CS 5234 –Spring 2013 Advanced Parallel Computing

Introduction

Yong Cao

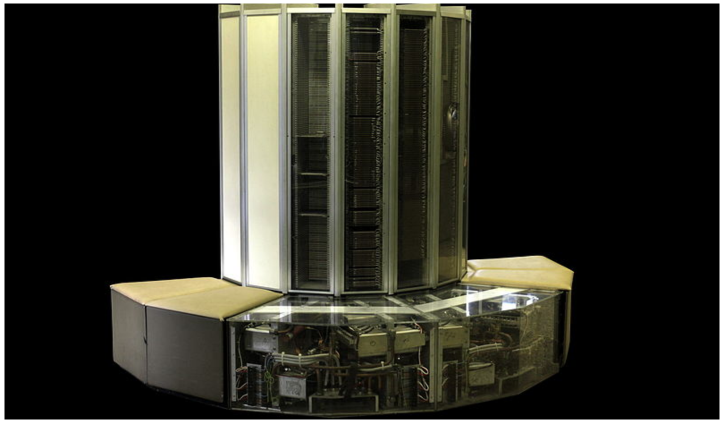
Course Goals

- Understand the massive parallel architecture of Graphics Processing Units (GPUs)
  - Features and Constrains
- Program on GPUs
  - Programing APIs, tools, and techniques
  - Achieve high performance and scalability
- Analyze parallel computing problems
  - Principles and paradigms for parallel algorithm design
  - Ability to apply to real life application and algorithms

Why Parallel Computing?

- Chase of Performance

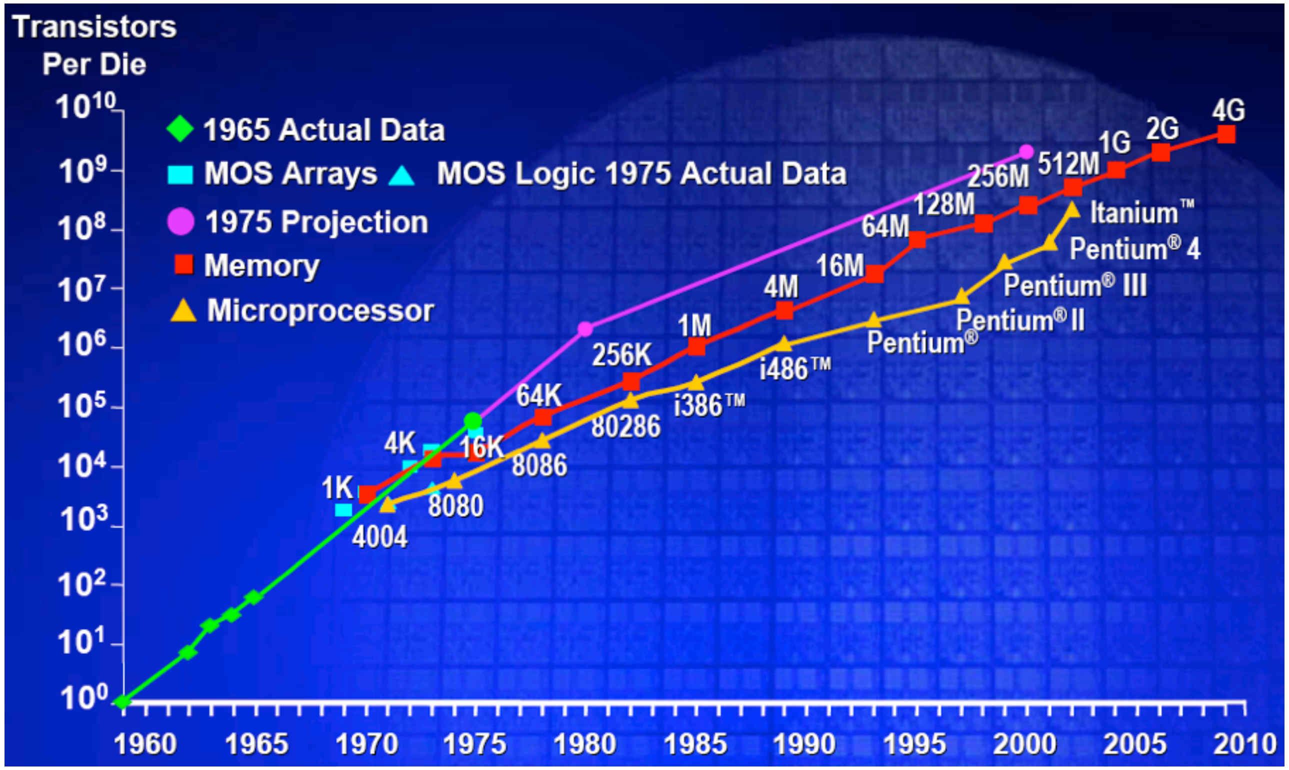
IBM 360/91
5 Units/“Cores”, 2M memory
1968



