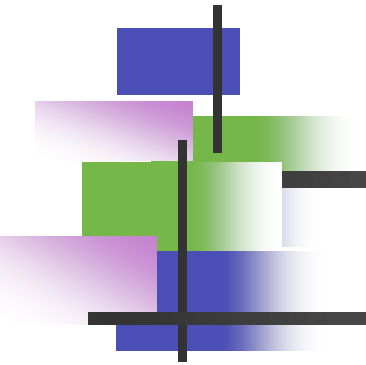


1. Parallel Programming – the Context

Parallel Programming

Dr Hamidreza Khaleghzadeh
School of Computing
University of Portsmouth

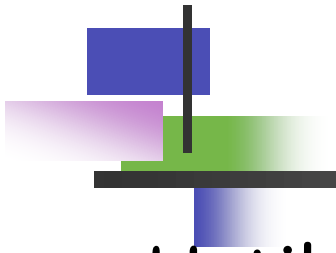
Goals

- Set the context for learning parallel programming
  - the emergence of *multicore* and *many-core* commodity systems
  - the nature of modern *supercomputers*.
- Structure of module and assessment.

Parallel Computing

- This module will focus on parallel computing and especially parallel programming.
- We interpret "parallel computing" as:

  Harnessing the nodes of multi-core or many-core processors, or computer clusters including supercomputers, to work cooperatively on individual problems, with a goal of getting results ***fast****.*



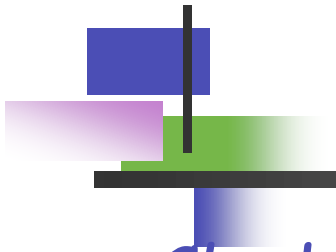
Context

- Until quite recently parallel computing has been the province of researchers with access to *supercomputers* – e.g. in university departments, national labs, weather forecasting centres, etc.
- The situation has changed in recent years with the widespread availability of *multi-core processors* and *many-core* accelerators.
- Let's reprise some of that context
  - If you attended the final TB1 lecture of COSINE in the second year, some of this may look familiar…

MULTICORE AND MANYCORE

The Era of the Free Lunch

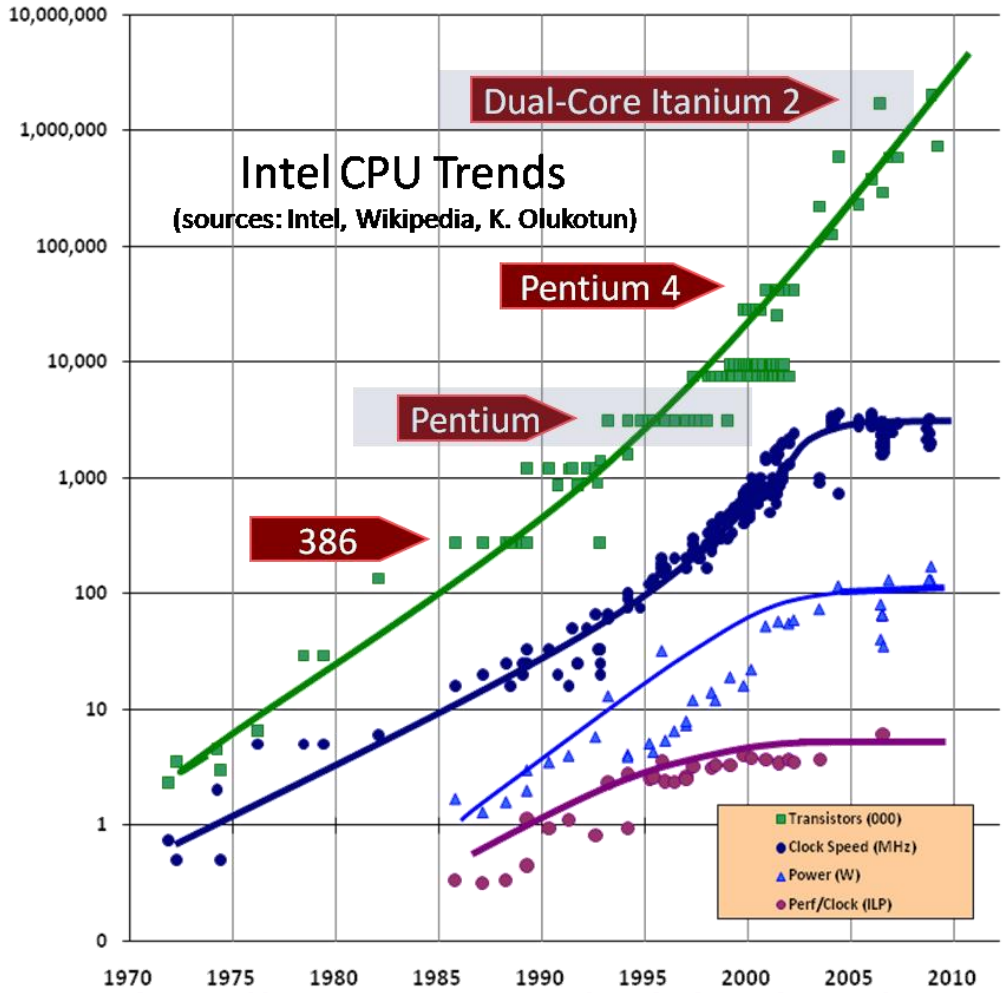
- Increasing *clock speeds* (combined with increasing *Instruction Level Parallelism*) over the decades leading up to the early 2000s meant that programs would run faster and faster with no extra effort
- ... or, equivalently, that applications of greater and greater complexity could be executed on hardware of fixed or reducing cost.



