Powering Success
Pweru Llwyddiant

Using HPC Wales

Ewrop & Chymru:
Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop

Europe & Wales:
Investing in your future
European Regional Development Fund

ERDF

Llywodraeth Cymru
Welsh Government

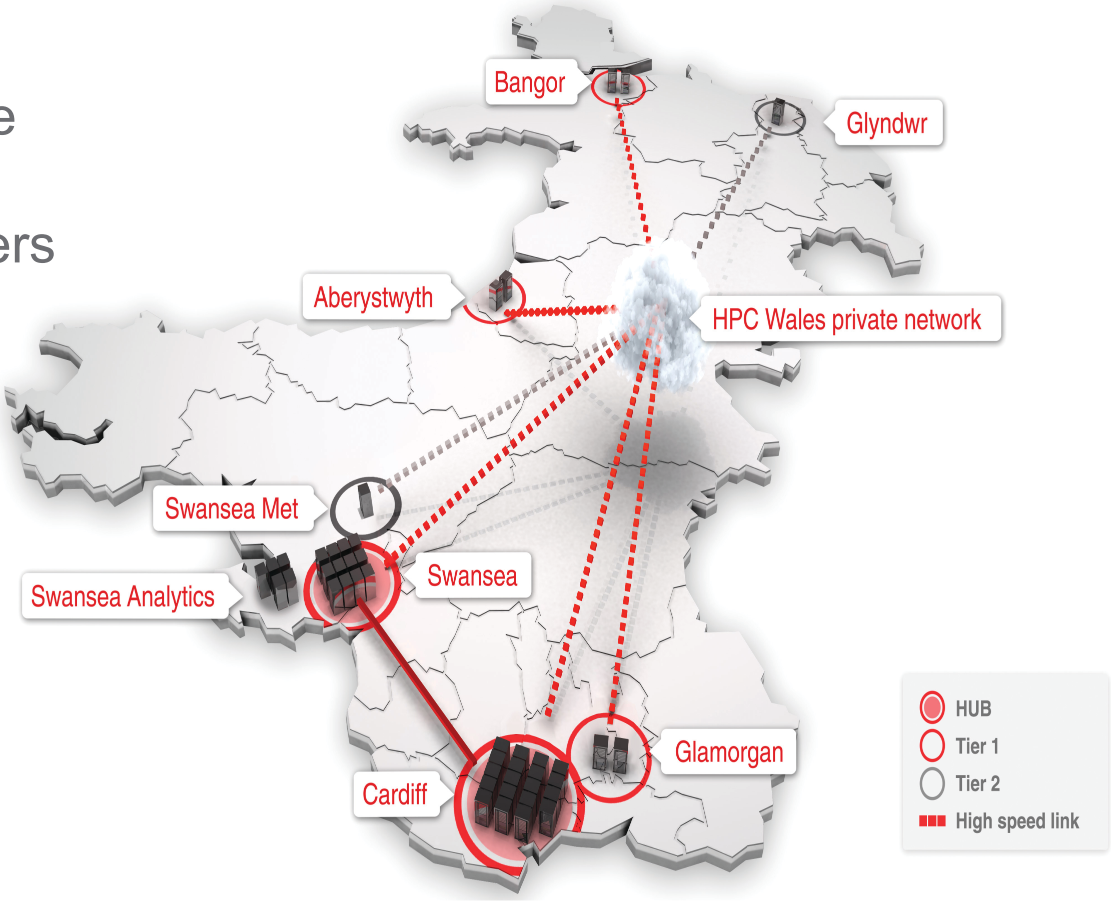
Agenda

- Infrastructure : An Overview of our Infrastructure
- Logging in : Command Line Interface and File Transfer
- Linux Basics : Commands and Text Editors
- Using Modules : Managing Software and the Environment
- Submitting Jobs : Using the Job Scheduler
- Examples : Hello World, Matrix, and IMB

INFRASTRUCTURE

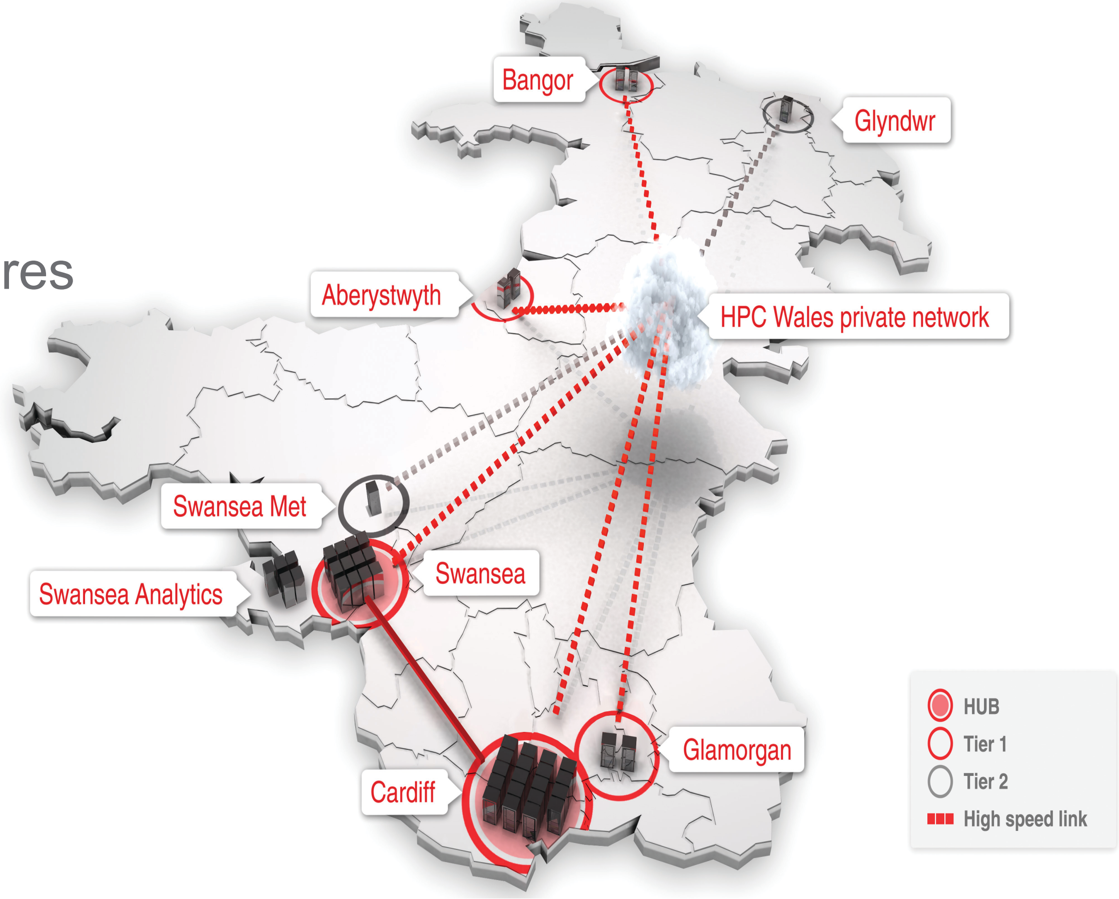
The Network

- HPC Wales offers a secure pan Wales distributed network of computer clusters
- HPC Wales technology provision is based on a distributed hub and spoke model
- That model provides resilience, shared storage, and a rich application environment



