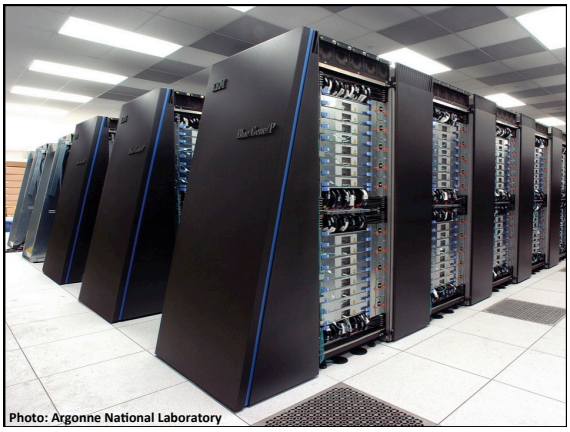


Photo: Argonne National Laboratory

High Performance Computing (HPC)

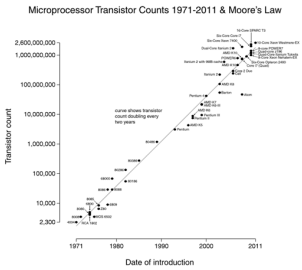
- Introduction to HPC and why it's useful
- How do you parallelize your code?
- The practice of running software on a cluster

Introduction to HPC and why it's useful

Gordon Moore

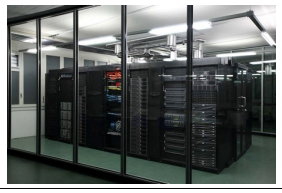
Born	January 3, 1929 (age 84) San Francisco, California, USA
Nationality	American
Alma mater	University of California, Berkeley; California Institute of Technology
Occupation	Chairman Emeritus, Intel Corporation
Net worth	▲$4 billion USD (2011)[1]

Jan 2015 update: $6.7 billion



Introduction to HPC and why it's useful

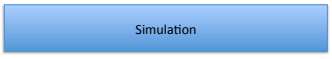
- Embarrassingly parallel problems
  - Graphics
  - Simulations with multiple parameters
- Non embarrassingly parallel problems
  - Fluid dynamics
  - A lot of the tasks run by a single program



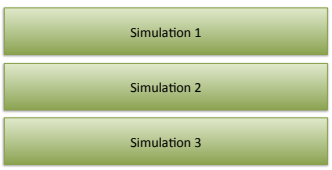
How do you parallelize your code? !

- `as.numeric(Sys.getenv("PBS_ARRAY_INDEX"))`
  - Should be used in your code to give a simulation number

Before



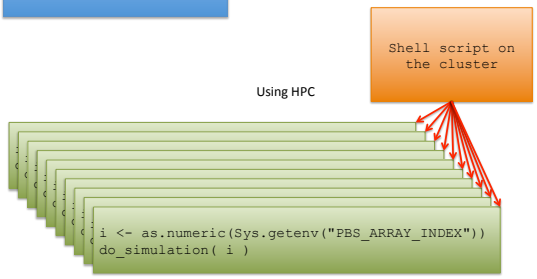
Now



Using your PC

```
for ( i in 1 : 10 )
{
   do_simulation( i )
}
```

Using HPC

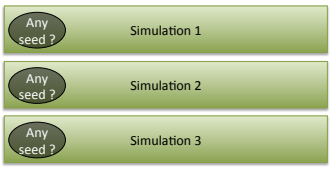


How do you parallelize your code?

- `as.numeric(Sys.getenv("PBS_ARRAY_INDEX"))`
  - Should be used in your code to give a simulation number

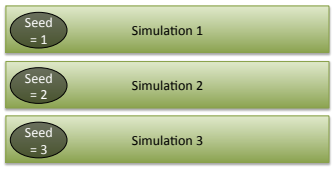
Before

Now

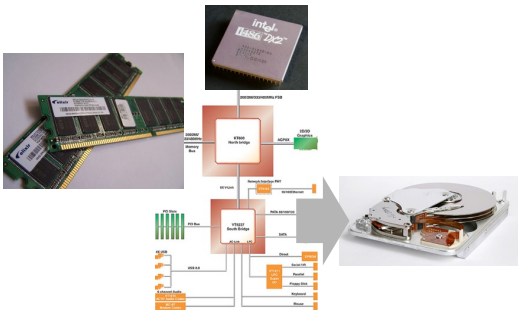


How do you parallelize your code?

