

CSE 591/392: GPU Programming

Introduction

Klaus Mueller

Computer Science Department

Stony Brook University

First: A Big Word of Thanks!

... to the millions of computer game enthusiasts worldwide



Who demand an utmost of performance and realism of their game engines

And who create a market force for high performance computing that beats any federal-funded effort (DOE, NASA, etc.)

High Performance Computing on the Desktop

PC graphics boards featuring GPUs:

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



the latest board:
NVIDIA GeForce GTX 960

“Just” Computing

Compute-only (no graphics): NVIDIA Tesla K 40 (Kepler)



True GPGPU
(General Purpose
Computing using
GPU Technology)

12 GB memory
per card, 2,880
processors
~\$3,500

Bundle 8 cards into a server: 23k processors, 96 GB memory

At SUNY Korea

NVIDIA Kepler K20 Workstation

MW 7047-GPU Workstation SYSTEM

Key Features



- Intel® Xeon® processor E5-2600 Dual Socket(LGA 2011)
QPI up to 8GT/s (up to 150W TDP)
- Intel C602 Chipset
- Up to 512GB DDR3 1600MHz ECC Registered DIMM 16x DIMM sockets
- 2x Intel i350 Port Gigabit Ethernet LAN ports
1x RJ45 Dedicated IPMI LAN port
- 6x SATA (3 Gbps) Ports vialCH10R Controller
- Expansion Slots
4 PCI-E 3.0 x16(4x NVIDIA Tesla GPU cards),
2 PCI-E 3.0 x8 (1 in x16),
1 PCI-E 2.0 x4 (in x8)
- Integrated Matrox G200eW Graphics
- 1620W Redundant Power Supplies 80 PLUS, Platinum Level (94%)
- Form Factor : 178mm x 452 x 29.4" (HxWxD)

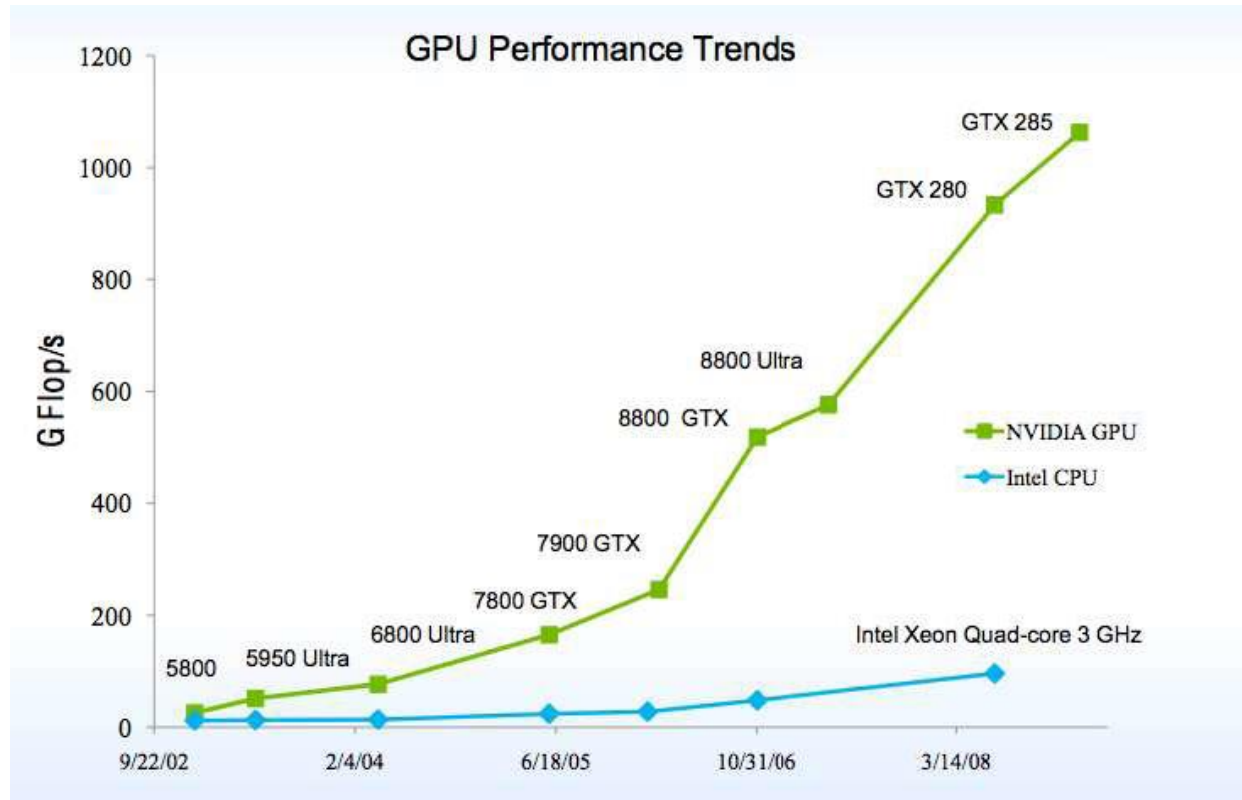
At SUNY Korea

NVIDIA HP ProLiant Server

Dual rack server with 12 Tesla K 20 cards



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

	<i>GeForce 8800 GTX</i>	<i>GeForce GTX 580</i>
Codename	G80	GF118
Release date	11/2006	11/2010
Transistors	681 M (90nm)	3,000 M (40nm)
Clock speed	1,350 MHz	1,544 MHz
Processors	128	512
Peak pixel fill rate	13.8 Gigapixels/s	37.6 Gigapixels/s
Pk memory bandwidth	86.4 GB/s (384 bit)	192 GB/s (384 bit)
Memory	768 MB	1536 MB
Peak performance	520 Gigaflops	1,581 Gigaflops