Cray-1A Supercomputer
“SIMD” Architecture- Vector
1M 72-bits words
1979

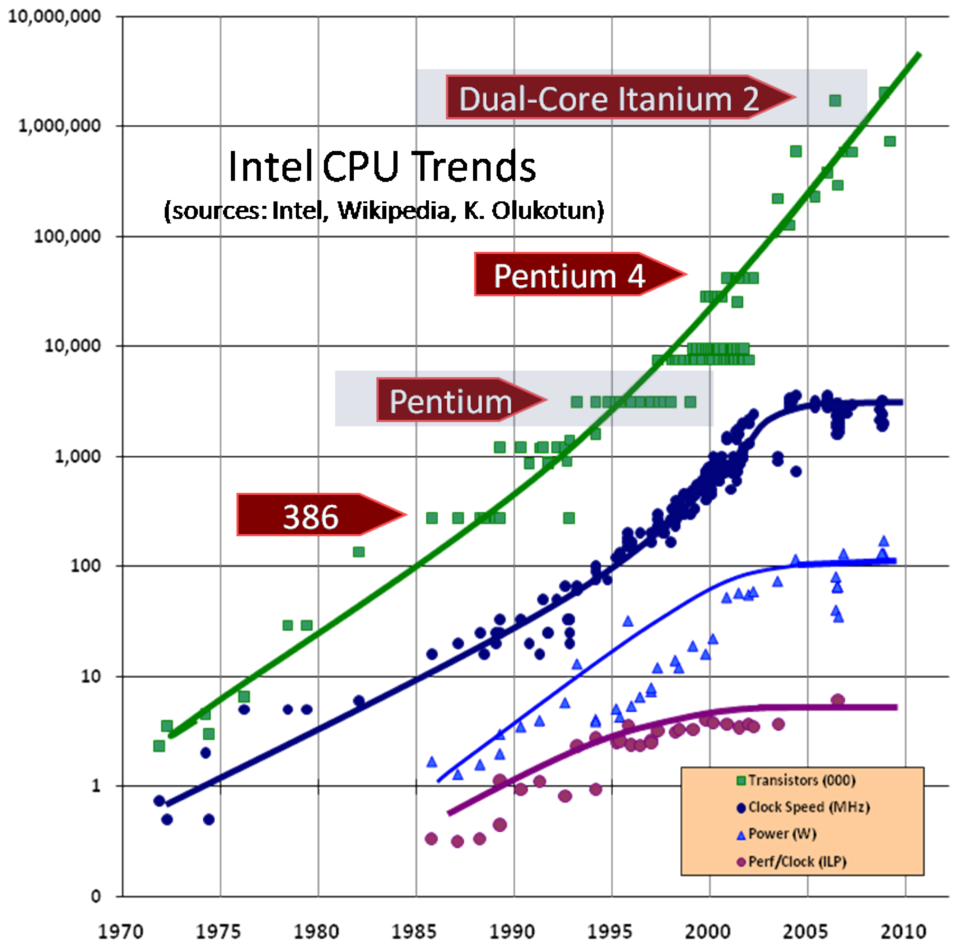
Microprocessors

- Semiconductors: "Moore's Law": a "Free Lunch"
  - The number of transistors/inch^2 in these circuits roughly doubled every 18 month



End of "Free Lunch"

- "The size of transistors … approaching the size of atoms …"
  --- Gordon Moore, April 13, 2005.
- Problem: Quantum tunneling

