GPGPU

General Purpose computation on Graphics Processing Units

Mark Harris, 2002

Perform demanding calculations on the GPU instead of the CPU!

At first, appeared to be a wild idea, but is now a very serious technology! Results were highly varied in the early years, but the GPU advantage has grown bigger and bigger.

GPGPU using shaders

Has less attention now, due to CUDA.

Still interesting:

- Apart from some reuable standard code, it is not very complicated.
- Portable to most GPUs with no extra software.
- Excellent performance.

Key components of the GPGPU trend

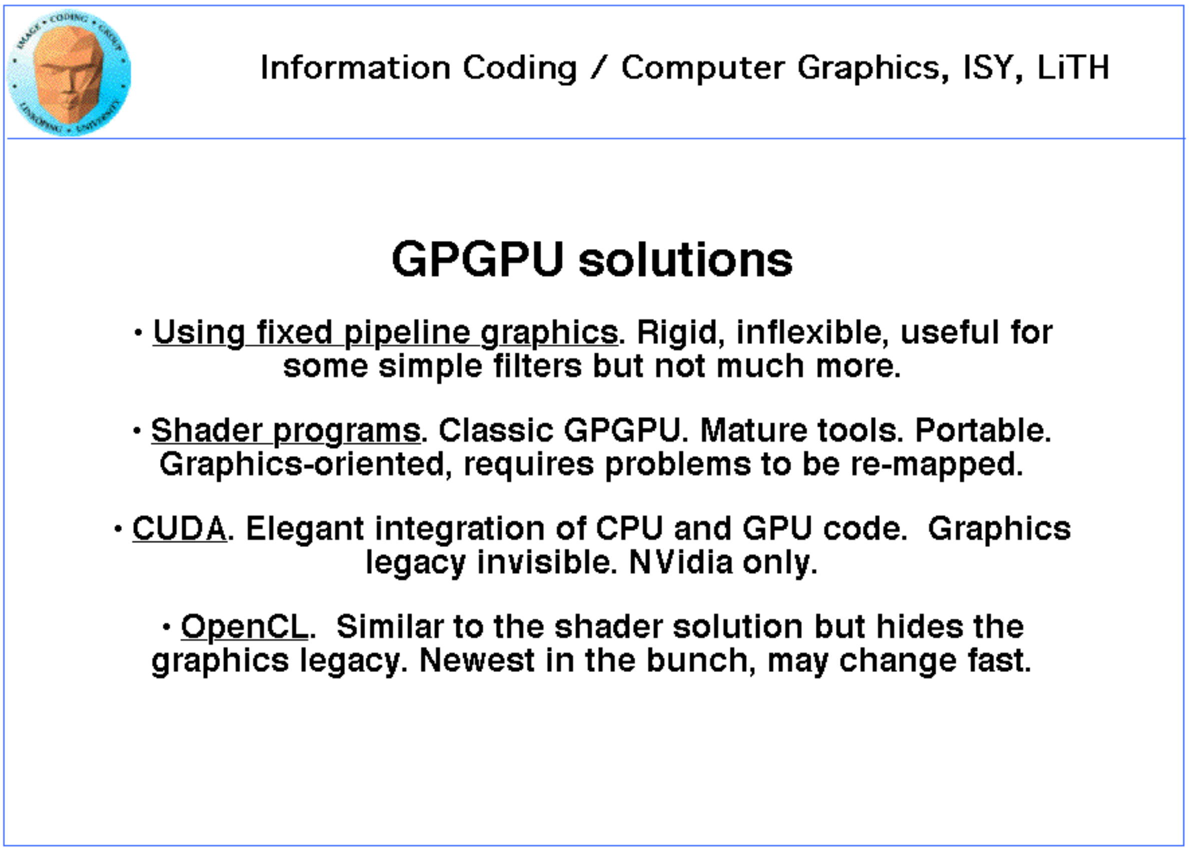
High processing power in parallel

Programmability: Introduction of shader programs, much more flexible, programmable for any problem.

Floating-point buffers: Vital! Initially with poor precision. 32-bit floating-point decent... but not really impressive.

