GPU Computing with CUDA

Dortmund, June 4, 2009
SFB 708, AK "Modellierung und Simulation"

Dominik Göddeke

Angewandte Mathematik und Numerik

TU Dortmund

dominik.goeddeke@math.tu-dortmund.de // http://www.mathematik.tu-dortmund.de/~goeddeke

Acknowledgements

- Slides based on previous courses by
  - Mark Harris, Simon Green, Gregory Ruetsch (NVIDIA)
  - Robert Strzodka (MPI Informatik)
  - Dominik Göddeke (TU Dortmund)

  - ARCS 2008 GPGPU and CUDA Tutorials
    http://www.mathematik.tu-dortmund.de/~goeddeke/arcs2008/
  - University of New South Wales Workshop on GPU Computing with CUDA
    http://www.cse.unsw.edu.au/~pls/cuda-workshop09/

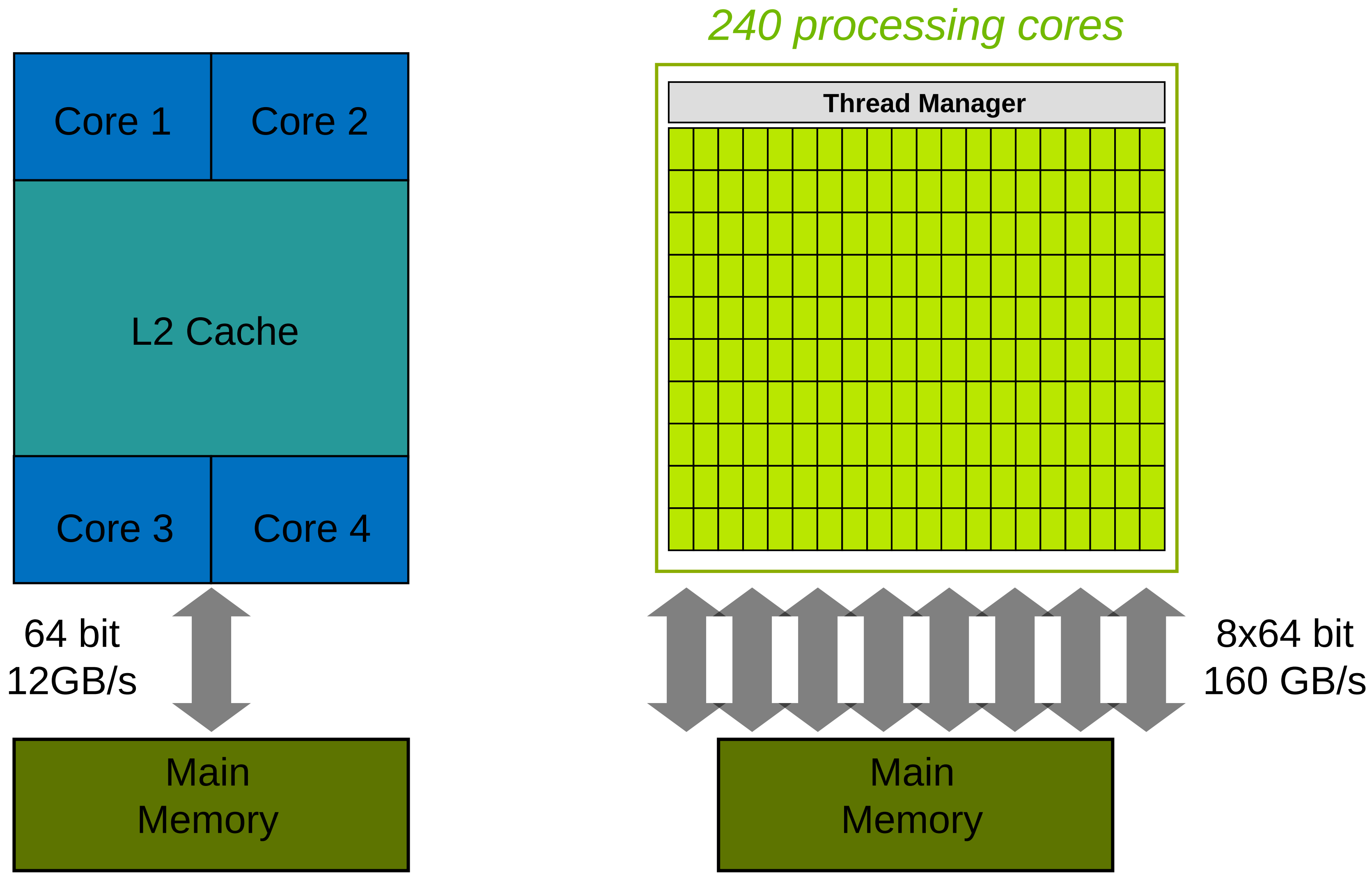
What is this all about

- Paradigm change in scientific computing
  - Frequency scaling is over, we are now scaling cores
  - Memory wall continues to get worse
- Many-core fine-grained parallel architectures