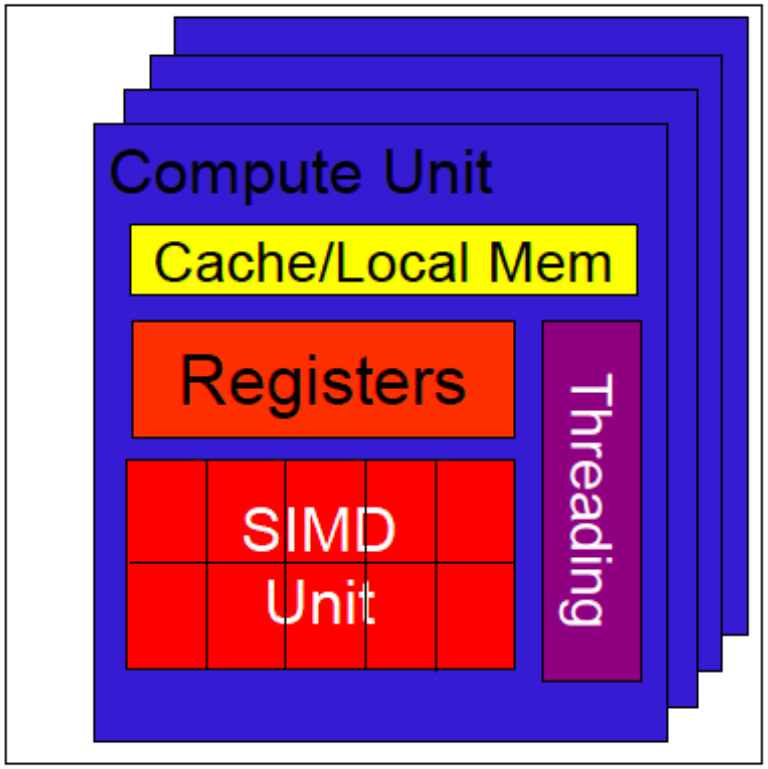


Recent Parallel Processors

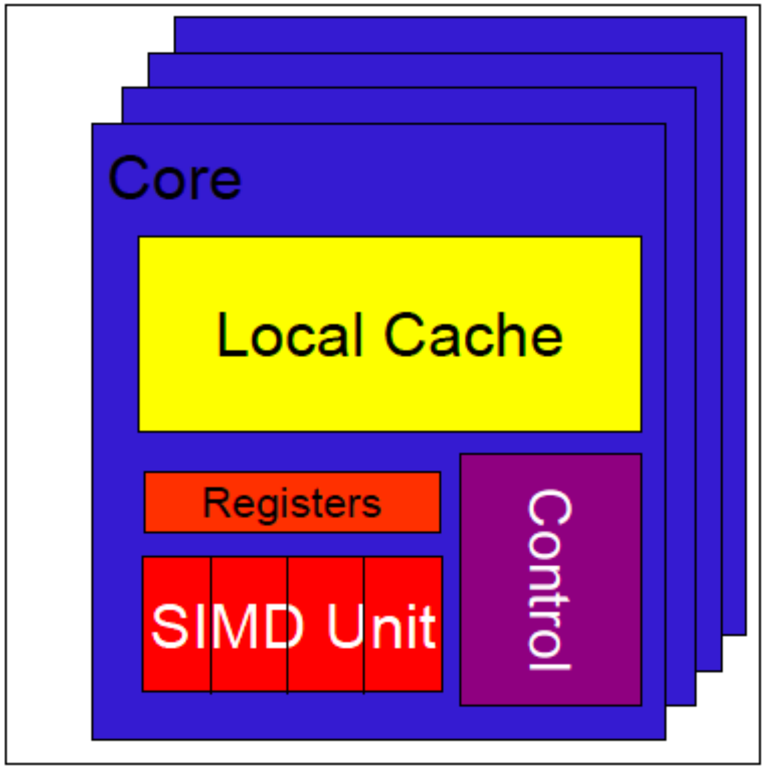
- General CPU
  - Blue Gene/Q: 17 Cores, 4-way SMT
  - AMD Interlagos: 8 FP cores, 16 Integer cores
  - Intel Xeon E7: 10 cores, 2-way SMT
  - Sparc T4: 8 cores, 8-way fine-grain MT per core
- Accelerators
  - Intel Xeon Phi: 60 cores
  - NVIDIA Kepler K20X: 2688 cores, 3.95 Tflops, 7.1B transistors!!!
- CPU + GPU Hybrid
  - AMD Trinity: 4 CPU cores + 384 GPU cores
  - Intel Ivy Bridge: 4 CPU cores + 6-16 GPU units

