High Performance Scientific computing
Lecture 4

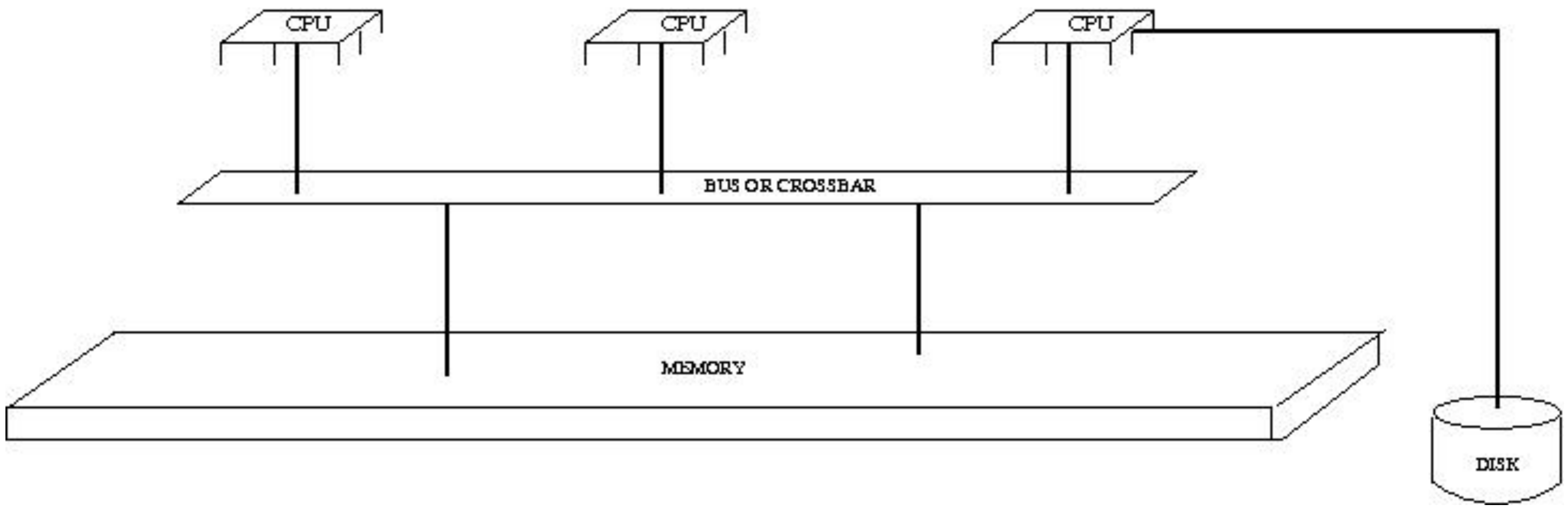
S. Gopalakrishnan

Introduction to Parallel Programming

Shared-Memory Processing

Each processor can access the entire data space

- Pro's
  - Easier to program
  - Amenable to automatic parallelism
  - Can be used to run large memory serial programs
- Con's
  - Expensive
  - Difficult to implement on the hardware level
  - Processor count limited by contention/coherency (currently around 512)
  - Watch out for "NU" part of "NUMA"

