# Introduction to High Performance Scientific Computing

**Autumn, 2017**

**Lecture 1**

**Imperial College**
**London**

# Instructor

**Prasun Ray**
**Teaching Fellow**
**Department of Mathematics**
**p.ray@imperial.ac.uk**
**Huxley 6M20**
**Office hours: Mondays 5-6pm, MLC**
**Thursdays 4-5pm, MLC**
**(First office hour on Monday, 9/10)**

# Weekly schedule

- **Lectures:**
  **Monday, 11-12, Huxley 340**
  **Thursday, 11-12, Huxley 340**

- **Labs:**
  **Tuesday, 5-6pm, MLC (Huxley 414)**
  **or**
  **Wednesday, 10-11am, Huxley 340**

- **Only need to attend *one* lab session**

  **CDT-only lab: 11am-12pm, RSM 3.38**

- **Wednesday 10am lab requires laptop with necessary software installed (more on this later)**

# Syllabus

**Lectures 1-2: Unix basics, version control with git/bitbucket**

**Lectures 3-6: Programming and scientific computing with Python**

**Lectures 7-10: Modular programming with Fortran, libraries, makefiles, coupling Fortran+Python**

# Syllabus

**Lectures 11-14: Introduction to parallel computing and OpenMP**

**Lectures 15-16: Distributed memory computing with MPI, parallel libraries**

**Lectures 17-20: Basic computer architecture, cloud computing, cluster computing with Python and Spark**

# Assessment

**3 Programming assignments**
HW1: Assigned 23/10, due 2/11 (**20%**)
HW2: Assigned 6/11, due 16/11 (**20%**)
HW3: Assigned 20/11, due 29/11 (**15%**)

**1 Programming Project (45%)**
Assigned 30/11, due 15/12

**Submitting** HW1 **commits you to the course**

**CDT students:** Will be contacted about assessment separately

# Online material

- Main resource is course webpage:

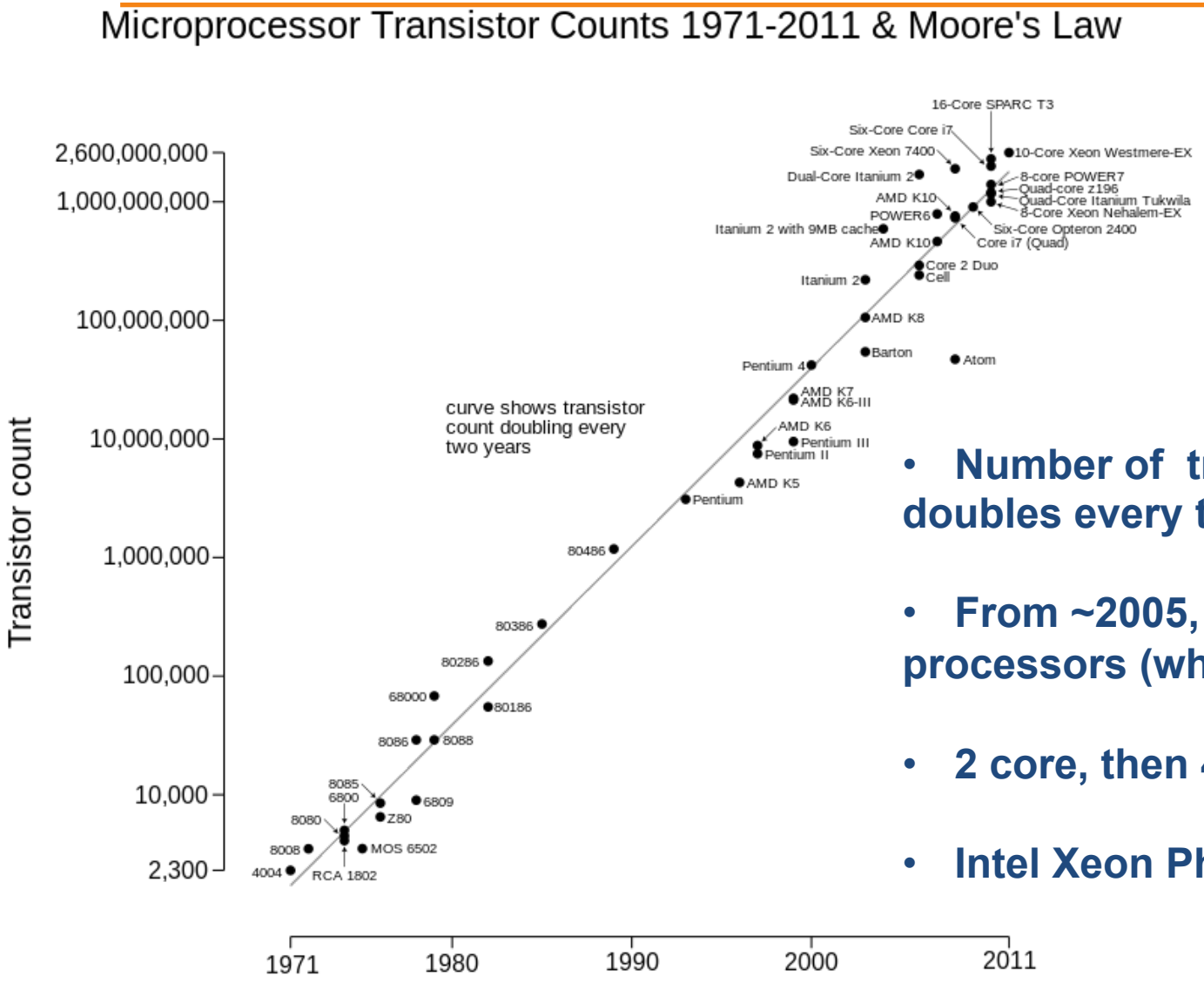
  http://imperialhpsc.bitbucket.io/

- Slides will be available before every lecture

  All course material will be available on course bitbucket page (more on this later):

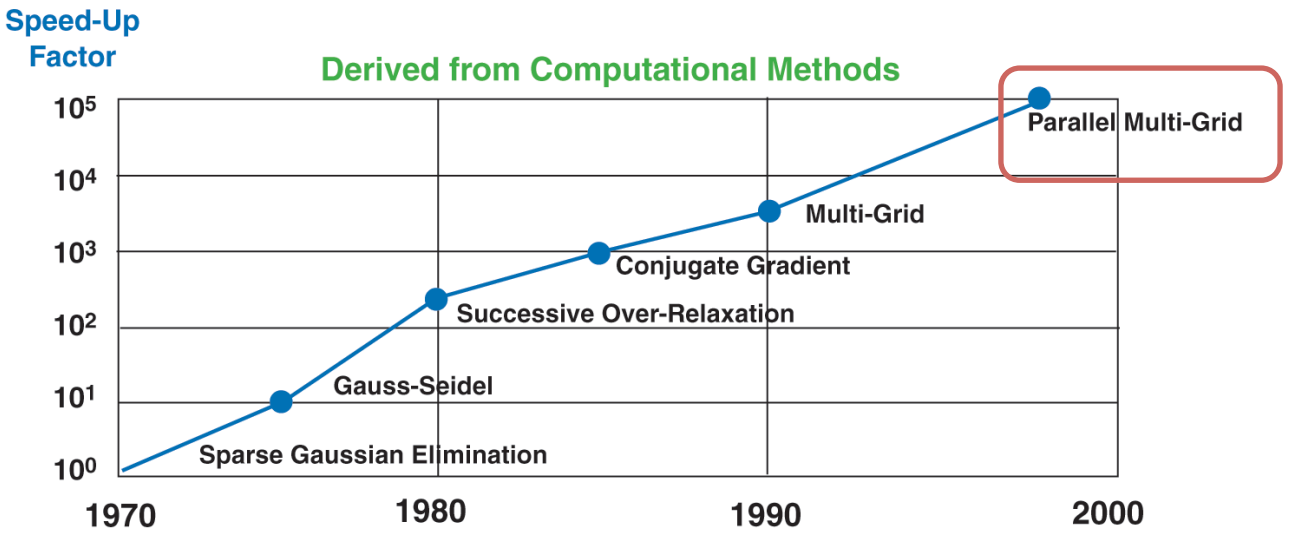
  https://bitbucket.org/ImperialHPSC/m3c2017

