Lecture 0: CPUs and GPUs

Prof. Mike Giles

mike.giles@maths.ox.ac.uk

Oxford University Mathematical Institute

Oxford e-Research Centre

Money – root of all innovation

Money drives computing, as much as technology.

If there's a big enough market, someone will develop the product.

Need huge economies of scale to make chips cheaply, so very few companies and competing products.

To anticipate computing trends, look at market drivers, key applications.

CPUs

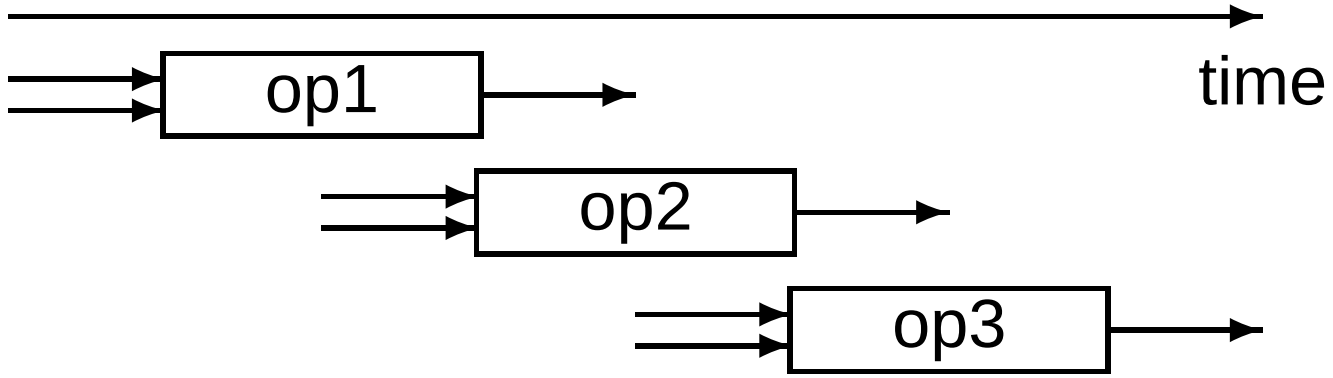
- chip size (amount of circuitry) continues to doubles every 18-24 months (Moore's Law)
- similar growth in other hardware aspects, but memory bandwidth struggles to keep up
- safe to assume that this will continue for at least the next 10 years, driven by:
  - multimedia applications (streaming video, HD)
  - image / video processing
  - "intelligent" software
  - internet applications (Google, Bing, Facebook)

CPU Parallelism

- instruction parallelism (e.g. addition)
- pipeline parallelism, overlapping different instructions
- multiple pipelines, each with own capabilities
- multiple cores (CPUs) within a single chip
- multiple chips within a single shared-memory computer
- multiple computers within a distributed-memory system
- multiple systems within an organisation