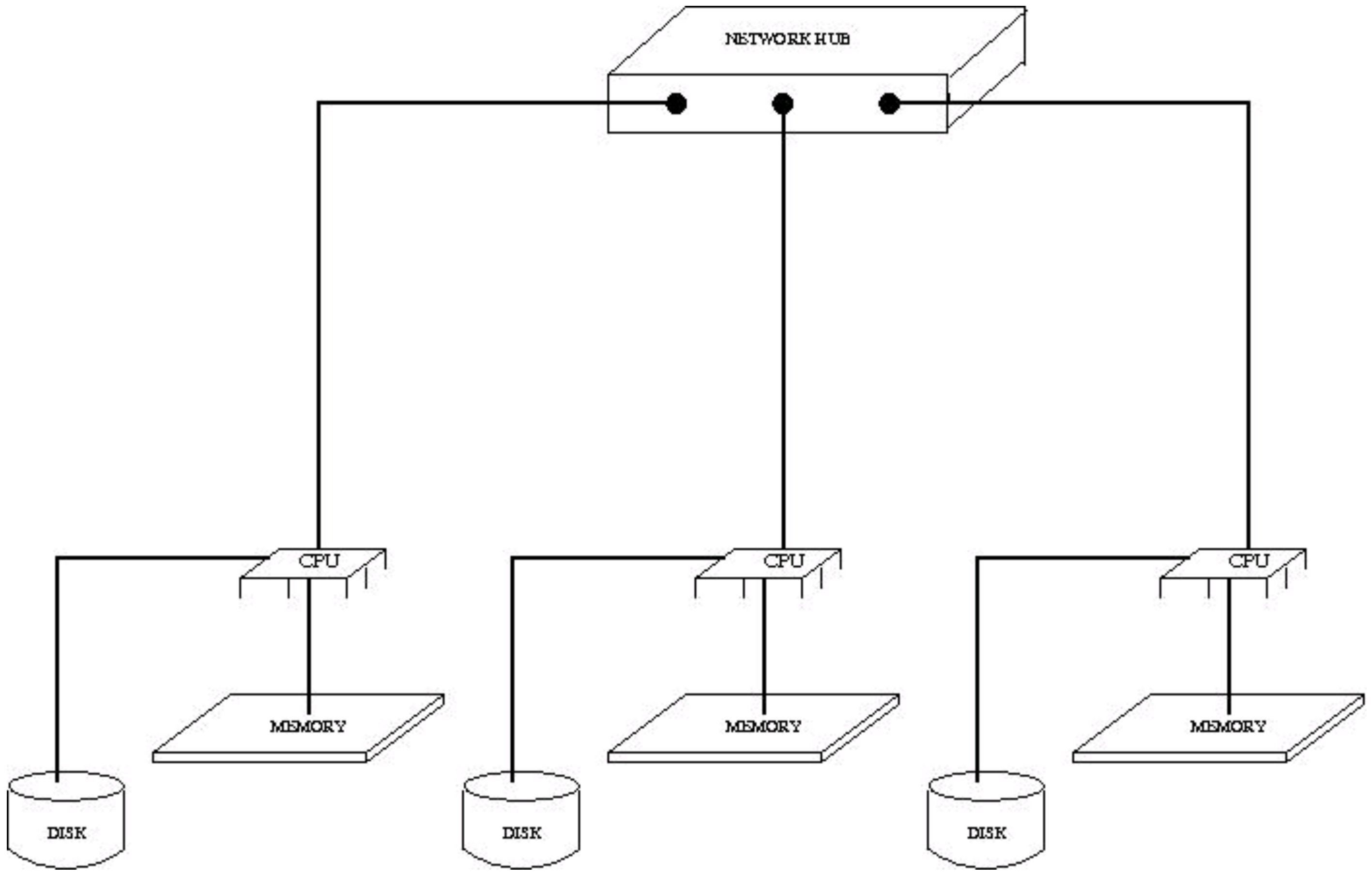


Distributed – Memory Machines

- Each node in the computer has a locally addressable memory space
- The computers are connected together via some high-speed network
  - Infiniband, Myrinet, Giganet, etc..

- Pros
  - Really large machines
  - Size limited only by gross physical considerations:
    - Room size
    - Cable lengths (10's of meters)
    - Power/cooling capacity
    - Money!
  - Cheaper to build and run
- Cons
  - Harder to program

    Data Locality



MPPs (Massively Parallel Processors)

Distributed memory at largest scale. Often shared memory at lower hierarchies.

- IBM BlueGene/L (LLNL)
  - 131,072 700 Mhz processors
  - 256 MB or RAM per processor
  - Balanced compute speed with interconnect