Concerning Processor Speed

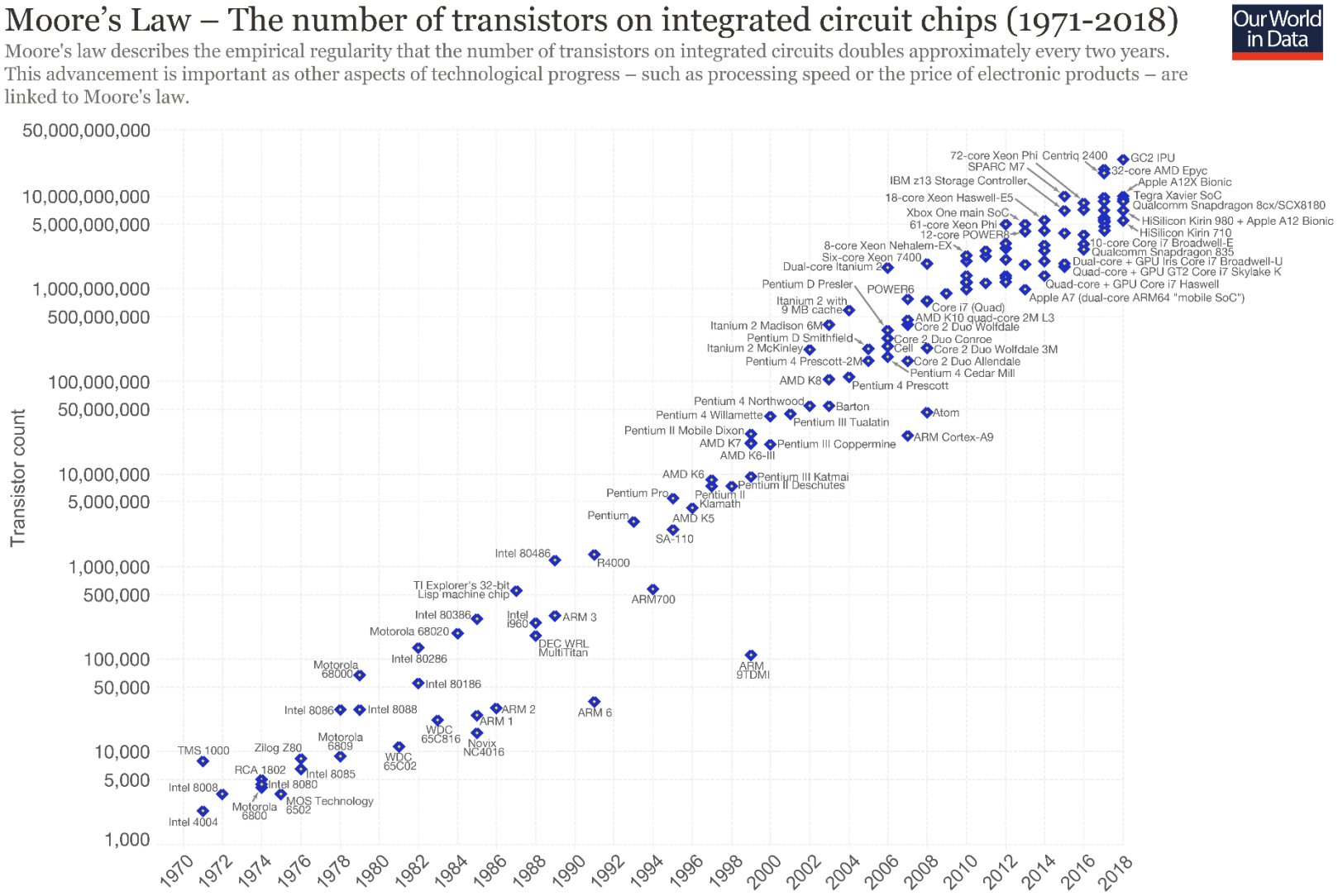
- *Clock speed* of microprocessors increased exponentially through the 1990s and beyond; but for now they appear to have reached a limit.
- In 2005 Herb Sutter pointed out that clock speed for Intel CPUs had *stuck* around 3.5 GHz.
  - Fifteen years on, observation still good.
  - Clock speed is now limited by power consumption/heat dissipation.
- Moore's Law for *transistor count* continues (for now).