The Cardiff Hub

- 2010 – 2012 – Phase 1
  - Capability system
    - ~ 2000 Westmere Cores
- 2013 – 2015 – Phase 2
  - Capacity system
    - ~ 6000 Sandy Bridge Cores



hpcwales.co.uk

The Cardiff Capability System

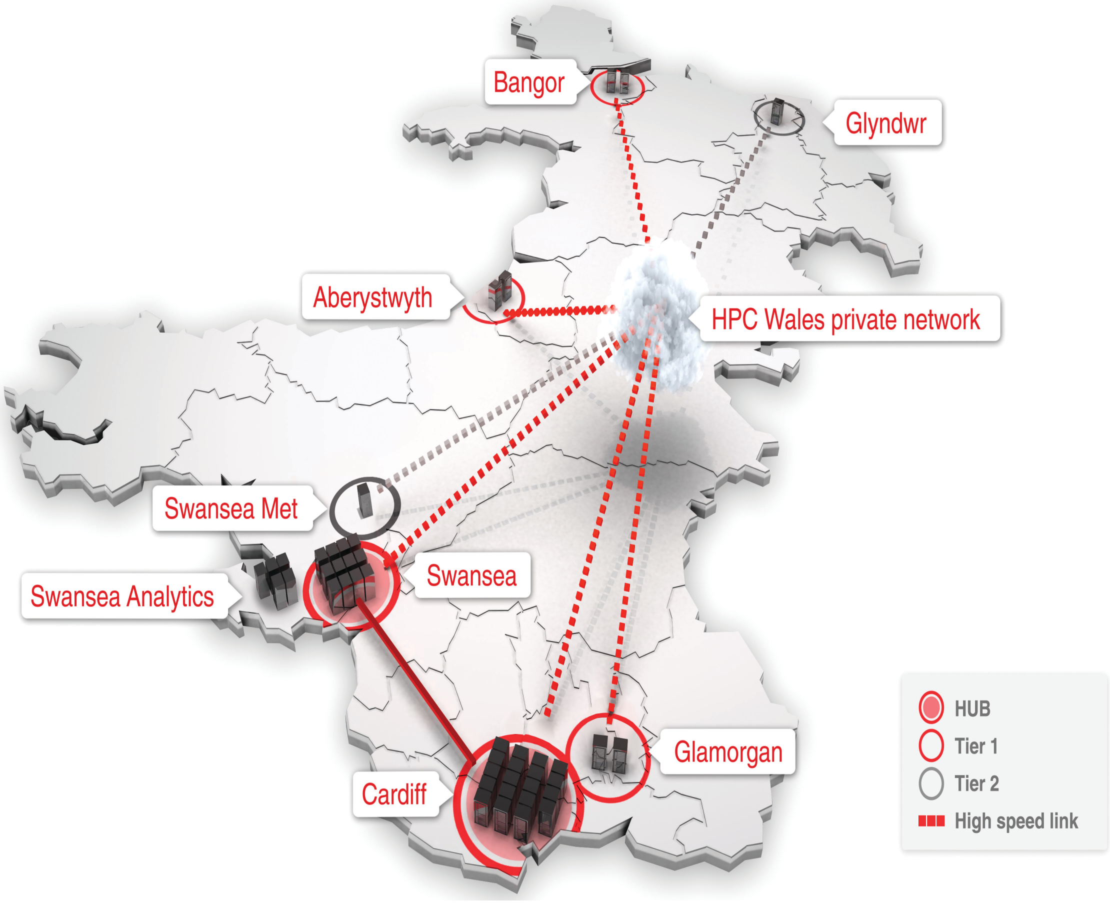
- 162 BX922 Nodes (Windows and Linux OS)
- 12 cores and 36 Gb memory per node
- Intel Westmere X5650 at 2.67 GHz
- Mellanox Infiniband (1.2 usec latency and 40 Gbps bandwidth)
- 75 Tb NFS File system (providing /home)
- 200 Tb Lustre File system (providing /scratch)

The Cardiff Capacity System

- 384 CX250 Nodes
- 16 cores and 64Gb memory per node
- Intel Sandy Bridge E5-2670 at 2.6GHz
- Mellanox Infiniband (1.2 usec latency and 40 Gbps bandwidth)
- 75 Tb NFS File system (providing /home)
- 200 Tb Lustre File system (providing /scratch)

The Swansea Hub

- 2013 – 2015 – Phase 2
  - Capability system
    - ~ 4000 Sandy Bridge Cores
  - Capacity system
    - ~ 2000 Sandy Bridge Cores



hpcwales.co.uk

The Swansea Capability System

- 240 CX250 Nodes
- 16 cores and 64 Gb memory per node
- Intel Sandy Bridge E5-2690 at 2.9 GHz
- Mellanox Infiniband (1.2 usec latency and 40 Gbps bandwidth)
- 100 Tb NFS File system (providing /home)
- 400 Tb Lustre File system (providing /scratch)