CPU vs GPUs

GPUs
Throughput Oriented

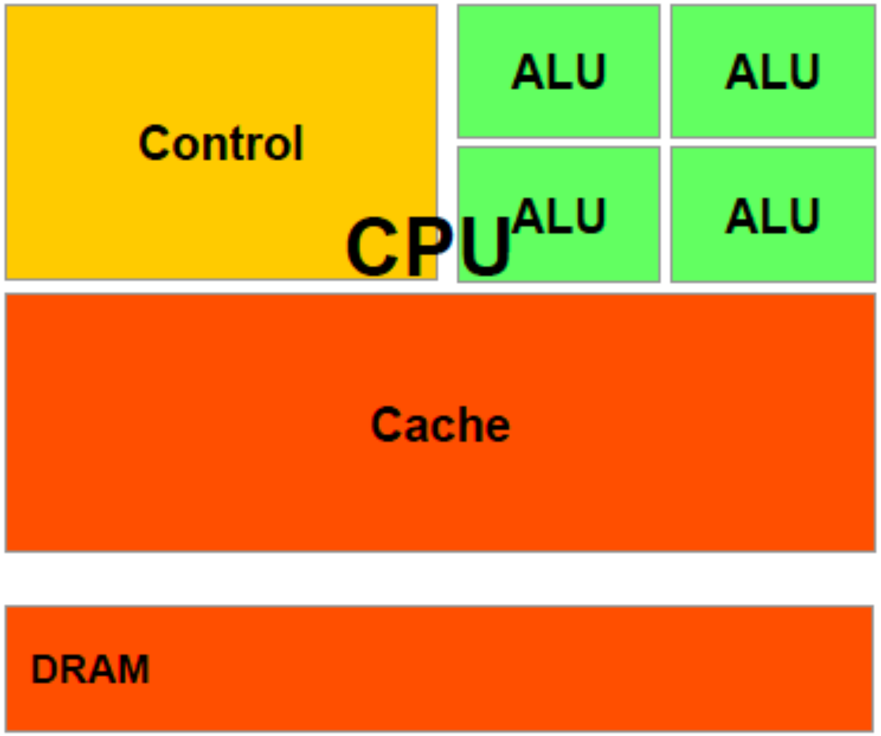


CPUs
Latency Oriented



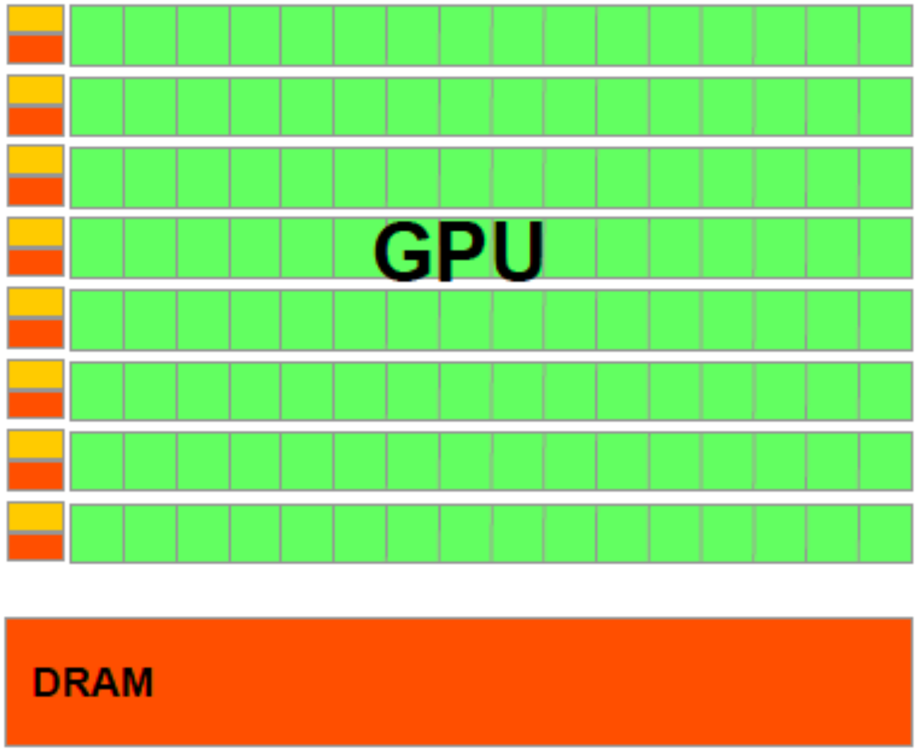
CPU: Latency Oriented Cores

- Large caches
  - Convert long latency memory accesses to short latency cache accesses
- Sophisticated control
  - Branch prediction for reduced branch latency
  - Data forwarding for reduced data latency
- Powerful ALU
  - Reduced operation latency



