CUDA

Session 1

Introduction to CUDA

Hadi Salimi and Nima Ghaemian
Distributed Systems Lab.
School of Computer Engineering,
Iran University of Science and Technology,
hsalimi@iust.ac.ir and nima@comp.iust.ac.ir

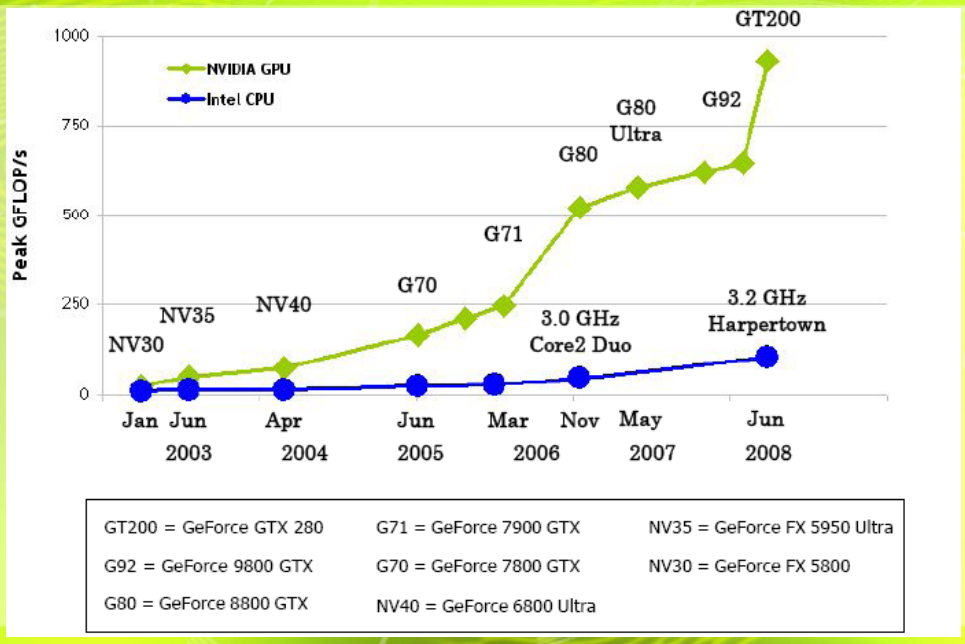
Traditional CPUs

- **Traditional CPUs:**
  - **Performance increases**
  - **Cost reductions**
  - **Brought GFLOPS to the desktop**
  - **Brought hundreds of GFLOPS to clusters**
- **This increase has slowed since 2003 due to power consumption issues that limited the increase of the clock frequency.**
- **Thus processor vendors have switched to multi-core and many-core models.**

Traditionally Programs

- **Traditionally, the vast majority of software applications are written as sequential programs.**
- **A sequential program will only run on one of the processor cores.**
- **To use the whole power of the cores of the new multi-core processors, parallel programs are needed.**

GPUs and CPUs Performance Gap



1 teraflop on your desktop

In June 2008, NVIDIA introduced the GT200 chip, which delivers almost 1 teraflop (1,000 gigaflops) of single precision and almost 100 gigaflops of double precision performance.

GPGPU & CUDA

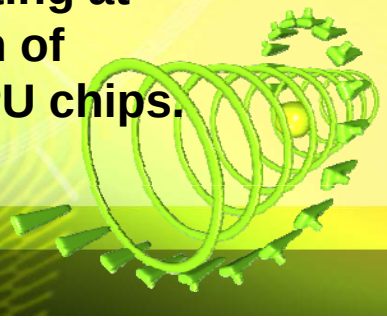
- **What is GPGPU?**
  - **General-Purpose computing on a Graphics Processing Unit**
  - **Using graphic hardware for non-graphic computations**
- **What is CUDA?**
  - **Compute Unified Device Architecture**
  - **Software architecture for managing data-parallel programming**

CPU vs. GPU

- CPU
  - Fast caches
  - Branching adaptability
  - High performance
- GPU
  - Multiple ALUs
  - Fast onboard memory
  - High throughput on parallel tasks
    - Executes program on each fragment/vertex
- CPUs are great for *task* parallelism
- GPUs are great for *data* parallelism

CPU vs. GPU (Cont.)

- The design of a CPU is optimized for sequential code performance.
- The general philosophy for GPU design is to optimize for the execution of massive number of threads.
- Graphics chips have been operating at approximately 10x the bandwidth of contemporaneously available CPU chips.



CPU vs. GPU (Cont.)

CPU-GPU Design philosophy

CPU vs. GPU (Cont.)

- GPU is designed as a numeric computing engine and it will not perform well on some tasks that CPUs are designed to perform well.

- Therefore, most applications will use both CPUs and GPUs, executing the sequential parts on the CPU and numeric intensive parts on the GPUs.

- CUDA programming model is designed to support joint CPU-GPU execution of an application.

