GPUs: Engines for Future High-Performance Computing

John Owens

Assistant Professor, Electrical and Computer Engineering

Institute for Data Analysis and Visualization

University of California, Davis

GPUs as Compute Engines

10 years ago:

- Graphics done in software

5 years ago:

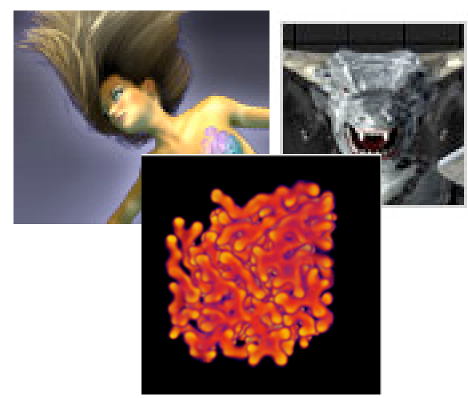
- Full graphics pipeline

Today:

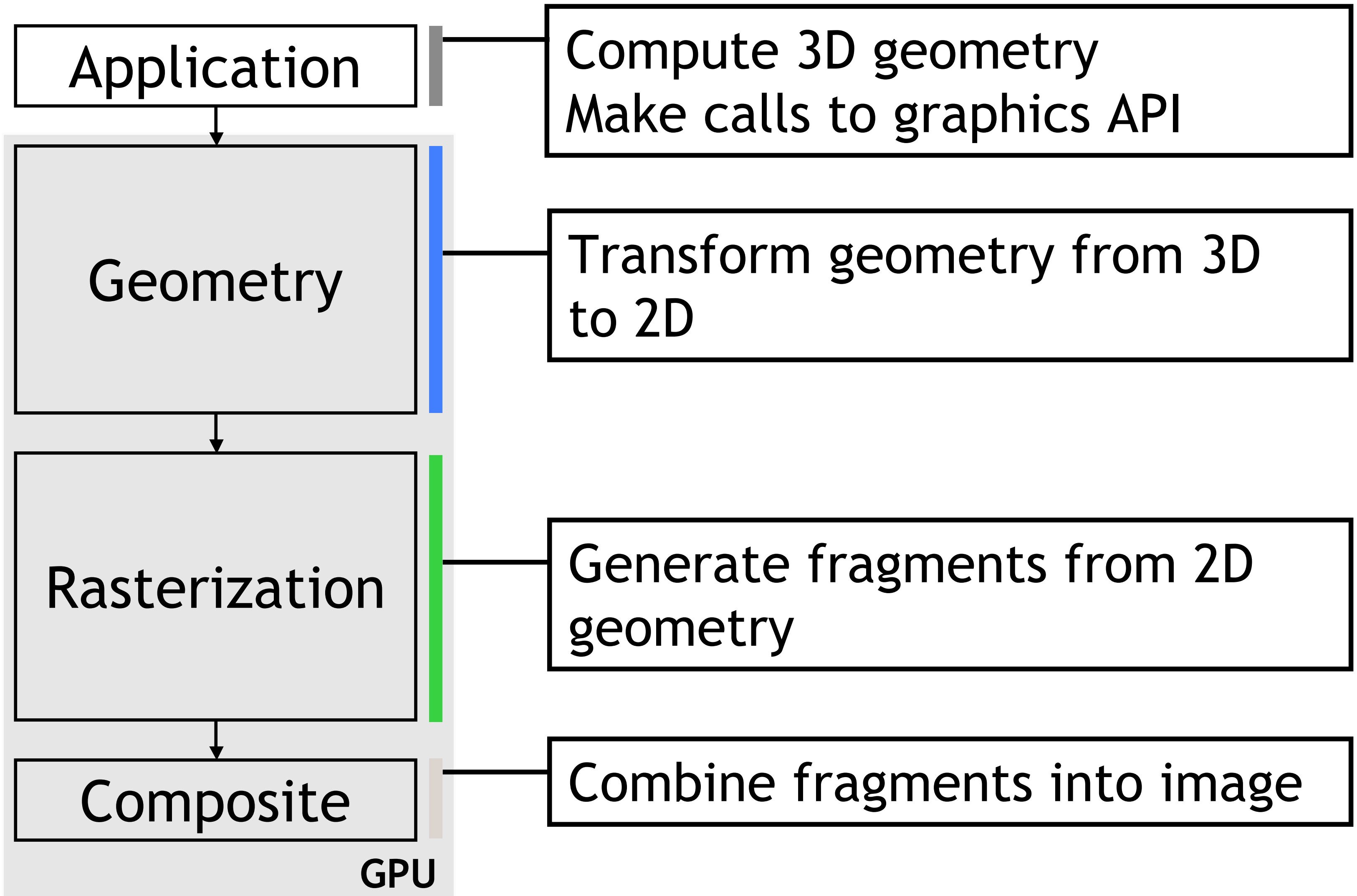
- 40x geometry, 13x fill vs. 5 yrs ago
- Programmable!