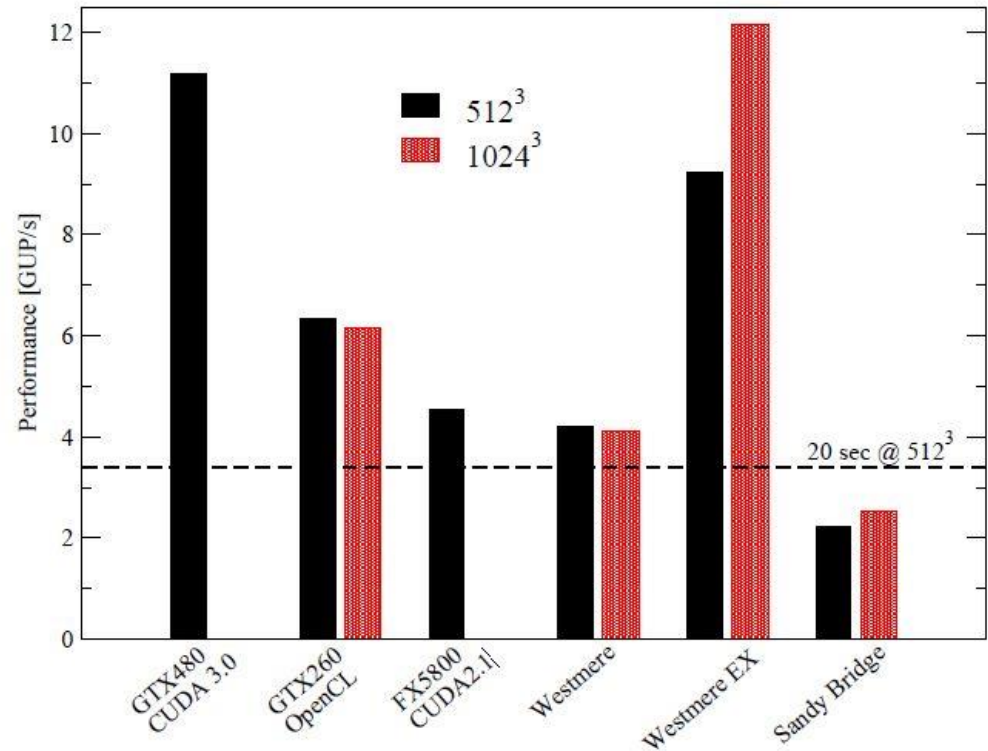
Comparison with CPUs

	Intel Xeon Westmere X5670	GeForce GTX 580
Price	\$800	\$500
Cores / Chip	6	16
ALUs / Core	1	32
Managed threads / Core	2	1536
Clock speed	3 GHz	1.5 GHz
Performance	96 Gigaflops	1,581 Gigaflops

Comparison with CPUs

Backprojection task

- 496 projections
- size 1248x960 each



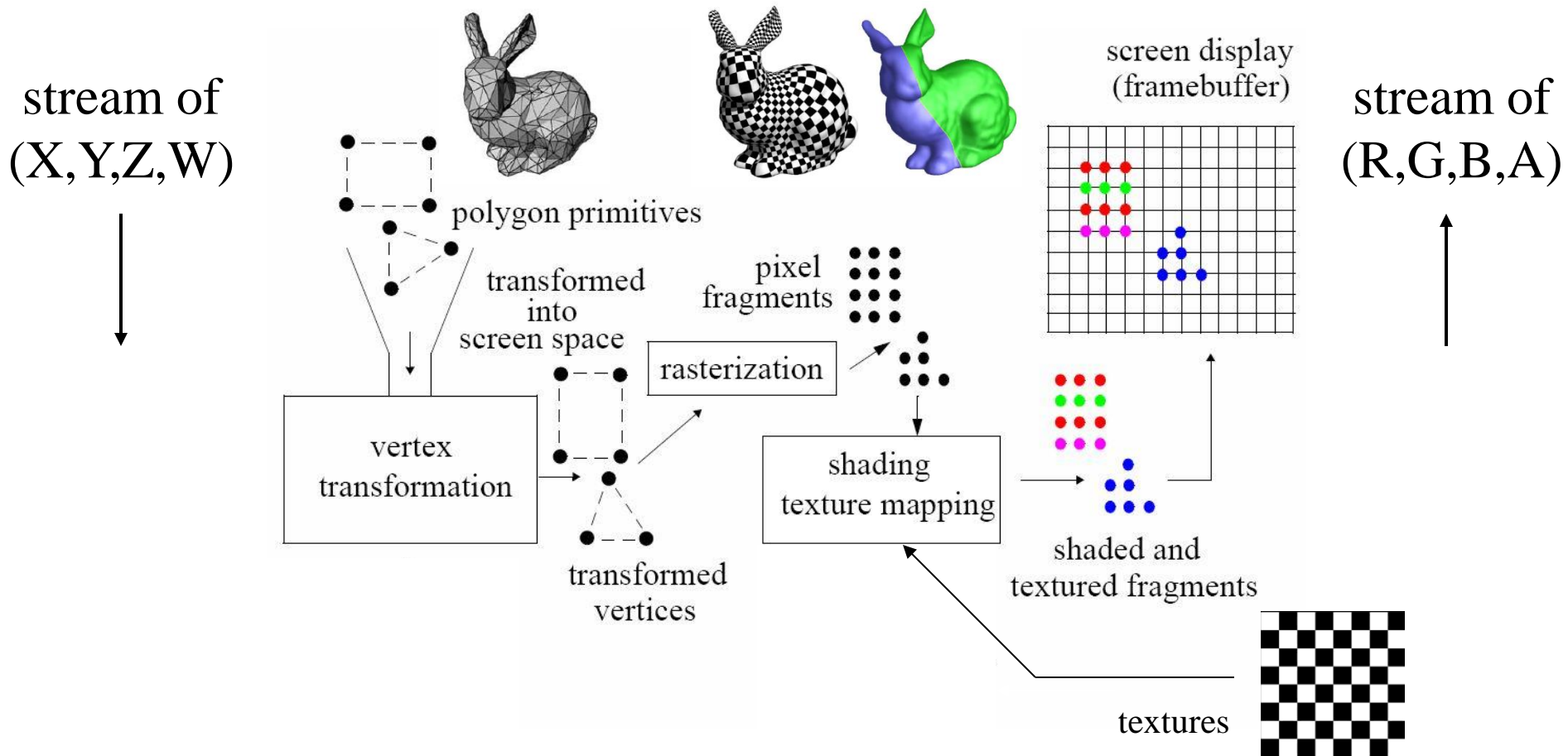
from Treibig et al. "Pushing the limits for medical image reconstruction on recent standard multicore processors," *International Journal of High Performance Computing Applications*