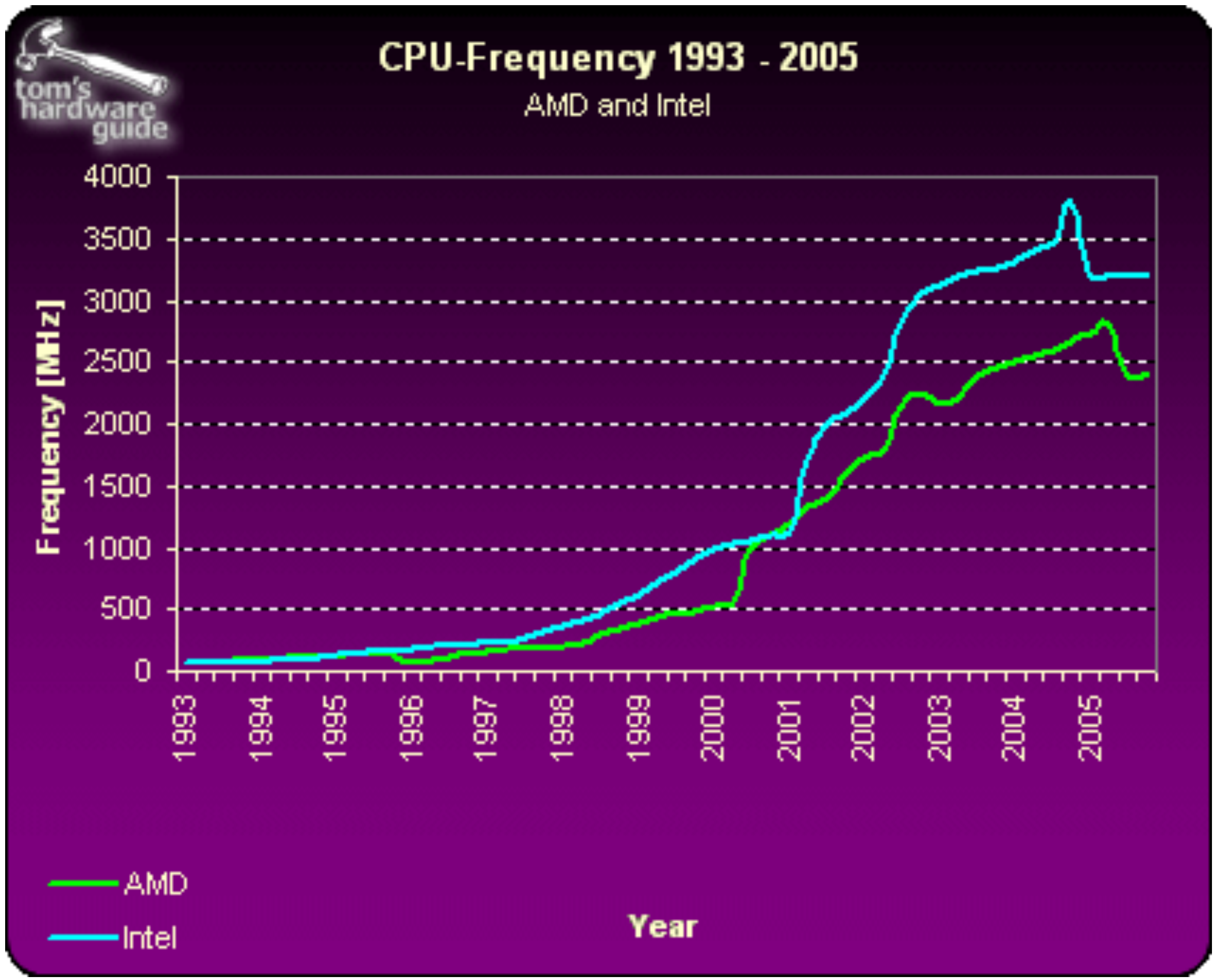
- Red Storm (Sandia National Labs)
  - 12,960 Dual Core 2.4 Ghz Opterons
  - 4 GB of RAM per processor
  - Proprietary SeaStar interconnect

GPGPU Computing
(General Purpose Graphics Processing unit)