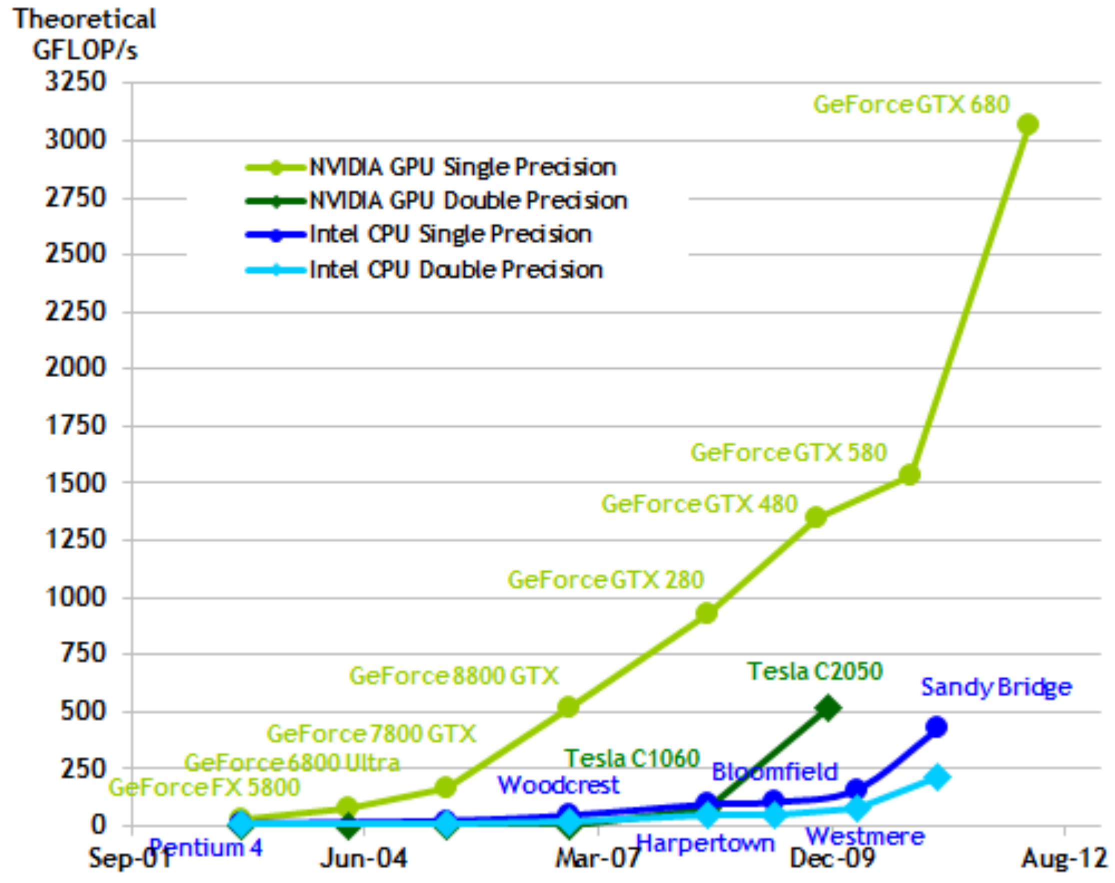
GPU: Throughput Oriented Cores

- Small caches
  - To boost memory throughput
- Simple control
  - No branch prediction
  - No data forwarding
- Energy efficient ALUs
  - Many, long latency but heavily pipelined for high throughput
- Require massive number of threads to tolerate latencies



Why GPUs?

➢ It's powerful!



Why GPUs?

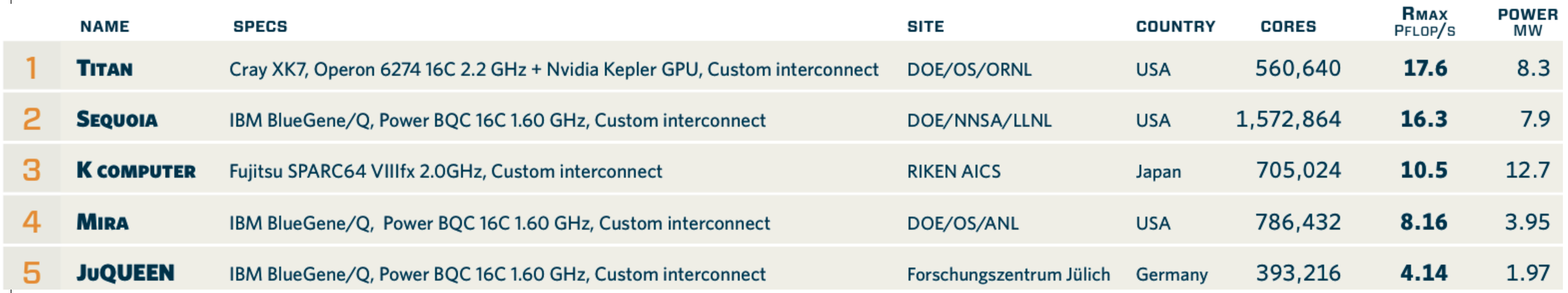
- It's powerful!
  - NVIDIA K20X
    - 2688 Cores; About 16 TFLOPs (More than the top 1 super computing 11 years ago)

4 NVIDIA K20X GPUs
16 Tera FLOPs
About 1,500 Watts
Cost: $13,000

IBM ASCI White at 2000
512 Nodes
8192 Processors
7.266 Tera FLOPs
106 tons
3 Million Watts
Cost: $110 Millions

Why GPUs?

- It's cheap and everywhere.
  - E.g. NIVIDA sold more than 200 Million high-end GPGPU devices.
  - A 1536-core Geforce GTX 680 is $550 on Newegg.com.
  - Supercomputer, Desktop, Laptop, Mobile devices



	NAME	SPECS	SITE	COUNTRY	CORES	RMAX PFLOP/S	POWER MW
1	TITAN	Cray XK7, Operon 6274 16C 2.2 GHz + Nvidia Kepler GPU, Custom interconnect	DOE/OS/ORNL	USA	560,640	**17.6**	8.3
2	SEQUOIA	IBM BlueGene/Q, Power BQC 16C 1.60 GHz, Custom interconnect	DOE/NNSA/LLNL	USA	1,572,864	**16.3**	7.9
3	K COMPUTER	Fujitsu SPARC64 VIIIfx 2.0GHz, Custom interconnect	RIKEN AICS	Japan	705,024	**10.5**	12.7
4	MIRA	IBM BlueGene/Q, Power BQC 16C 1.60 GHz, Custom interconnect	DOE/OS/ANL	USA	786,432	**8.16**	3.95
5	JUQUEEN	IBM BlueGene/Q, Power BQC 16C 1.60 GHz, Custom interconnect	Forschungszentrum Jülich	Germany	393,216	**4.14**	1.97