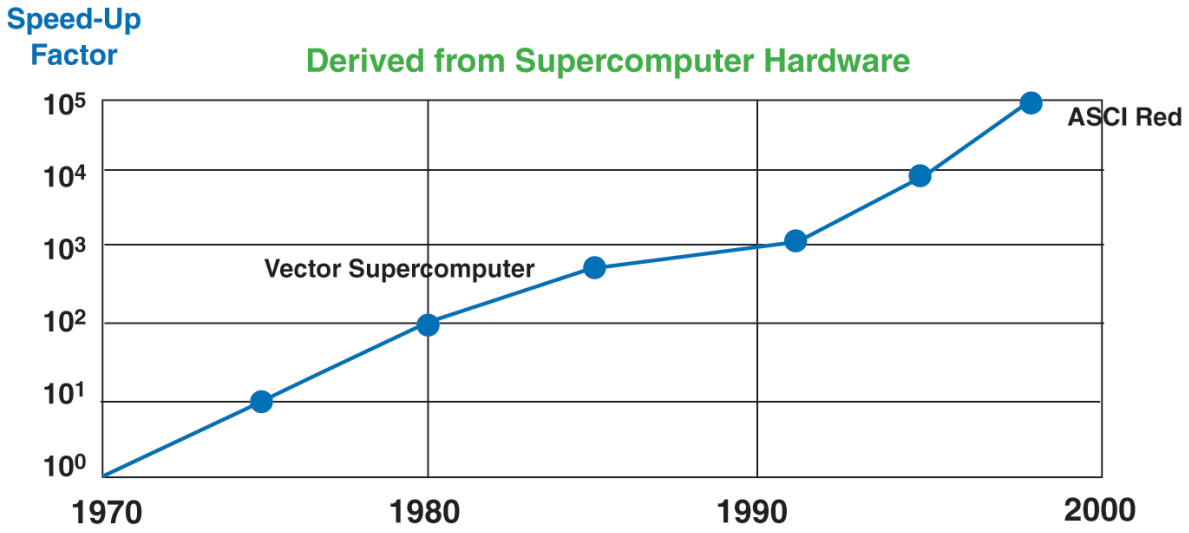
# Moore's law




- **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!**

# Algorithms and hardware






*SIAM Rev (2001)*

# High-end HPC

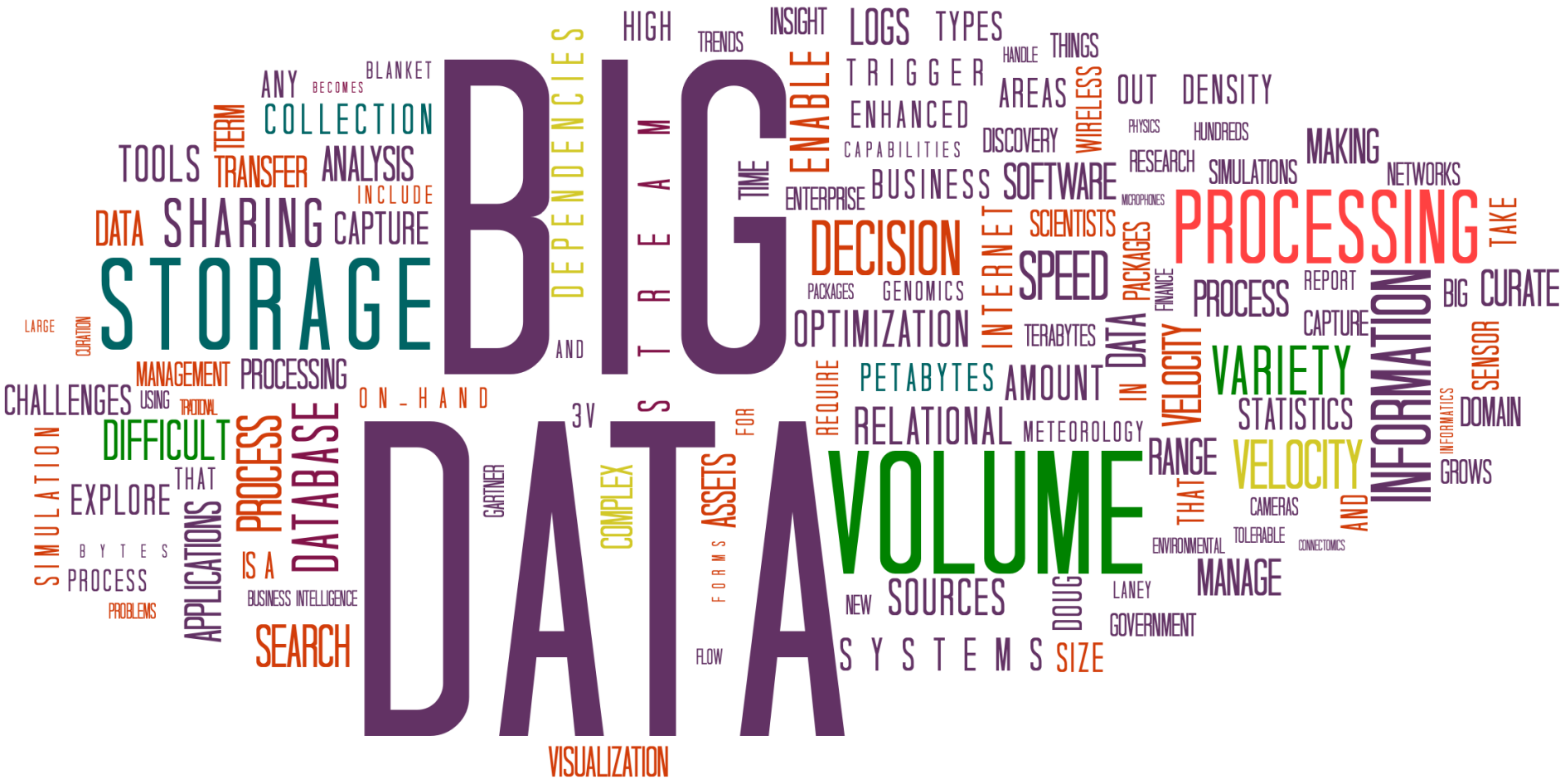
| RANK | SITE | SYSTEM | CORES | RMAX (TFLOP/S) | RPEAK (TFLOP/S) | POWER (KW) |
|---|---|---|---|---|---|---|
| 1 | National Super Computer Center in Guangzhou China | **Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT | 3,120,000 | 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…**

# Cluster computing is mainstream

## Big data means big computers!



# Cluster computing is mainstream


Google Cloud Platform

# Course objective

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

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

# Software tools

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

# Software tools

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

**Examples:**

| Scientific | General purpose |
|---|---|
| Matlab | Python |
| Fortran | C++ |
| R | Java |

# Software tools

**Languages are *compiled* or *interpreted***

# Software tools

## Languages are *compiled* or *interpreted*

| Compiled | Interpreted |
| --- | --- |
| Fortran | Python |
| C++ | Matlab |
| Java | R |