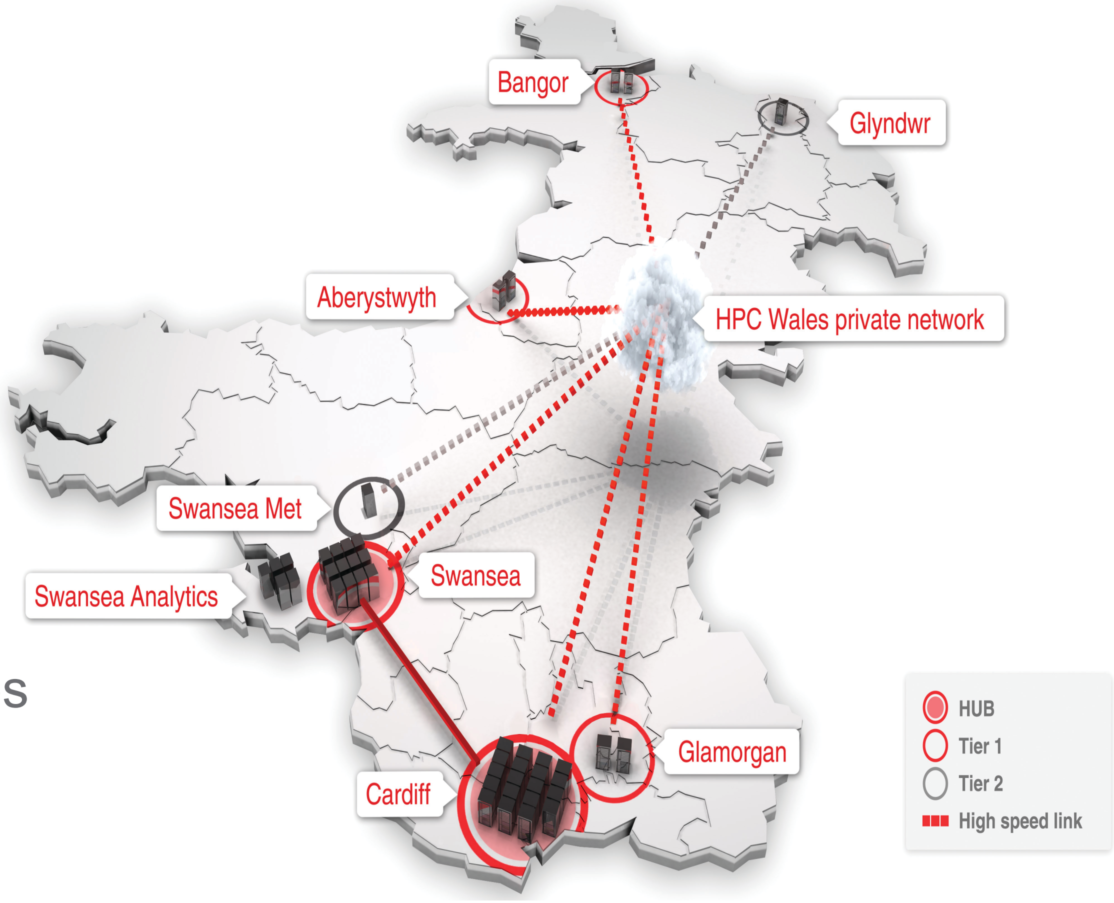
The Swansea Capacity System

- 128 CX250 Nodes
- 16 cores and 64 Gb memory per node
- Intel Sandy Bridge E5-2670 at 2.6 GHz
- Mellanox Infiniband (1.2 usec latency and 40 Gbps bandwidth)
- 100 Tb NFS File system (providing /home)
- 400 Tb Lustre File system (providing /scratch)

The Tier 1 Sites

- Aberystwyth
- Bangor
- Glamorgan

- 2010 – 2012 – Phase 1
  - Capacity systems
    - ~ 650 Westmere cores



The Tier 1 Systems – Aber – Bangor – Glamorgan

- 54 BX922 Nodes
- 12 cores and 36 Gb memory per node
- Intel Westmere X5650 at 2.67 GHz
- Mellanox Infiniband (1.2 usec latency and 40 Gbps bandwidth)
- 8 Tb NFS File system (providing /home)

LOGGING IN

Logging In

- You login to the cluster using something called a Terminal Emulator which allows you to connect your keyboard and screen to the remote system
- The protocol used is called Secure Shell or SSH
- On the Windows platform you can install and then use the Putty Terminal Emulator

http://www.chiark.greenend.org.uk/~sgtatham/putty/

- On the Linux and Mac platforms you can use the Terminal which is usually already installed

Transferring Files

- You transfer files to and from the cluster using something called a File Transfer Program which allows you to connect your computer to the remote system
- The protocol used is called Secure FTP or SFTP
- On Windows, Linux, and Mac platforms you can install and then use the FileZilla File Transfer Program

http://filezilla-project.org/

Portable Applications

- If you cannot install Putty or Filezilla due to a lack of administrator rights on your machine, then you may be able to use portable applications instead

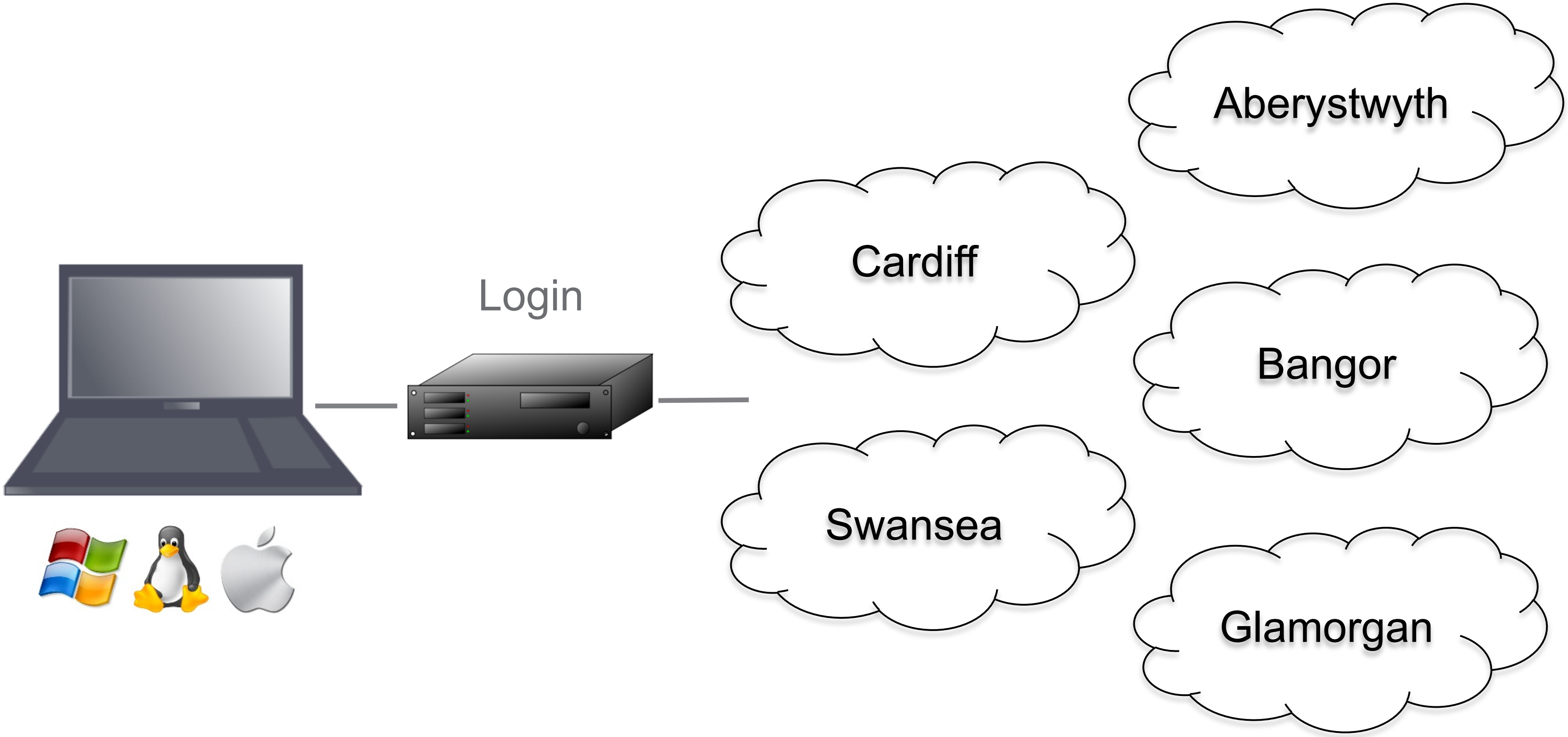
http://portableapps.com/apps/internet/putty_portable

http://portableapps.com/apps/internet/filezilla_portable

Logging In

- Logging into the cluster is a two stage process
- First you login to a generic front end machine e.g. login.hpcwales.co.uk