Intel CPU Clock Speed (to 2009)†



† Graph taken from: http://www.gotw.ca/publications/concurrency-ddj.htm

Moore's Law up to 2018

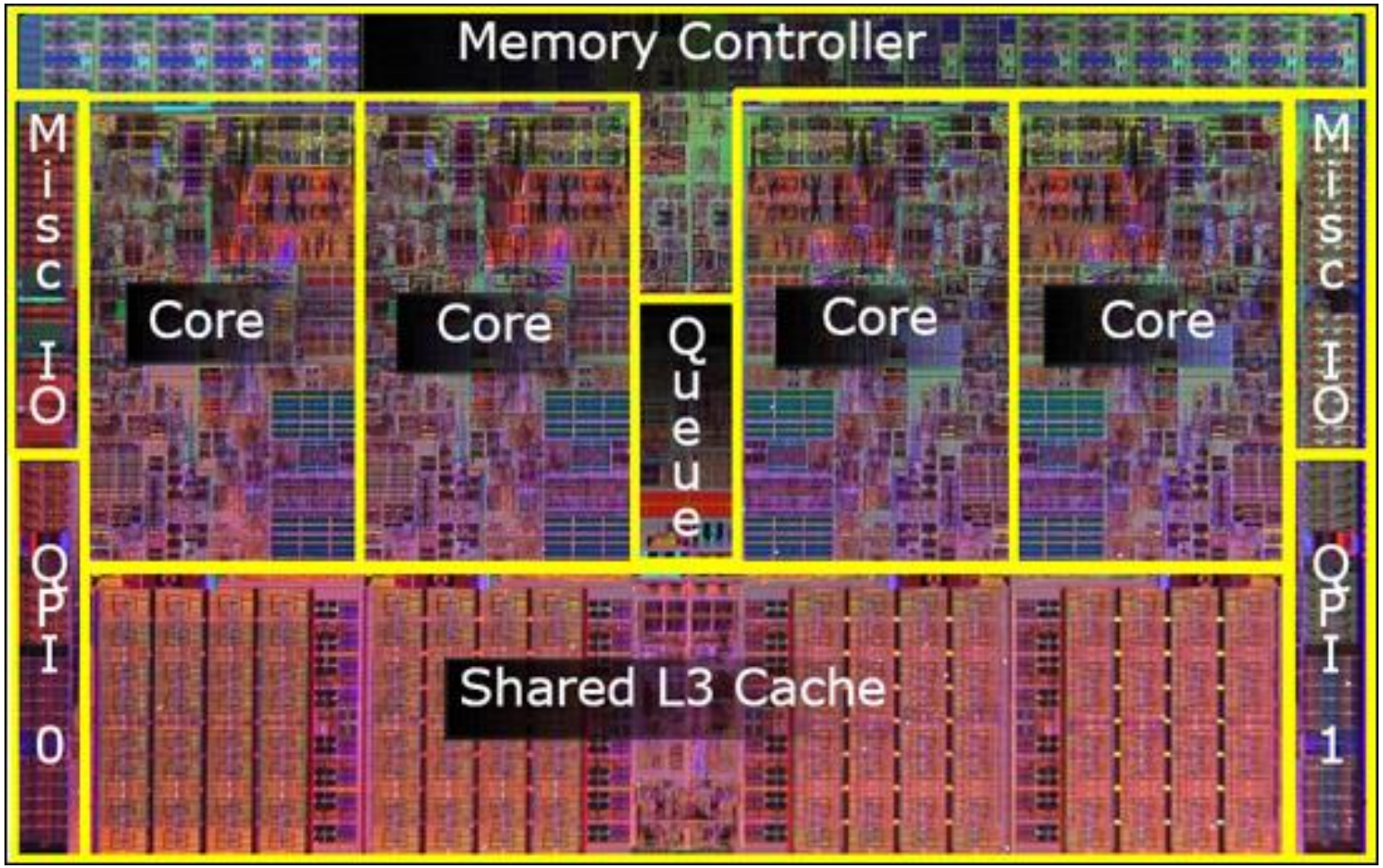


Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Multicore

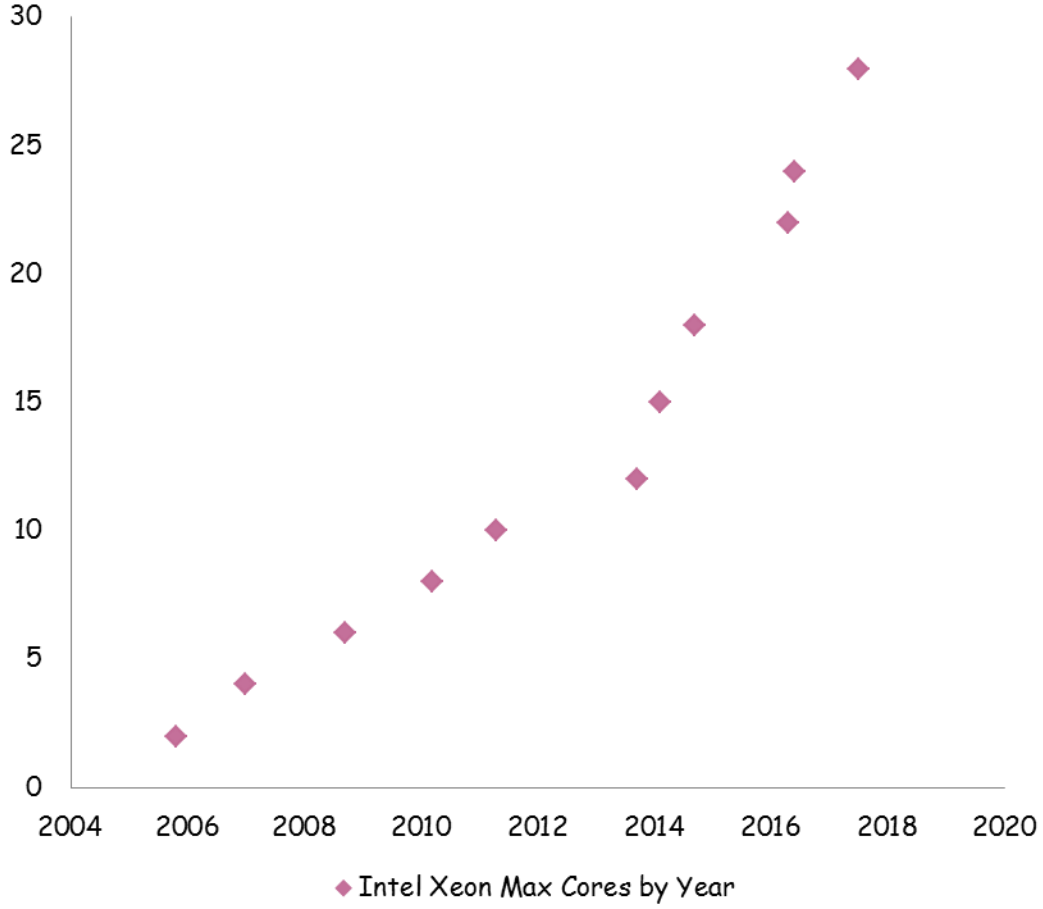
- Response of chip manufacturers has been *multicore* – multiple independent CPUs on a single chip.
  - Already commonplace – all but cheapest new desktops quad-core; laptops at least dual-core.
  - 8-core Nehalem-EX chip (for servers) released in March 2010:
    http://www.theregister.co.uk/2010/03/30/intel_nehalem_ex_launch/
  - 15-core Intel Xeons released early 2014:
    http://techreport.com/news/26056/intel-releases-15-core-xeon-e7-v2-processor
  - 28-core Xeons July 2017
    - (56-core available April 2019, but dual processor)

