GPU Computing & Architectures

1. Introduction

Ezio Bartocci
Vienna University of Technology
Aim of this course

Objectives:

• Gaining understanding of GPU computing architecture
• Getting familiar with GPU programming environments
• Learning Compute Unified Device Architecture (CUDA)
• Implementing programs solving problems that would classically have been run on a supercomputer

Requirements

• C/C++
Resources

Website:


Books

CUDA by Example: An Introduction to General-Purpose GPU Programming
Jason Sanders, Edward Kandrot, Addison Wesley

Programming Massively Parallel Processors: A Hands-on Approach
David B. Kirk, Wen-mei W. Hwu, Morgan Kaufmann

GPU Computing GEMS Emerald Edition
Wen-Mei w. Hwu, Morgan Kaufmann
# Schedule of the course

<table>
<thead>
<tr>
<th>Mon</th>
<th>Material</th>
<th>Title of the lesson</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 16</td>
<td>10.00-11.00</td>
<td>GPU Technology: Introduction and Motivation</td>
</tr>
<tr>
<td></td>
<td>11.00-12.30</td>
<td>CUDA Programming Model</td>
</tr>
<tr>
<td></td>
<td>12.30-14.00</td>
<td>Lunch Break</td>
</tr>
<tr>
<td></td>
<td>14.00-16.00</td>
<td>Coding in CUDA Lab: (CUDA Examples, Using nVidia, nvcc compiler, setting up a simple application)</td>
</tr>
<tr>
<td>Tue</td>
<td>10.00-12.30</td>
<td>CUDA Threads, Atomics and Memories</td>
</tr>
<tr>
<td>February 17</td>
<td>12.30-14.00</td>
<td>Lunch Break</td>
</tr>
<tr>
<td></td>
<td>14.00-16.00</td>
<td>Coding in CUDA Lab Exercise: Matrix Multiplication</td>
</tr>
<tr>
<td>Wed</td>
<td>10.00-12.30</td>
<td>CUDA Libraries (CURAND, THRUST, CUBLAS, CUSP, CUFFT)</td>
</tr>
<tr>
<td>February 18</td>
<td>12.30-14.00</td>
<td>Lunch Break</td>
</tr>
<tr>
<td></td>
<td>14.00-16.00</td>
<td>Coding in CUDA Lab Exercise: Monte Carlo Simulation and FFT</td>
</tr>
<tr>
<td>Thu</td>
<td>10.00-12.30</td>
<td>Advanced Topic: CUDA Profiling, CUDA Stream and Concurrency</td>
</tr>
<tr>
<td>February 19</td>
<td>12.30-14.00</td>
<td>Lunch Break</td>
</tr>
<tr>
<td></td>
<td>14.00-16.00</td>
<td>Coding in CUDA Lab Exercise: Reaction Diffusion System</td>
</tr>
<tr>
<td>Fri</td>
<td>10.00-12.30</td>
<td>Advanced Topic: CUDA Multi-GPU</td>
</tr>
<tr>
<td>February 20</td>
<td>12.30-14.00</td>
<td>Lunch Break</td>
</tr>
<tr>
<td></td>
<td>14.00-16.00</td>
<td>Coding in CUDA Lab Exercise: to be decided</td>
</tr>
</tbody>
</table>
Moore’s Law

“The number of the transistors of the integrated circuits double every 18 months”
Moore’s Law

More transistors, **but how to get them really fast?**

“The number of the transistors of the integrated circuits double every 18 months”

Gordon Moore, co-Founder
Intel Corporation
Moore’s Law

More transistors, but how to get them really fast?

Increasing the processor speed

- i386 DX (1985): 12 MHz
- Sandy Bridge (2011): 3.7 GHz

Power ~ Freq\(^3\)

- Single Core: 1 GHz / 1 Watt → Single Core: 4 GHz / 4\(^3\)=64 Watts

Increasing the parallelism (concurrency, multithreading)

- CPU Intel Xeon E5 v3 (2013): 12 Cores
- GPU NVIDIA TITAN Z (2013): 5760 Cores

Power ~ Num. of Cores

- Single Core: 1 GHz / 1 Watt → 4 Cores: 1 GHz / 4 Watts
GPU vs CPU

Graphic Processing Unit vs Central Processing Unit

Theoretical GFLOP/s
GPU vs CPU

Graphic Processing Unit vs Central Processing Unit

Theoretical GB/s

- GeForce 780 Ti
- Tesla K40
- Tesla K20X
- Tesla M2090
- Tesla C2050
- NVIDIA GeForce GTX 680
- NVIDIA GeForce GTX 480
- NVIDIA GeForce GTX 280
- NVIDIA GeForce 8800 GTX
- NVIDIA GeForce 7800 GTX
- NVIDIA GeForce 6800 GT
- NVIDIA GeForce FX 5900