Major problem

The major problem with traditional parallel processing applications:

Applications that can be run on a processor with a small market place will not have a large customer base!

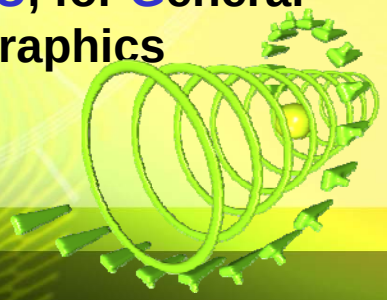
The G80 family of CUDA-capable processors and its successors have shipped almost 100 million units to date.

IEEE Floating-Point Standard

- **IEEE Floating-Point Standard makes it possible to have predictable results across processors from different vendors.**
- **Support for it was not widespread in early GPUs, but this has changed for the GeForce 8 series.**
- **The GPUs floating-point arithmetic units are primarily single precision today but we have already seen many applications where single precision floating is sufficient.**

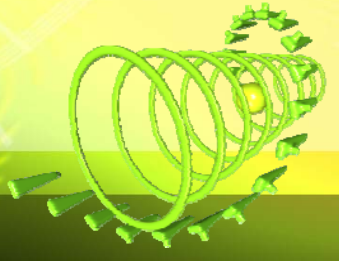
Traditionally GPGPU

- Until 2006, graphics chips were very difficult to use because programmers had to use the equivalent of graphic API to access the processor cores, (by OpenGL or direct3D techniques)
- This technique was called GPGPU, for General Purpose Programming using a Graphics Processing Unit.

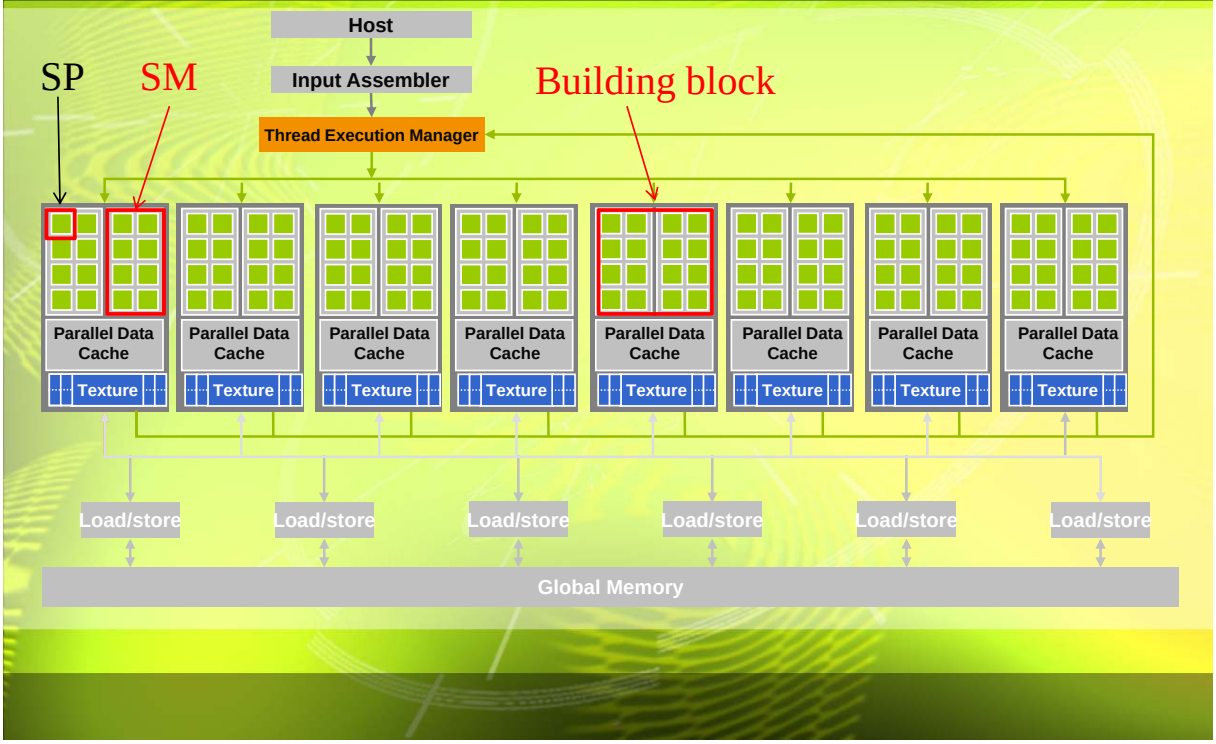


CUDA

- But everything changed in 2007 with the release of CUDA.
- NVIDIA actually devoted silicon area to facilitate the ease of parallel programming, so this does not represent software changes alone; additional hardware was added to the chip.



