Topics

GPUs

Graphics co-processors have grown immensely in power in the last few years
Graphics co-processors have grown immensely in power in the last few years.

Originally intended to offload graphical work from the main CPU they have become recognised as powerful processors in their own right and people have tried to tap into their potential.
Graphics co-processors have grown immensely in power in the last few years.

Originally intended to offload graphical work from the main CPU they have become recognised as powerful processors in their own right and people have tried to tap into their potential.

General-Purpose computing on Graphics Processing Units (GPGPU) has emerged as an important example of parallel processing.
Graphics co-processors have grown immensely in power in the last few years.

Originally intended to offload graphical work from the main CPU they have become recognised as powerful processors in their own right and people have tried to tap into their potential.

*General-Purpose computing on Graphics Processing Units* (GPGPU) has emerged as an important example of parallel processing.

So hardware, originally intended to support gamers, is now being used in general purpose computations.
Topics

GPUs

Graphics co-processors have grown immensely in power in the last few years.

Originally intended to offload graphical work from the main CPU they have become recognised as powerful processors in their own right and people have tried to tap into their potential.

General-Purpose computing on Graphics Processing Units (GPGPU) has emerged as an important example of parallel processing.

So hardware, originally intended to support gamers, is now being used in general purpose computations.

GPU-based computing appears strongly in the Top 500 largest computers in the world.
GPUs naturally do certain things very well: in particular data-parallel pixel rendering (colouring, shading and so on)
GPUs naturally do certain things very well: in particular, data-parallel pixel rendering (colouring, shading and so on).

The computations you typically do on pixels can be quite intensive, but are fairly restricted in nature.
GPUs naturally do certain things very well: in particular data-parallel pixel rendering (colouring, shading and so on)

The computations you typically do on pixels can be quite intensive, but are fairly restricted in nature

And the data-parallel nature of the computations on the millions of pixels on your screen is very relevant
GPUs naturally do certain things very well: in particular data-parallel pixel rendering (colouring, shading and so on)

The computations you typically do on pixels can be quite intensive, but are fairly restricted in nature

And the data-parallel nature of the computations on the millions of pixels on your screen is very relevant

Over time GPUs became more and more programmable as they needed to do more and more complex manipulations
GPUs naturally do certain things very well: in particular data-parallel pixel rendering (colouring, shading and so on)

The computations you typically do on pixels can be quite intensive, but are fairly restricted in nature

And the data-parallel nature of the computations on the millions of pixels on your screen is very relevant

Over time GPUs became more and more programmable as they needed to do more and more complex manipulations

Graphics libraries (like OpenGL and DirectX) that were originally developed to draw pictures now supported programmable sequences of operations via shader languages such as GLSL and HLSL
So people soon realised that GPUs are powerful multicore SIMD processors, but just tuned for certain intensive data-parallel computations.
So people soon realised that GPUs are powerful multicore SIMD processors, but just tuned for certain intensive data-parallel computations.

GPU companies like NVIDIA and AMD/ATI have seen the possibilities of using this power and now put hardware into their GPUs specifically to help GPGPU computations.
So people soon realised that GPUs are powerful multicore SIMD processors, but just tuned for certain intensive data-parallel computations.

GPU companies like NVIDIA and AMD/ATI have seen the possibilities of using this power and now put hardware into their GPUs specifically to help GPGPU computations.

This means putting in hardware to support generic computation, not just graphics oriented stuff.
And NVIDIA have also produced a language, *Compute Unified Device Architecture* (CUDA), to aid in the general programming of these devices.
And NVIDIA have also produced a language, *Compute Unified Device Architecture* (CUDA), to aid in the general programming of these devices.

There is also an open standard, *Open Computing Language* (OpenCL), that is not vendor based.
And NVIDIA have also produced a language, *Compute Unified Device Architecture* (CUDA), to aid in the general programming of these devices.

There is also an open standard, *Open Computing Language* (OpenCL), that is not vendor based.

CUDA is quite popular right now, but only runs on Nvidia cards.
And NVIDIA have also produced a language, *Compute Unified Device Architecture* (CUDA), to aid in the general programming of these devices.

There is also an open standard, *Open Computing Language* (OpenCL), that is not vendor based.

CUDA is quite popular right now, but only runs on Nvidia cards.

OpenCL is strong, and is supported by NVIDIA, AMD, Intel and ARM amongst others.
CUDA looks a lot like C and C++
CUDA looks a lot like C and C++

Dangerously close, as there are several important differences between CUDA and these languages
CUDA looks a lot like C and C++

Dangerously close, as there are several important differences between CUDA and these languages

CUDA is a modified C/C++ with a *syntactic* addition to notate parallel execution and various semantic additions to support parallelism
CUDA looks a lot like C and C++

Dangerously close, as there are several important differences between CUDA and these languages

CUDA is a modified C/C++ with a *syntactic* addition to notate parallel execution and various semantic additions to support parallelism

OpenCL is a library that runs on plain C or C++ (and any other language that can call C functions)
A GPU has several *multiprocessors* each containing a bunch of SIMD cores: thus a GPU is a MIMD of SIMD
A GPU has several *multiprocessors* each containing a bunch of SIMD cores: thus a GPU is a MIMD of SIMD

It works best when there are thousands of threads, even if there are only hundreds of cores
A GPU has several *multiprocessors* each containing a bunch of SIMD cores: thus a GPU is a MIMD of SIMD.

It works best when there are thousands of threads, even if there are only hundreds of cores.

This is to overlap communications with computation: a core that would be waiting for some data can pick up another thread and work on it instead on doing nothing.
A GPU has several *multiprocessors* each containing a bunch of SIMD cores: thus a GPU is a MIMD of SIMD.

It works best when there are thousands of threads, even if there are only hundreds of cores.

This is to overlap communications with computation: a core that would be waiting for some data can pick up another thread and work on it instead on doing nothing.

Memory access is *very slow*, so there would be a lot of waiting otherwise.
Threads in a GPU are hardware managed and extremely lightweight, meaning they have tiny creation and scheduling overhead.
Threads in a GPU are hardware managed and extremely lightweight, meaning they have tiny creation and scheduling overhead.

Thus there is no need to worry about making and destroying large numbers of threads.
Threads in a GPU are hardware managed and extremely lightweight, meaning they have tiny creation and scheduling overhead.

Thus there is no need to worry about making and destroying large numbers of threads.

Very different from normal CPU threads.
Threads in a GPU are hardware managed and extremely lightweight, meaning they have tiny creation and scheduling overhead.

Thus there is no need to worry about making and destroying large numbers of threads.

Very different from normal CPU threads.

Exercise. Why don’t normal CPUs do the same: have hardware support for threads? Find out.