Introduction to High Performance Scientific Computing

Autumn, 2016

Lecture 1

Imperial College London

6 October, 2016

Instructor

Prasun Ray
Teaching Fellow
Department of Mathematics
p.ray@imperial.ac.uk
Huxley 6M20
Office hours: Mondays 4-5pm, MLC
Thursdays 5-6pm, MLC
(First office hour on Monday, 10/10)

Imperial College London

Weekly schedule

- Lectures:
  Monday, 11-12, Huxley 139
  Thursday, 9-10, Huxley 139
- Labs:
  Tuesday, 5-6pm, MLC (Huxley 414)
  or
  Wednesday, 10-11am, Huxley 139
- Only need to attend *one* lab session
- Wednesday lab requires laptop with necessary software installed (more on this later)

Imperial College London

Syllabus

Week 1: Unix basics, version control with git/bitbucket

Weeks 2-3: Programming and scientific computing with Python

Week 4: Modular programming with Fortran

Week 5: Libraries, makefiles, coupling Fortran+Python

Imperial College London

Syllabus

Weeks 6-7: Introduction to parallel computing and OpenMP

Weeks 8-9: Distributed memory computing with MPI, parallel libraries

Weeks 9-10: Basic computer architecture, cloud computing, cluster computing with Python and Spark

Imperial College London

Assessment

4 Programming assignments
HW1: Assigned 19/10, due 26/10 (**5%**)
HW2: Assigned 27/10, due 7/11 (**15%**)
HW3: Assigned 10/11, due 21/11 (**20%**)
HW4: Assigned 24/11, due 1/12 (**20%**)

1 Programming Project (40%)
Assigned 2/12, due 15/12

Submitting HW2 **commits you to the course**

Imperial College London

Online material

- Main resource is course webpage:

  http://imperialhpsc.bitbucket.org/

- Slides will be available before every lecture

  On blackboard 1st two weeks only

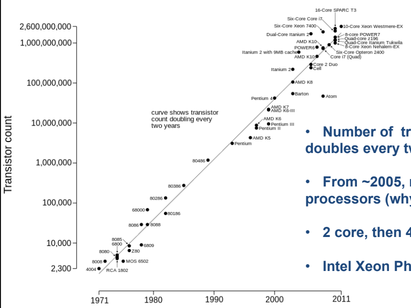
  - Afterwards only on course bitbucket page (more on this later):

  https://bitbucket.org/ImperialHPSC/m3c2016

Imperial College London

Moore's law

Microprocessor Transistor Counts 1971-2011 & Moore's Law

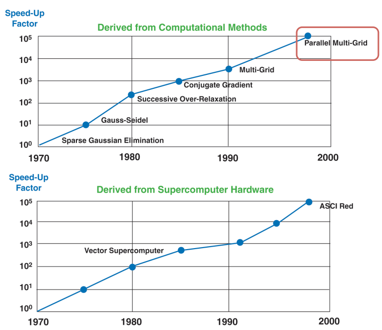


- Number of transistors on chip doubles every two years
- From ~2005, multicore processors (why?)
- 2 core, then 4, 6, 8, 16
- Intel Xeon Phi: 60+ cores!

Imperial College London

"Transistor Count and Moore's Law - 2011" by Wgsimon - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Transistor_Count_and_Moore%27s_Law_-_2011.svg#/media/File:Transistor_Count_and_Moore%27s_Law_-_2011.svg

Algorithms and hardware



Imperial College London

SIAM Rev (2001)

High-end HPC

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660

Historically: cluster computing limited to national labs, research universities

But now…

Imperial College London

Cluster computing is mainstream

Big data means big computers!

Imperial College London

Cluster computing is mainstream

amazon web services™

Google Cloud Platform

Microsoft | Cloud Services
We're all in.

Imperial College London

Course objective

- Cluster computing is not free!
- Important to:
  - choose right tools
  - use them effectively

This course provides foundation for "intelligent, informed" computing.

Imperial College London

Software tools

Useful to classify tools as *scientific* or *general purpose*

Imperial College London

Software tools

Useful to classify tools as *scientific* or *general purpose*

Examples:

Scientific	General purpose
Matlab	Python
Fortran	C++
R	Java

Imperial College London

Software tools

Languages are *compiled* or *interpreted*

Software tools

Languages are *compiled* or *interpreted*

Compiled	Interpreted
Fortran	Python
C++	Matlab
Java	Java

Software tools

This course:

Python: interpreted, general purpose

Fortran: compiled, scientific

Operating systems

Most HPC and scientific computing requires Unix (or Unix-like terminals)

Linux and Mac OS are built on Unix (and have terminal apps)
- Fairly straightforward to install course software

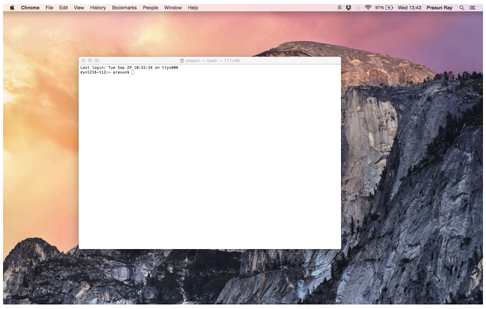
Windows:
- Not well-suited for HPC
- Can get Unix terminal with cygwin
- For this course: Should install Linux virtual machine (VM) and install software within the VM
- MLC computers have Linux VMs installed (go try them out!)