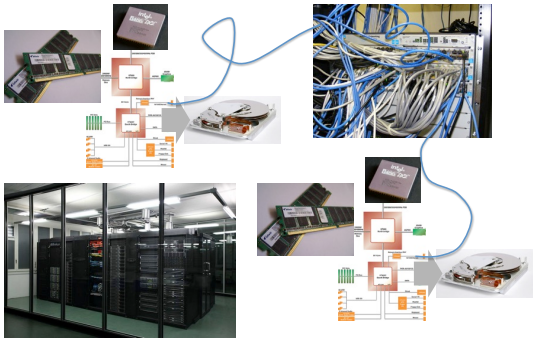
- `as.numeric(Sys.getenv("PBS_ARRAY_INDEX"))`
  - Should be used in your code to give a simulation number
- Pseudo random numbers
  - Given a certain random number seed, you get the same sequence of random numbers every time
  - Each simulation should have a different seed



Handling memory

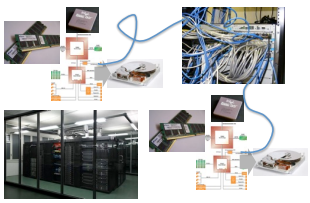


Handling memory



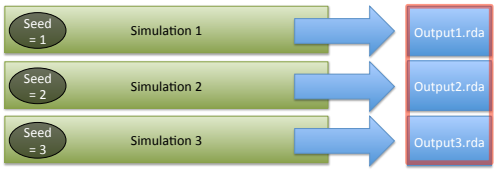
Handling memory

- Save your results in memory and then write to disk at the end



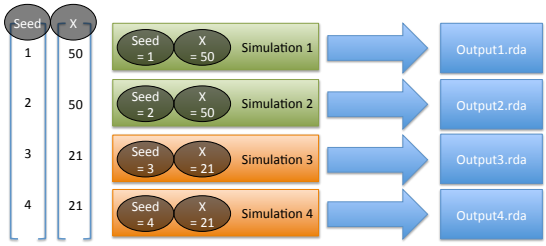
Handling memory

- Save your results in memory and then write to disk at the end
- Output your code to a series of files
- Write local code to read in your series of files automatically



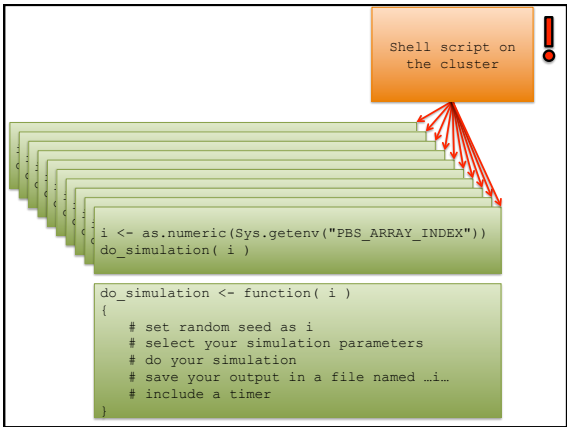
Handling memory

- Save your results in memory and then write to disk at the end
- Output your code to a series of files
- Write local code to read in your series of files automatically



Handling memory

- Save your results in memory and then write to disk at the end
- Output your code to a series of files
- Write local code to read in your series of files automatically
- Build a timer into your code
- Test your code locally to know your memory and time requirements



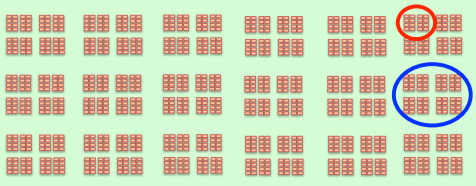
The practice of running code on a cluster

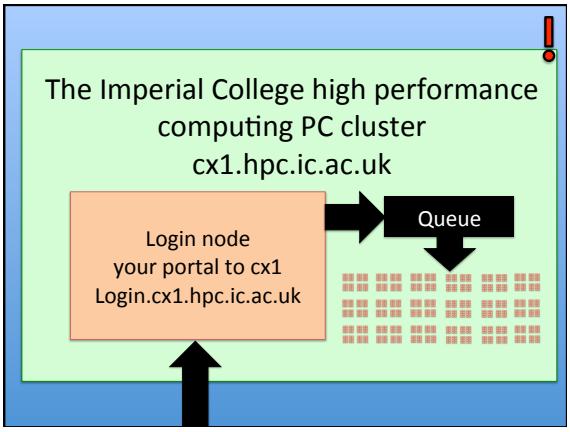
Imperial College high performance computing hpc.ic.ac.uk