\$500

\$4,500

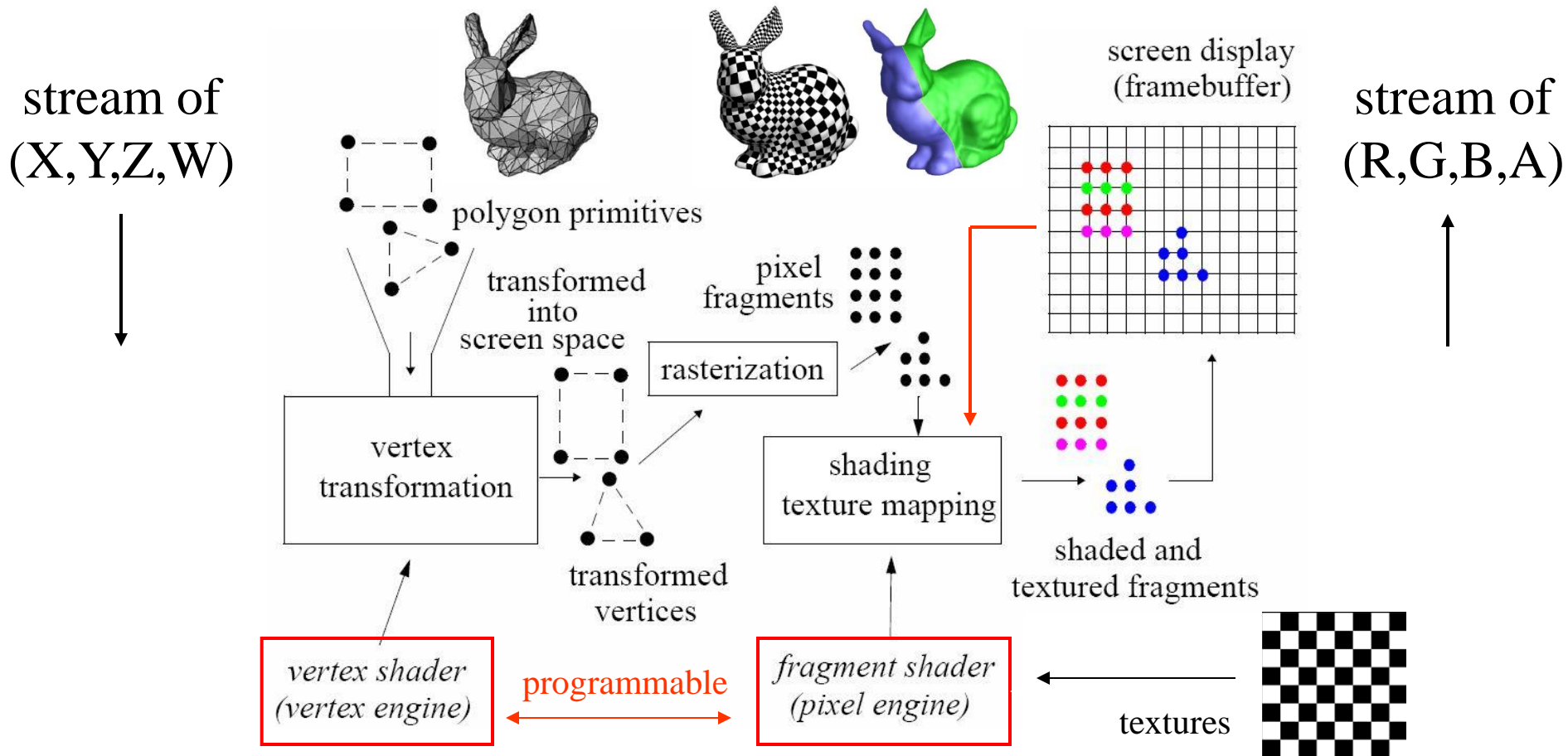
The Graphics Pipeline

Old-style, non-programmable:



The Graphics Pipeline

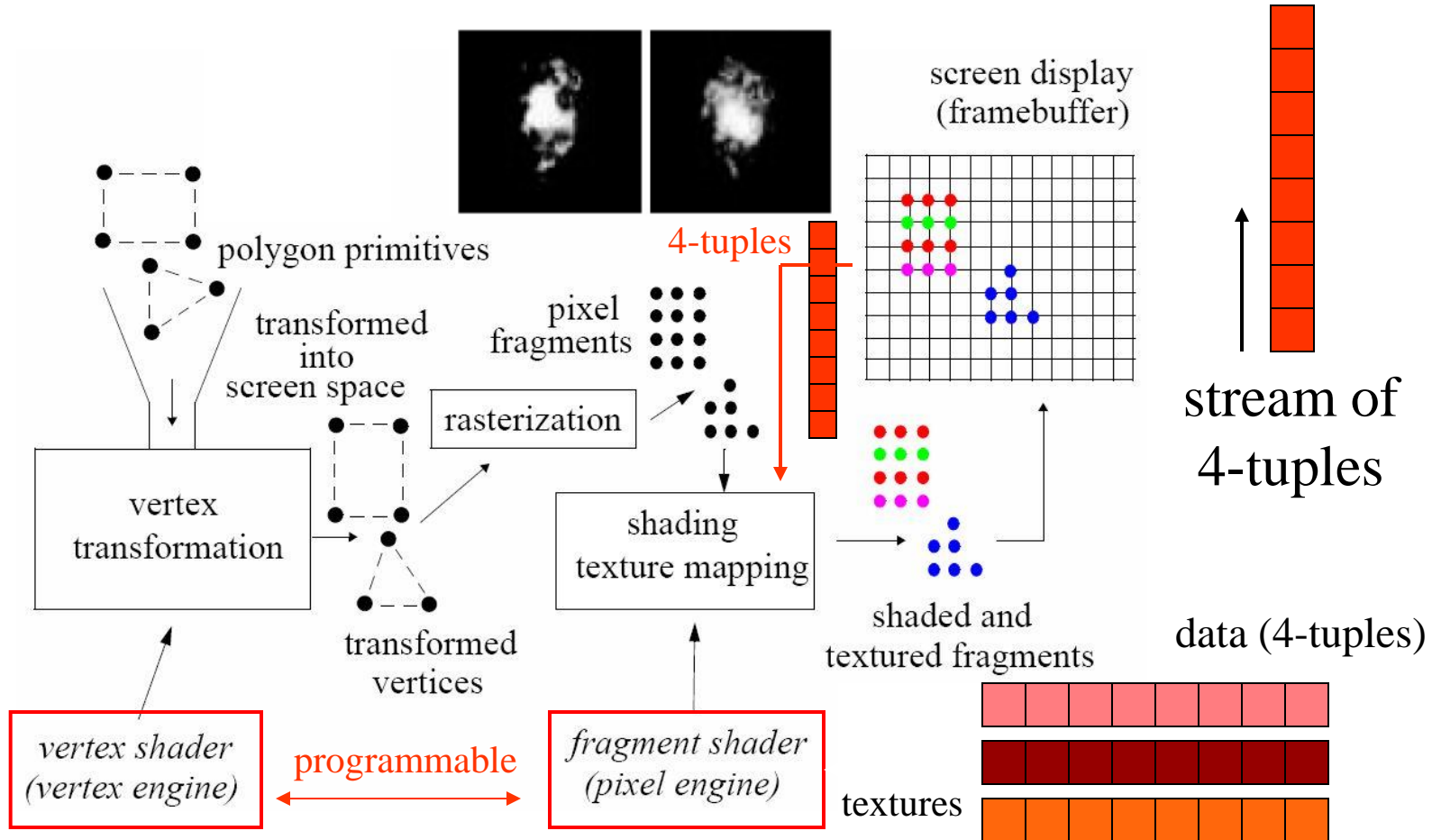
Modern, programmable:



The Graphics Pipeline

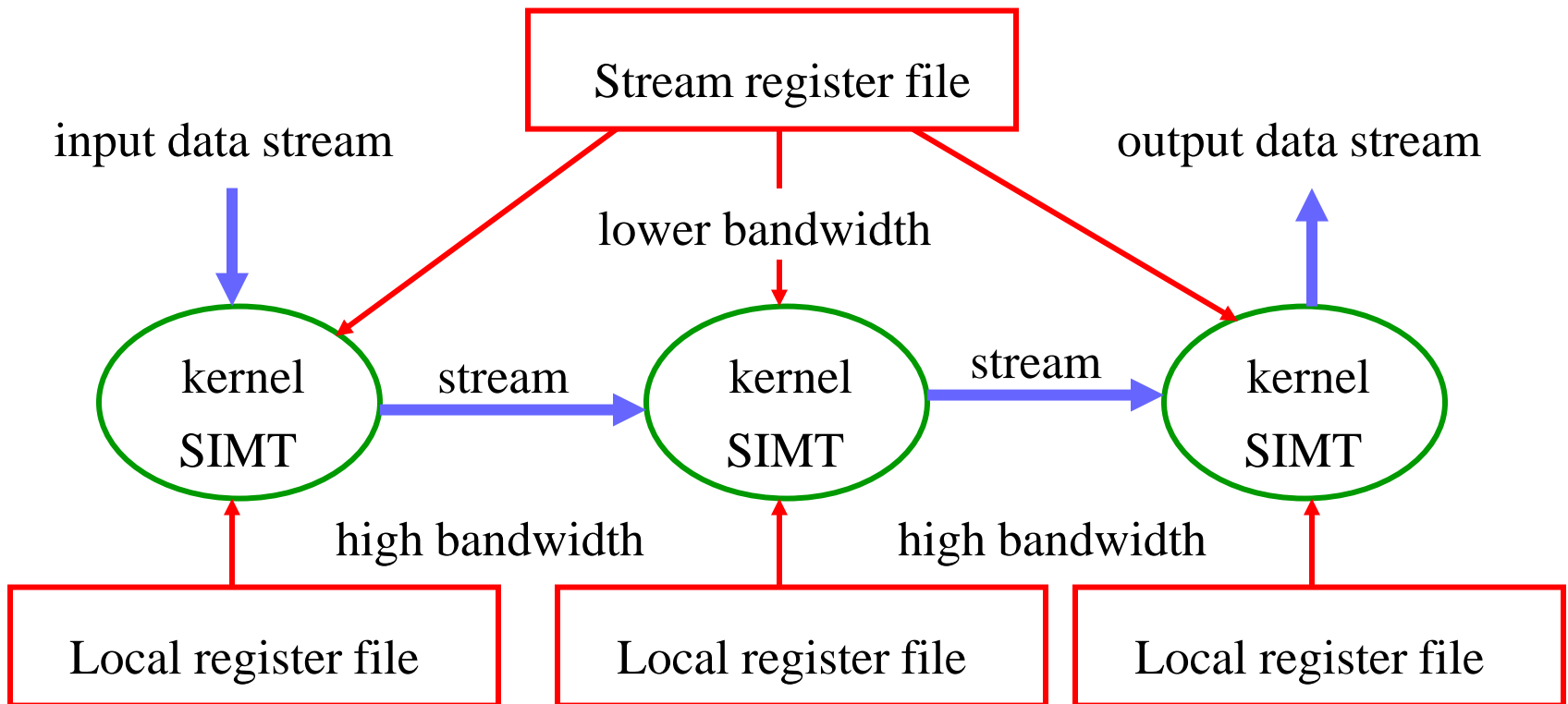
From a computational view:

stream of
control
primitives



Stream Processing

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



History: Accelerated Graphics

1990s: accelerated graphics

- Silicon Graphics (SGI)
- expensive and non-programmable



Late 1990s: rise of consumer graphics chips

- Voodoo, ATI Rage, NVIDIA Riva
- chips still separate from memory

End 1990s: consumer graphics boards with high-end graphics

- the world's first GPU: NVIDIA GeForce 256 (NV 10)
- inexpensive, but still non-programmable

2000s: *programmable* consumer graphics hardware

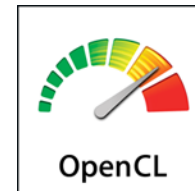
- graphics cards: NVIDIA GeForce 3, ATI Radeon 9700
- HW programming languages: CG, GLSL, HLSL



Now: Focus Parallel Computing

2006: parallel computing languages appear

- dedicated SDK and API for parallel high performance computing (GPGPU)
- CUDA (Compute Unified Device Architecture)
 - developed by NVIDIA
- OpenCL (Open Computing Language)
 - initially developed by Apple
 - now with the Khronos Compute Working Group
- specific GPGPU boards: NVIDIA Tesla, AMD FireStream



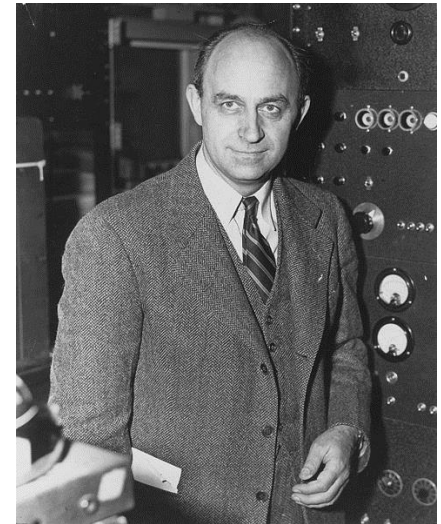
Right Now: Focus “Serious” Parallel Computing

2009: next generation CUDA architectures announced

- NVIDIA Fermi, AMD Cypress
- substrate for supercomputing
- focused on “serious” high performance computing (clusters, etc)

Enrico Fermi (1901-1954)

- Italian physicist
- one of the top scientists of the 20th century
- developed the first nuclear reactor
- contributed to
 - quantum theory, statistical mechanics
 - nuclear and particle physics
- Nobel Prize in Physics in 1938 for his work on induced radioactivity



GPU vs. CPU

One instruction-decode per kernel stream

- CPU needs a decode for each data item

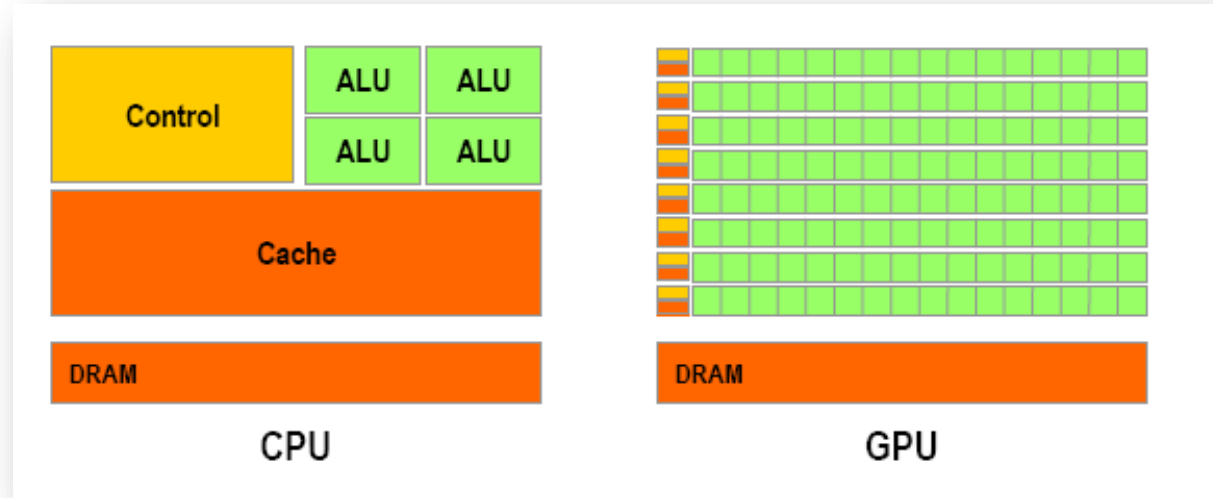
Highly parallel

- GTX 580 has 512 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)