GPU Evolution

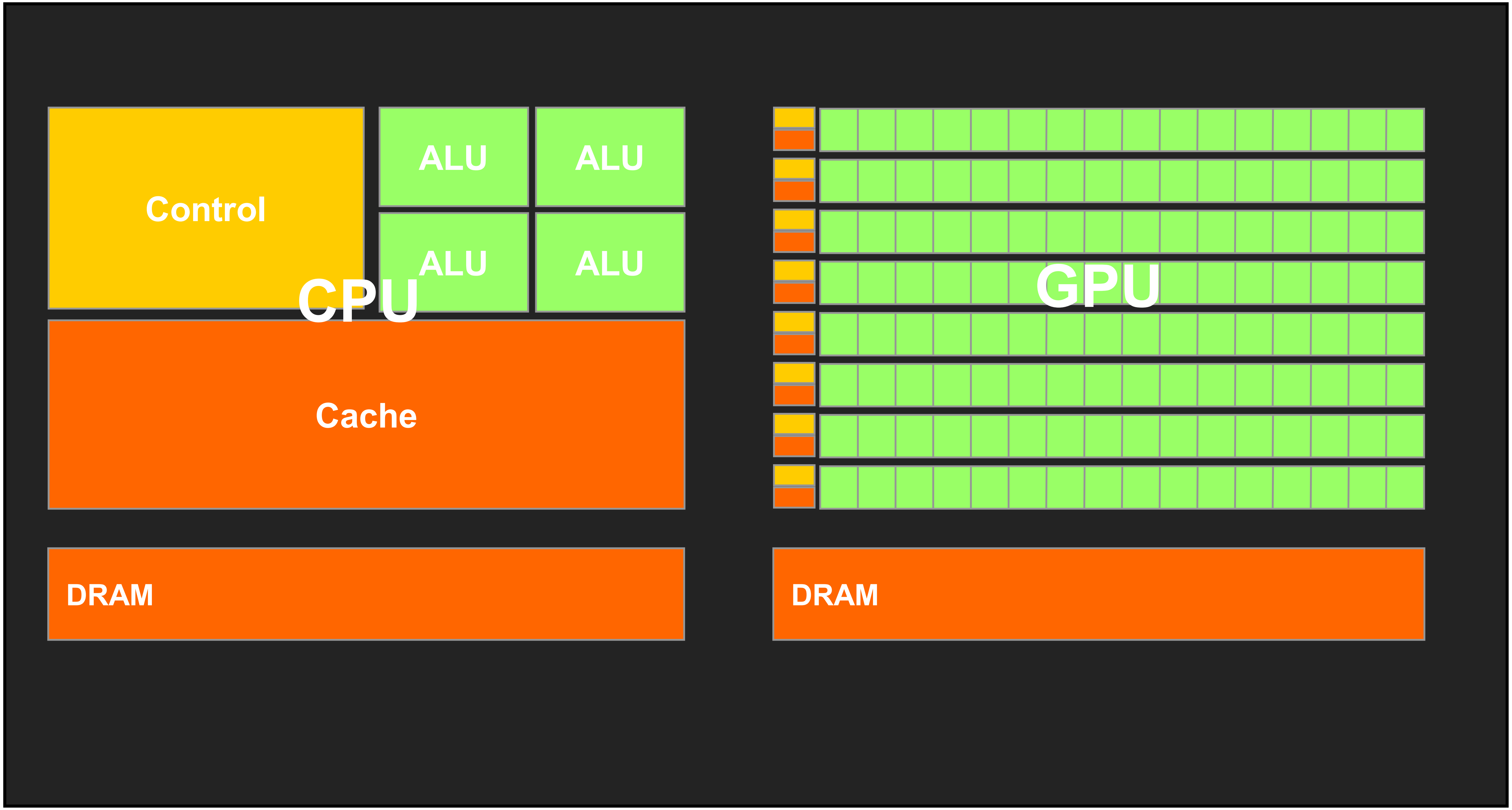
- The **Graphic Processing Unit** (GPU) is a processor that was **specialized** for processing graphics.
- The GPU has recently **evolved** towards a **more flexible** architecture.
- Opportunity: We can implement ***any algorithm*,** not only graphics.
- Works on SIMD (Single Instruction Multiple Data) approach.
- Challenge: obtain **efficiency** and **high**