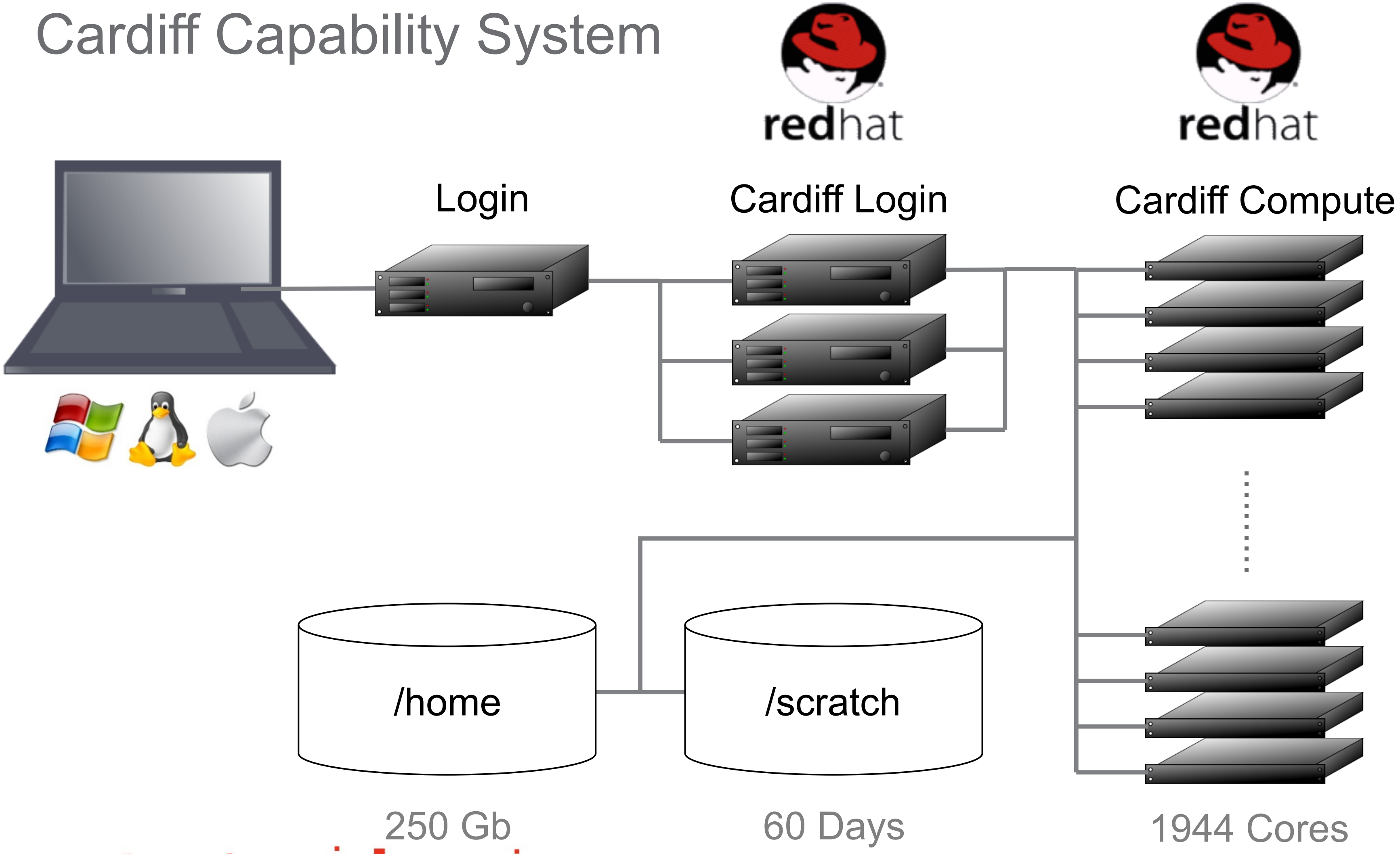
Accessing HPC Wales



Logging In

- From there you can list the available clusters
  - e.g. **hpcwhosts**
- Then you login to the specific cluster of interest
  - e.g. **ssh cf-log-001**

Cardiff Capability System



LINUX BASICS

Command Prompt Basics

- **man man**
  - Displays manual information on the manual command
- **man [command]**
  - Displays manual information on command
- **clear**
  - Clears the screen
- **exit**
  - Exits the command interpreter

Manipulating Directories

- **cd ..**
  - Change to the parent directory
- **cd [directory]**
  - Change to directory [directory]
- **mkdir [directory]**
  - Create directory [directory]
- **rmdir [directory]**
  - Remove directory [directory]

Listing Files

- **ls**
  - Display list of files and sub directories in standard format < name > excluding hidden files
- **ls -a**
  - Display list of files and sub directories in standard format < name > including hidden files
- **ls -l**
  - Display list of files and sub directories in long format < permissions owner group size date time name >

Listing Files

- **ls -lh**
  - Display list of files and sub directories in long format < permissions owner group size date time name > with human readable size
- **ls -lt**
  - Display list of files and sub directories in long format < permissions owner group size date time name > sorted by time
- **ls -lr**
  - Display list of files and sub directories in long format < permissions owner group size date time name > in reverse order

Listing Files

- **ls -ltrh**
  - Display list of files and sub directories in long format < permissions owner group size date time name > sorted by time, in reverse order, with human readable size

Moving Files

- **mv [source] [dest]**
  - Move file [source] to file [dest]
- **mv -i [source] [dest]**
  - Move file [source] to file [dest]
  - Prompt before overwriting [dest] if it exists
- **mv -f [source] [dest]**
  - Move file [source] to file [dest]
  - Overwrite [dest] if it exists

Removing Files

- **rm [file]**
  - Remove file [file]
- **rm -i [file]**
  - Remove file [file]
  - Prompt before removing
- **rm -R [directory]**
  - Remove directory [directory]
  - Remove all sub directories and files

Copying Files

- **cp [source][dest]**
  - Copy file [source] to file [dest]
- **cp -i [source][dest]**
  - Copy file [source] to file [dest]
  - Prompt before overwriting [dest] if it exists
- **cp -R [source][dest]**
  - Copy directory [source] to directory [dest]
  - Copy all sub directories and files

Displaying Files

- **more [file]**
  - Display [file] on the screen
  - Will scroll through one screen at a time
  - Press space to scroll one screen at a time
  - Press enter to scroll one line at a time

Editing Files with Nano

- A simple text editor
- Installed on HPC Wales clusters
- Not installed on all Linuxes by default
- Commands in CTRL key format
- A list of commands is not required

- **nano**
  - Open the nano file editor
- **nano [file]**
  - Open [file] in the nano file editor

Editing Files with Emacs

- A powerful / complicated text editor
- Installed on HPC Wales clusters
- Not installed on all Linuxes by default
- Commands in CTRL key format
- A list of commands will be provided