Ideal Von Neumann Processor

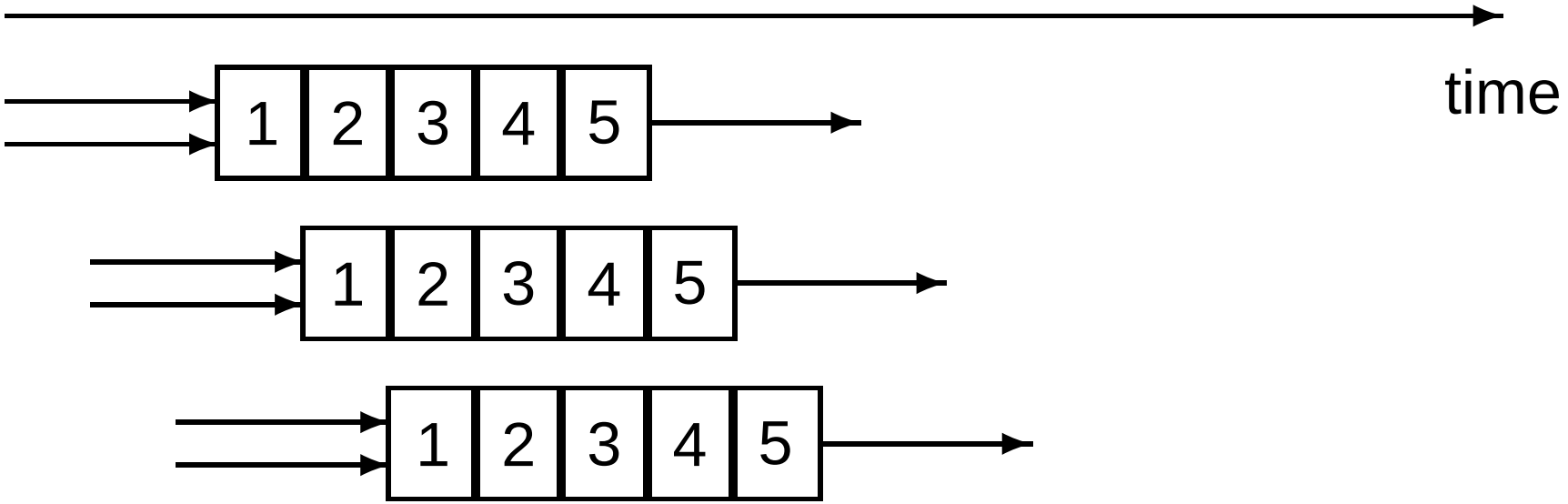
- each cycle, CPU takes data from registers, does an operation, and puts the result back
- load/store operations (memory $\longleftrightarrow$ registers) also take one cycle
- CPU can do different operations each cycle
- output of one operation can be input to next



CPU's haven't been this simple for a long time!

Pipelining

Pipelining is a technique in which multiple instructions are overlapped in execution.



- 1 result per cycle after pipeline fills up
- improved utilisation of hardware
- major complication – an output can only be used as input for an operation starting later