Peak CPU Performances have reached a bottleneck



Source: www. tomshardware.com

Comparison of CPU vs GPU Architecture



Source: Prof. Wen-mei W. Hwu UIUC

GPU CPU Analogy

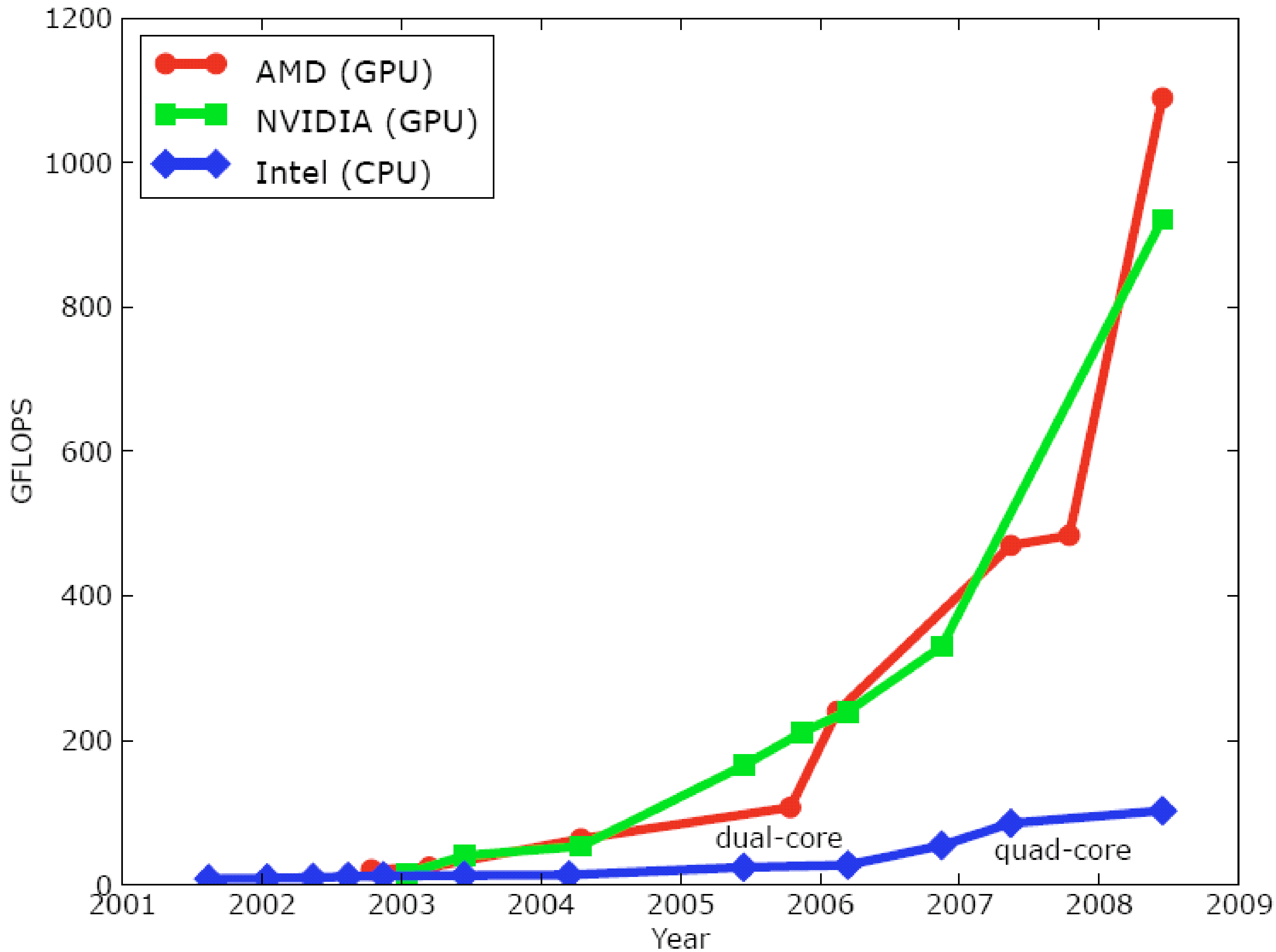
GPU

CPU

It is more effective to deliver Pizza's through light duty scooters rather than big truck. Similarly effective to use several lightweight GPU processors for parallel tasks.