GPGPU solutions

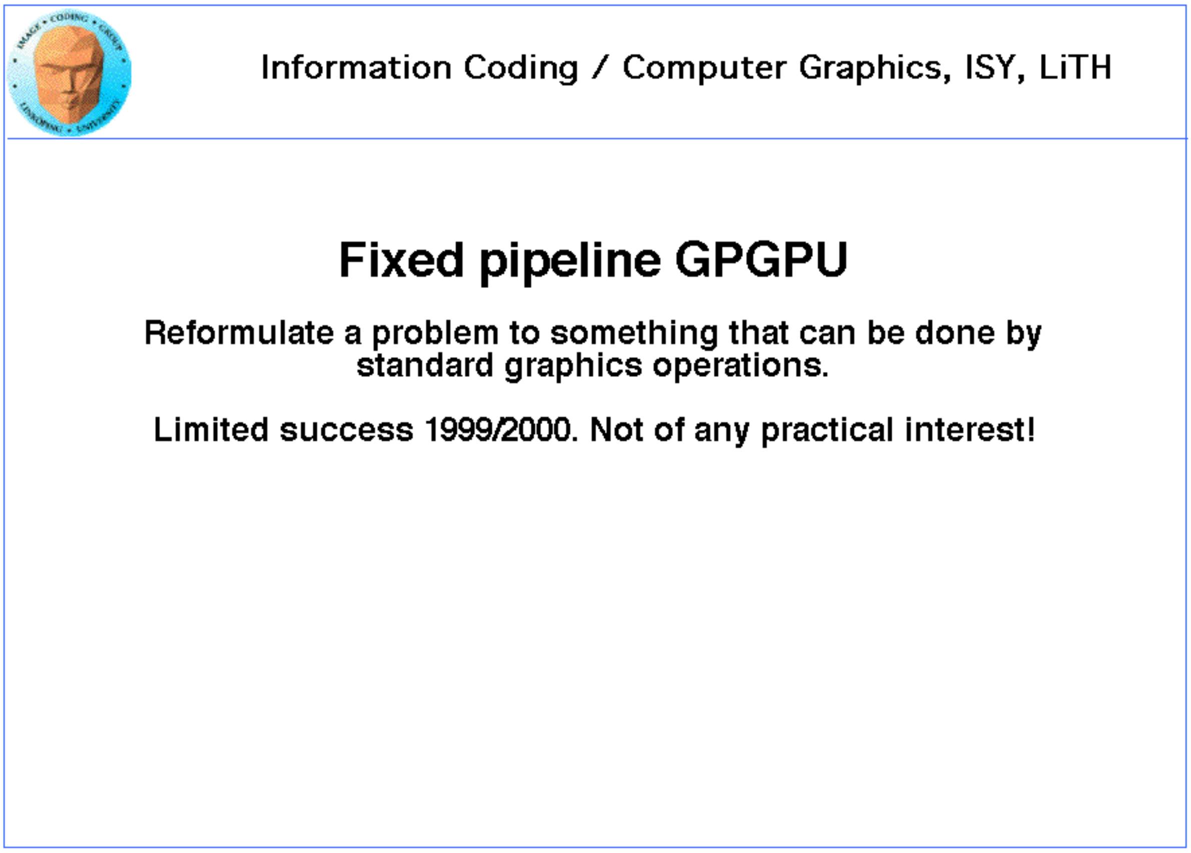
- Using fixed pipeline graphics. Rigid, inflexible, useful for some simple filters but not much more.
- Shader programs. Classic GPGPU. Mature tools. Portable. Graphics-oriented, requires problems to be re-mapped.
- CUDA. Elegant integration of CPU and GPU code. Graphics legacy invisible. NVidia only.
- OpenCL. Similar to the shader solution but hides the graphics legacy. Newest in the bunch, may change fast.



Information Coding / Computer Graphics, ISY, LiTH

GPGPU solutions

- Using fixed pipeline graphics. Rigid, inflexible, useful for some simple filters but not much more.
- Shader programs. Classic GPGPU. Mature tools. Portable. Graphics-oriented, requires problems to be re-mapped.
- CUDA. Elegant integration of CPU and GPU code. Graphics legacy invisible. NVidia only.
- OpenCL. Similar to the shader solution but hides the graphics legacy. Newest in the bunch, may change fast.



