XJCO3221 Parallel Computation

University of Leeds

Lecture 1: Introduct

[1] p.k.jimack@leeds.ac.uk

This lecture

This lecture we will cover:

-
-
-

  the current ubiquity of parallel machines.
- The three classes of parallel architecture w
- Module overview

Module adminstration

Lecture slides

- All lectures will be made available on Minerva at least 24 hours in advance of the timetabled slots

Wor

- and 14 respectively.
- Specimen answers will appear on Minerva r after the corresponding lecture.

Computer resources

- All computer assignments will be undertaken using the Cloud
- Accounts have been created for you: ONLY use for work relating to THIS module
- Access details provided as part of worksheets/assignments

Other support

Discussion on Minerva

- There is a forum for each part of the module.
-
-

- Check first your query has not already been an

Labs

- For you to practice coding exercises (worksheets) and courseworks.
- Please see Joint School for locations/times.

Assessments (summative)

Assessment is by both **coursework** and **exam**.

- 50% spread over 3 items of coursework.
-

Each

- **Please test your submissions on the clou if you develop and run on your own compu**

Each worksheet covers similar material to each coursework, so *attempting the current worksheet prior to the assignment will help you significantly* with the next coursework.

Coursework schedule and deadlines

Coursework	Weight	Release	Deadline
1	15%	15^{th} March	Tuesday 29^{th} March
			pril
			y

Before attempting the courseworks you should f
with the relevant material:

Coursework	Up to and i
1	Lecture 6
2	Lecture 11
3	Lecture 16

Language

For this module we will use C.

- We will cover three different parallel libraries/API's, and the

- Coursework submissions must be in C.

If you have not programmed in C for a while you may lik
XJCO1711 Procedural Programming.

We will mostly use **loops**, **conditionals**, **arrays** and **pointers**.

Books

For additional information on for parallel programming **in general**:

- **Parallel Programming**, *Wilkinson and Allen* (Pearson).
  - nd
  -
  -

- **Structured Parallel Programming**
  Reinders (Morgan-Kauffman, 2012
  - Modern, focuses on patterns of parallel al
  - Few code examples, mainly for shared m
  - eBook available *via* the library.

Books for specific architectures will also be mentioned when introduced. **You do not need to buy any of these books**.

Why this module?

- Almost all[1] modern computers and devices fall into one of three classes of **parallel architecture**.
-

  -
-
  - No point focusing on any one as it may not last

- Need to develop **portable** skills in that can be applied to current **an** s and architcetures.

[1]With very few exceptions, *e.g.* feature phones *etc.*

Objectives and learning outcomes

Objectives: This module will introduce the fundamental skills and knowledge required to develop parallel computer software.

Lear

stud

-
- Apply parallel design paradigms to serial algorithms.
- Evaluate and select appropriate parallel s problems.
- Generalise parallel concepts to future har developments.

Skills outcomes: Programming, design, performance measurement, evaluation.

Syllabus

This module covers the following 3 topic areas:

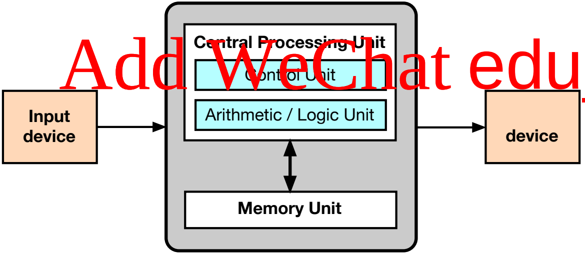
-
- **Parallel computation models**: sha
(SMP), distributed memory parallelis
graphics processing unit (GPGPU).

- **Common frameworks**: OpenMP, M
(MPI) and OpenCL.

Background and motivation

Early computers followed the so-called **von Neumann architecture** (1946):

- Based on Turing's **universal machine** (1936).
- F
  s



Turing (top), von Neumann (bottom) *(from Wikipedia)*

Moore's law

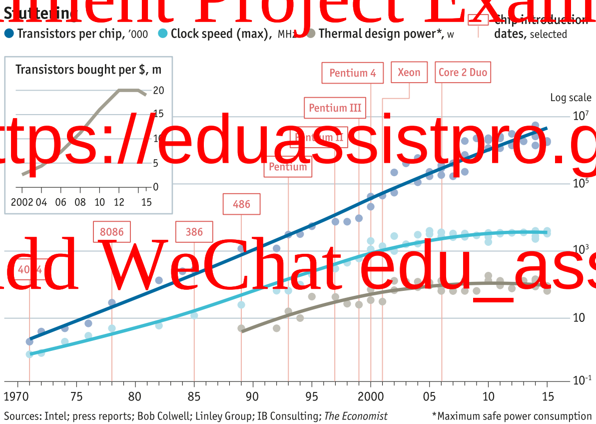
In 1965 Gordon Moore made the empirical observation that the number of transistors on a chip doubles every 18-24 months.

