Parallel & Distributed Processing II: *parallel processing on manycore chips*

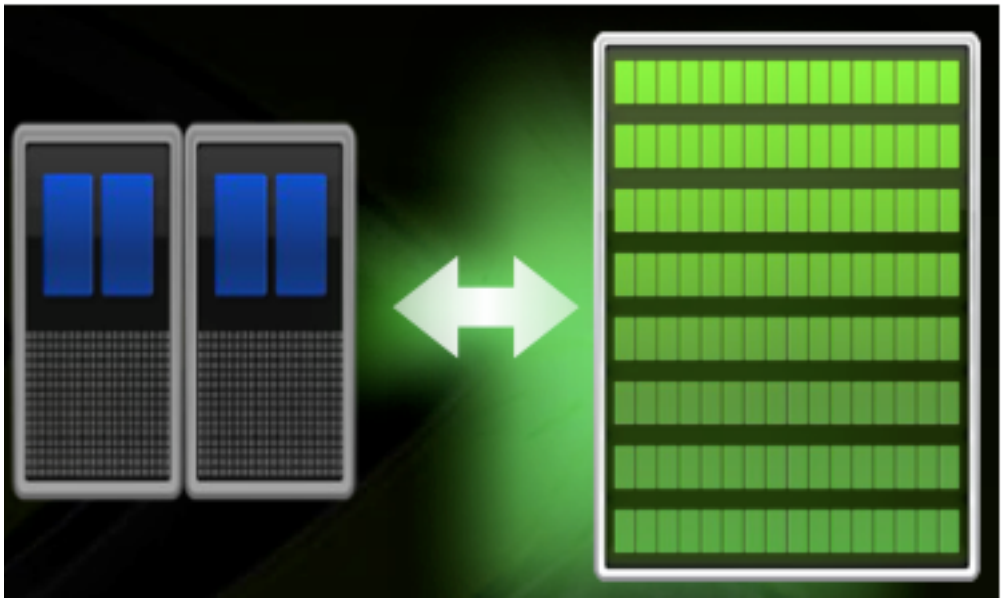
Introduction

Eric Aubanel
Winter 2010, UNB Fredericton

Multicore vs. Manycore

"*Regarding multicore versus manycore*: We believe that manycore is the future of computing. Furthermore, it is unwise to presume that multicore architectures and programming models suitable for 2 to 32 processors can incrementally evolve to serve manycore systems of 1000s of processors."