- **emacs**
  - Open the emacs file editor
- **emacs [file]**
  - Open [file] in the emacs file editor

Starting Emacs

Command	Description
Emacs	run emacs
emacs /home/user/myfile.txt	run emacs and open myfile.txt

Leaving Emacs

Command	Description
CTRL-x, CTRL-c	quit emacs
CTRL-x, CTRL-s	save open file

File Operations

Command	Description
CTRL-x, CTRL-f, /home/user/myfile.txt	find and open myfile.txt (tab completion works)
CTRL-x, CTRL-s	save open file

Cursor Operations

Command	Description
ESC-f	move forwards one word
ESC-b	move backwards one word
CTRL-a	move to the beginning of the line
CTRL-e	move to the end of the line
ESC-a	move backwards one sentence
ESC-e	move forwards one sentence
ESC-{	move backwards one paragraph
ESC-}	move forwards one paragraph

Edit Operations

Command	Description
CTRL-x, u	Undo
ESC-d	kill (cut) a word
CTRL-k	kill (cut) a line
CTRL-w	kill (cut) highlighted region
ESC-w	kill (copy) highlighted region
CTRL-y	yank (paste) highlighted region

Search and Replace

Command	Description
CTRL-s	search forwards for instances of string entered at prompt
CTRL-r	search backwards for instances of string entered at prompt
ESC-SHIFT-5	interactively replace string entered at prompt with next string
<space>	replace text and find next occurrence
<del>	leave text and find next occurrence
.	replace text then stop looking
!	replace all occurrences without asking again

Editing Files with Vi

- A powerful / complicated text editor
- Installed on HPC Wales clusters
- Installed on all Linuxes by default
- Commands in COLON key format
- A list of commands will be provided

- **vi**
  - Open the vi file editor
- **vi [file]**
  - Open [file] in the vi file editor

Cursor Operations

Command	Description
[repeat]w	move forwards [repeat] words
[repeat]b	move backwards [repeat] words
^	move to the beginning of the line
0	move to the beginning of the line
[repeat]f [letter]	move forwards to the [repeat] instance of [letter]
[repeat]F [letter]	move backwards to the [repeat] instance of [letter]
[number] G	move to line [number]
H	move to the home line (first line on the screen)
M	move to the middle line (on the screen)
L	move to the last line (on the screen)
(	move backwards one sentence
)	move forwards one sentence
{	move backwards one paragraph
}	move forwards one paragraph

Starting Vi

Command	Description
vi	run vi
vi /home/user/myfile.txt	run vi and open myfile.txt
vim	run vim
vim /home/user/myfile.txt	run vim and open

Leaving Vi

Command	Description
ZZ	quit vi
:q:wq	quit vi
:wq	Write open file and quit
q!	quit vi and do not write open file

File Operations

Command	Description
e/home/user/myfile.txt	edit myfile.txt (tab completion works)
:w/home/user/myfile.txt	write myfile.txt (tab completion works)
:w	write open file

Modes

Command	Description
ESC	return to command mode
j	change to insert mode
a	change to append mode
o	change to open mode

Edit Operations

Command	Description
u	undo
[repeat]dw	delete (cut) [repeat] lines
[repeat]dd	delete (cut) [repeat] lines
dG	delete (cut) to end of file
[repeat]yw	yank (copy) [repeat] words
[repeat]yy	yank (copy) [repeat] lines
p	put (paste)

Search and Replace

Command	Description
[repeat]/[string]	search forwards to the [repeat] instance of [string]
:s/[old-string]/[new-string]	search and replace the first instance of [old-string] with [new-string] on this line
:s/[old-string]/[new-string]/g	search and replace all instances of [old-string] with [new-string] on this line
:%s/[old-string]/[new-string]/g	search and replace all instances of [old-string] with [new-string] in this file

Comparing Files

- **diff [file1] [file2]**
  - Display differences between [file1] and [file2]
- **fgrep “string” [file]**
  - Find “string” in [file]
- **sort [file]**
  - Sort [file]

Command Modifiers

- Wildcards allow you to specify multiple items to operate on
  - ls *.txt   rm *.txt
- Redirection allows you to direct the output of one command to a file
  - sort unsorted.txt > sorted.txt
- Filters are external commands that change data in some manner
  - fgrep "string" [file]
- Pipes let you direct the output of one command as input to another
  - ls | find "txt"

Other Commands

- **who**
  - Show who is logged on
- **top**
  - Show which tasks are running
- **watch**
  - Run a task repeatedly
- **history**
  - Show which tasks you ran
- **date**
  - Display or set the date and time

Other Commands

- **cat**
  - Concatenate files and print on screen
- **head**
  - Print top of file on screen
- **tail**
  - Print bottom of file on screen
- **uniq**
  - Report or omit repeated lines

USING MODULES

What are Modules ?

- A consistent way of setting up your environment, which contains important information
  - In particular the locations of the specific versions of the compilers, libraries and applications you want to use whilst logged in or running a job through the scheduler
  - You might want to load a different combination of compilers, libraries and applications for each computation you want to run

Module Commands

- **module avail**
  - List all of the available modules
- **module list**
  - List the modules in your environment
- **module load module_name**
  - Load module_name into your environment

Module Commands

- module unload module_name
  - Unload module_name from your environment
- module purge
  - Unload all modules from your environment

JOB SCHEDULER

What is LSF ?

- The job scheduler that runs on the clusters
- It tracks the status of all compute nodes
- It tracks the status of all jobs
- It queues jobs until there are free nodes
- It runs jobs and monitors their progress
- It is what you use to run jobs

LSF Commands

- **bjobs**
  - List the status of my jobs
- **bjobs –l**
  - As above plus list the compute nodes used
- **bjobs –u all**
  - List the status of all jobs in all queues
- **bjobs –u all –r**
  - List all currently running jobs
- **bjobs –u all –p**
  - List all currently pending jobs

LSF Commands

- **bsub < jobscript**
  - Submit jobscript to the queue
- **bkill jobid**
  - Remove jobid from the queue

EXAMPLES

Example

- The first example is a hello world program that shows you how to compile and run a parallel program

Hello World

- > **cd Onboarding**
- > **ls**
- Hello IMB Matrix
- > **cd Hello**
- > **ls**
- clean.sh hello.f90 make.sh run.lsf

Hello.f90

```
program hello
  include 'mpif.h'
  integer mpierr, rank, procs
  call MPI_Init ( mpierr )
  call MPI_Comm_size ( MPI_COMM_WORLD , procs , mpierr )
  call MPI_Comm_rank ( MPI_COMM_WORLD , rank , mpierr )
  write (*,*) 'Hello world from ', rank, 'of', procs
  call MPI_Finalize ( mpierr )
end program hello
```

Run.lsf

```
#!/bin/bash --login
#BSUB -x                 # give this job exclusive access
#BSUB -n 12                      # give this job 12 cores
#BSUB -o HELLO.out           # put the output stream here
#BSUB -e HELLO.err            # put the error stream here
#BSUB -J HELLO                      # give the job a name
#BSUB -W 01:00      # run the job for no more than 1 hour
#BSUB -R "span[ptile=12]"       # fully populate the node
#BSUB -q q_cf_htc_work    # run on the cardiff htc system
```