- *sus*

-
- **Exponential increase** of the most cost-effective number of components.

http://www.startupinnovation.org/research/moores-law

Processor speeds also used to follow Moore's law, but stopped around 15 years ago at $\approx$ 3.3GHz (ignoring overclocking).



http://www.economist.com/technology-quarterly/2016-03-12/after-moores-law

Limitations on clock speed

Increased frequencies result in greater **leakage** and greater **power consumption**[1]:

$$P = \alpha C_{\mathrm{L}} V^2 f$$

- f is the frequency.

However, $V \propto f$, so $\mathbf{P} \propto \mathbf{f^3}$.

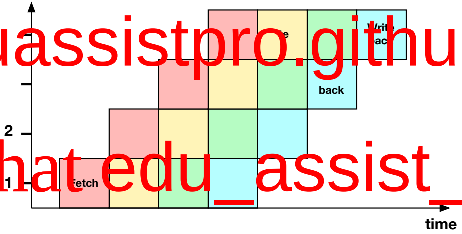
- **Rapid increase** that exceeds 100W fo
- Unsustainable even with sophisticated cooling technology.

[1] You don't need to learn this equation [which was taken from *Parallel Programming*, 2$^{\mathrm{nd}}$ ed., Rauber and Rünger (Springer, 2013)].

ILP: Instruction Level Parallelism

Chip designers have tried various **architectural improvements** to increase performance (*memory cache, speculative execution etc.*)

One is
differe
subse
are over



- Only one *fetch, decode etc.* at any given time.

This **instruction level parallelism** (ILP) is **limited** to around 10-20 instructions.

- We say it does not **scale**.

Multi-core CPUs (Lectures 2-7)

These architectural improvements did not require changes to code.

- Legacy sequential code **automatically** benefited.

Each i .

Star
mult

- For a few cores, can run applications simulta

With new chips having many cores (6, 8, 12, 16, 24, 28
running one application per core is not feasible.

- **Single** applications need to use **multiple** cores.
- **Requires new program logic.**

Clusters / Supercomputers (Lectures 8-13)

Even before clock speeds plateaued, some applications used multiple **machi**

- Sc
- W
- ...

PFlops = **petaflops** = 10^{15} floating point operations per second;
EFlops = **exaflops** = 10^{18} floating point operations per second.

https://www.top500.org/statistics/perfdevel

GPGPU (Lectures 14-19)

In the mid-1990s, the rise of graphical applications (especially games) drove the development of **graphics accelerators**:

-

In 200 ... **eneral purp**

- GPGPU = General Purpose Grap
- Now supported by most manufacturers

Suitable for other applications **including machine learning**.

- GPUs are part of the **deep learning** revolution.
- Now have dedicated **neural processing units** (NPUs).

Precedent from nature

Arguably the most complex system known is the **human brain**

If re

- would be less than 1kHz.
- We have about 10^{11} neurons, each connected to 10^4 others.
- The current fastest supercomputer has $\approx 10^7$ cores.

http://scitechconnect.elsevier.com

Parallel *versus* concurrent

Two or more applications run '**concurrently**' if they both execute 'in the same time frame.'

-
-

Whereas **parallel** applications actually per **simultaneously** on a parallel architecture.

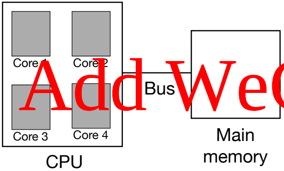
Parallelism implies concurrency, but **not** *vice versa*, *i.e.*

Parallel $\subset$ Concurrent

Shared *versus* distributed memory

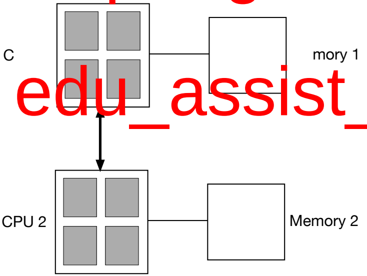
Shared memory

- All cores can 'see' whole of main memory.
-



Distributed memory

- Cores only see a fraction of the total memory.
-



Computation *versus* communication

Moving data to and from the cores also affects performance:

Fast: **R**

-
-
- Fast communication in **high-perf** InfiniBand, Gigabit Ethernet)

- **Local area** network communicatio
- **File I/O**.

Slow: **Wide area** network communication (*e.g.* the internet).

Flynn's taxonomy[1]

Characterises parallel architectures by **data** and **control flows**.

Acronym	**Instruction/data**	**Examples**
SI[illegible]	[illegible]	[illegible]
SIMD	Single Instruction, Multiple Data	GPU *(also SIMT; c.f. Lec-t*
MIMD	Multiple Instruction, Multiple Data	[illegible] p
MISD	Multiple Instruction, Single Data	Specialist hardware only

[1] Flynn, *IEEE Transactions on Computers* **21**, 948 (1972).

Module overview

Lectures	Content
1	Introduction
8-13	**Distributed memory parallelism** MPI-C Worksheet 2 and Cours
14-19	**General purpose GP** OpenCL *(a C-based language)* Worksheet 3 and Coursework 3
20	Module review

Next lecture

Next l :

-
- Overview typical hardware **architecture**, including **memory cache**.
- Look at some language and library support
- How to install and run OpenMP programs.