Example Quad-core Processor†



†Intel Core i7 “Nehalem”, from: http://www.hardwarecanucks.com

Intel Xeon Maximum Cores by Year†



†Data from:
https://en.wikipedia.org/wiki/List_of_Intel_Xeon_microprocessors

Graphics Processing Units

- GPUs are widely available *Massively Parallel Processors*.
- Rather than a few complex, superscalar cores, they use *many*, *simple* cores.
  - E.g. NVIDIA GTX 680:
    http://developer.nvidia.com/content/geforce-gtx-680-developers-perspective
    has 1536 "CUDA" cores.
- Notional performance for floating point: *3090 GFLOPS*
  - c.f. perhaps few 10s GFLOPS for a multicore *CPU*.

"General Purpose" GPUs

- Using APIs like *CUDA* and *OpenCL*, it is possible to use GPUs for general processing (not just graphics).
- Benchmark on next slide from machine identical to the PCs in BK0.19.

Running NVIDIA Sample code for Matrix Multiplication

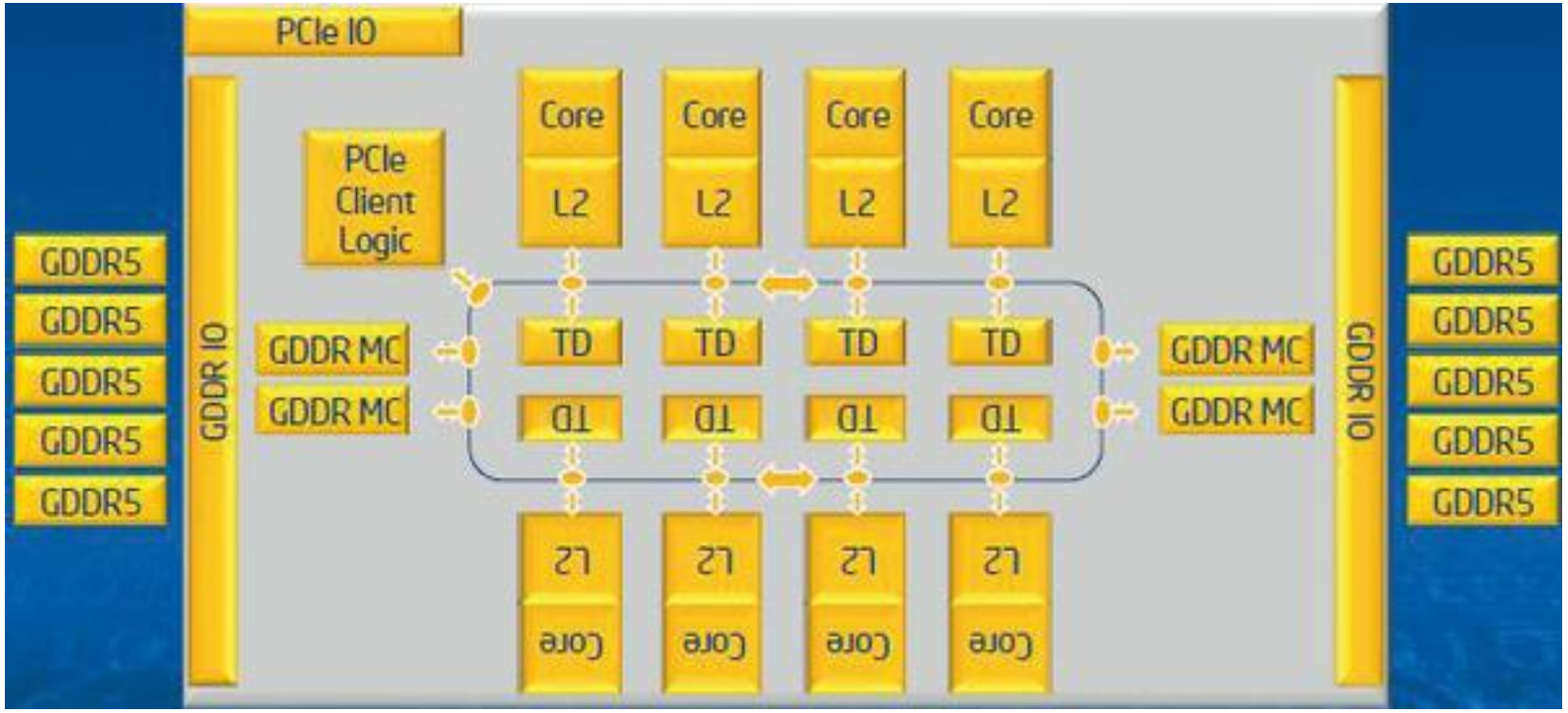
```
[Matrix Multiply CUBLAS] - Starting...
GPU Device 0: "GeForce RTX 2070" with compute capability 7.5
