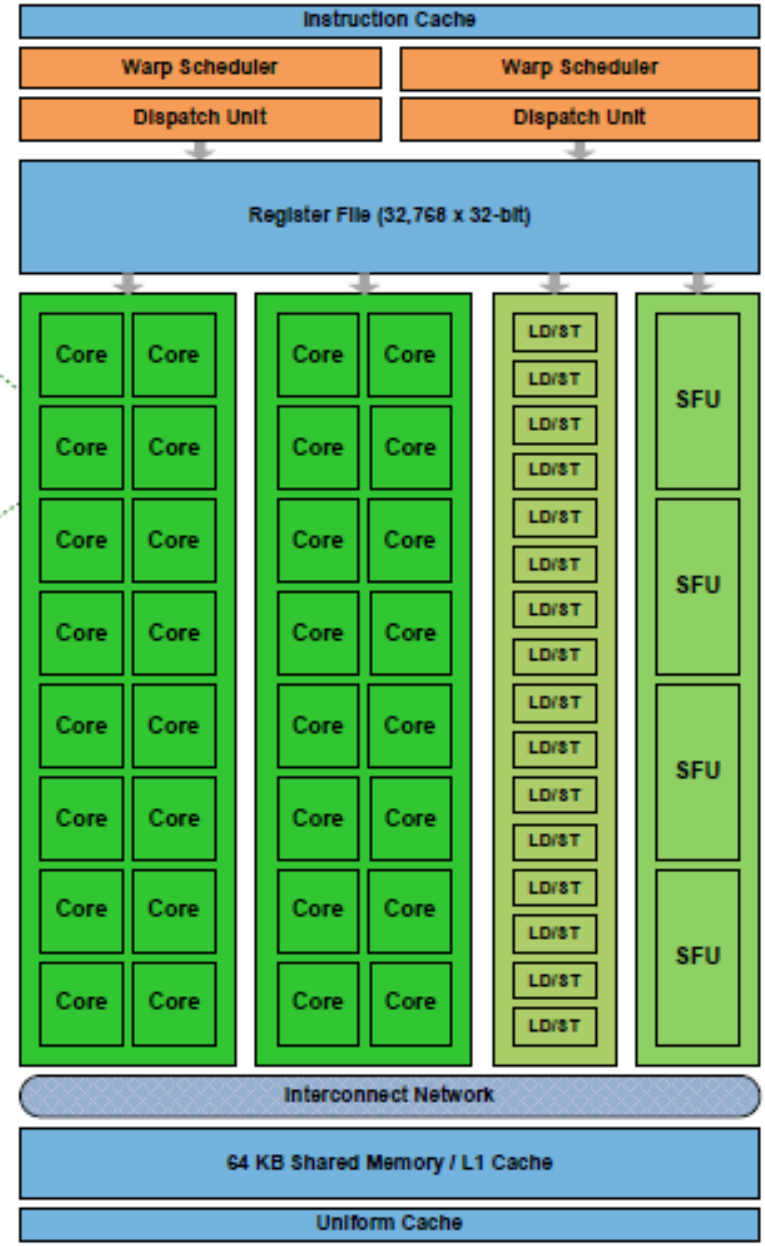
Serial Kernel Execution

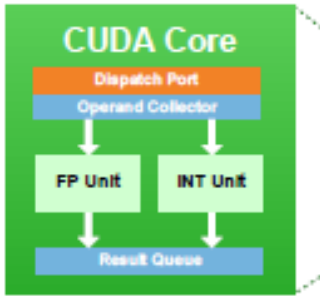


Concurrent Kernel Execution

Source: Nvidia

An SM in Fermi





- 32 cores
- SFU = Special Function Unit
- 64KB of SRAM split between cache and local mem

Each core can perform one single-precision fused multiply-add (FMA) operation in each clock period and one double-precision FMA in two clock periods