Programmable, data parallel processing on every desktop

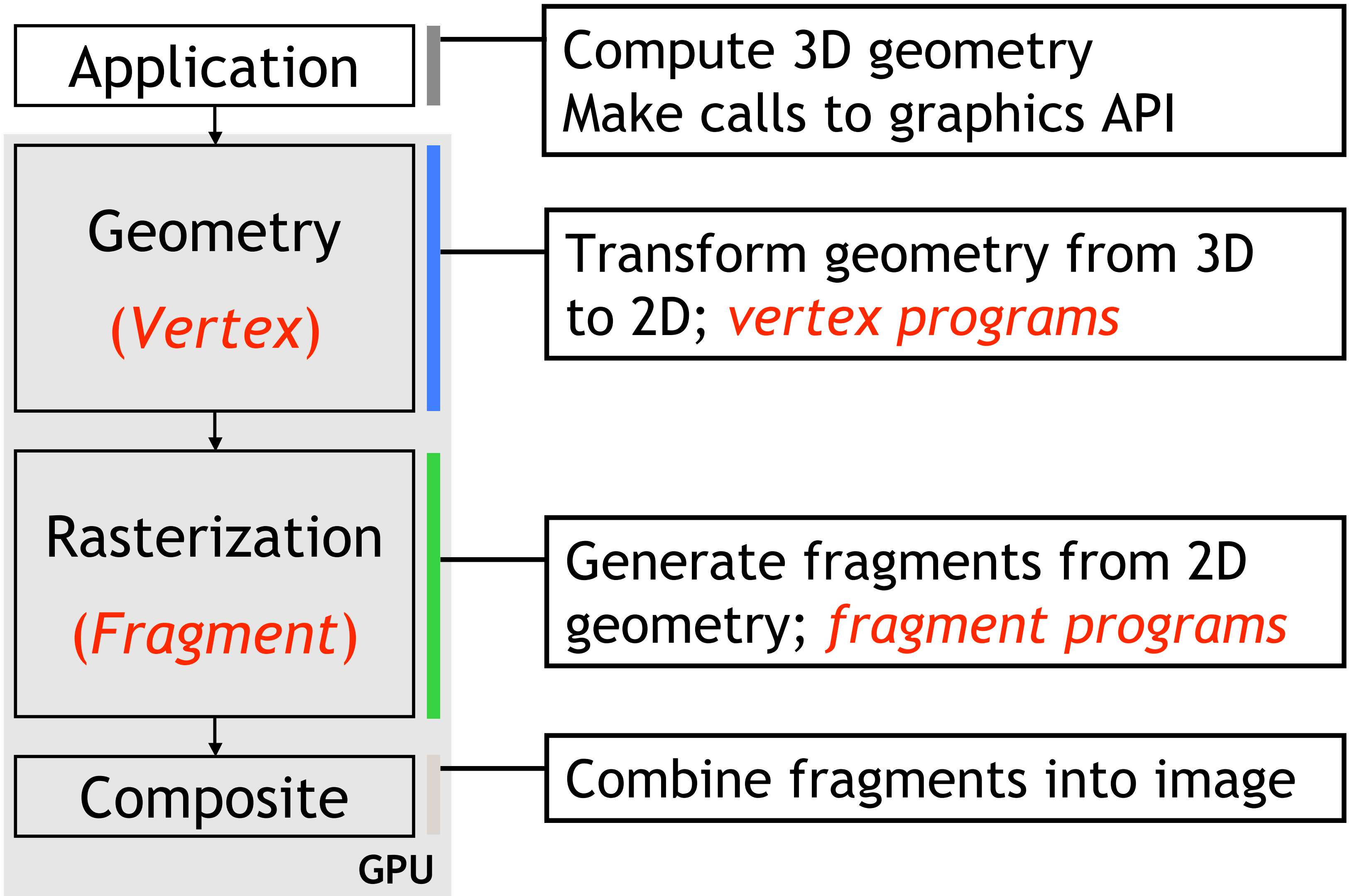
Enormous opportunity to change the way commodity computing is done!