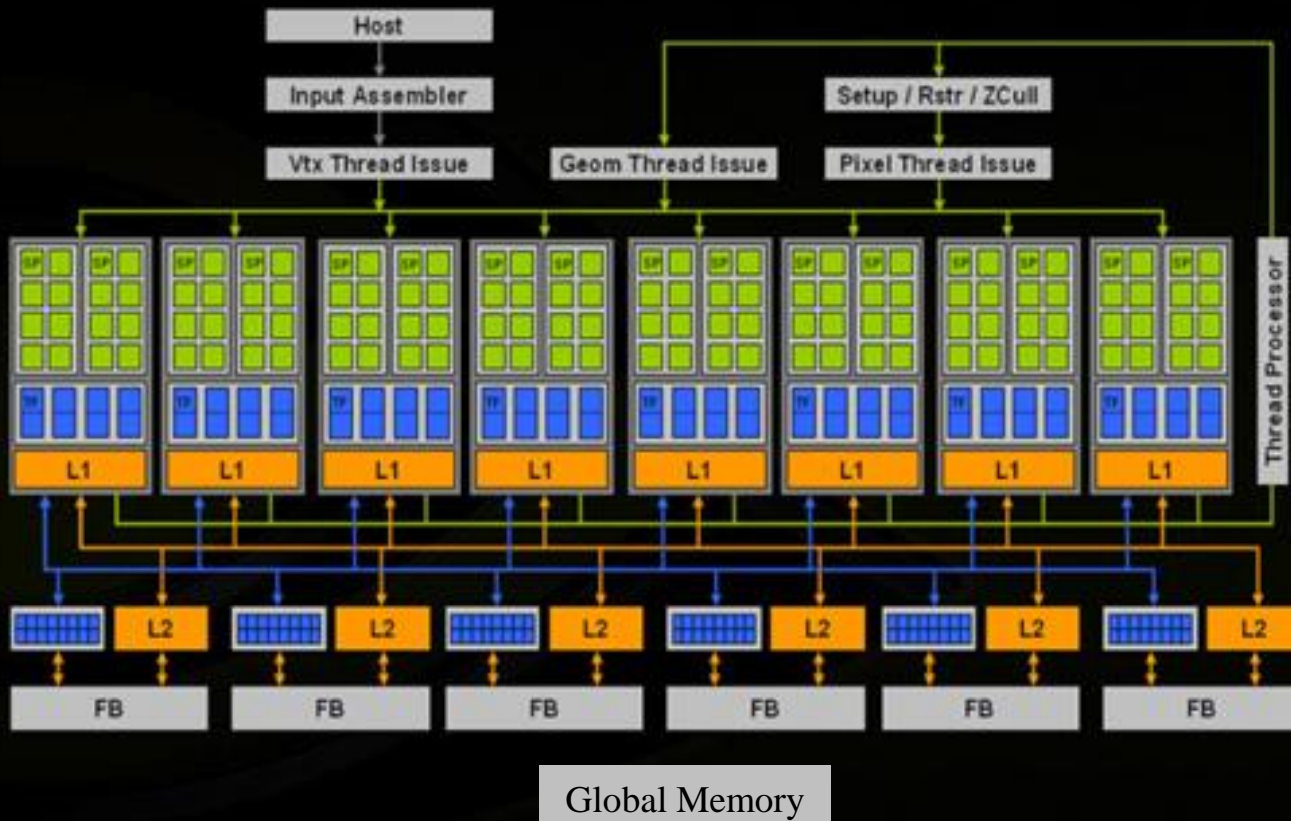


In many cases speedups of 1-2 orders of magnitude can be obtained by porting to GPU