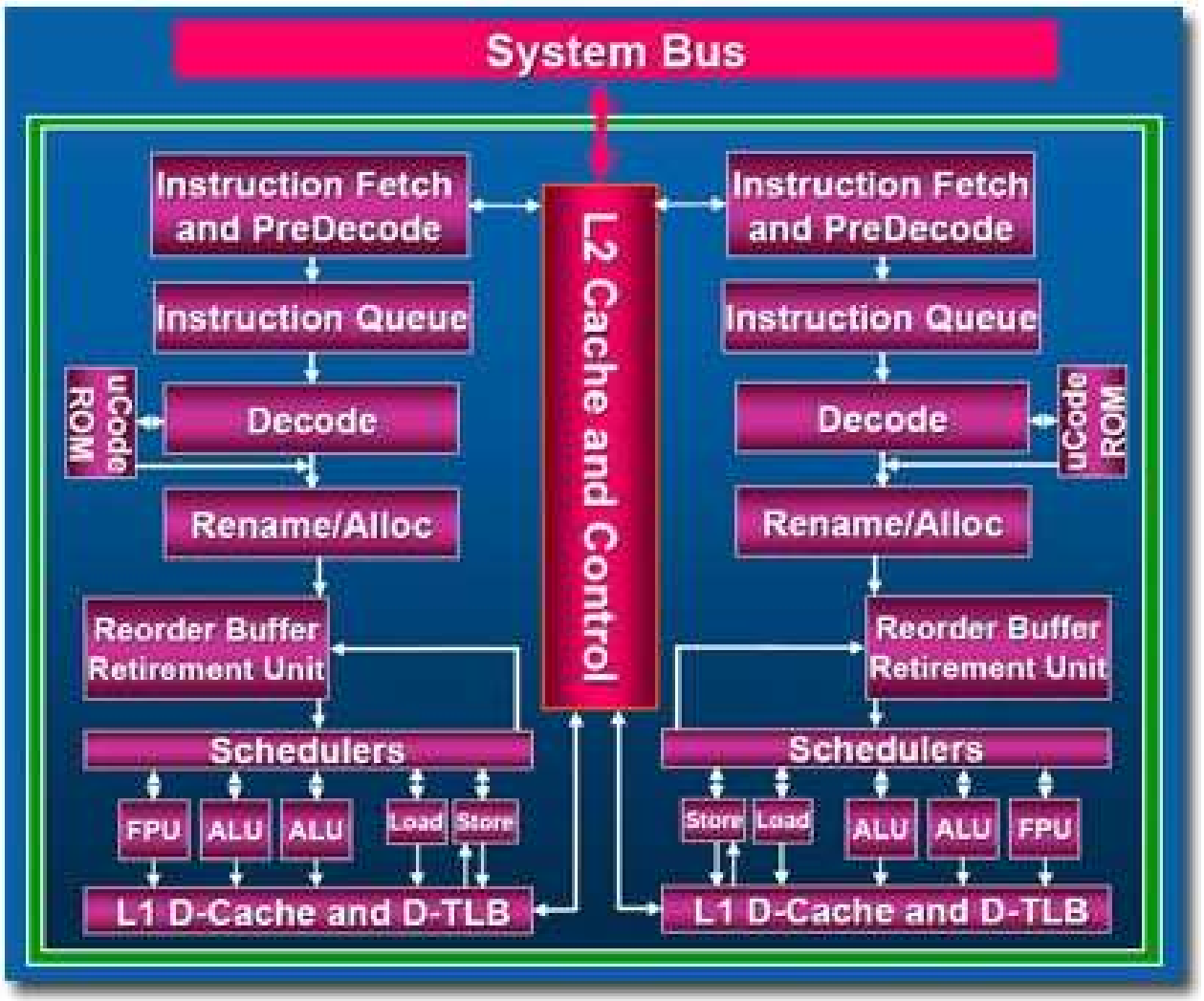
Superscalar Processors

Most processors have multiple pipelines for different tasks, and can start a number of different operations each cycle.

Example: each core in an Intel Core 2 Duo chip

- 14-stage pipeline
- 3 integer units (ALU)
- 1 floating-point addition unit (FPU)
- 1 floating-point multiplication unit (FPU)
- 2 load/store units
- in principle, capable of producing 3 integer and 2 FP results per cycle
- FP division is very slow

Intel Core Architecture



Technical Challenges

- compiler to extract best performance, reordering instructions if necessary
- out-of-order CPU execution to avoid delays waiting for read/write or earlier operations
- branch prediction to minimise delays due to conditional branching (loops, if-then-else)
- memory hierarchy to deliver data to registers fast enough to feed the processor

These all limit the number of pipelines that can be used, and increase the chip complexity $\Longrightarrow$ 95% of Intel chip devoted to control and data?

Current Trends

- CPU clock stuck at about 3GHz since 2006 due to high power consumption (up to 130W per chip)
- chip circuitry still doubling every 18-24 months

  $\Rightarrow$ more on-chip memory and MMU (memory management units)

  $\Rightarrow$ specialised hardware (e.g. multimedia, encryption)

  $\Rightarrow$ multi-core (multiple CPU's on one chip)
- peak performance of chip still doubling every 18-24 months

Intel chips

“Nehalem”:

- four cores, each running 2 threads
- integrated memory-management unit (QuickPath)

“Westmere”:

- 6 cores in 2010
- 10 cores in 2011?
- 4-way SMP system (40 cores, 80 threads)

“Sandy Bridge”:

- 8 cores in 2011?

Intel chips

All current chips support SSE vector instructions

- added for graphics on systems without a GPU
- mini-vector of 4 `floats` or 2 `doubles`