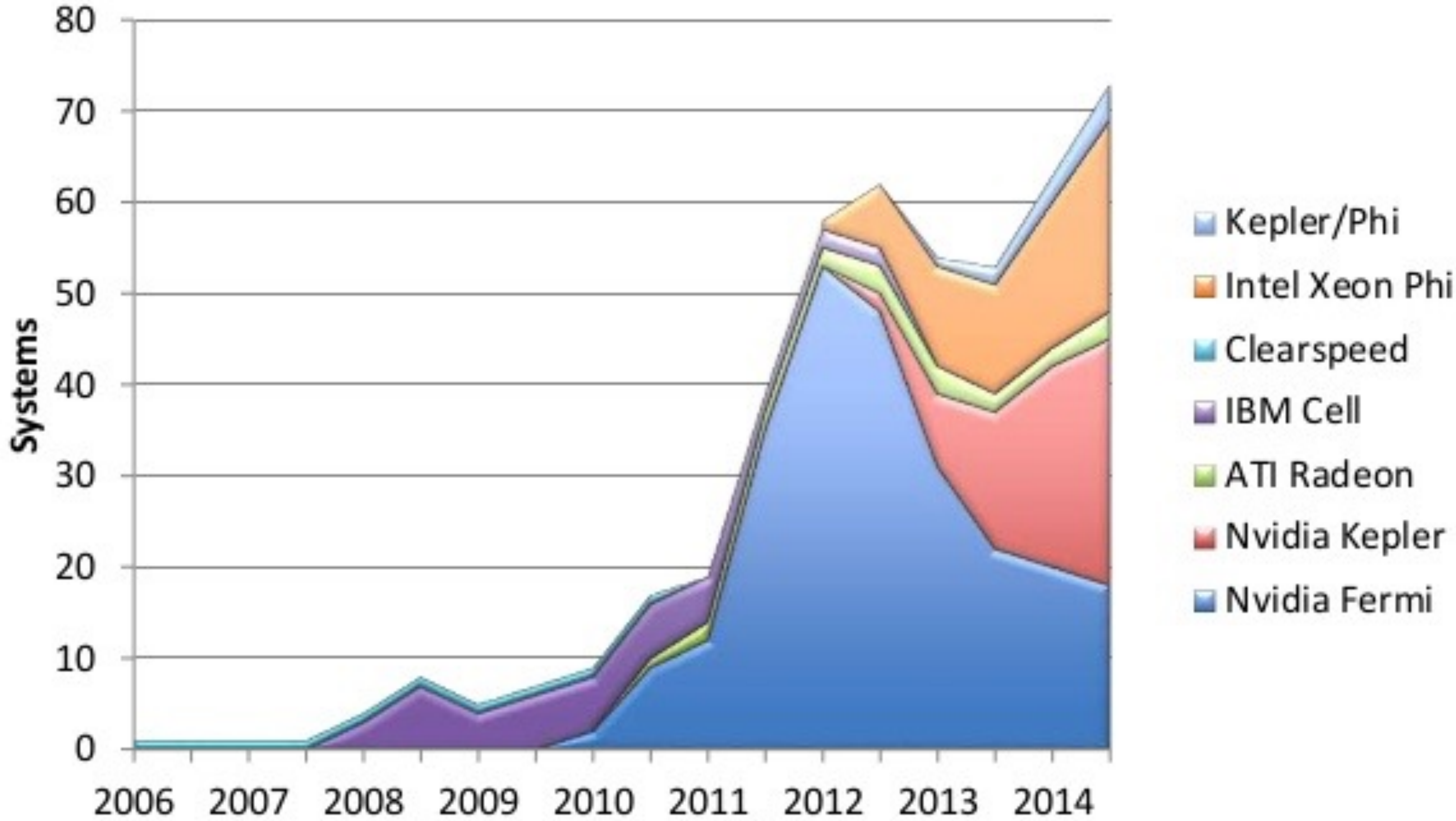
GPU Performance

Peak performance increase
- Calculation ~ 1 TFlop on Desktop
- Memory Bandwidth ~ 150 GB/s