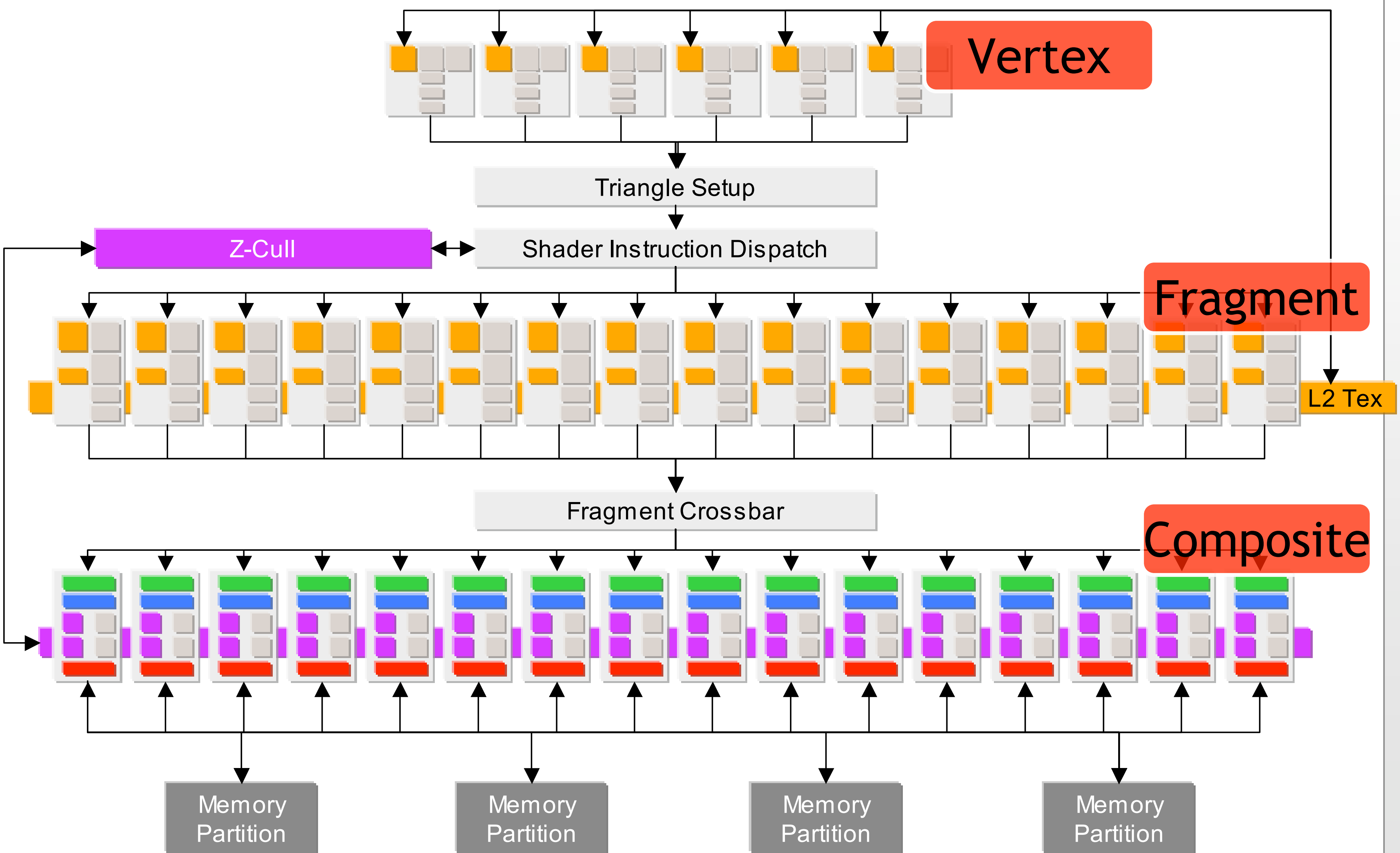
The Rendering Pipeline



The Programmable Rendering Pipeline

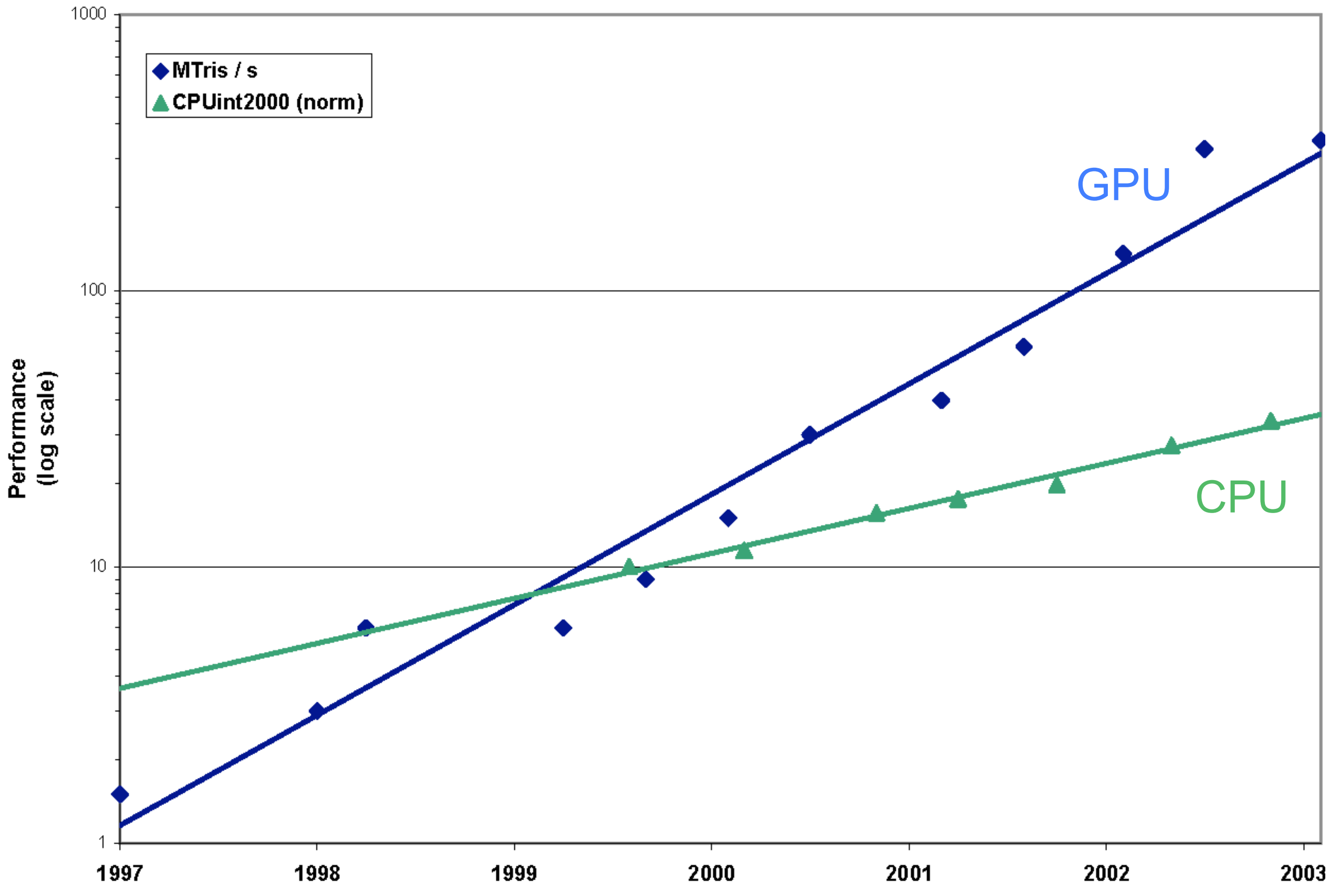


NVIDIA GeForce 6800 3D Pipeline



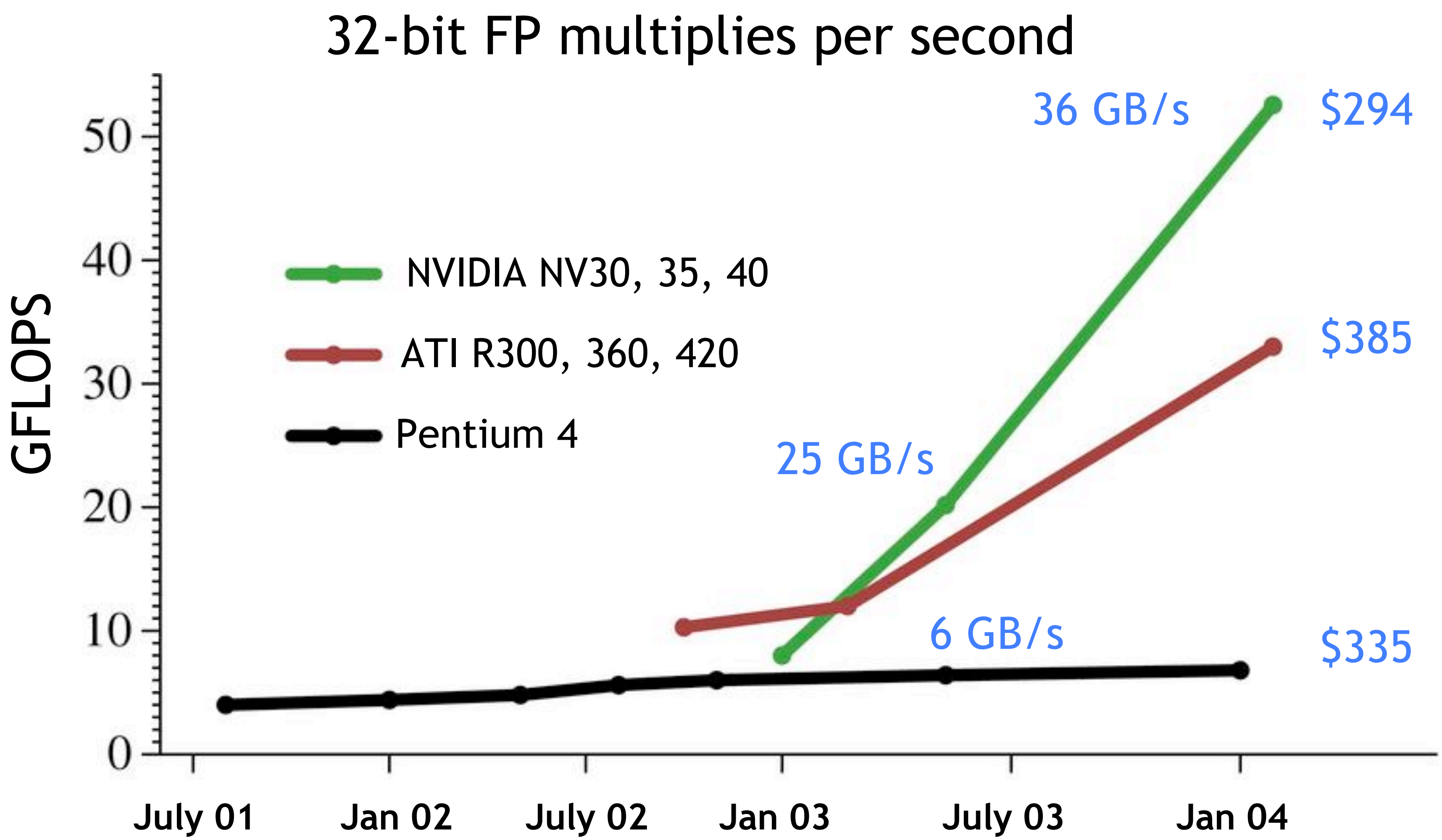