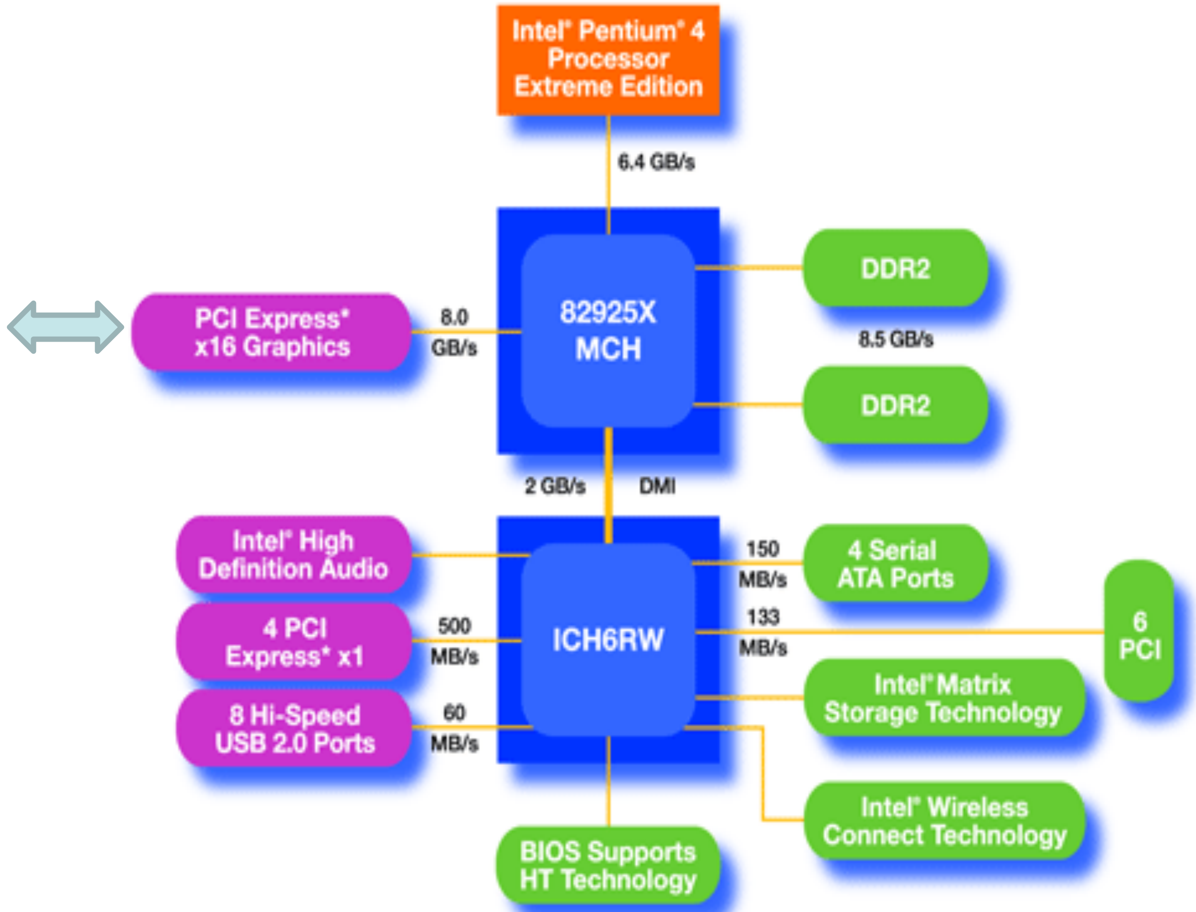
Why GPU NOW?

- Before:
  - 5-6 years ago, everyone used Graphics API (Cg, GLSL, HLSL) for GPGPU programming.
  - Restrict random-read (using Texture), NOT be able to random-write. (No pointer!)