Architecture of a CUDA-capable GPU



It is organized into

- SP: Streaming Processor
- Each SP has a multiply-add (MAD) unit, and an additional multiply (MUL) unit, all running at 1.35 GHz.
- SMs: highly threaded Streaming Multiprocessors
  - Each SM has 8 streaming processors (SPs)
- Building block: A pair of SMs

Threads per core

- Intel supports 2 or 4 threads per core that is 16 threads on a quad-core!
- The G80 chip supports up to 768 threads per core that is 12,000 threads on 128 core!
- It's very important to use it efficiently!

GeForce 8800 Hardware

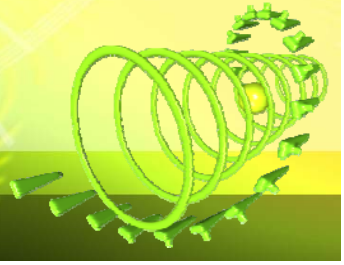
- The GeForce 8800 of the G8 series contains 16 GPU chips, or *streaming multiprocessors* (SMs).
- Each SM chip has eight cores, which currently run at 1.35 GHz.
- The off-chip video memory is currently limited to 1.5 GB.
- A single core has a 32-bit single-precision floating point unit that can also handle integers. (The G9 series will allow 64-bit operations.)
- Each SM chip also has two special functional unites (SFUs) that are presumably shared among the eight cores.

GeForce 8800 Hardware

- The SFUs perform reciprocal, square root, since and cosine.
- The 16 SMs and 2*nbsp;SFUs per chip imply a peak theoretical performance of (16 SM * 18 functional units * 1.35 GHYz) FLOPs, or 388.8 gigaflops (388.8 billion floating point operations per second).
- The aggregate bandwidth to off-chip global memory is 86.4 GB/s. (In comparison, bandwidth of CPU to RAM is typically between 5 GB/s and 10 GB/s.)

Speed up

- A good implementation on a GPU can achieve more than 100 times (100x) of speedup over a CPU.
- If the application includes “data parallelism,” it’s a simple task to achieve a 10x speedup with just a few hours of work.

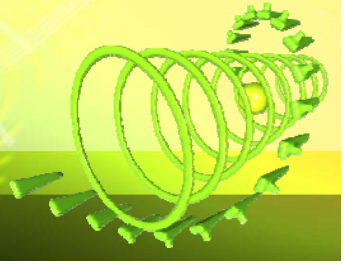


Why more speed up?

- The exciting applications of the future will be what we currently consider “supercomputing applications”.
- Microscopes with a supercomputing application for simulation can help biology!
- Massively parallel processors will continue to enhance the size and fidelity of the pictures of HDTVs in the coming years.

Why more speed up?

- Much better user interfaces with three-dimensional perspectives.
- Physics simulation in the games that can be based on dynamic simulation rather than pre-arranged scenes.
- And so on!



How much speed-up?

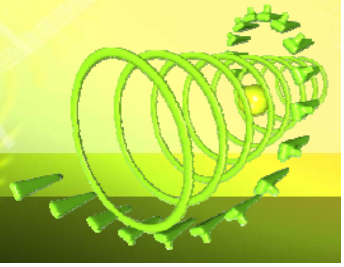
It depends on the portion of the application that can be parallelized.

Example 1

- **if 40% of the execution time is in the parallel portion, and we have a 50X speedup, how much speed-up can be expected from this application?**

- **Answer:**

  It will reduce the application execution to 60.8% that means 1.6X speedup.

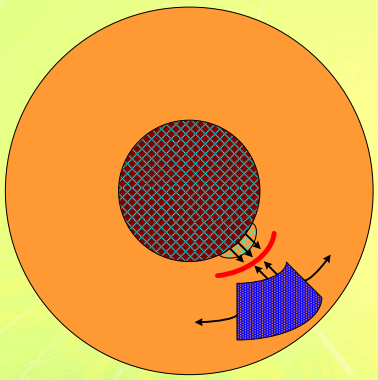


Example 2

- **if 99.9% of the execution time is in the parallel portion, and we have a 50X speedup, how much speed-up can be expected from this application?**
- **Answer:**

  It will reduce the application execution to 2.098% that means 47.5X speedup.

key parts of a typical application



Traditional applications

Current architecture coverage

New applications

Domain-specific architecture coverage

Obstacles

Web Resources

- **CUDA ZONE: http://www.nvidia.com/object/cuda_education.html**
- ***David Kirk and Wen-mei W. Hwu,* Lecture Notes of Programming Massively Parallel Processors, University of Illinois, Urbana-Champaign, *2009***
- **Rob Farber, "CUDA, Supercomputing for the Masses", *Dr. Dobb's Journal, can be found at:* http://www.ddj.com/hpc-high-performance-computing/**

Acknowledgment

- **Special Thanks to Mr. Ebrahim Khademi for his technical comments.**


Any Questions?
Any Question?