

CSE 564: Visualization

High Performance Computing with GPUs

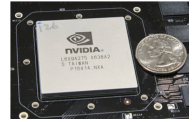
Klaus Mueller

Computer Science Department
Stony Brook University

High Performance Computing on the Desktop

PC graphics boards featuring GPUs:

- NVIDIA GeForce, ATI Radeon
- available at every computer store for less than \$500
- set up your PC in less than an hour and play



the latest board:
NVIDIA GeForce GTX 480

“Just” Computing

Compute-only (no graphics): NVIDIA Tesla c2050/c2070



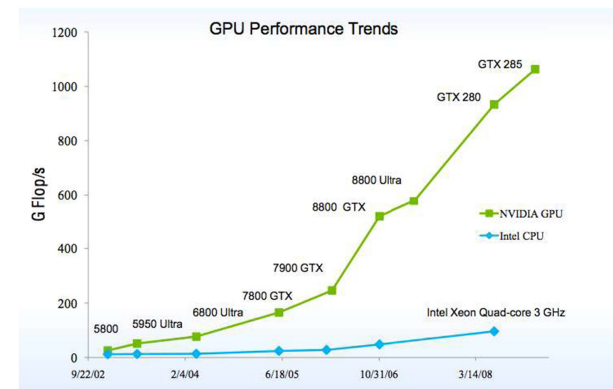
True GPGPU
(General Purpose
Computing using
GPU Technology)

3/6 GB memory
per card, 448
processors

\$2,500/\$3,500

Bundle 8 cards into a server: 3,584 processors, 48 GB memory

Incredible Growth



Performance gap GPU / CPU is growing

- currently 1-2 orders of magnitude is achievable (given appropriate programming and problem decomposition)

GPU Vital Specs

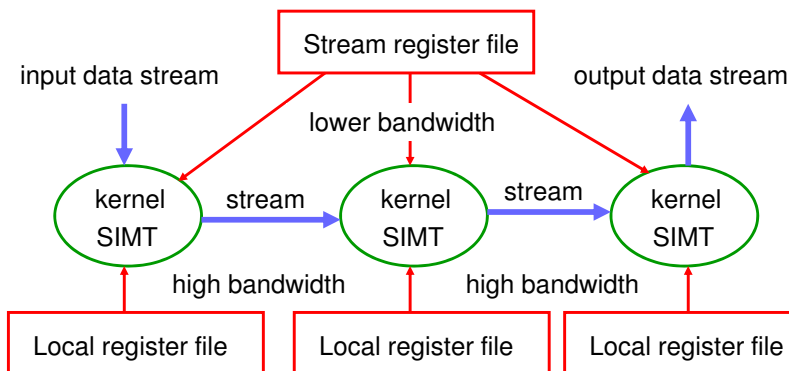
| | GeForce 8800 GTX | GeForce GTX 480 |
|----------------------|---------------------|----------------------|
| Codename | G80 | GF108 |
| Release date | 11/2006 | 3/2010 |
| Transistors | 681 M (90nm) | 3200 M (40nm) |
| Clock speed | 1,350 MHz | 1,401 MHz |
| Processors | 128 | 480 |
| Peak pixel fill rate | 13.8 Gigapixels/s | 33.6 Gigapixels/s |
| Pk memory bandwidth | 86.4 GB/s (384 bit) | 177.4 GB/s (512 bit) |
| Memory | 768 MB | 1536 MB |
| Peak performance | 520 Gigaflops | 1,344 Gigaflops |

Comparison with CPUs

| | Intel Xeon Nehalem | GeForce GTX 480 |
|------------------------|--------------------|-----------------|
| Cores / Chip | 8 | 15 |
| ALUs / Core | 1 | 32 |
| Managed threads / Core | 2 | 1536 |
| Clock speed | 2 GHz | 1.4 GHz |
| Performance | 96 Gigaflops | 1,344 Gigaflops |

Stream Processing

GPUs are *stream processors* [Kapasi '03]
(with some restrictions) [Venkatasubramanian '03]



GPU vs. CPU

One instruction-decode per kernel stream

- CPU needs a decode for each data item

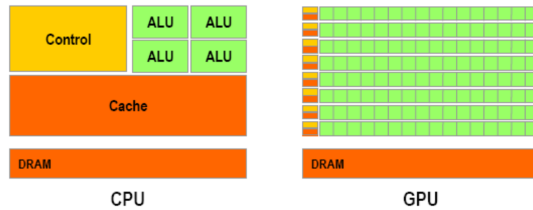
Highly parallel

- GTX 480 has 480 processors
- Memory very close to processors → fast data transfer
- Threads are cheap to switch (light-weight)
→ use this to swap out waiting threads, swap in ready threads
- CPU requires lots of cache logic and communication to manage resources
- GPU has the resources close by