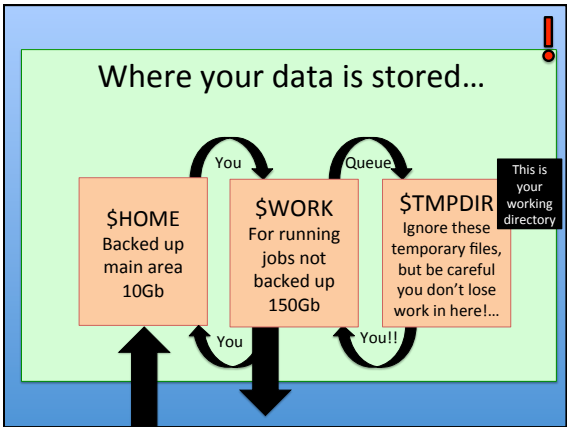
cx1	cx2	ax3
A cluster of many ordinary computers	~~Massively parallel system~~	~~Large shared memory tasks~~

The Imperial College high performance computing PC cluster cx1.hpc.ic.ac.uk

The Imperial College high performance computing PC cluster cx1.hpc.ic.ac.uk







Step 1: get your code onto the cluster

- Use sftp: from the directory of your code in a shell window type ….
  - `sftp username@login.cx1.hpc.ic.ac.uk`
  - You will be asked for your standard cluster password
  - `cd $HOME`
  - `put filename.R`
  - `exit`
- Your software is now on your home directory of the cluster which is backed up regularly

Step 2: log into the cluster

- Use ssh: from a shell window type ….
  - `Ssh -l username login.cx1.hpc.ic.ac.uk`
  - You will be asked for your standard cluster password
  - Now it's as though you were sitting with a shell open at the login node.
  - `ls` (will list the files in $HOME)
  - `mkdir foldername` (make a new folder)
  - `mv filename $HOME/foldername` (move)
  - `cd foldername` (change directory)
  - `cat filename` (see your file to check it's contents)
  - `cp filename $WORK` (make a copy in $WORK)
  - `cd $WORK` (change directory to $WORK)
- Your software is now in two copies: in a folder in your home directory of the cluster and in the work area ready for running.

Step 3: make a file for your shell script

- This is the list of instructions to be executed when you get to the front of the queue is written in shell script. It should be a .sh file
- ***Never*** run code on the login node – always write a shell script and wait in the queue.
- If you type `cat > filename.sh`
- You will then get the chance to type text (pressing enter for new lines) and the cat command will make the file containing the text that you typed.
- When you are finished typing the contents of your new file press control and D to complete the process.
- Type `cat filename.sh` to check that your file is correct before submitting it to the queue.

Step 3 continued: your shell script file

```
#!/bin/bash
#PBS -l walltime=12:00:00
#PBS -l select=1:ncpus=1:mem=800mb
module load R/2.13.0
module load intel-suite
echo "R is about to run"
R --vanilla < $WORK/Rtest/ForwardsNTC_V5.R
mv datafilename* $WORK
echo "R has finished running"
# this is a comment at the end of the file
```

You can run Python code too –
just use different commands here

Step 4: submitting your job to the cluster

- You are now in the $WORK directory with your code and shell script both written.
- To submit your job type
  `qsub -J 1-32 filename.sh`
  `qstat` (S changes from Q to B when running)
- If you want to delete a job
  `qstat`
  `qdel job-id[]` (the [] is for array jobs only)
- qstat will give you a list of jobs and you would get the job-id from there.

Step 5: check that all is well

- Wait 5-10 minutes then check that nothing has gone wrong.
- `qstat` (is your job running still)
- `ls` (are output files as expected)
- `cat filename.sh.ejob-id.index` (are error files empty?)
- `cat filename.sh.ojob-id.index` (are standard output files as expected)
- `qstat` (is your job running still)
- `exit` (you're done for now come back later)

Step 6: Getting your results back

- `qstat` (is your job running still)
- `cd $WORK`
- `ls` (output files as expected?)
- `cat output filename` (contents as expected?)
- `cat filename.sh.ejob-id.index` (error files empty?)
- `cat filename.sh.ojob-id.index` (standard output files as expected?)
- `tar czvf filename.tgz *`
- `mv filename.tgz $HOME/foldername`

Step 6 continued: sftp to get results

- `exit`
- Use sftp: from a new directory on your own computer of where you want the results to be. Open a shell and type …
  - `sftp username@login.cx1.hpc.ic.ac.uk`
  - You will be asked for your standard cluster password
  - `cd $HOME/foldername`
  - `get filename.tgz`
  - `exit`
- Your results are now all on your own computer
  - `tar xzvf filename.tgz`
- Your results are now complete uncompressed and ready for use. Now you need to write some R code to read in and analyze all those file.