- more details on the rules for effective porting later

GPU Architecture: Overview

GeForce 8800 GTX Block Diagram



128 processors \rightarrow 8 multi-processors of 16 processors each

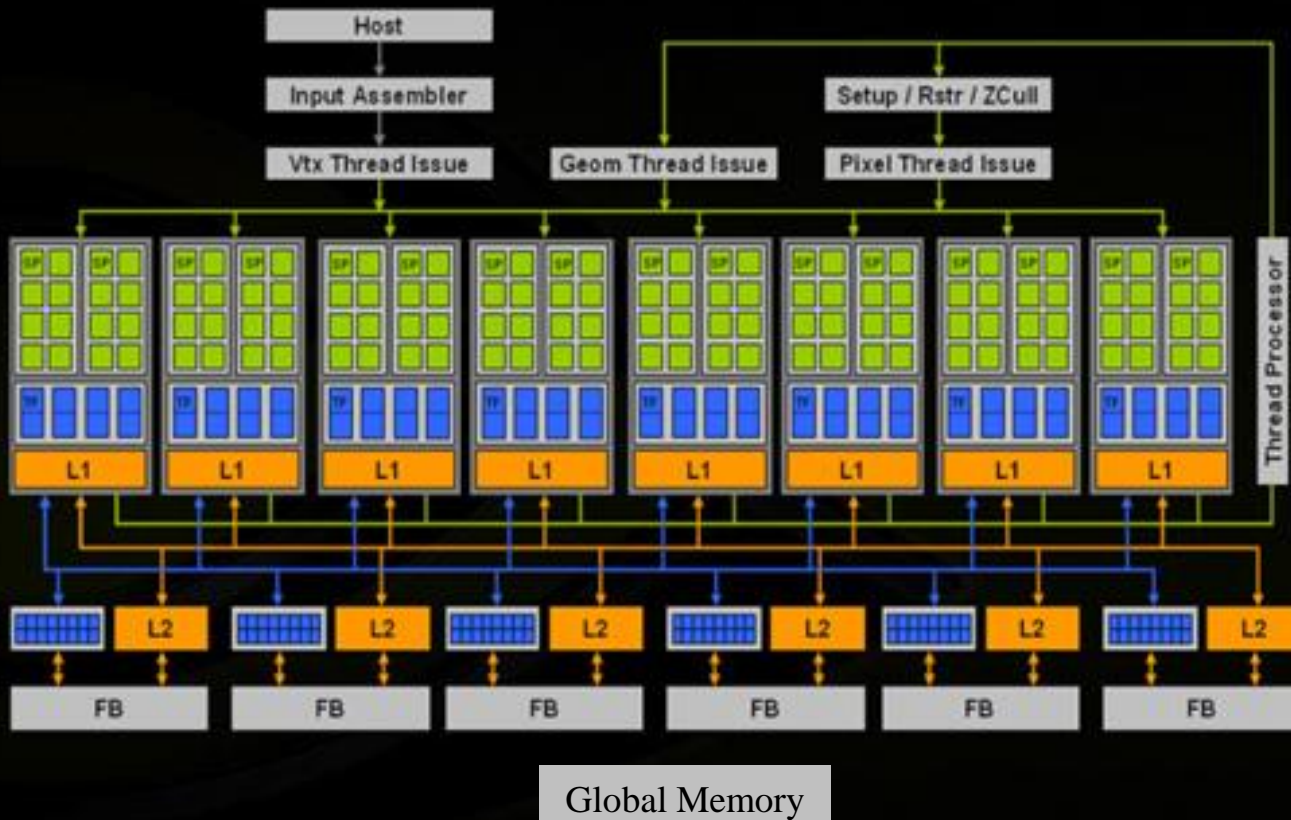
local cache L1 (4k)

shared cache L2 (1M)

DRAM (global memory)

GPU Architecture: Overview

GeForce 8800 GTX Block Diagram



Memory management
is key!

Thread management is
key!

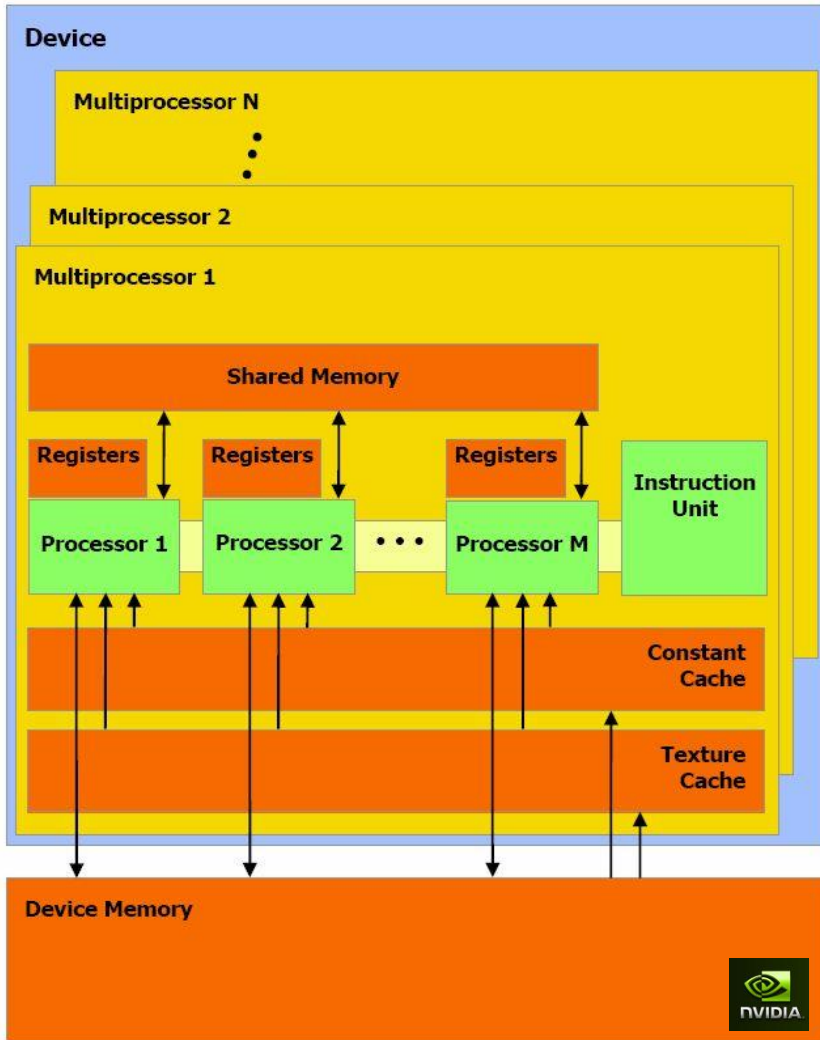
128 processors → 8
multi-processors of
16 processors each

local cache L1 (4k)

shared cache L2
(1M)