GPU Device 0: "GeForce RTX 2070" with compute capability 7.5
MatrixA(640,480), MatrixB(480,320), MatrixC(640,320)
Computing result using CUBLAS...done.
Performance= 3650.41 GFlop/s, Time= 0.054 msec, Size= 196608000 Ops
Computing result using host CPU...done.
Comparing CUBLAS Matrix Multiply with CPU results: PASS

NOTE: The CUDA Samples are not meant for performance measurements.
    Results may vary when GPU Boost is enabled.
```

Xeon Phi

- An alternative to the GPU co-processor approach, being marketed by Intel, is their *Multiple Integrated Core Architecture* (MIC).
- In this product line, a large number of Pentium-based cores are integrated on a single chip.
  - Unlike GPU, each core is a *general x86 processor* with private level 1 and 2 cache.
  - Unlike conventional multicore CPU, there is no *shared* cache on chip, but there *is* cache coherent access to external memory.
  - Fairly conventional programming tools apply.

MIC Architecture†



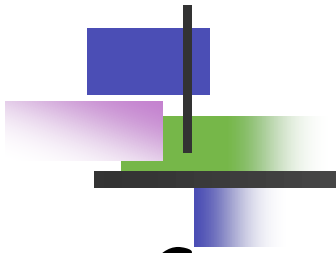
- Current generation Xeon Phis have *up to 72 cores* on chip.