The Landscape of Parallel Computing Research: A View from Berkeley, 2006

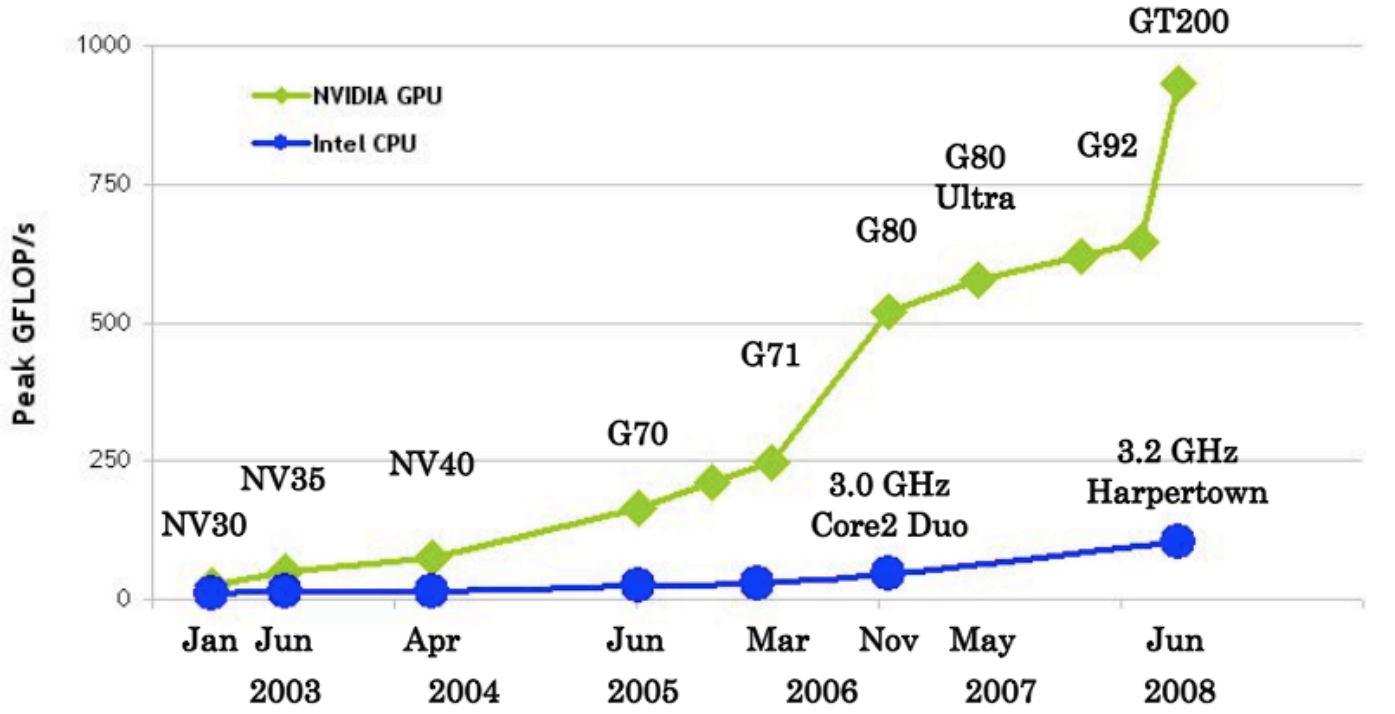


Source: Patrick LeGresley, NVIDIA

Manycore Chips

- Specialized processors
  - E.g., network processors, ASICs for bioinformatics, DSP chips, IBM Cell Broadband Engine
  - GPUs are dominant
- Manycore chips on separate "accelerator" cards, working together with CPU
  - Consequence for parallel programming: incremental parallelism
  - Amdahl's law!
- High Performance Computing and Embedded Computing coming together