  - 100s of cores, 1000s of threads in flight in parallel
  - SIMD characteristics
- It has started
  - 4-core commodity CPUs by AMD and Intel are abundant
  - AMD RV770: 800 stream processing units
  - NVIDIA GT200: 30 multiprocessors with 8+3 processing cores each
  - Do you know how your code scales with 100s of cores?
- GPUs are getting faster, faster
  - 1 TFLOP/s and 160 GB/s on a single GPU

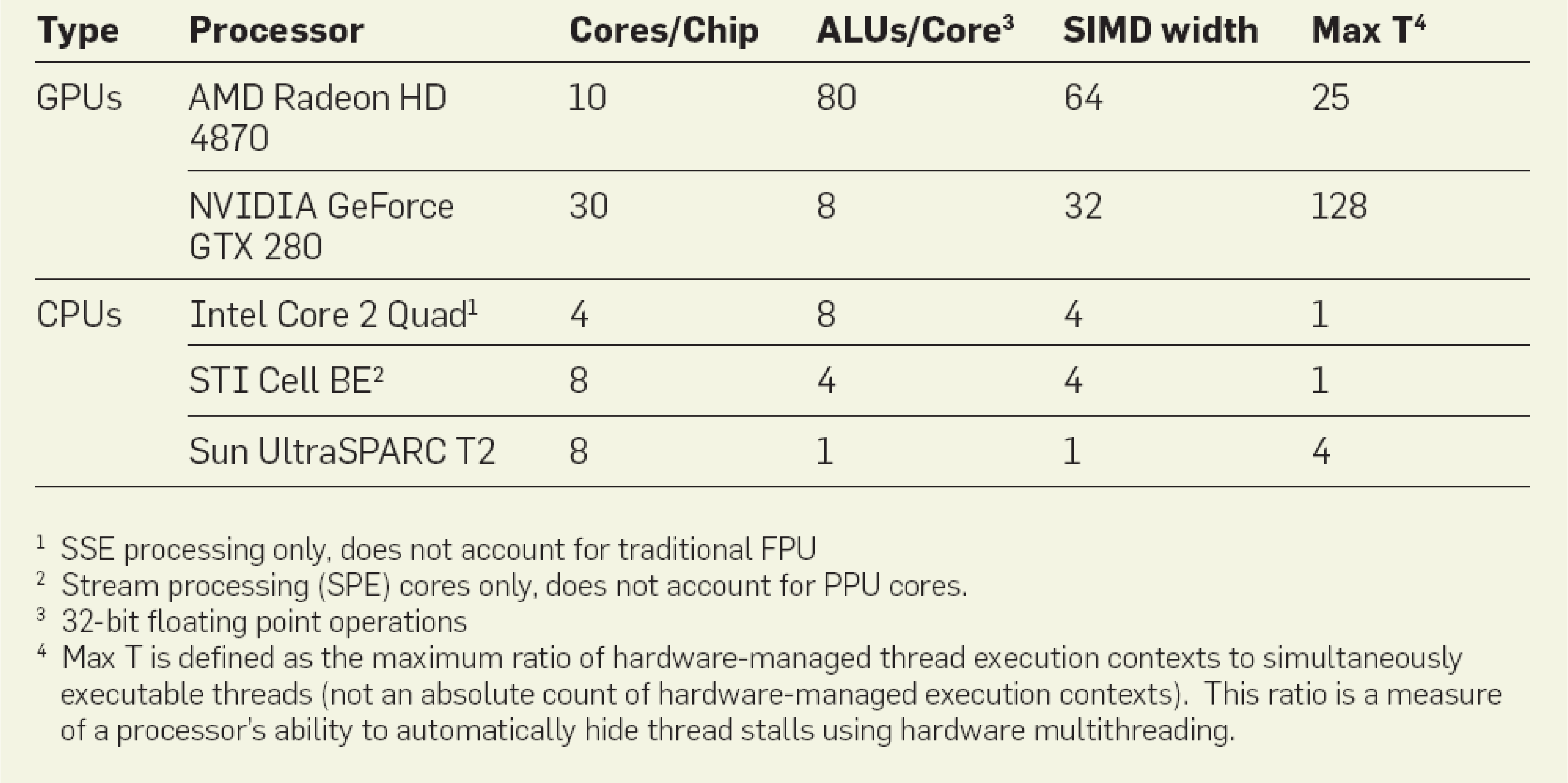
What is the GPU good at?