†From: http://www.admin-magazine.com/Articles/Exploring-the-Xeon-Phi

Software Implications

- *But*, as Sutter said, move to multicore and many cores marks "the end of the free lunch".
  - Applications no longer *automatically* speed up with increasing clock rate.
- To exploit multiple cores, apps must be *rewritten*, adopting *parallel programming* techniques.
- This may eventually impact much or all of the software industry!

SUPERCOMPUTERS



Supercomputers and Parallelism

- Supercomputers have been exploiting massive parallelism for some decades, and many parallel programming techniques were developed for these.
- Increasing numbers of cores in commodity chips does *not* mean the need for supercomputers has gone away!
- In fact most supercomputers today are built by bolting together *large* numbers of "ordinary" multicore CPUs, to get even more parallelism.

Benchmarking Supercomputers

- Many different benchmarks suites have been developed to measure the speed of supercomputers.
- One of the simplest and most widely used numerical benchmarks today is called *HPL*†.
- In HPL, one solves a system of linear equations:

$$A\,x = b$$

A is an N by N random matrix, b is a random vector, and x a vector that must be solved for.

†http://www.netlib.org/benchmark/hpl/

High-Performance Linpack Benchmark(HPL)

- Solve for all x_i:

$$A_{11}x_1 + A_{12}x_2 + \ldots + A_{1N}x_N = b_1$$
$$A_{21}x_1 + A_{22}x_2 + \ldots + A_{2N}x_N = b_2$$
$$\ldots \qquad \ldots \qquad \ldots$$
$$A_{N1}x_1 + A_{N2}x_2 + \ldots + A_{NN}x_N = b_N$$

  - A and b are randomly generated.
  - The solution must be done by an LU scheme involving $2/3\ N^3 + O(N^2)$ floating point operations.
  - The benchmarker must adjust N to get the *best performance* in FLOPS for the system being tested.

FLOPS Reminder

- Number of *floating point operations* executed *per second* (1 FLOP = a single + , -, * , etc operation)
  - *megaFLOPS* (= 10^6 FLOPS) - historic processors
  - *gigaFLOPS* (= 10^9 FLOPS) - current CPUs
  - *teraFLOPS* (= 10^{12} FLOPS) – current GPUs
  - *petaFLOPS* (= 10^{15} FLOPS) - world class, massively parallel supercomputers
  - *exaFLOPS* (= 10^{18} FLOPS) - not quite yet – see later!

TOP500

- The *TOP500 list* is published twice a year†
- Often regarded as a ranking of the most powerful supercomputers in the world
- Technically speaking, it only ranks their speed at executing HPL!
  - There seems to be some move afoot to shift emphasis to a new benchmark High Performance Conjugate Gradients, claimed to be representative of a wider range of real applications.

†http://www.top500.org/

Sierra

- *Number 5* on the June 2022 TOP500 list.
- Located at Lawrence Livermore National Lab in US.
  - Commissioned for nuclear weapons simulation.
  - LINPACK performance of 94.64 petaFLOPS.
  - Consists 8,640 of 22-core *POWER9* processors for (only!) *190,080 cores*
  - …connected to 17,280 NVIDIA V100 GPUs.

Summit

- *Number 4* on the June 2022 TOP500 list.
- A *IBM* developed computer at Oak Ridge National Lab in US.
  - For general use by scientific community – initial apps likely to include cosmology, medicine and climatology.
  - LINPACK performance of 143.5 petaFLOPS.
  - Consists 9,216 of 22-core POWER9 processors for *202,752 CPU cores*
  - *But* POWER9s connected to 27,648 Nvidia Tesla V100 GPUs, each with 5120 CUDA cores – so notional total of *141,557,760 CUDA cores!*