Courtesy Nick Triantos, NVIDIA

Long-Term Trend: CPU vs. GPU



Courtesy Naga Govindaraju

Recent GPU Performance Trends



Courtesy Pat Hanrahan/David Luebke

Why Are GPUs Fast?

Characteristics of computation permit efficient hardware implementations

- High amount of parallelism …
- … exploited by graphics hardware
- High latency tolerance and feed-forward dataflow …
- … allow very deep pipelines
- … allow optimization for bandwidth not latency

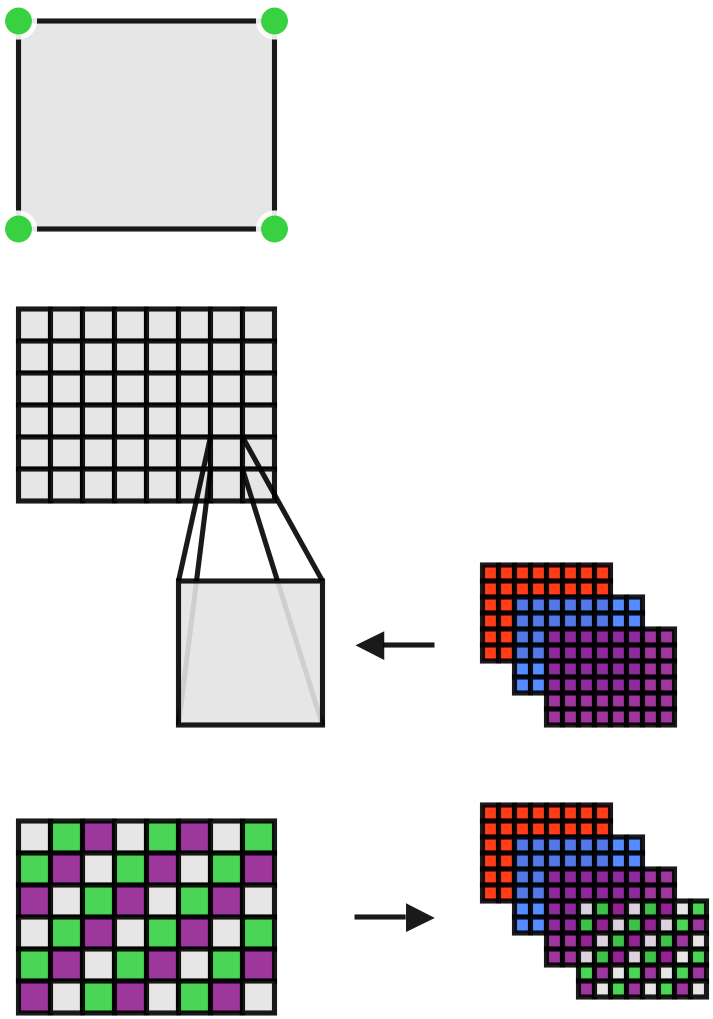
Simple control

- Restrictive programming model

Competition between vendors

What about programmability? Effect on performance? How hard to program?