DRAM (global
memory)

GPU Architecture: Different View



each multiprocessor is a SIMT (Same Instruction, Multiple Thread) architecture equipped with a set of local 32-bit registers (L1 and L2 caches)

the (multi-processor level) shared Constant Cache and Texture Cache are read-only

the (device-level shared) Device Memory (Global Memory) has read-write access (with caching soon)

GPU Specifics

All standard graphics ops are hardwired

- linear interpolations
- matrix and vector arithmetic (+, -, *)

Arithmetic intensity

- the ratio of ALU arithmetic per operand fetched
- needs to be reasonably high, else application is memory-bound

GPU memory 1-2 orders of magnitude slower than GPU processors

- computation often better than table look-ups
- indirections can be expensive

Be aware of GPU 2D-caching protocol (for texture memory)

- data is fetched in 2D tiles (recall graphics bilinear texture filtering)
- promote data locality in 2D tiles

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

Programmability

GPU hardware can be programmed with

- shading languages (NVIDIA CG, OpenGL GLSL, Microsoft HLSL)
- parallel programming language (CUDA, OpenCL)

Shading languages

- require computer graphics knowledge
- give access to all fixed function pipelines (fast, ASIC-accelerated)
 - texturing: data interpolation, filtering
 - rasterization: mapping into the data domain
 - culling: clipping, early thread removal (early fragment kill)
- this can provide performance benefits

Parallel programming languages

- ease programming, eliminate need to study (some) graphics

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

- Single Instruction Multiple Threads
- 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

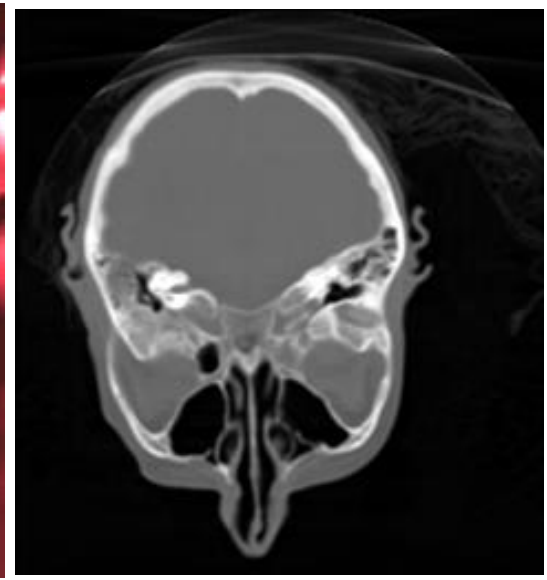
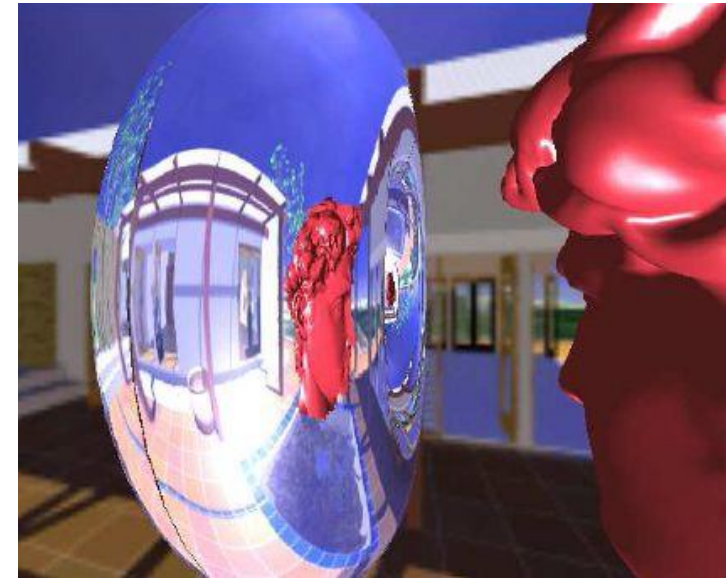
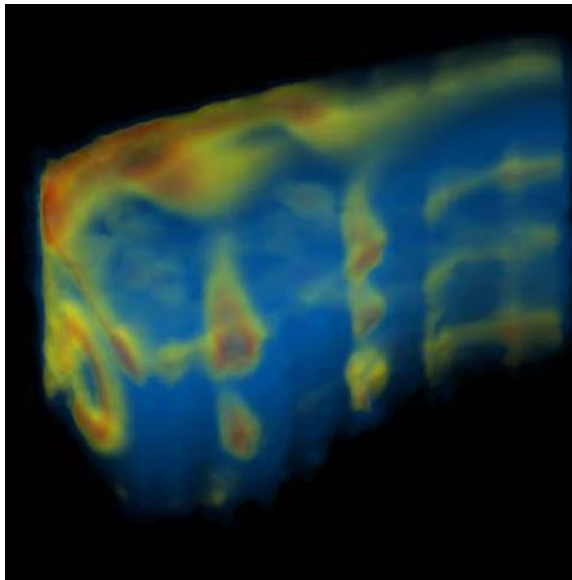
GPGPU = **G**eneral **P**urpose Computation on Graphics hardware (**GPU**)

- 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



Organization

Will have 4-5 lab projects

- practice what you have learned in class

Either use your own PC or lab machines

- ideally use the lab server

Final project

- there will be a larger final project
- choose from a number of topics or choose your own
- ideally develop some multi-GPU application