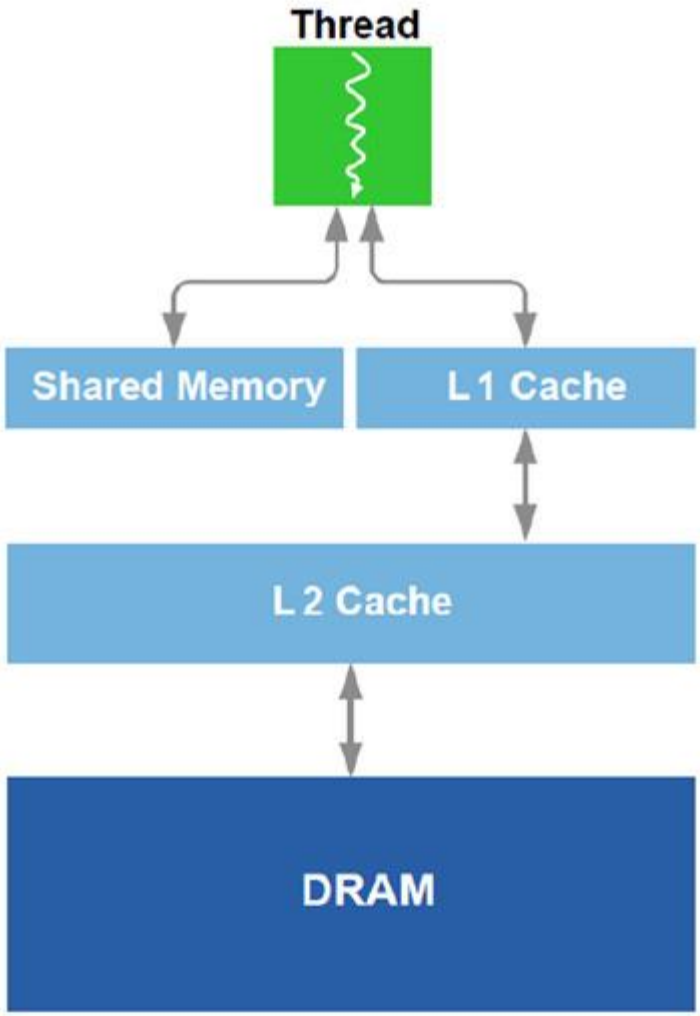
The Memory Hierarchy

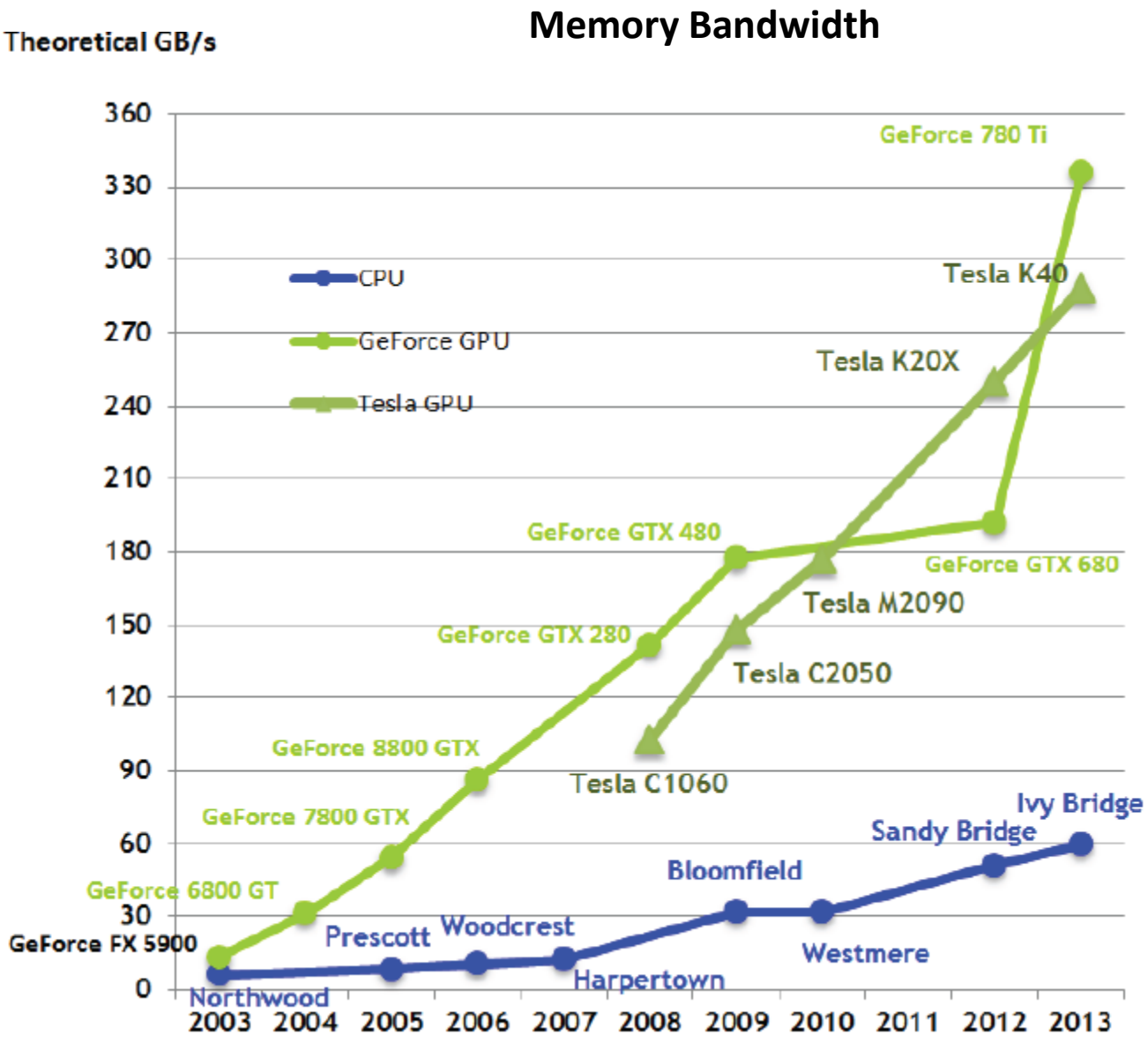
- All addresses in the GPU are allocated from a continuous 40-bit (one terabyte) address space.
- Global, shared, and local addresses are defined as ranges within this address space and can be accessed by common load/store instructions.
- The load/store instructions support 64-bit addresses to allow for future growth.