Supercomputer Fugaku

- *Number 2* on the June 2022 TOP500 list.
- Developed at RIKEN Centre for Computational Science, Kobe Japan.
  - Ultimate goal to break the *exaFLOP* barrier.
  - LINPACK performance of 415.5 petaFLOPS.
  - Consists 158,976 Fujitsu A64FX processors (48-core *ARM*-based processor, with *Scalar Vector Extensions* SIMD capabilities) for *7,640,848 CPU cores.*

Supercomputer Frontier

- *Number 1* on the June 2022 TOP500 list.
- Developed at Hewlett Packard Enterprise (HPE)
  - LINPACK performance of 1,102 petaFLOPS.
  - Consists 136,408 AMD Optimized 3rd Generation EPYC 64C 2GHz for *8,730,112 CPU cores.*

Supercomputing at UP

- The university's Institute of Cosmology and Gravitation run a smaller supercomputer called *SCIAMA* (SEPNet Computer Infrastructure for Astrophysical Modelling and Analysis)†
  - Primarily for cosmological simulation and data analysis, but available for researchers throughout the university.
  - Originally assembled from 6-core Intel Xeon Westmere processors for *1008 CPU cores*.
  - Recently upgraded to SCIAMA II, adding Ivy Bridge processors to more than double total number of cores.

† http://www.port.ac.uk/research/news/title,149485,en.htm
http://www.sciama.icg.port.ac.uk/

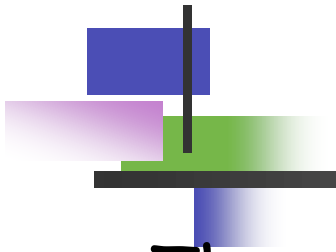
The Millennium Simulation

- Slightly old now, but nice example of a supercomputer application in cosmology
  - See *Simulating the Joint Evolution of Quasars, Galaxies and their Large-Scale Distribution* at http://www.mpa-garching.mpg.de/gadget
  - Also see movies at that site!
- Follows evolution of 10^{10} dark matter "particles" from early Universe (z = 127) to current day.
  - Performed on 512 nodes of IBM p690.
  - Used 1TB of distributed memory.
  - 350,000 CPU hours – 28 days elapsed time.
  - Floating point performance around 0.2 TFLOPS.

PLAN FOR THIS TEACHING BLOCK

Module Structure

- General plan:
- Learning driven by *practical work*; assessment takes form of a "lab book".
- Early lab sessions will be "scripted", and introduce some basic skills for parallel programming.
- Some later lab sessions will be dedicated to an individual development project that you will select about half-way through the term.



Scripted Lab Sessions

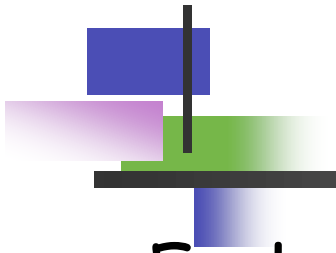
- These will use Java, starting with some simple examples of parallel programming using Java threads on the multi-core machines in the labs.
- Subsequent scripted sessions will use an *MPI*-like Java interface called *MPJ Express* to introduce parallel programming on "clusters" (hopefully using subsets of machines in the lab as our clusters).

Development Project

- I will provide a few possible choices of programming project, or you can propose your own (I must agree to it!)
  - You *may* use Java/MPJ Express for implementation in the teaching labs.
  - If you wish to use C/C++ or other programming environments. *Should* be possible in labs using CentOS – up to you to assess feasibility!
  - Or you might bring your own platform – perhaps you have an NVIDIA GPU in your laptop that you can use for CUDA programming?!

Lab Books

- 4000 words, which should record all lab work, and the outcome of your development project.
- Details of marking scheme, etc, will follow shortly. For now, make sure you keep a record of all you do in lab sessions!



Lectures

- Early lectures concentrate on introducing programming environments, paradigms and algorithms that are likely to be useful in your lab work (including development project), with an emphasis on MPI programming.
- Later lectures will cover more diverse topics, including more novel approaches to programming parallel systems, with recent developments.

Summary

- Set the context for parallel programming and this unit.
- Next week: *parallel programming with threads*

Further Reading

- See links embedded in these slides.