Programming a GPU for GP Programs



- Draw a screen-sized quad
- Run a SIMD program over each fragment
- “Gather” is permitted from texture memory
- Resulting buffer can be treated as texture on next pass

GPU Programming is Hard

Must think in graphics metaphors

Requires parallel programming (CPU-GPU, task, data, instruction)

Restrictive programming models and instruction sets

Primitive tools

Rapidly changing interfaces

Challenge: Programming Systems

Programming Model	High-Level Abstractions/ Libraries	Low-Level Languages	Compilers

Performance Analysis Tools

Docs

CPU	GPU
Scalar	Stream?
STL, GNU SL, MPI, …	-
C, Fortran, …	GLSL, Cg, HLSL, …
gcc, vendor-specific, …	Vendor-specific
gdb, vtune, Purify, …	Shadesmith, NVPerfHUD
Lots	None
→ *applications*	→ *kernels*

Brook: General-Purpose Streaming Language

Stream programming model

- Treats GPU as streaming coprocessor
- Streams enforce data parallel computing
- Kernels encourage arithmetic intensity
- Streams and kernels explicitly specified

C with stream extensions

Open-source: www.sf.net/projects/brook/

Ian Buck et al., "Brook for GPUs: Stream Computing on Graphics Hardware", Siggraph 2004

Challenge: GPU-to-Host Bandwidth

GPUs lack band-width to the host, so we won't use it!

No one uses host bandwidth, so we won't optimize it!

PCI EXPRESS

- **PCI-E optimizes GPU-to-CPU bandwidth**
  - 16-lane card: 8 GB/s
  - Scalable in future
- **Major vendors support PCI-E cards now**
- **Multiple GPUs supported per CPU - opportunity!**
  - Cheap and upgradable

Challenge: Mobile/embedded market

Why?

- UI, messaging/screen savers, navigation, gaming (location based)

Typical specs (cell-phone class):

- 200-800k gates, ~100 MHz, ~100 mW
- 1-10M vtx/s, 100+M frags/s

What’s important?

- Visual quality
- Power-efficient (ops/W)
  - Avoid memory accesses, unified shaders …
- Low cost