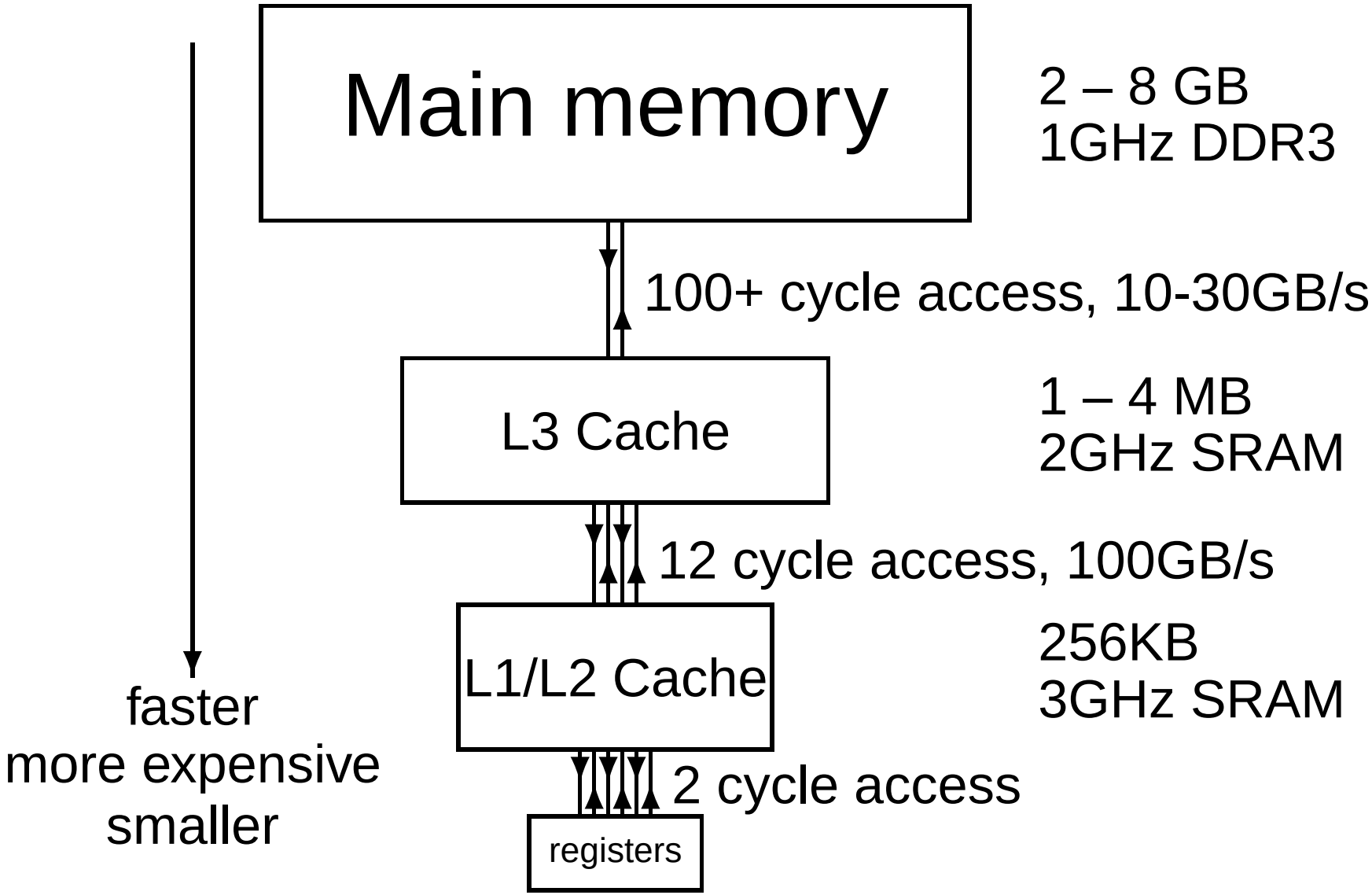
New “Sandy Bridge” CPUs support AVX vector instructions:

- mini-vector of 8 `floats` or 4 `doubles`

and later products may increase the vector length.

Code vectorisation will become essential for good performance

Memory Hierarchy



Memory Hierarchy

Execution speed relies on exploiting data *locality*

- temporal locality: a data item just accessed is likely to be used again in the near future, so keep it in the cache
- spatial locality: neighbouring data is also likely to be used soon, so load them into the cache at the same time using a ‘wide’ bus (like a multi-lane motorway)
- from a programmer point of view, all handled automatically – good for simplicity but maybe not for best performance

GPUs – the big development

Economics is again the key:

- produced in vast numbers for computer graphics
- increasingly being used for
  - computer games “physics”
  - video (e.g. HD video decoding)
  - audio (e.g. MP3 encoding)
  - multimedia (e.g. Adobe software)