# 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

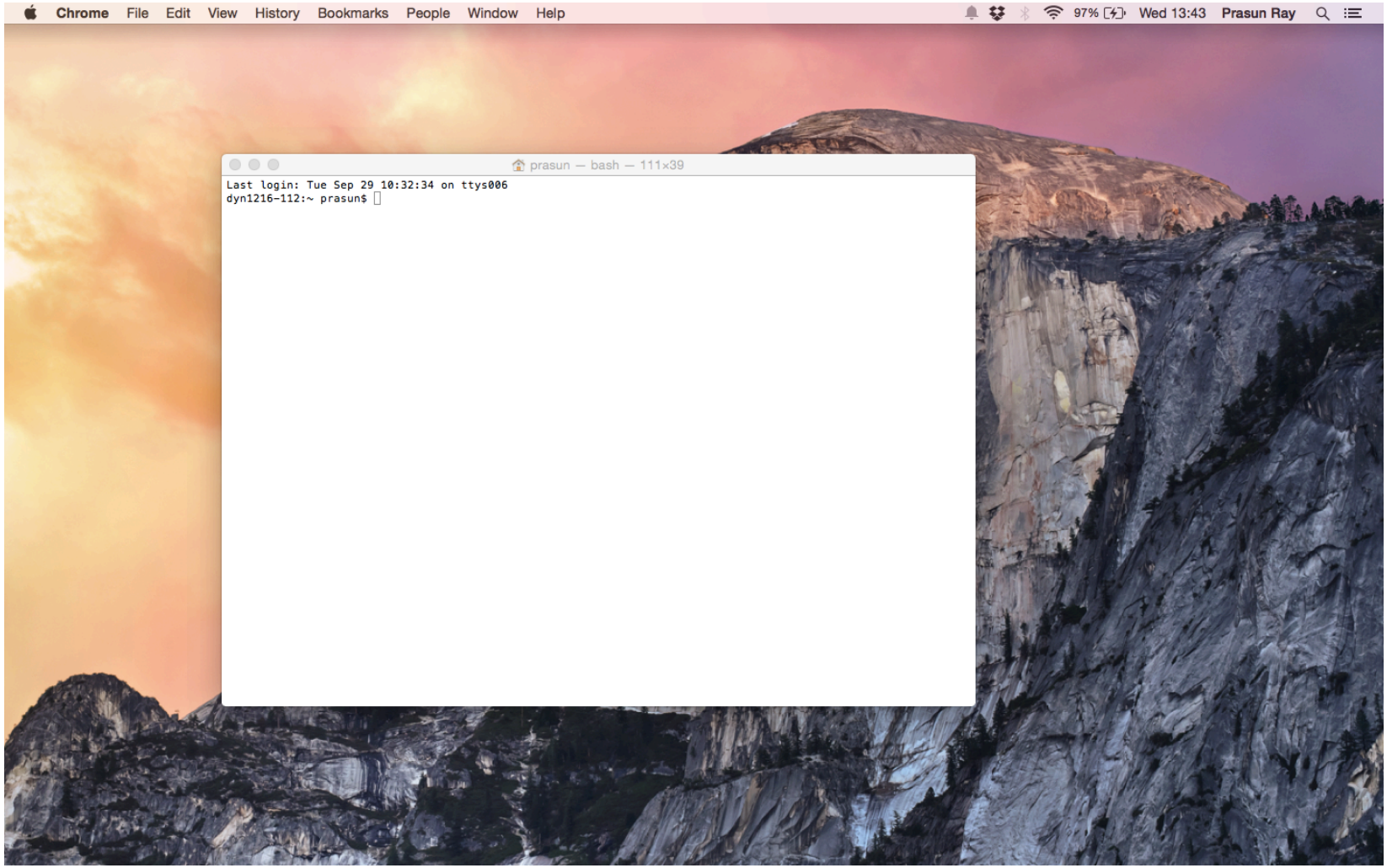
Windows:

- 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.io/

# Unix terminal

## Terminal on a mac:



# 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/m3c2017
$
$ ls
Readme.md lectures
$
$ cd lectures
$
$ ls
lecture1
```

# 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
```

# 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.
```

# 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.**

# 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

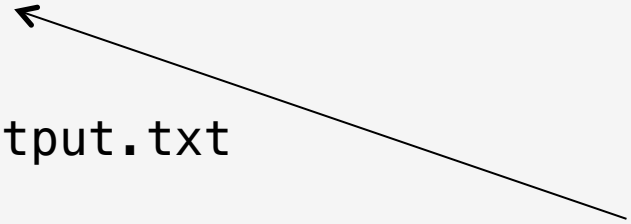
**Instead of outputting to screen, can output to file using “>”**

```
$ 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?**

# What next?

- If you have your own laptop/desktop: start installing course software – see webpage for instructions
- If you don't, try out the virtual machines and these Unix commands on the MLC computers -- look for the "oel" icon on the desktop
- Start working through introductory Python videos and exercises
  - 1st video should be online Friday morning