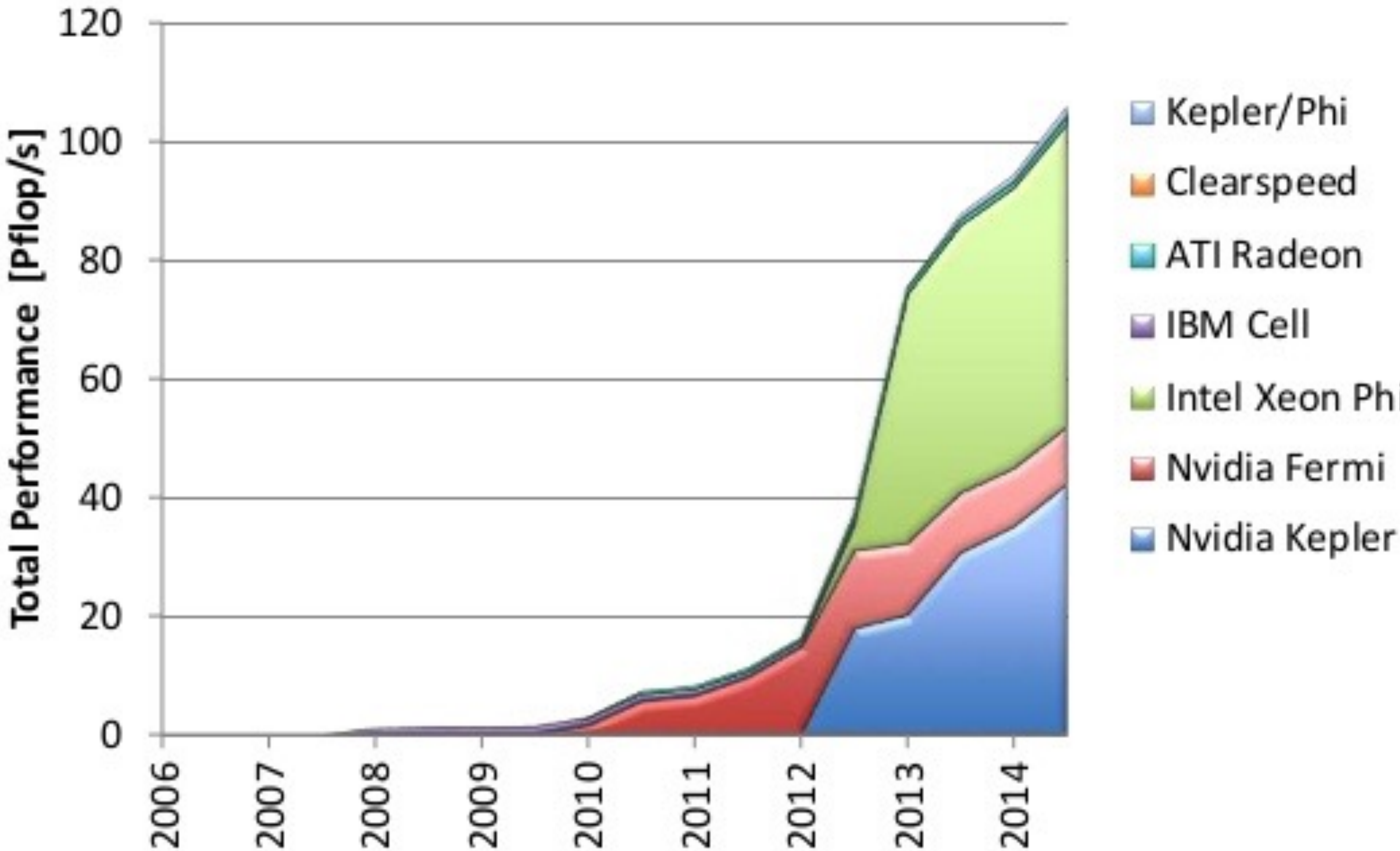
Courtesy: John Owens

ACCELERATORS



source: top500.org

PERFORMANCE OF ACCELERATORS



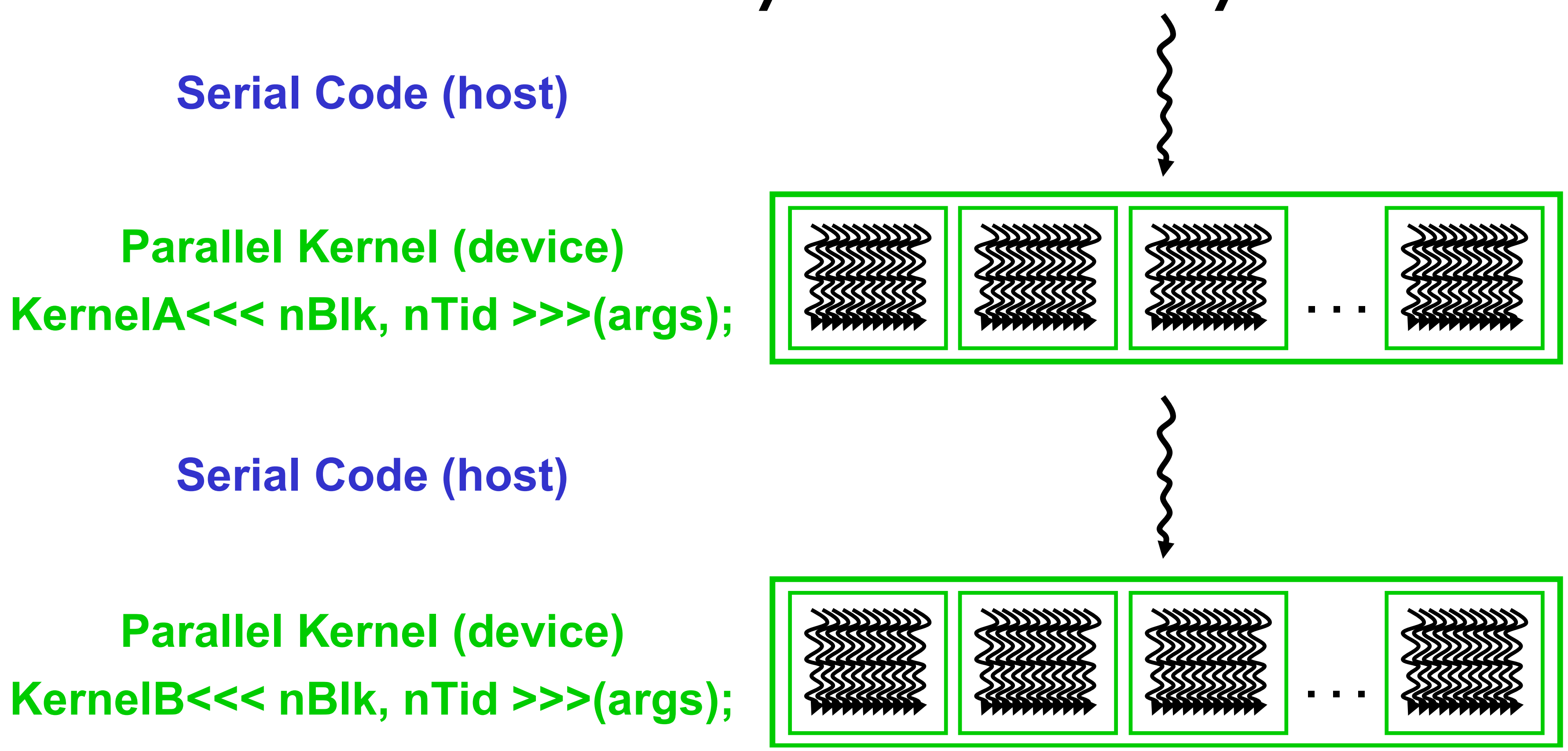
source: top500.org

Compute Unified Device Architecture (CUDA)

- CUDA set of APIs (application program interface) to use GPU’s for general purpose computing
- Developed and released by NVIDIA Inc. Works only on NVIDIA GPU hardware
- Works on commercial GPU’s and as well as specialized ones for scientific computing (Tesla)
- CUDA compiler supports C programming language. Extensions to FORTRAN are possible.
- Opensource alternative is OpenCL.