- Data-parallel processing
  - The same computation is executed on many data elements in parallel
  - Low control flow overhead
- High arithmetic intensity
  - Many calculations per memory access
- Throughput-oriented architecture
  - Hardware keeps 1000s of threads in flight simultaneously
  - Latency of an individual operation (especially memory access) is HIGH
  - Thread scheduler hides latencies for maximum throughput
  - No need for cache hierarchies

CPUs vs. GPUs


Core 1
Core 2
L2 Cache
Core 3
Core 4
64 bit
12GB/s
Main Memory
240 processing cores
Thread Manager
8x64 bit
160 GB/s
Main Memory

CPUs vs. GPUs



Type	Processor	Cores/Chip	ALUs/Core[3]	SIMD width	Max T[4]
GPUs	AMD Radeon HD 4870	10	80	64	25
	NVIDIA GeForce GTX 280	30	8	32	128
CPUs	Intel Core 2 Quad[1]	4	8	4	1
	STI Cell BE[2]	8	4	4	1
	Sun UltraSPARC T2	8	1	1	4

[1] SSE processing only, does not account for traditional FPU

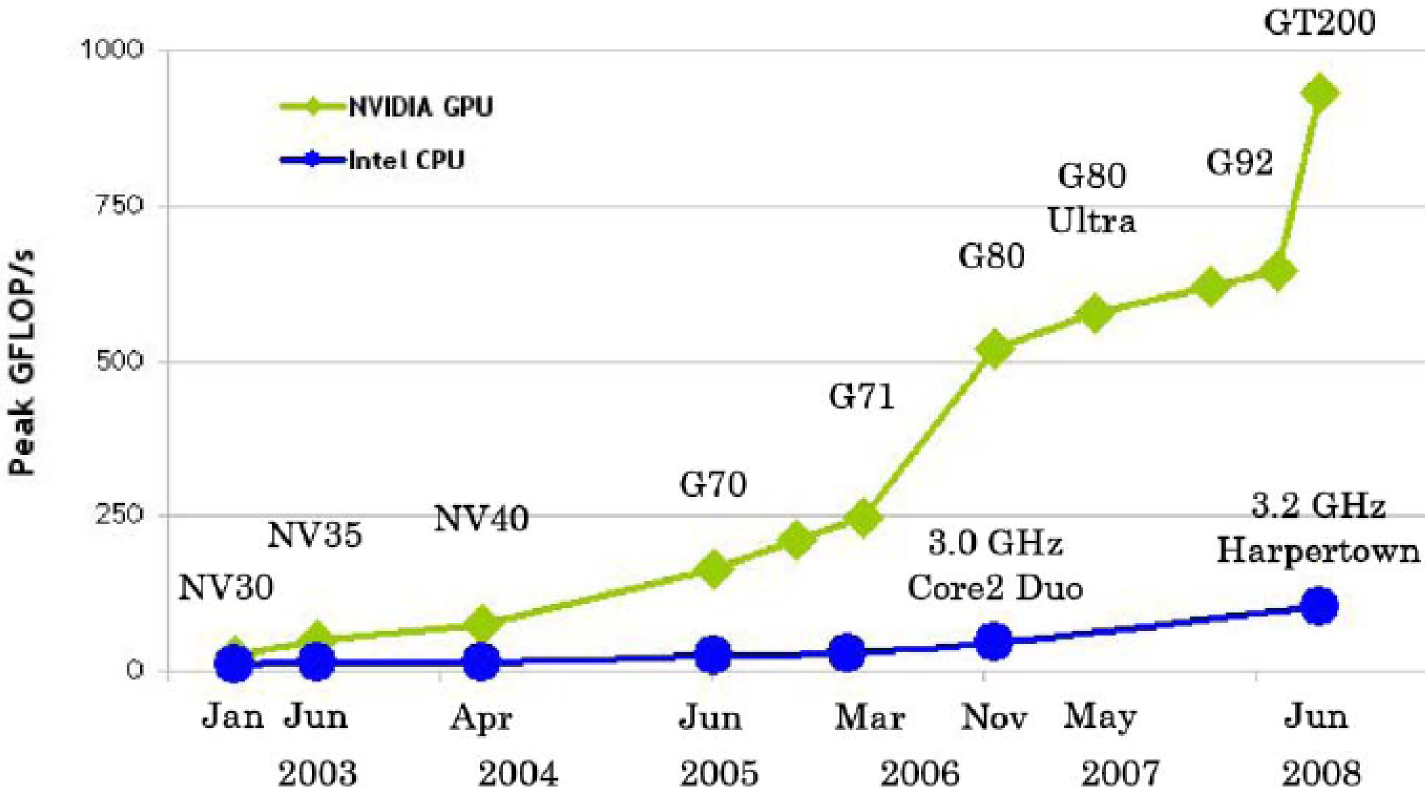
[2] Stream processing (SPE) cores only, does not account for PPU cores.

[3] 32-bit floating point operations

[4] Max T is defined as the maximum ratio of hardware-managed thread execution contexts to simultaneously executable threads (not an absolute count of hardware-managed execution contexts). This ratio is a measure of a processor's ability to automatically hide thread stalls using hardware multithreading.

Kayvon Fatahalian and Mike Houston:
A closer look at GPUs,
Communications of the ACM,
Vol 51 No 10, October 2008