Run.lsf

```
# Load the Environment
module purge                       # purge any loaded modules
module load compiler/intel-12.0.084   # use this compiler
module load mpi/intel-4.0.0.028            # use this MPI

# Run the Program
mpirun -n $LSB_DJOB_NUMPROC ./hello.exe >& log.HELLO.
$LSB_JOBID
```

Hello World

- > **./clean.sh**
- > **./make.sh**
- > **bsub < run.lsf**
- Job <…> is submitted to queue <…>
- > **bjobs**

Log.Hello.<>

1

```
Hello world from          0 of          12
Hello world from          1 of          12
Hello world from          5 of          12
Hello world from          3 of          12
Hello world from          2 of          12
Hello world from          6 of          12
Hello world from         10 of          12
Hello world from         11 of          12
Hello world from          8 of          12
Hello world from          9 of          12
Hello world from          4 of          12
Hello world from          7 of          12
```

Example

- The second example is a series of matrix multiplication programs that form a simple benchmark and show you the effect of using various compiler options

Matrix

- **> cd ..**
- **> cd Matrix**
- **> ls**
- clean.sh make.sh nodgemm1k.f90 nodgemm2k.f90 nodgemm3k.f90 nodgemm4k.f90 nodgemm5k.f90 run.lsf

Run.lsf

```
#!/bin/bash --login
#BSUB -x                 # give this job exclusive access
#BSUB -n 1                         # give this job 1 core
#BSUB -o MATRIX.out          # put the output stream here
#BSUB -e MATRIX.err           # put the error stream here
#BSUB -J MATRIX                     # give the job a name
#BSUB -W 03:00     # run the job for no more than 3 hours
#BSUB -R "span[ptile=12]"      # fully populate the node
#BSUB -q q_cf_htc_work    # run on the cardiff htc system
```

Run.lsf

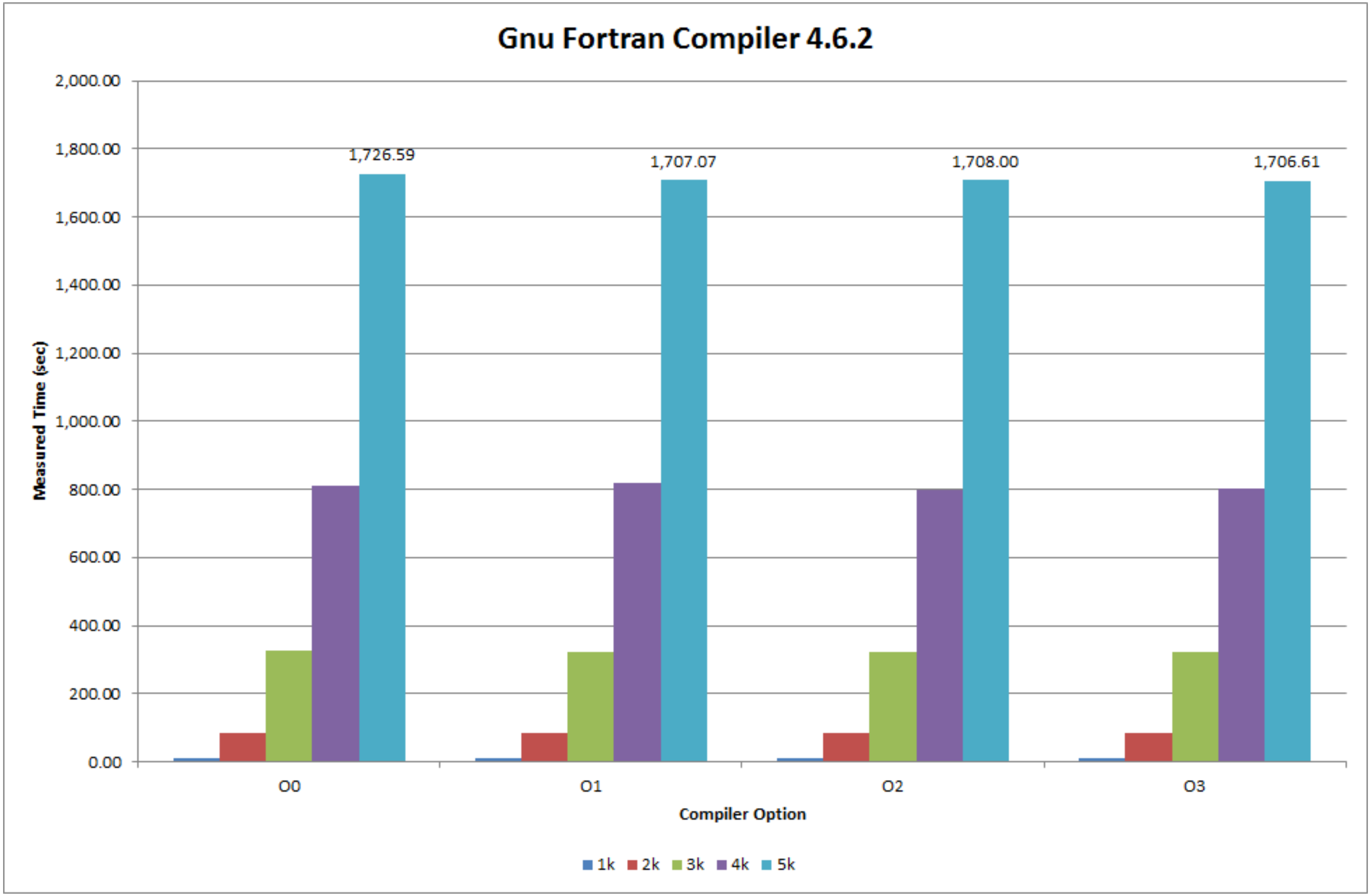
```
# Load the Environment
module purge                   # purge any loaded modules
module load compiler/intel-12.0.084   # use this compiler

# Run the Program
for PROG in $( ls *.exe )
  do
    echo $PROG
    ./$PROG
  done
```