Instructions for installing course software available online:
http://imperialhpsc.bitbucket.org/

Imperial College London

Unix terminal

Terminal on a mac:



Imperial College London

12 Unix commands

Navigation:

pwd: print working directory (where am I?)
ls: list of directory contents (what is here?)
cd: change directory (let's go somewhere else)

```
$ pwd
/Users/prasun/Documents/repos/m3c2016
$
$ ls
Readme.md lectures
$
$ cd lectures
$
$ ls
lecture1
```

Imperial College London

12 Unix commands

Manipulate files and directories:

cp: Make copy of a file
mv: Move or rename a file
rm: Remove a file
rm –r : Remove directory and all of its contents (dangerous!)

```
$ ls
Readme.md lectures
$
$ cp Readme.md Readme.md_copy
$
$ ls
Readme.md Readme.md_copy
    lectures
$
$ mv Readme.md_copy
Readme.md_copy2
$
$ ls
Readme.md Readme.md_copy2
    lectures
$
$ rm Readme.md_copy2
$
$ ls
Readme.md lectures
```

Imperial College London

12 Unix commands

Info about contents of file:

cat: List contents of file
head -n: List first n lines
tail -n: list last n lines
grep: search within file for a string

```
$ cat example.txt
This is an example text file.
This is line 2.
This is line 3.
This is the last line.
$
$ head –1 example.txt
This is an example text file.
$
$ tail –2 example.txt
This is line 3.
This is the last line.
$
$ grep last example.txt
This is the last line.
```

Imperial College London

12 Unix commands

Getting help:
man: manual page for a command

Try *man ls*. What does *ls -l* do? *ls -a*?

What if you don't know name of command?

https://en.wikipedia.org/wiki/List_of_Unix_commands

or google.

Imperial College London

12 Unix commands

The 12 commands:

1. pwd
2. ls
3. cd
4. cp
5. mv
6. rm
7. rm –r
8. cat
9. head –n
10. tail –n
11. grep
12. man

This is "basic" Unix. Can do much more!

A little more Unix

Instead of outputting to screen, can output to file using ">"

```
$ ls
example.txt  lecture1

$ grep last example.txt > output.txt

$ ls
example.txt  lecture1  output.txt

$ cat output.txt
This is the last line.
```

Lines in example.txt containing "last" are written to output.txt

A little more Unix

Command can be executed sequentially (they can be "piped") using "|"

```
$ head -2 example.txt | grep line > output.txt
$
$ cat output.txt
This is line 2.
```

First two lines in example.txt are searched for the string "line" with results being written to output.txt

An example

You run optimization software that gives output that looks like:

```
INPUT:endgeom
INPUT:azimuthal 9 0.1
INPUT:polar 5
INPUT:begin

 k-cactus is   1.402458


TIMING: Module: cpu        10.03 wall        10.04 Overall: cpu        29.00 wall        29.29
=================================================================================
==================
INPUT:EDIT 4

CALLING EDIT(INTERFACE_NO=  4)

INPUT:begin
*********************************************************************************
INTERFACE    4  EIGENVALUE  1.402458  OVERALL MWd/t   0.0000E+00 BURNUP TIME   0.0000E+00
DAYS

=================================================================================
=======================================
RUN SET    1
```


An example

We only care about the "k-cactus" values which appear several times. How do we extract them?

```
INPUT:endgeom
INPUT:azimuthal 9 0.1
INPUT:polar 5
INPUT:begin

 k-cactus is   1.402458


TIMING: Module: cpu        10.03 wall        10.04 Overall: cpu        29.00 wall        29.29
=================================================================================
==================
INPUT:EDIT 4

CALLING EDIT(INTERFACE_NO=  4)

INPUT:begin
*********************************************************************************
INTERFACE    4  EIGENVALUE  1.402458  OVERALL MWd/t   0.0000E+00 BURNUP TIME   0.0000E+00
DAYS

=================================================================================
=======================================
RUN SET    1
```


An example

Using grep:

```
$ grep cactus datafile.out
 k-cactus is    1.402458
 k-cactus is    1.386050
 k-cactus is    1.377296
 k-cactus is    1.352324
 k-cactus is    1.328779
```

But what if we only want the numbers?

An example

Using grep:

```
$ grep cactus datafile.out
 k-cactus is   1.402458
 k-cactus is   1.386050
 k-cactus is   1.377296
 k-cactus is   1.352324
 k-cactus is   1.328779
```

But what if we only want the numbers?

Use "cut":
```
$ grep cactus datafile.out | cut -d s -f 3
   1.402458
   1.386050
   1.377296
   1.352324
   1.328779
```

Questions: How do we store these numbers in a file? How do we find out what the flags after "cut" are doing?

Imperial College
London

Overview of Lab 1

- Practice with these Unix commands
- Learn a bit more: path, top, which, setenv
- Tuesday: Familiarize yourselves with VMs in MLC
- Wednesday: If you have a Macbook or Linux laptop, you don't need to install anything (yet)
  - Windows: Need to install VirtualBox software and Linux virtual machine within VirtualBox
  - Instructions on course webpage
- Unix notes will be provided after lecture: make sure you are comfortable with these commands!

Imperial College
London