Information Coding / Computer Graphics, ISY, LiTH

Fixed pipeline GPGPU

Reformulate a problem to something that can be done by standard graphics operations.

Limited success 1999/2000. Not of any practical interest!

Shader-based GPGPU

Portable! Most GPUs can use shaders, no need for extra software, run using standard software/drivers.

All modern shader languages (GLSL, Cg, HLSL) are similar and easy to program in.

Requires a re-mapping of data to textures.

Very good results already in 2005: 8x speedups overall reported!

CUDA-based GPGPU

Only works on NVidia hardware.

Requires extra software - which isn't very elegant.

Nice integration of CPU and GPU code in the same program.

Excellent results! 100x speedups are common - before optimizing! Even low-end GPUs give significant boosts.

OpenCL-based GPGPU

Works on various hardware - not only GPUs.

Developed by Khronos Group, pushed by Apple.

Harder to get started, software looks pretty much like programming shaders.

Use the source, Luke!

Three trivial examples:

Hello World! for CUDA

Hello World! for OpenCL

Hello World for GLSL

So what GPU should you get?

For CUDA, go for Fermi boards!

GTX560 good middle-range board, best Fermi price/performance and reasonable power consumption.

GTX570 also good choice!

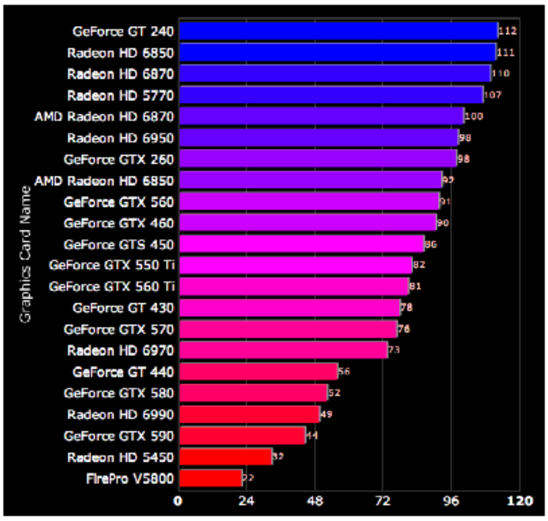
GTX590, expensive but powerful - and hot

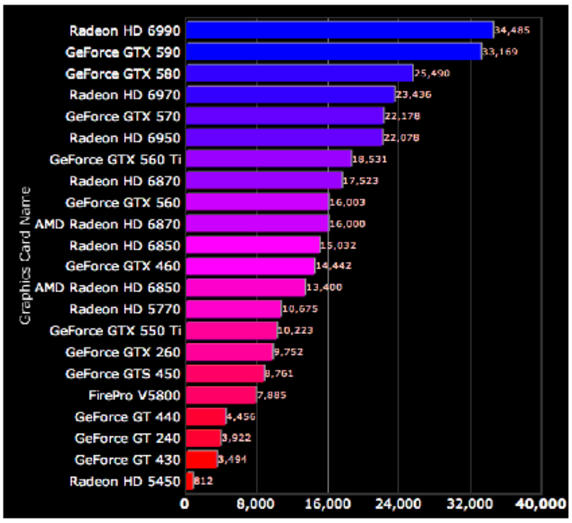
GT240 cheap low-power board (not Fermi)

Avoid overclocked boards!

Don't bother with "professional" Quadro boards.

AMD is DEFINTELY an option for shaders or OpenCL but not CUDA.





Price/performance: old GT240 on top, 260 and 560 good, top performers low

Performance: GTX590 and Radeon 6990 on top

Good on both charts: 560/570 and several good Radeon boards (6870 etc) 570 and 6850/6970 on "sweet spot"?

Power/performance not included (and it should)

In the lab

GTS250

Not the latest and greatest but 128 cores and modern enough for most our purposes.

That’s all, folks!

Next friday: Introduction to CUDA