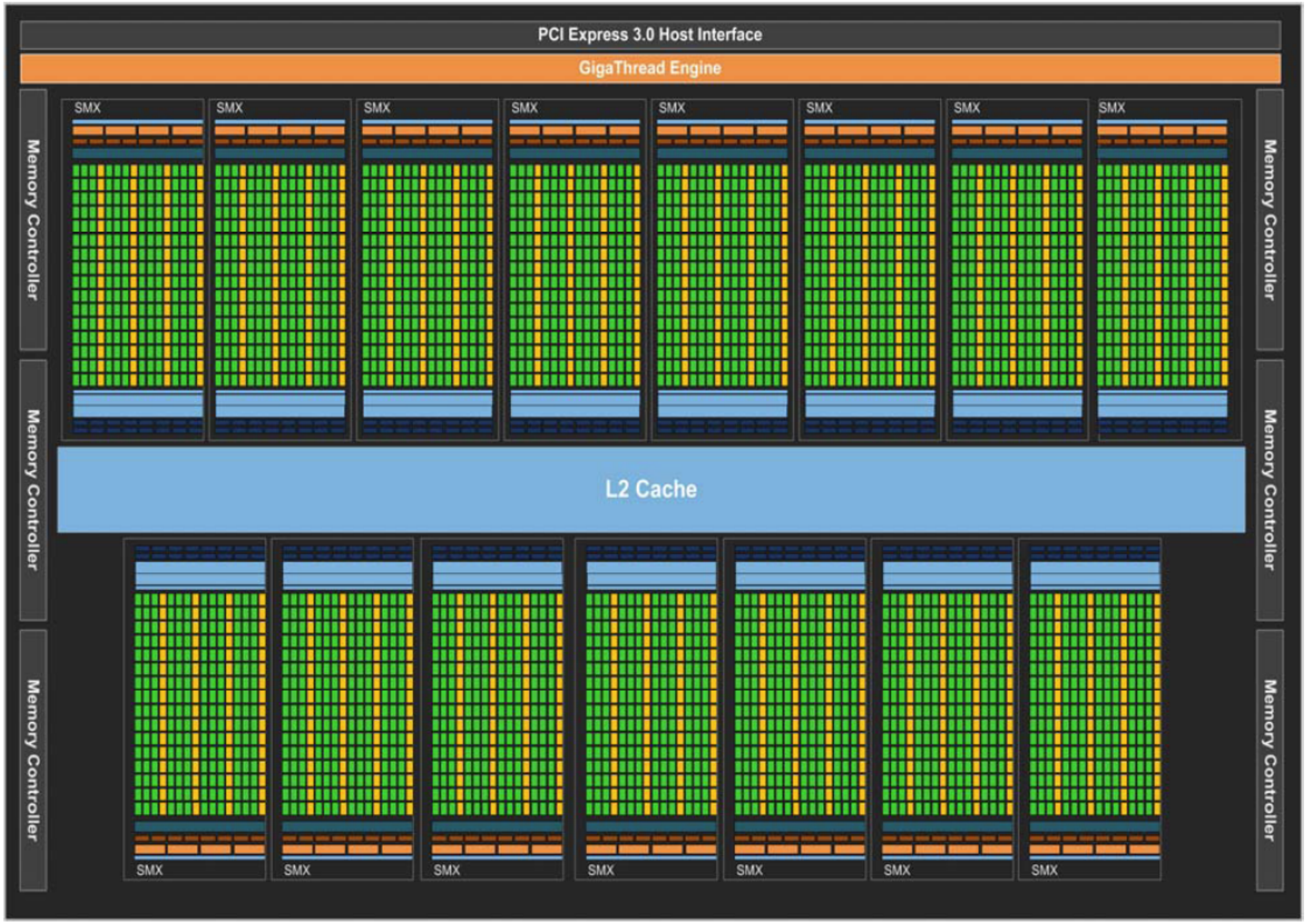
Why GPU NOW?

- Now:
  - NIVIDA released CUDA 6.5 years ago, since then
    - Hundreds of Thousands of CUDA software engineers
    - New job title "CUDA programmer"
    - 2150 publications with "CUDA" in their title since 2006. (Google Scholar today)
    - 8560 publications with "GPU" in their title since 2006.
  - Why?
    - Standard C language
    - Support Pointer! Random read and write on GPU memory.
    - Work with C++, Fortran