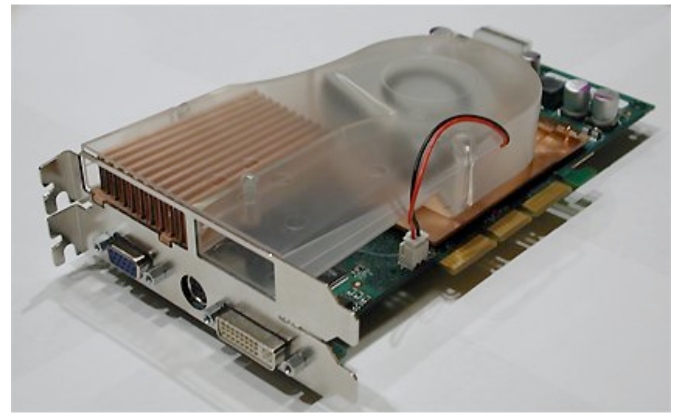
Challenge: Power

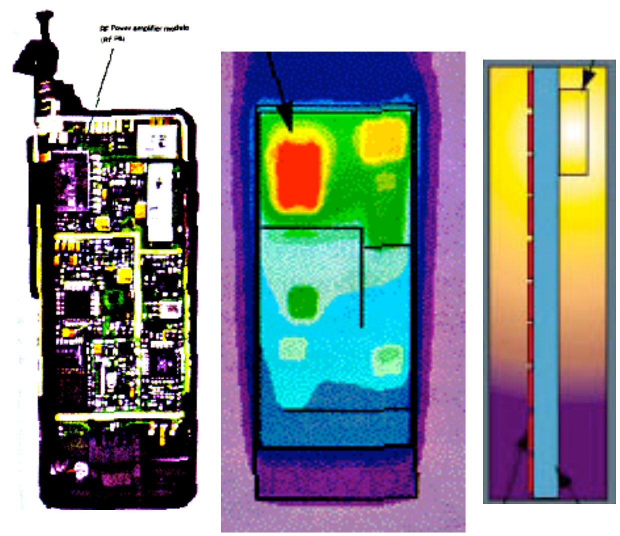
Desktop:

- Double-width cards
- Workstation power supplies; draw power from motherboard

Mobile:

- Batteries improving 5-10% per year
- Ops/W most important

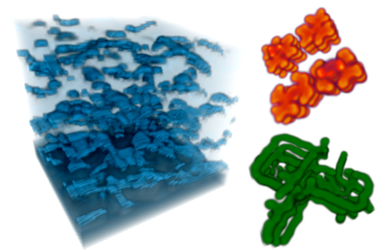


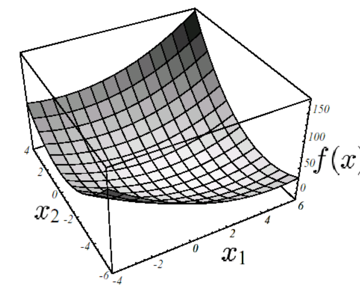


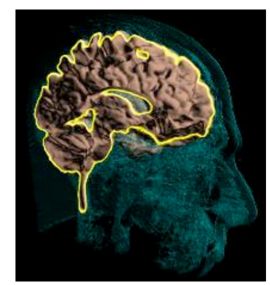
www.coolingzone.com

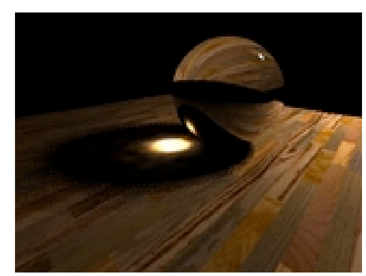
Current GPGPU Research

- Image processing [Johnson/Frank/Vaidya, LLNL]
- Alternate graphics pipelines [Purcell, Carr, Coombe]
- Visual simulation [Harris]
- Volume rendering [Kniss, Krüger]
- Level set computation [Lefohn, Strzodka]
- Numerical methods [Bolz, Krüger, Strzodka]
- Molecular dynamics [Buck]
- Databases [Sun, Govindaraju]
- …









Grand Challenges

Architecture: Increase features and performance without sacrificing core mission

Interfaces: Abstractions, APIs, programming models, languages

- *Many* approaches needed
- Goal: C programs compiling to dynamically-balanced CPU-GPU clusters
- Academic and research community

Applications: Killer app needed!

Acknowledgements

Craig Lund: Mercury Computer Systems

Jeremy Kepner: Lincoln Labs

Nick Triantos, Craig Kolb: NVIDIA

Mark Segal: ATI

Kari Pulli: Nokia

Aaron Lefohn: UC Davis

Ian Buck: Stanford

Funding: DOE Office of Science, Los Alamos National Laboratory, ChevronTexaco, UC MICRO, UC Davis

For more information …

GPGPU home: http://www.gpgpu.org/

- Mark Harris, UNC/NVIDIA

***GPU Gems* (Addison-Wesley)**

- Vol 1: 2004; Vol 2: 2005

Conferences: Siggraph, Graphics Hardware, GP²

- Course notes: Siggraph '04, IEEE Visualization '04

University research: Caltech, CMU, Duisberg, Illinois, Purdue, Stanford, SUNY Stonybrook, Texas, TU München, Utah, UBC, UC Davis, UNC, Virginia, Waterloo