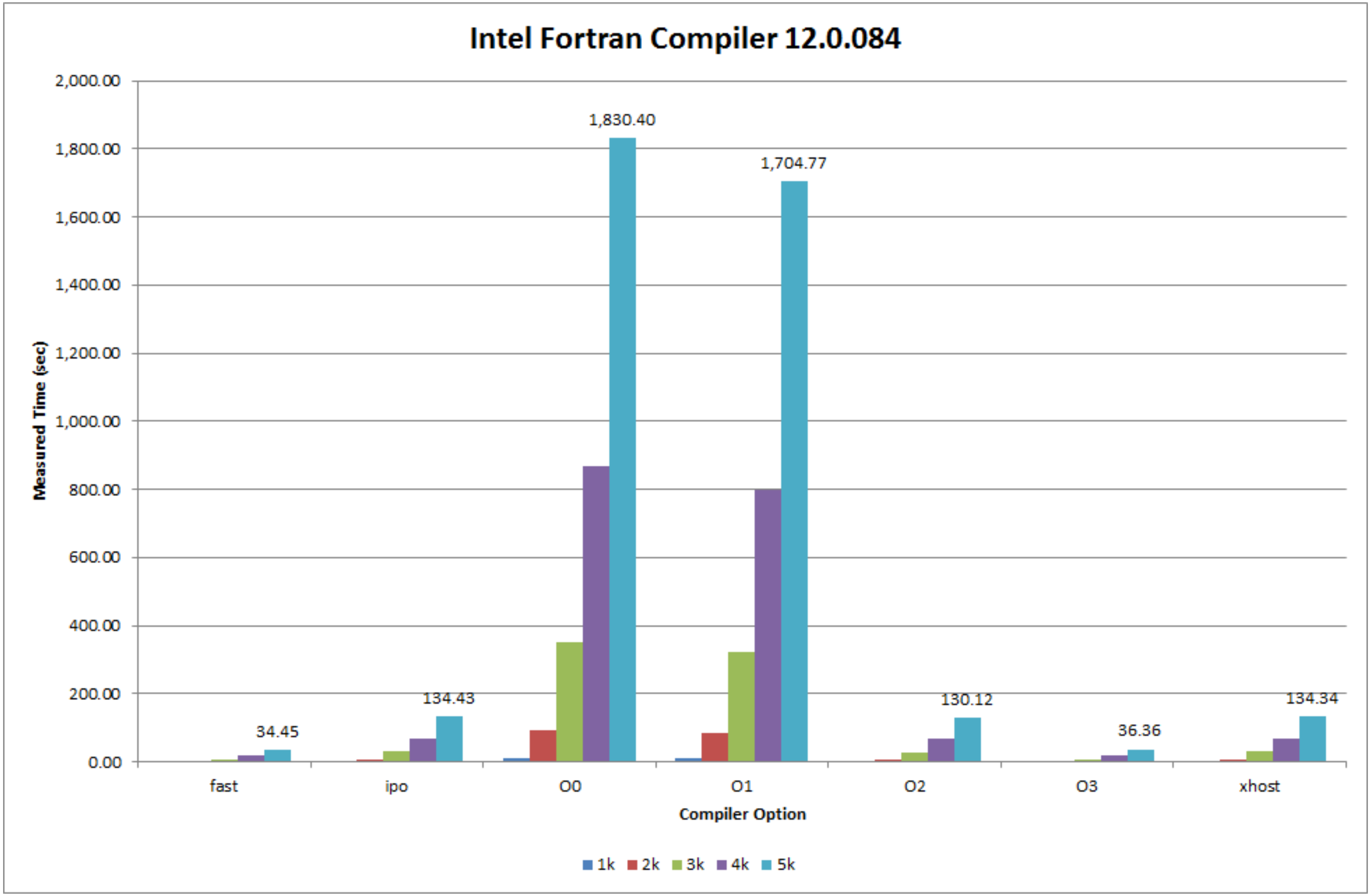
Matrix

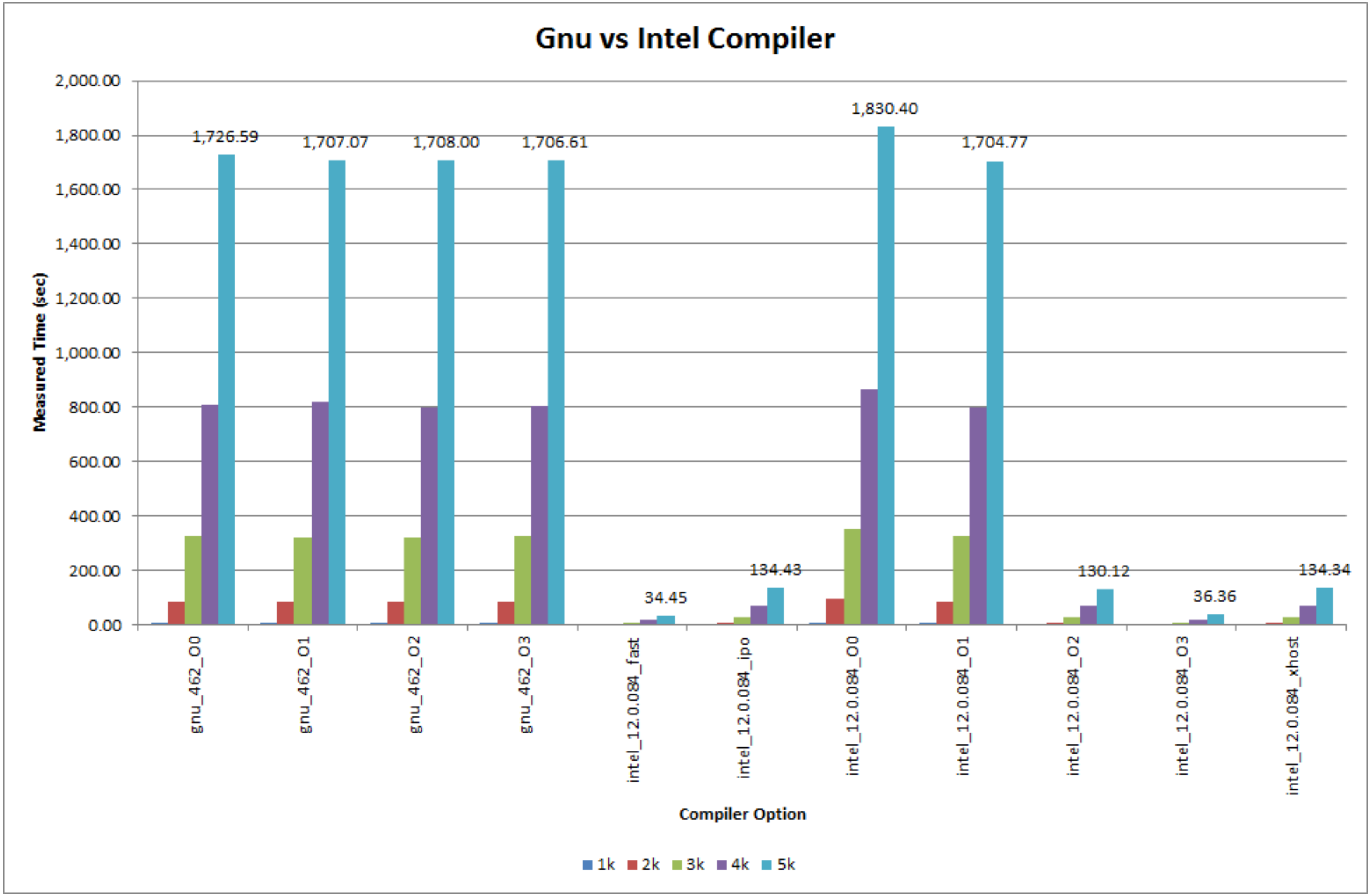
- > **./clean.sh**
- > **./make.sh**
- > **bsub < run.lsf**
- Job <…> is submitted to queue <…>
- > **bjobs**