NVIDIA: Northwood, Woodcrest, Harpertown, Westmere

Intel: Sandy Bridge, Ivy Bridge, Bloomfield, Westmere
GPU vs CPU

Graphic Processing Unit vs Central Processing Unit

Chip Design

ALU: Arithmetic Logic Unit

GPU devotes more transistors to data processing
NVIDIA GPU Roadmap

- **Volta**: Stacked DRAM
- **Maxwell**: Unified Virtual Memory
- **Kepler**: Dynamic Parallelism
- **Fermi**: FP64
- **Tesla**: CUDA

Timeline:
- 2008
- 2010
- 2012
- 2014
NVIDIA Tesla Architecture

- TPCs: Texture/Processor Clusters
- SMs: Stream Multiprocessors
- SPs: Streaming Processors
- SFU: Special Function Unit (4 floating-point multipliers)

MT Issue: Multithreaded instruction fetch
The GPU core is called **stream processor**

Stream processors are grouped in **Stream Multiprocessors**

- SM is a SIMD processor
- It is also called **Single Instruction Multiple Threads**
What about branches?

Time (clocks)

1 2 ... 8

ALU1 ALU2 ... ... ALU8

... If (x > 0) {
  y = x * x;
  y += 3;
  a = y + 3;
} else {
  x = 0;
  a = 3;
} ...
What about branches?

<table>
<thead>
<tr>
<th>Time (clocks)</th>
<th>ALU1</th>
<th>ALU2</th>
<th>...</th>
<th>...</th>
<th>ALU8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>...</td>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

```
  If (x > 0) {
      y = x * x;
      y += 3;
      a = y + 3;
  } else {
      x = 0;
      a = 3;
  }
```
What about branches?

Not all the ALUs do useful work!

Worst case: 1/8 performance
NVIDIA Fermi Architecture

Diagram showing the layout of the Fermi architecture with various components labeled such as DRAM, Host Interface, GigaThread, and L2 Cache.
NVIDIA Fermi Architecture

Each CUDA processor has a fully pipelined integer arithmetic logic unit (ALU) and floating point unit (FPU).

More cores per multiprocessors and faster arithmetic operations

Memory Protection Support: Memory are protected by a Single - Error Correct Double - Error Detect (SECDED) ECC code
GPU Design trend

GPU are designed to host many simple cores:

- The cores are designed to perform very simple tasks:
  high throughput (number of tasks in a fixed time)

- They have few control units: this may introduce
  latency (time to complete a task)

Objectives of a good programmer are:

- Maximize throughput of all threads

- Minimize latency of a thread
Hiding latency

A group of threads that want to access to the global memory or texture may introduce latency

Credits for the slide: M. Houston
Hiding latency

A group of threads that want to access to the global memory or texture may introduce latency.
Hiding latency

A group of threads that want to access to the global memory or texture may introduce latency.
Hiding latency

A group of threads that want to access to the global memory or texture may introduce latency.

Credits for the slide:
M. Houston
Hiding latency

A group of threads that want to access to the global memory or texture may introduce latency.
Increasing High-throughput

A group of threads that want to access to the global memory or texture may introduce latency.
NVIDIA Kepler Architecture
Dynamic Parallelism

GPU Adapts to Data, Dynamically Launches New Threads

NVIDIA Kepler Architecture
NVIDIA Kepler Architecture

Dynamic Parallelism
Makes GPU Computing Easier & Broadens Reach

Too coarse  Too fine  Just right
NVIDIA Kepler Architecture

GPUDirect™
Direct Transfers between GPU and 3rd Party Devices
CUDA

Computer Unified Device Architecture

- Parallel computer architecture developed by NVIDIA
- General purpose programming model:
  - Offers a computed designed API
  - Explicit GPU memory managing

CUDA Plymouth
by zede
### GPU Computing Applications

#### Libraries and Middleware
- CUFFT
- CUBLAS
- CURAND
- CUSPARSE
- CULA
- MAGMA
- Thrust
- NPP
- VSIPPL
- SVM
- OpenCurrent
- PhysX
- OptiX
- iray
- MATLAB
- Mathematica

#### Programming Languages
- C
- C++
- Fortran
- Java
- Python
- Wrappers
- DirectCompute
- Directives (e.g. OpenACC)

#### CUDA-Enabled NVIDIA GPUs

<table>
<thead>
<tr>
<th>Kepler Architecture (compute capabilities 3.x)</th>
<th>GeForce 600 Series</th>
<th>Quadro Kepler Series</th>
<th>Tesla K20, Tesla K10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fermi Architecture (compute capabilities 2.x)</td>
<td>GeForce 500 Series</td>
<td>Quadro Fermi Series</td>
<td>Tesla 20 Series</td>
</tr>
<tr>
<td></td>
<td>GeForce 400 Series</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tesla Architecture (compute capabilities 1.x)</td>
<td>GeForce 200 Series</td>
<td>Quadro FX Series</td>
<td>Tesla 10 Series</td>
</tr>
<tr>
<td></td>
<td>GeForce 9 Series</td>
<td>Quadro Plex Series</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GeForce 8 Series</td>
<td>Quadro NVS Series</td>
<td></td>
</tr>
</tbody>
</table>

- Entertainment
- Professional Graphics
- High Performance Computing