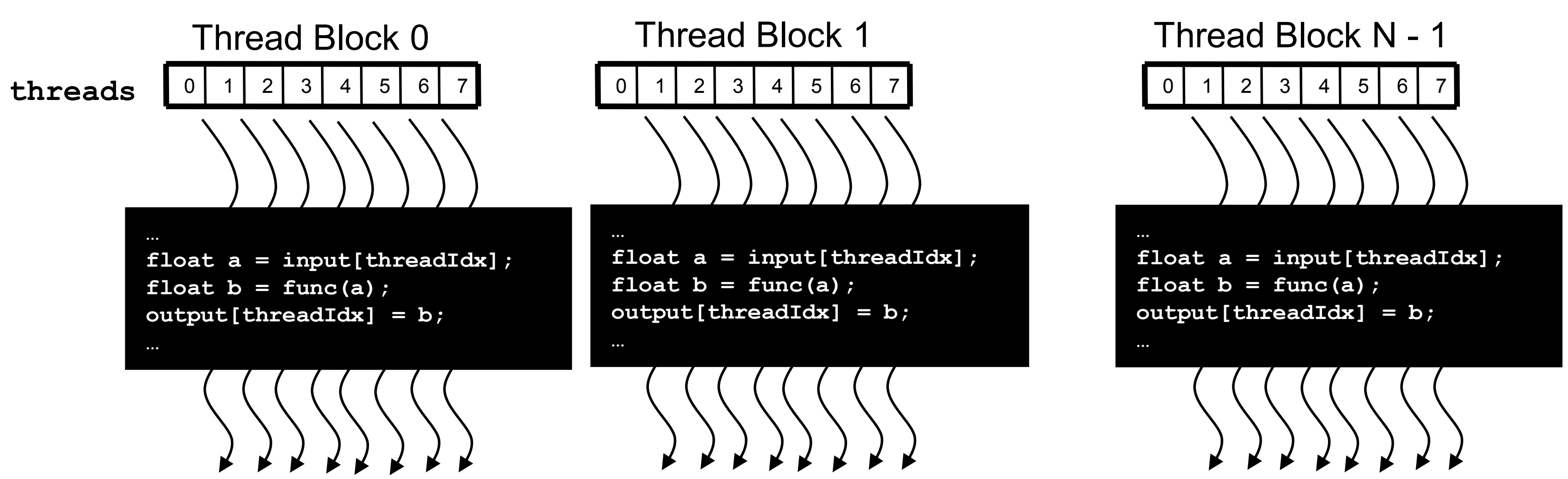
CUDA/ OpenCL Execution Model

- Combination of Host (CPU) and Device(GPU) code
- Parallel portions of the code are expressed as device kernels and they run on many threads



GPU threads vs CPU threads

- GPU threads are very lightweight and hence have less overheads
- For efficiency hundreds of GPU threads are required. Few in case of CPU
- We can subdivide threads into multiple blocks for shared memory cooperation.



NVIDIA Tesla Specificatons

Features	Tesla K20X	Tesla K20	Tesla K10	Tesla M2090	Tesla M2075
Number and Type of GPU	1 Kepler GK110		2 Kepler GK104s	1 Fermi GPU	1 Fermi GPU
GPU Computing Applications	Seismic processing, CFD, CAE, Financial computing, Computational chemistry and Physics, Data analytics, Satellite imaging, Weather modeling		Seismic processing, signal and image processing, video analytics	Seismic processing, CFD, CAE, Financial computing, Computational chemistry and Physics, Data analytics, Satellite imaging, Weather modeling	
Peak double precision floating point performance	1.31 Tflops	1.17 Tflops	190 Gigaflops (95 Gflops per GPU)	665 Gigaflops	515 Gigaflops
Peak single precision floating point performance	3.95 Tflops	3.52 Tflops	4577 Gigaflops (2288 Gflops per GPU)	1331 Gigaflops	1030 Gigaflops
Memory bandwidth (ECC off)	250 GB/sec	208 GB/sec	320 GB/sec (160 GB/sec per GPU)	177 GB/sec	150 GB/sec
Memory size (GDDR5)	6 GB	5 GB	8GB (4 GB per GPU)	6 GigaBytes	6 GigaBytes
CUDA cores	2688	2496	3072 (1536 per GPU)	512	448

Source: NVIDIA Inc.

Hybrid Programming!

- Need to use distributed memory design to reach large scales.
- Leverage commodity processor shared memory at the board level.
- It *is* an efficient way to get a pile of flops.
- But, a little more "interesting" from programming perspective