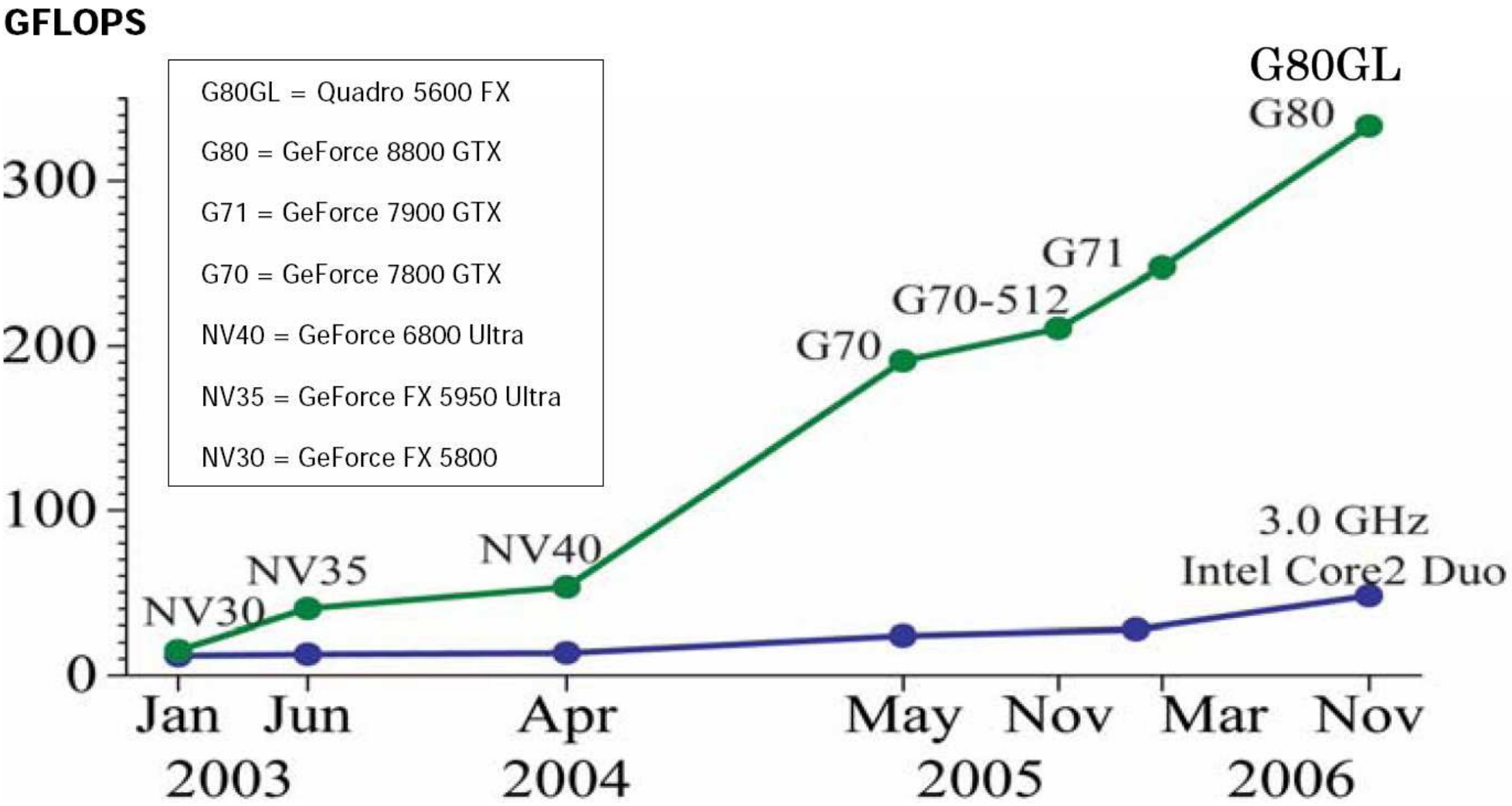
  - computational finance
  - oil and gas
  - medical imaging
  - computational science

GPUs – the big development

4 major vendors:

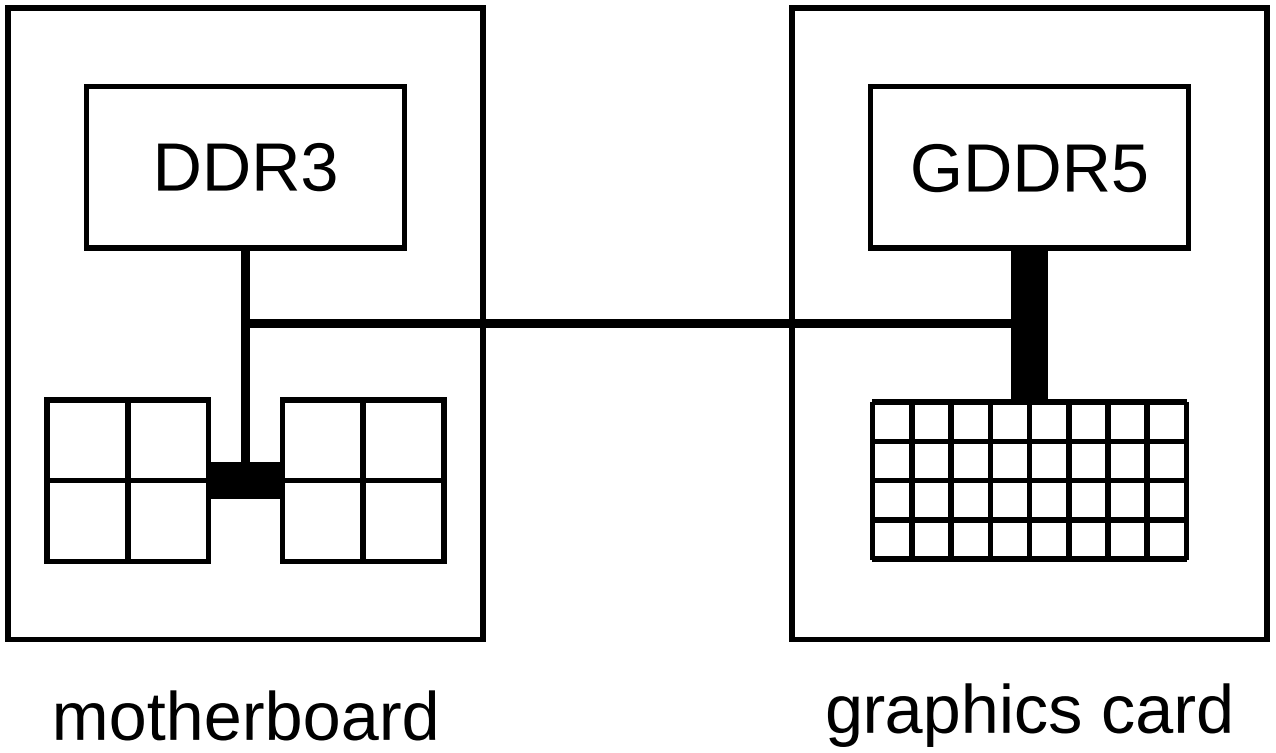
- NVIDIA
  - turnover about 10% of Intel's
- AMD
  - bought ATI several years ago
- IBM
  - co-developed Cell processor with Sony and Toshiba for Sony Playstation, but now dropped it for HPC
- Intel
  - was developing "Larrabee" chip as GPU, but now aimed for HPC

CPUs and GPUs



GPUs

The GPU sits on a PCIe graphics card inside a standard PC/server with one or two multicore CPUs:



GPUs

- up to 512 cores on a single chip
- simplified logic (no out-of-order execution, no branch prediction) means much more of the chip is devoted to floating-point computation
- arranged as multiple units with each unit being effectively a vector unit, all cores doing the same thing at the same time
- very high bandwidth (up to 190GB/s) to graphics memory (up to 6GB)
- not general purpose – for parallel applications like graphics, and Monte Carlo simulations

GPUs

Can also build big clusters out of GPUs:

- NVIDIA S2070 1U server holds four GPUs which can be connected to a 1U twin-node Intel server
- HP / IBM / Dell all have systems with built-in GPUs
- CRAY is now designing huge supercomputers based on GPUs
- cluster management software still evolving to support such servers

High-end HPC

- Tianhe-1A (China) – #1 supercomputer
  - 7168 NVIDIA Fermi GPUs
- Nebulae (China) – #3 supercomputer
  - 4640 NVIDIA Fermi GPUs
- TSUBAME-2 (Japan)– #4 supercomputer
  - 4224 NVIDIA Fermi GPUs