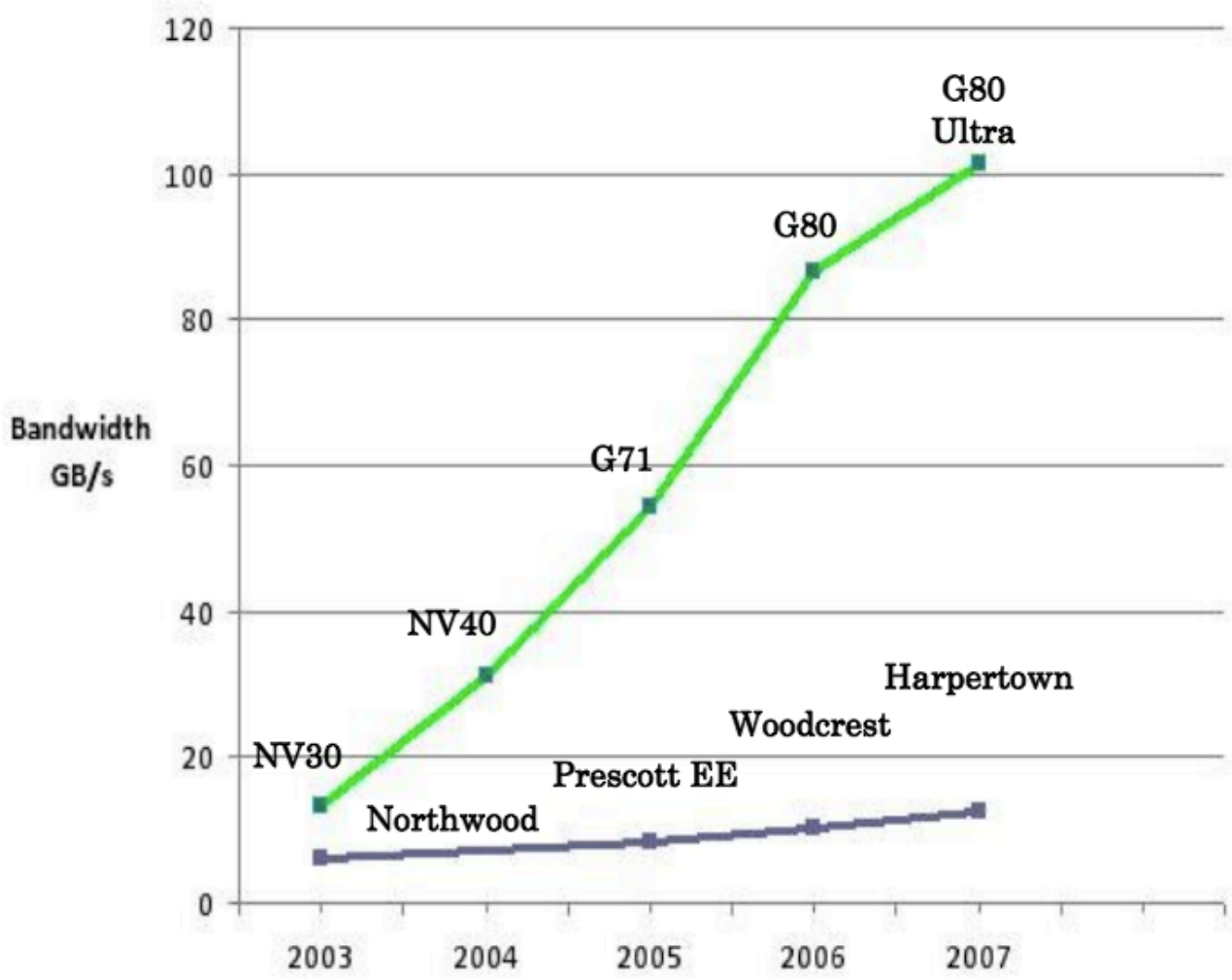
Why GPUs? *flops*



GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

Source: NVIDIA CUDA Programming Guide 2.2.1

Why GPUs? *bandwidth*

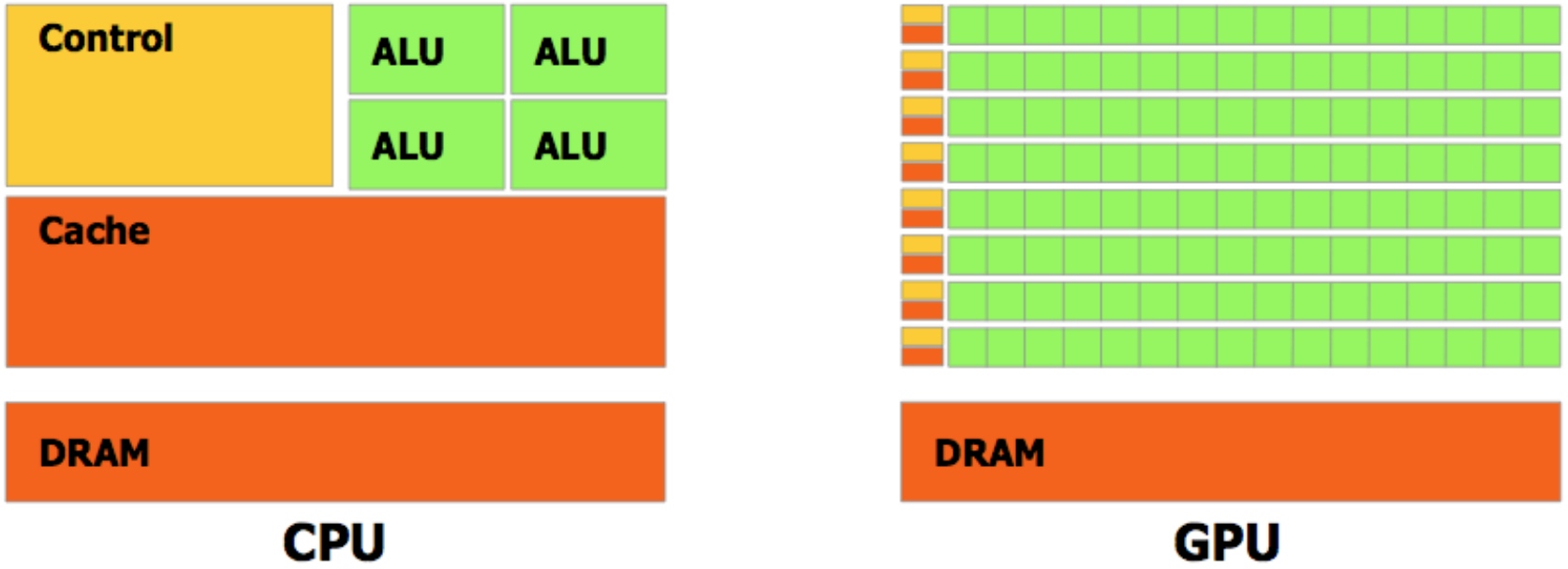


Source: NVIDIA CUDA Programming Guide 2.2.1

Why GPUs? *general purpose programming*

- 1999: First GPU, NVIDIA GeForce 256
- Graphics pipeline:
  - Vertex processors & Pixel fragment processors
    - Combination of task and data parallelism
  - mid 2000s: migration to single unified data parallel programmable unit
- Until recently, General Purpose GPU (GPGPU) programming only possible with graphics API, such as OpenGL
- Then, in 2007:
  - ATI's Close to Metal (assembly-level), Compute Abstraction Layer (CAL), support for Brook
  - NVIDIA's Compute Unified Device Architecture (CUDA) parallel programming model

GPU vs CPU



Source: NVIDIA CUDA Programming Guide 2.2.1

If you were plowing a field, which would you rather use?... Two strong oxen or 1024 chickens?
Seymour Cray

GPU vs. CPU

- "GPUs are throughput computers. They're optimized to deliver performance per dollar, per unit power, and per unit area for [applications] with plenty of parallelism.
- In contrast, CPUs optimize for latency. They spend a lot of die area and a lot of power optimizing single thread performance and doing speculative computation: predicting branches and so on. All of that costs area and power, and doesn't give you more FLOPS."

Interview with Bill Daly, HPC Wire Oct. 5 2009

CS6035 Goals

- Critical understanding of the manycore architecture landscape, with an emphasis on the algorithmic and programming challenges.
- In-depth understanding of the NVIDIA Tesla architecture and of the CUDA programming model, including the challenges involved in optimizing performance.
- Practical experience with a substantial programming project involving CUDA or OpenCL

CS6035 Questions

- What manycore chips have been developed?
- What types of applications do they target?
- Are they more suited to speeding up existing problems or solving larger problems?
- What factors are considered when making design decisions about manycore chips?
- How does programming manycore chips differ from programming multiprocessors or clusters?
- What are the major factors affecting performance on manycore chips?

CS6035 Questions

- What types of applications have been successfully ported to these chips?
- What software support is available (or needed) to aid the development of applications?
- Manycore architectures are changing very rapidly. What general principles can one apply to programming current and future chips? Do we have to start from zero with each new architecture?
- What algorithmic approaches will yield more efficient solutions on manycore chips?
- Are there good theoretical models for designing algorithms for manycore chips?
- ...