CSCI-GA.3033-004

Graphics Processing Units (GPUs): Architecture and Programming

Lecture 1: Gentle Introduction to GPUs

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

- Mohamed Zahran (aka Z)
- Computer architecture/OS/Compilers Interaction
- [http://www.mzahran.com](http://www.mzahran.com)
- Office hours: Tue 4:00-6:00 pm
  - Or by appointment
- Room: WWH 320
Formal Goals of This Course

• Why GPUs
• GPU Architecture
• GPU-CPU Interaction
• GPU programming model
• Solving real-life problems using GPUs
Informal Goals of This Course

• To get more than an A
• To learn GPUs and enjoy it
• To use what you have learned in MANY different contexts
• To have a feeling about how hardware and software evolve and interact
The Course Web Page

• Lecture slides
• Reading assignments
• Info about mailing list, labs, and homework assignments.
• Useful links (manuals, tools, book errata, ... )
Programming Massively Parallel Processors: A Hands-on Approach

By

David B. Kirk & Wen-mei W. Hwu

2nd Edition
Grading

• Homework assignments : 20%
• Project : 20%
• Programming assignments : 20%
• Final : 40%
Computer History

Eckert and Mauchly

- 1st working electronic computer (1946)
- 18,000 Vacuum tubes
- 1,800 instructions/sec
- 3,000 ft$^3$
Computer History

- Maurice Wilkes

EDSAC 1 (1949)

1st stored program computer
650 instructions/sec
1,400 ft³

http://www.cl.cam.ac.uk/UoCCL/misc/EDSAC99/
Intel 4004 Die Photo

- Introduced in 1970
  - First microprocessor
- 2,250 transistors
- 12 mm²
- 108 KHz
Intel 8086 Die Scan

- 29,000 transistors
- 33 mm²
- 5 MHz
- Introduced in 1979
  - Basic architecture of the IA32 PC
Intel 80486 Die Scan

- 1,200,000 transistors
- 81 mm²
- 25 MHz
- Introduced in 1989
  - 1st pipelined implementation of IA32
Pentium Die Photo

- 3,100,000 transistors
- 296 mm$^2$
- 60 MHz
- Introduced in 1993
  - 1$^{st}$ superscalar implementation of IA32
Pentium III

- 9,500,000 transistors
- 125 mm²
- 450 MHz
- Introduced in 1999

Pentium 4

- 55,000,000 transistors
- 146 mm²
- 3 GHz
- Introduced in 2000

http://www.chip-architect.com
Pentium 4

Core 2 Duo (Merom)

Intel Core i7 (Nehalem)

IBM Power 7

Montecito

Cell Processor

Niagara
(SUN UltraSparc T2)
The Famous Moore’s Law
Hardware Improvement

Positive Cycle of Computer Industry

Better Software

People ask for more improvements

People get used to the software
The Status-Quo

• We moved from single core to multicore
  – for technological reasons
• Free lunch is over for software folks
  – The software will not become faster with every new generation of processors
• Not enough experience in parallel programming
  – Parallel programs of old days were restricted to some elite applications -> very few programmers
  – Now we need parallel programs for many different applications
Two Main Goals

- Maintain execution speed of old sequential programs
- Increase throughput of parallel programs
Two Main Goals

- Maintain execution speed of old sequential programs
- Increase throughput of parallel programs
Figure 1.1. Enlarging Performance Gap between GPUs and CPUs.

Multi-core CPU

Many-core GPU

Courtesy: John Owens
CPU is optimized for sequential code performance
Almost 10x the bandwidth of multicore (relaxed memory model)
How to Choose A Processor for Your Application?

- Performance
- Very large installation base
- Practical form-factor and easy accessibility
- Support for IEEE floating point standard

- 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

Streaming Multiprocessor (SM)
A Glimpse at A Modern GPU

SPs within SM share control logic and instruction cache
A Glimpse at A Modern GPU

- Much higher bandwidth than typical system memory
- A bit slower than typical system memory
- Communication between GPU memory and system memory is slow
Amdahl's Law

\[
\text{Execution Time After Improvement} = \\
\text{Execution Time Unaffected} + \left( \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} \right)
\]

• Example:

"Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?"

How about making it 5 times faster?

Improvement in your application speed depends on the portion that is parallelized.
Things to Keep in Mind

• Try to increase the portion of your program that can be parallelized
• Figure out how to get around limited bandwidth of system memory
• When an application is suitable for parallel execution, a good implementation on GPU can achieve more than 100x speedup over sequential implementation.
• You can reach 10x fairly easy, beyond that ... stay with us!
Enough for Today

• We are done with Chapter 1
• Some applications are better run on CPU while others on GPU
• If you don’t care about performance, parallel programming is easy!
• Main limitations
  – The parallelizable portion of the code
  – The communication overhead between CPU and GPU
  – Memory bandwidth saturation