- within UK, very little based on latest Fermi GPUs:
  - BAESystems cluster has 32 GPUs
  - Oxford Skynet cluster has 16 GPUs
  - new HPC Wales cluster will have 16 GPUs
  - HECToR cluster has 8 GPUs

Chip Comparison

Intel “Sandy Bridge” i7

- 4 MIMD cores
- few registers, multilevel caches
- 30 GB/s bandwidth to main memory

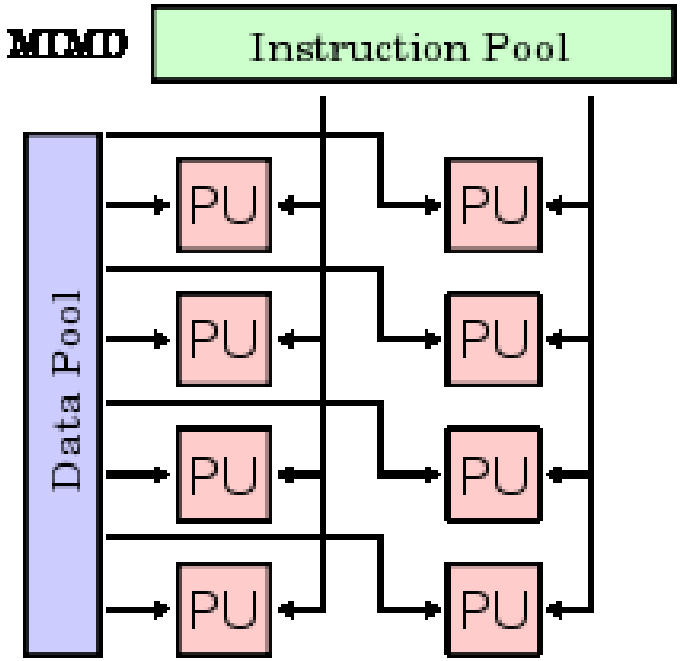
NVIDIA GTX580:

- 512 cores, arranged as 16 units each with 32 SIMD cores
- lots of registers, almost no cache
- 5 GB/s bandwidth to host processor (PCIe x16 gen 2)
- 190 GB/s bandwidth to graphics memory

Chip Comparison

MIMD (Multiple Instruction / Multiple Data)

- each core operates independently
- each can be working with a different code, performing different operations with entirely different data

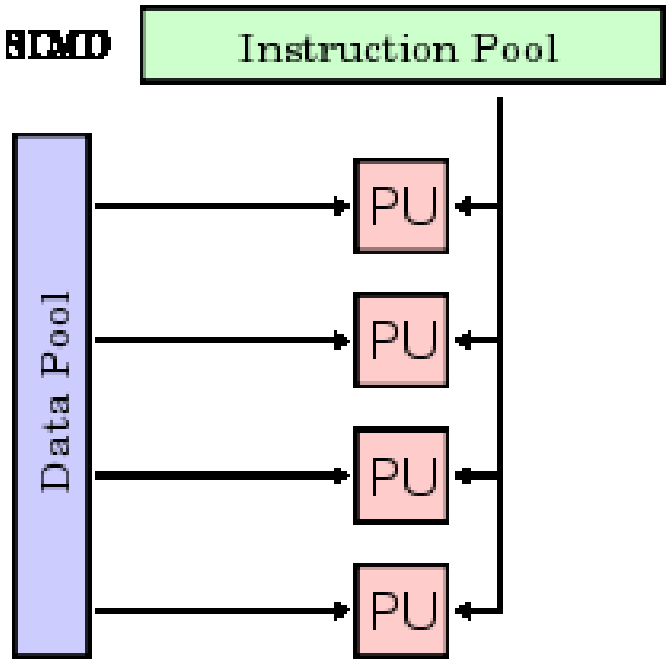


Wikipedia figure

Chip Comparison

SIMD (Single Instruction / Multiple Data)

- all cores executing the same instruction at the same time, but working on different data
- only one instruction de-coder needed to control all cores
- functions like a vector unit



Wikipedia figure

Chip Comparison

One issue with SIMD / vector execution: what happens with if-then-else branches?