Peak performance



GT200 = GeForce GTX 280

G92 = GeForce 9800 GTX

G80 = GeForce 8800 GTX

G71 = GeForce 7900 GTX

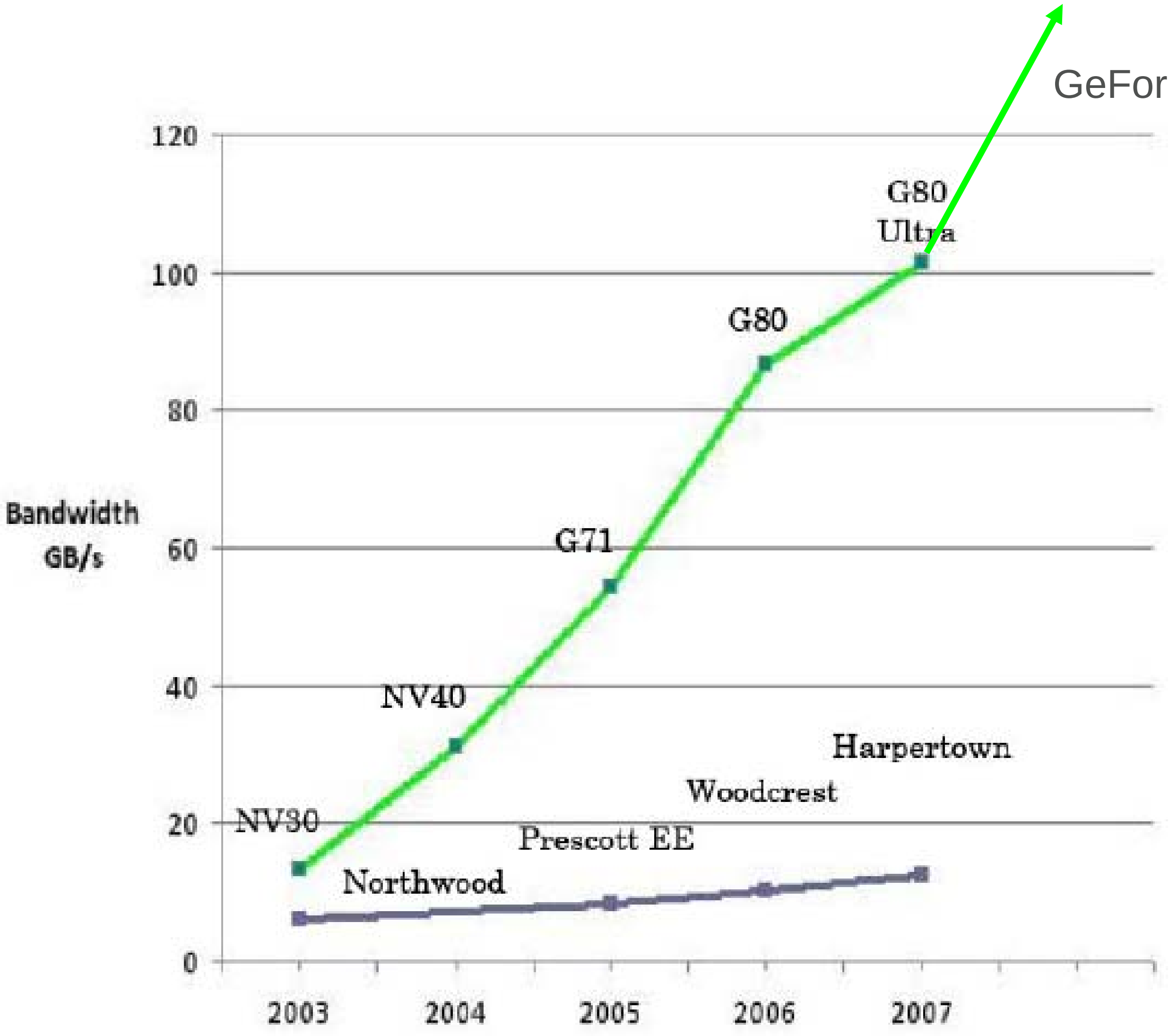
G70 = GeForce 7800 GTX

NV40 = GeForce 6800 Ultra

NV35 = GeForce FX 5950 Ultra

NV30 = GeForce FX 5800

Memory bandwidth



GeForce GTX 280: 140 GB/s

Current high-end model: GTX 285, 1.04 TFLOP/s, 160 GB/s

Outline

- GPU computing evolution (~30 minutes)
  - Why GPUs need to be fast
  - The graphics pipeline
  - Evolution towards programmability
  - The first wave of GPGPU: 2003-2006
  - Example architecture: GeForce 6800 Ultra (2004)
  - Consolidation with DirectX 10?

- CUDA introduction (~60 minutes)
  - CUDA parallel hardware architecture
  - CUDA programming model
  - Code walkthrough
  - Libraries
  - Tool chain and OpenCL
  - Tesla compute hardware

Outline

- CUDA performance tips and tricks (90 minutes)
  - Hardware
  - Memory optimizations
  - Execution configuration optimizations
  - Instruction optimizations

Further reading

- http://gpgpu.org/developer
  - Recommended reading
  - Simple tutorial codes
  - Links to conference courses

- http://www.nvidia.com/cuda
  - Toolkit and driver downloads
  - CUDA SDK
  - CUDA library of results
  - Developer forums
  - ...