GPU vs. CPU

High % of GPU chip real-estate for computing

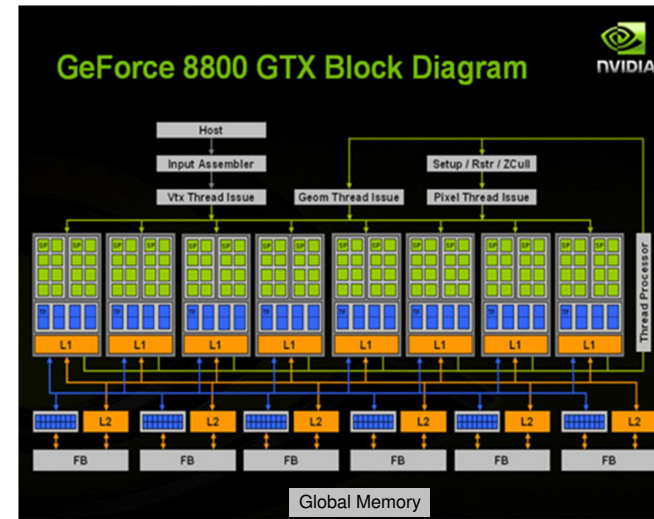
- small in CPUs (example, 6.5% in Intel Itanium)



In many cases speedups of 1-2 orders of magnitude can be obtained by porting to GPU

- more details on the rules for effective porting later

GPU Architecture: Overview



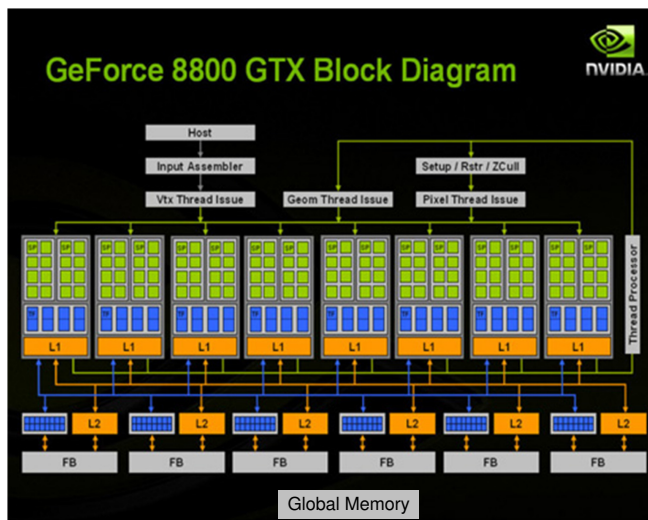
128 processors →
8 multi-processors
of 16 processors
each

local cache L1 (4k)

shared cache L2
(1M)

DRAM (global
memory)

GPU Architecture: Overview



Memory
management is key!

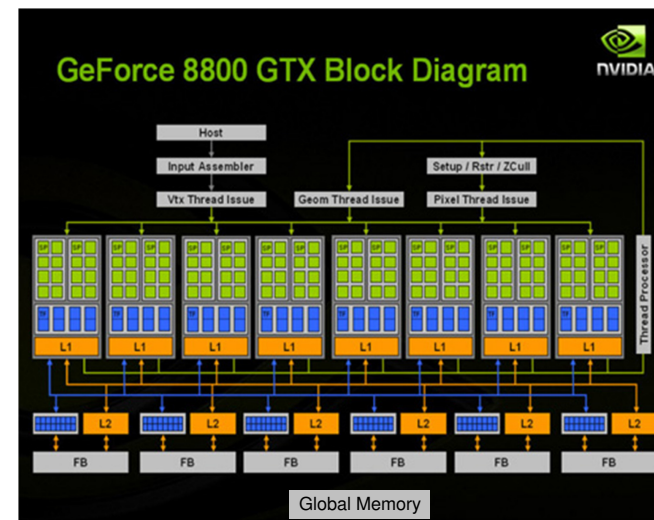
128 processors →
8 multi-processors
of 16 processors
each

local cache L1 (4k)

shared cache L2
(1M)

DRAM (global
memory)

GPU Architecture: Overview



Memory
management is key!

Thread
management is key!

128 processors →
8 multi-processors
of 16 processors
each

local cache L1 (4k)

shared cache L2
(1M)

DRAM (global
memory)

Latency Hiding

GPUs provide *hardware multi-threading*

- kicks in when threads within a core ALU stall (waiting for memory, etc)
- then another (light-weight) SIMT thread group is swapped in for execution
- this *hides the latency* for the stalled threads
- GTX 480 allows 48x more threads to be maintained than currently SIMT- executed

Hardware multi-threading requires memory

- contexts of all these threads must be maintained in memory
- this typically limits the amount of threads that can be simultaneously maintained for latency hiding

Parallel Programming Languages

CUDA (C-interface: *C for CUDA*, also Fortran), OpenCL

Expose details on GPU memory and thread management

- memory hierarchy, latencies, operation costs, etc
- shading languages don't make this explicit
- give programmers better control over memory, threads, and arithmetic intensity (via occupancy calculator, profiler)

Promote computations as SIMT threads, executed in kernels

- synonymous to fragments in shading languages

But still require (for optimal performance):

- careful computation flow planning, memory management, and analysis before coding
- no magic here; no pain, no gain

GPGPU

GPGPU = **G**eneral **P**urpose Computation on Graphics hardware (**GPU**)

- massive trend to use GPUs for main stream computing
- see <http://www.gpgpu.org>

Accelerate

- volume rendering and advanced graphics effects
- computer vision
- scientific computing and simulations
- audio and image & video processing
- database operations, numerical algorithms and sorting
- data compression
- medical imaging
- and many others

GPGPU Example Applications

