Parallel Computing  
CM30225

Russell Bradford

2022/23

### 1. GPUs

And there are many others. For example, Google provided *Renderscript* for Android (now being replaced by Vulkan Compute)

Coming from the low-power, mobile end of the world, it provides a C-like language and a CUDA-like API for programming GPUs “close to the metal” (high performance)

It compiles the program to a device-independent form (LLVM), which is then compiled to machine code and optimised for the device at run-time

At run-time, it decides where to run the code, on the CPU or on the GPU: thus all code can run everywhere, falling back to the CPU if the GPU is not up to it (or there is no GPU!)

### 2. GPUs

The is also *OpenGL Shading Language* (GLSL) that allows you to write chunks of C-like code embedded in the graphics library OpenGL

For the die-hards, there is also an assembly language for programming the graphics pipeline

Coming along is *WebGPU*, a JavaScript API for graphics and compute, providing a uniform Web interface to whatever is running underneath

### 3. GPUs

Microsoft have their own versions of everything, of course

Their *DirectCompute* is not a million miles from CUDA, but is based on their DirectX suite

It runs on Nvidia and AMD cards

But the portability to other operating systems is an open question

They also have C++ Accelerated Massive Parallelism (C++ AMP), an annotated version of C++ that is reputedly much easier to write code for

This is more like an OpenMP for GPUs

### 4. GPUs

#### OpenACC

In fact, there is also OpenACC, which is essentially OpenMP for GPUs

pragma annotations indicate code can be run on a GPU

#pragma acc kernels  
{  
 for (int i = 0; i < n; ++i)  
 z[i] = x[i] + y[i];  
}

runs the loop on the GPU. The programmer does not have to think about copying data back and forth or writing and calling kernels

**Exercise** Is that a good or a bad thing?

### 5. GPUs

#### OpenACC

Similar to other systems, simply ignoring the pragma and running on the CPU will produce equivalent results

OpenACC does for accelerators (co-processors) what OpenMP does for multi-core

OpenMP and OpenACC pragmas can sit side-by-side in the same code

In fact, OpenACC is supposed to merge with OpenMP at some point, but progress seems slow

A freely available version of OpenACC on Nvidia GPUs is available and GCC now also supports it: this may help OpenACC to become more popular

### 6. GPUs

GPUs have a great future ahead of them as they are excellent at certain kinds of problem, when programmed by really good programmers

There are CUDA bindings for Python and Java (of course), so you don’t have to use C

Another item to note is that GPUs use (relatively) very little energy for the amount of processing they deliver

In a world where supercomputer centres spend more on electricity than they do on the computers themselves, the operations per watt that GPUs provide turns out to be very attractive

See the current Top 500

### 7. GPUs

If we were starting from scratch, we probably wouldn’t design a GPU in the way it is

Just like the original CPU was based on existing integrated circuits that engineers noticed could be made programmable, the GPU is based on graphics co-processors that engineers noticed could be made programmable

So the accidents of history brought us to where we are today

### 8. GPUs

As previously mentioned, we are currently seeing multicore processors merging with GPUs

This is repeating the historical precedents of coprocessors merging with main processors

One processor that has had a lot of attention recently is the Apple M1

### 9. Coprocessors

#### Apple M1

The Apple M1 is a 4CPU+4CPU+GPU+NPU ARM architecture on a chip sharing memory and cache: a *unified memory* architecture

* 4 fast CPU cores
* 4 energy efficient CPU cores
* 8 GPU cores (2500 threads)
* 16 core neural processing unit (NPU) (11 trillion ops/sec)
* a digital signal processor (DSP)
* an image processing unit (ISP)
* a video encoder/decoder

This takes coprocessing to new levels!

### 10. Coprocessors

#### Apple M1

The various units share memory in the unified memory architecture

This is 16GB memory, on the same chip

Note that on-chip memory is fast(er), but not expandable

Advanced **Exercise** Read about the memory consistency features used by the M1 to support compatability with x86 code

### 11. Coprocessors

Incidentally, Intel has its *Gaussian and Neural Accelerator* (GNA) integrated into the main CPU chip

Initially for support of speech recognition, it could probably be used for more general deep learning

### 12. ARM Mali

At the low-power end of the scale, ARM have their Mali core

“Core” in the sense of a chunk of silicon design that can be incorporated into other system chips

With current generations having up to 32 processing cores, this is a GPU you will find in your phone

It supports OpenCL and 64-bit floating point

As well as doing some graphics…