Where's GPU in the system

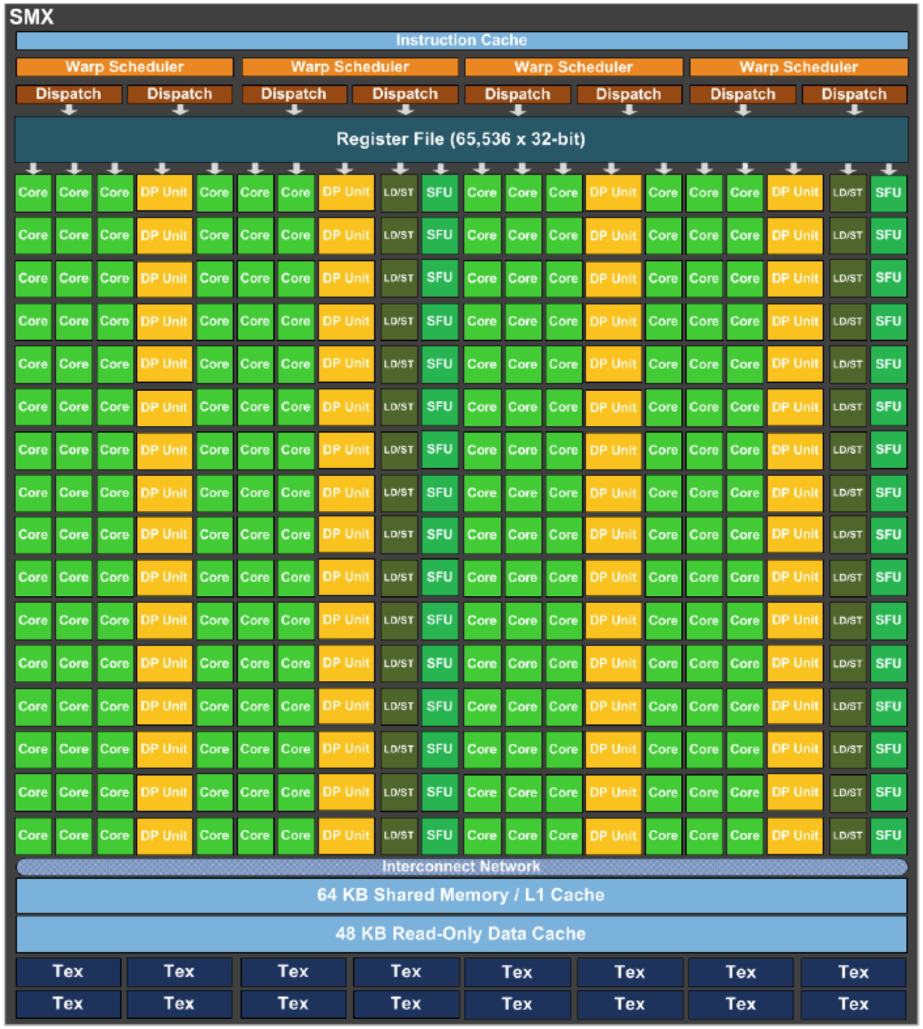

Intel® Pentium® 4 Processor Extreme Edition
6.4 GB/s
82925X MCH
DDR2
8.5 GB/s
DDR2
PCI Express* x16 Graphics
8.0 GB/s
2 GB/s
DMI
ICH6RW
Intel® High Definition Audio
4 PCI Express* x1
500 MB/s
8 Hi-Speed USB 2.0 Ports
60 MB/s
150 MB/s
4 Serial ATA Ports
133 MB/s
6 PCI
Intel® Matrix Storage Technology
Intel® Wireless Connect Technology
BIOS Supports HT Technology

NVIDIA GK110 (Kepler) Architecture



Stream Multi-Processor (SMX)

- 192 SP cores
- 32 SFUs
- 32 L/S units
- 4 Warp Scheduler
- 8 Instruction Dispatch units
- 2 instructions per warp


SMX
Instruction Cache
Warp Scheduler
Dispatch
Register File (65,536 x 32-bit)
Core
DP Unit
LD/ST
SFU
Interconnect Network
64 KB Shared Memory / L1 Cache
48 KB Read-Only Data Cache
Tex

About me

- Prof. Yong Cao
  - Office hour: By appointment at KWII 1127
  - Email: yongcao@vt.edu (Please use CS5234 in your subject line)
  - Phone: 540-231-0415
  - Website: www.cs.vt.edu/~yongcao

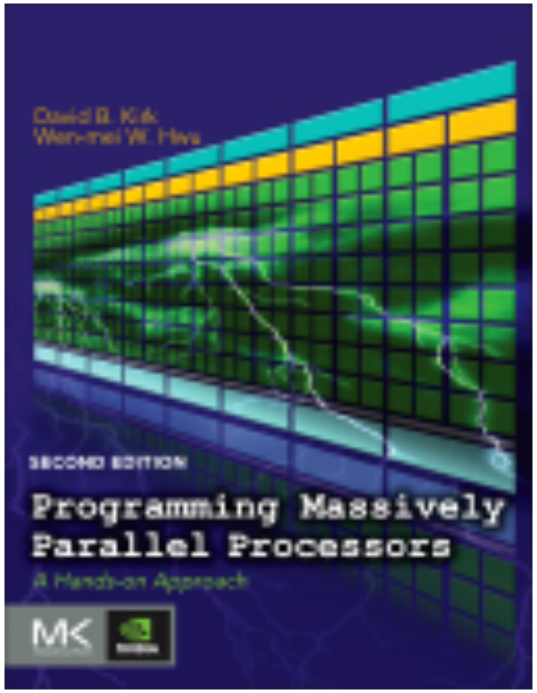
Course Website

- http://people.cs.vt.edu/~yongcao/teaching/cs5244/spring2013/index.html
- Or go to my website, and click on the course link.

- Five sections:
  - Home page
  - Syllabus/Schedule
  - Notes
  - Projects
  - Resources

Course Materials

- Textbooks:
  - Programming Massively Parallel Processors, Morgan Kaufmann, 2nd Edition. David Kirk and Wen-mei Hwu.



Course Materials

- Other Web Resources:
  - NVIDIA CUDA Programming Guide. NVIDA CUDA website, http://www.nvidia.com/object/cuda_home.html
  - UIUC Parallel Programming Course Website: http://courses.engr.illinois.edu/ece408/

Course Work (Tentative)

- Programming Assignments 60%
  - Assignment 1: Image convolution.
  - Assignment 2: Min, max, median
  - Assignment 3: Association rule mining
  - Assignment 4: Graph/tree traversal
  - Assignment 5: OpenGL interoperation
- Project Presentation & Report 40%
  - Problem statement and test data will be provided.
  - Oral presentation and final written report.

Academic Honesty

- You are allowed and encouraged to discuss assignments with other students in the class. Getting verbal advice/help from people who've already taken the course is also fine.
- Any reference to assignments from previous terms or web postings is unacceptable
- Any copying of non-trivial code is unacceptable
  - Non-trivial = more than a line or so
  - Includes reading someone else's code and then going off to write your own.

Late Assignment Policy

- Assignments will be downgraded 25% for each day late. No exception permitted.

Final Project

- Two-person team only!
- Presentation are required.
- Final report is required. (4-6 pages)
  - Please see class website for the detail.

Reading Material

- NVIDIA CUDA Programming Guide, Chapter One http://www.nvidia.com/object/cuda_develop.html and looking for documentation