The Memory Hierarchy

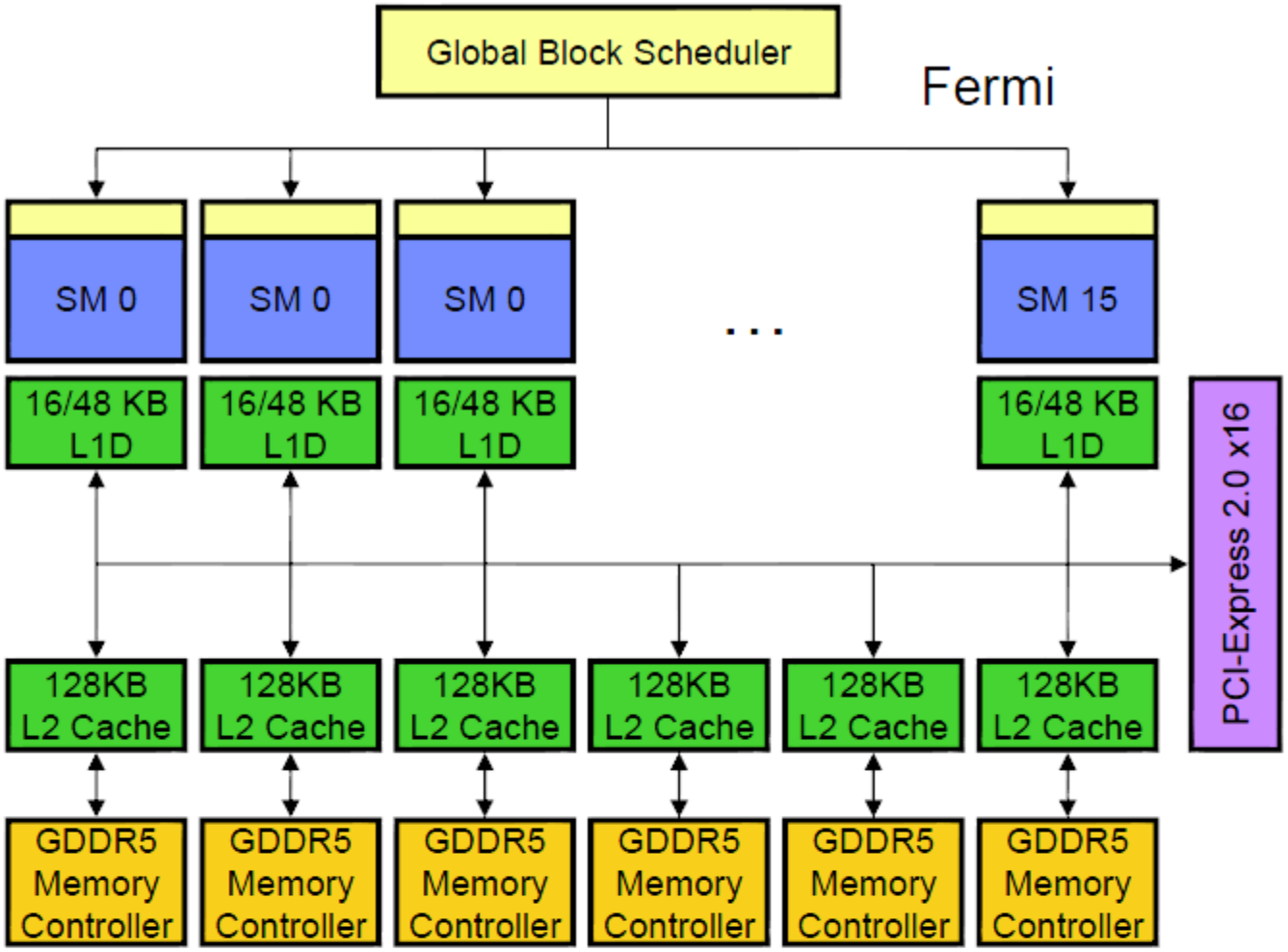
- Local memory in each SM
- The ability to use some of this local memory as a first-level (L1) cache for global memory references.
- Beside L1, each SM has also shared memory.
- Because the access latency to this memory is also completely predictable, algorithms can be written to interleave loads, calculations, and stores with maximum efficiency.
- Modern GPUs are also equipped with an L2 cache, shared among all SMs.

Fermi Memory Hierarchy





Source: NVIDIA CUDA C Programming Guide


Global Block Scheduler
Fermi
SM 0
SM 0
SM 0
SM 15
16/48 KB L1D
16/48 KB L1D
16/48 KB L1D
16/48 KB L1D
128KB L2 Cache
128KB L2 Cache
128KB L2 Cache
128KB L2 Cache
128KB L2 Cache
128KB L2 Cache
GDDR5 Memory Controller
GDDR5 Memory Controller
GDDR5 Memory Controller
GDDR5 Memory Controller
GDDR5 Memory Controller
GDDR5 Memory Controller
PCI-Express 2.0 x16

GPUs Today

- Are more and more general purpose and not only for graphics
- Discrete
  - separate chip on-board like all Nvidia GPUs and AMD GPUs
- Integrated
  - With the CPU on the same chip like the GPU in Intel Sandy Bridge and Ivy Bridge

Memory Bus

- Memory bus
  - Path between GPU itself and the video card memory
  - Bus width and speed of memory → bandwidth (GB/s) → more is better
  - Example:
    - GTX 680: 6GHz memory and 256-bit interface → 192.2 GB/s
    - GTX Titan: 6GHz memory and 384-bit interface → 288.4 GB/s
  - Since most modern GPUs use 6GHz memory, the bus width is the one that makes the difference.

Conclusions

- The main keywords:
  - data parallelism
  - kernel, grid, block, and thread
  - warp
- Some applications are better run on CPU while others on GPU
- Main limitations
  - The parallelizable portion of the code
  - The communication overhead between CPU and GPU
  - Memory bandwidth saturation