Standard vector treatment: execute both branches but only store the results for the correct branch.

NVIDIA treatment: works with sub-vectors (called warps) of length 32. If a particular warp only goes one way, then that is all that is executed.

Chip Comparison

How do NVIDIA GPUs deal with same architectural challenges as CPUs?

Very heavily multi-threaded, with at least 8 threads per core, and I often use 40 or more:

- solves pipeline problem, because output of one calculation can be used an input for next calculation *for same thread*
- switch from one set of threads to another when waiting for data from graphics memory

GPU Programming

Big breakthrough in GPU computing has been NVIDIA's development of CUDA programming environment

- initially driven by needs of computer games developers
- now being driven by new markets (e.g. HD video decoding)
- C plus some extensions and some C++ features (e.g. templates)

GPU Programming

- host code runs on CPU, CUDA code runs on GPU
- explicit movement of data across the PCIe connection
- very straightforward for Monte Carlo applications, once you have a random number generator
- harder for finite difference applications

GPU Programming

Next major step is development of OpenCL standard

- pushed strongly by Apple, which now has NVIDIA GPUs in its entire product range, but doesn't want to be tied to them forever
- drivers are computer games physics, MP3 encoding, HD video decoding and other multimedia applications
- multi-platform standard will encourage 3rd party developers (including ISVs and banks)
- based on CUDA and supported by NVIDIA, AMD, Intel, IBM and others, so developers can write their code once for all platforms

GPU Programming

- OpenCL compilers now available from NVIDIA, AMD and Intel
- Imagination Technologies (which developed chip for iPhone) is also assembling an OpenCL compiler team
- Microsoft is the one big name not involved
- may need to re-optimise OpenCL code for each new platform – auto-tuning is another big trend in scientific computing
- at present, not making any headway in scientific computing, because AMD does not have a competitive hardware product

My experience

- Random number generation (mrg32k3a/Normal):
  - 2000M values/sec on GTX 280
  - 70M values/sec on Xeon using Intel's VSL library
- LIBOR Monte Carlo testcase:
  - 180x speedup on GTX 280 compared to single thread on Xeon

My experience

- 3D PDE application:
  - factor 50x speedup on GTX 280 compared to single thread on Xeon
  - factor 10x speedup compared to two quad-core Xeons

GPU results are all single precision – double precision is roughly 2 times slower on latest hardware

Why GPUs will stay ahead

Technical reasons:

- SIMD cores (instead of MIMD cores) means larger proportion of chip devoted to floating point computation
- tightly-coupled fast graphics memory means much higher bandwidth

Economic reasons:

- CPUs driven by price-sensitive office/home computing; not clear these need vastly more speed
- CPU direction may be towards low cost, low power chips for mobile and embedded applications
- GPUs driven by high-end applications
  – prepared to pay a premium for high performance

Will GPUs have impact?

- I think they're the most exciting development in last 10 years
- Have generated a lot of interest/excitement in academia, being used by application scientists,not just computer scientists
- Gives at least $10\times$ improvement in energy efficiency and price / performance compared to $2\times$ quad-core Intel Xeons.
- Effectively a personal cluster in a PC under your desk

Course objectives

- learn about GPU hardware and software
- learn about most key parts of CUDA programming
- through the practicals, develop confidence in applying the knowledge to writing programs
- learn about some key challenges, and how to approach the GPU implementation of a new application
- learn about resources for future learning

Most of all, I hope to convince you it’s not difficult!

More information

Wikipedia overviews of GeForce and Tesla cards:
`en.wikipedia.org/wiki/GeForce_400_Series`
`en.wikipedia.org/wiki/GeForce_500_Series`
`en.wikipedia.org/wiki/Nvidia_Tesla`

NVIDIA's CUDA homepage:
`www.nvidia.com/object/cuda_home_new.html`

Microprocessor Report article:
`www.nvidia.com/docs/IO/47906/220401_Reprint.pdf`

My webpages:
`people.maths.ox.ac.uk/gilesm/`
`people.maths.ox.ac.uk/gilesm/hpc.html`