Log.Matrix.<>

```
nodgemm1k.f90-fast.exe
 time for         1000  by         1000  is   0.2849570
seconds
nodgemm1k.f90-ipo.exe
 time for         1000  by         1000  is   0.6099080
seconds
nodgemm1k.f90-O0.exe
 time for         1000  by         1000  is    8.743671
seconds
```


Gnu Fortran Compiler 4.6.2
Measured Time (sec)
2,000.00
1,800.00
1,600.00
1,400.00
1,200.00
1,000.00
800.00
600.00
400.00
200.00
0.00
1,726.59
1,707.07
1,708.00
1,706.61
O0
O1
O2
O3
Compiler Option
1k
2k
3k
4k
5k


Intel Fortran Compiler 12.0.084
Measured Time (sec)
2,000.00
1,800.00
1,600.00
1,400.00
1,200.00
1,000.00
800.00
600.00
400.00
200.00
0.00
1,830.40
1,704.77
34.45
134.43
130.12
36.36
134.34
fast
ipo
O0
O1
O2
O3
xhost
Compiler Option
1k
2k
3k
4k
5k


Gnu vs Intel Compiler
Measured Time (sec)
2,000.00
1,800.00
1,600.00
1,400.00
1,200.00
1,000.00
800.00
600.00
400.00
200.00
0.00
1,726.59
1,707.07
1,708.00
1,706.61
34.45
134.43
1,830.40
1,704.77
130.12
36.36
134.34
gnu_462_O0
gnu_462_O1
gnu_462_O2
gnu_462_O3
intel_12.0.084_fast
intel_12.0.084_ipo
intel_12.0.084_O0
intel_12.0.084_O1
intel_12.0.084_O2
intel_12.0.084_O3
intel_12.0.084_xhost
Compiler Option
1k
2k
3k
4k
5k

Example

3

- The third example is the Intel Message Passing Interface Benchmark or IMB which is a parallel program that stresses the InfiniBand backplane

IMB

- > **cd ..**
- > **cd IMB**
- > **ls**
- clean.sh make.sh … run.lsf

Run.lsf

```
#!/bin/bash --login
#BSUB -x                 # give this job exclusive access
#BSUB -n 24                      # give this job 24 cores
#BSUB -o IMB.out             # put the output stream here
#BSUB -e IMB.err              # put the error stream here
#BSUB -J IMB                        # give the job a name
#BSUB -W 02:00     # run the job for no more than 2 hours
#BSUB -R "span[ptile=12]"       # fully populate the node
#BSUB -q q_cf_htc_work    # run on the cardiff htc system
```

Run.lsf

```
# Load the Environment
module purge                   # purge any loaded modules
module load compiler/intel-12.0.084   # use this compiler
module load mpi/intel-4.0.0.028            # use this MPI

# Run the Program
mpirun -n $LSB_DJOB_NUMPROC ./IMB-MPI1 >& log.IMB.
$LSB_JOBID
```

IMB

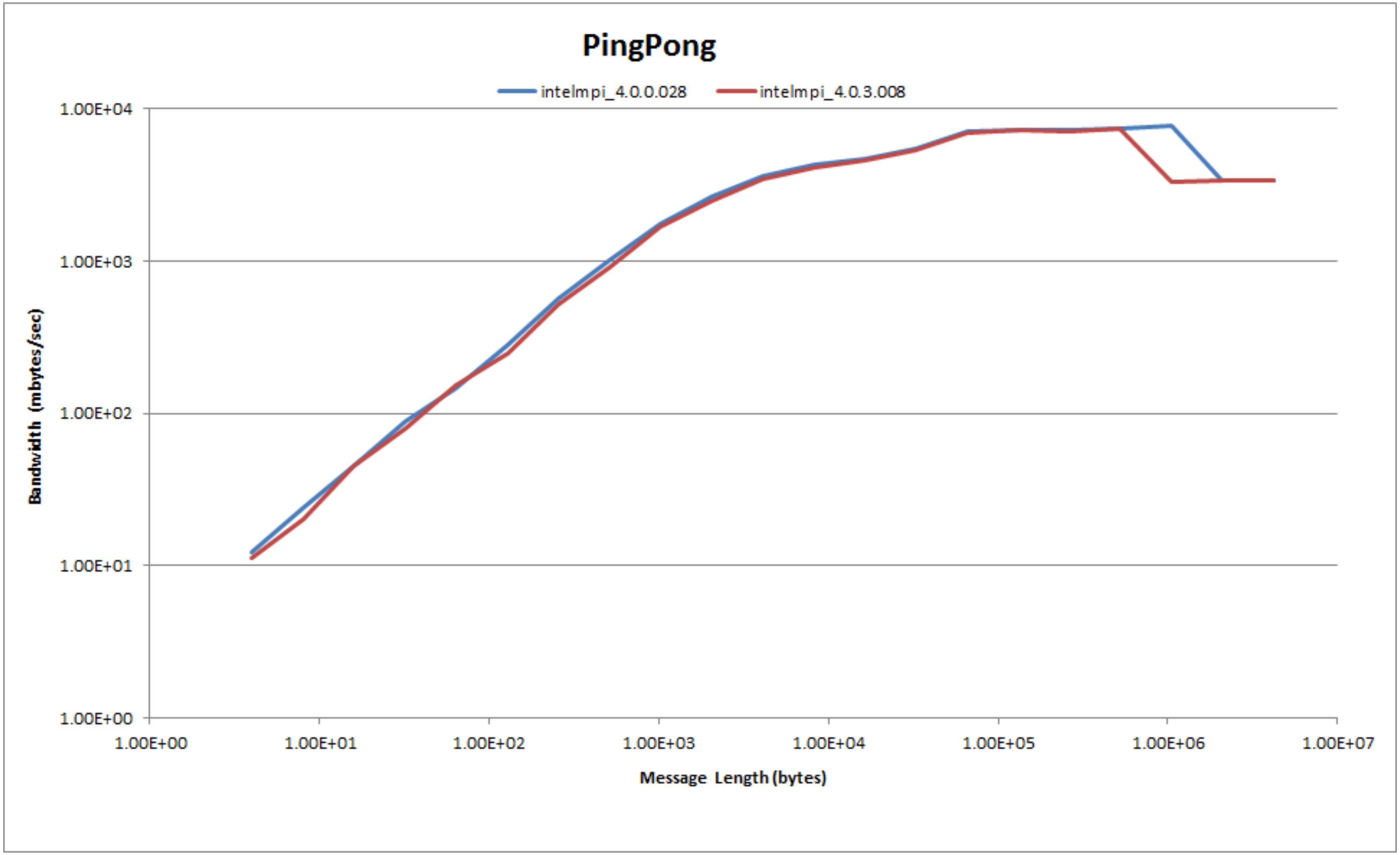
- > **./clean.sh**
- > **./make.sh**
- > **bsub < run.lsf**
- Job <…> is submitted to queue <…>
- > **bjobs**

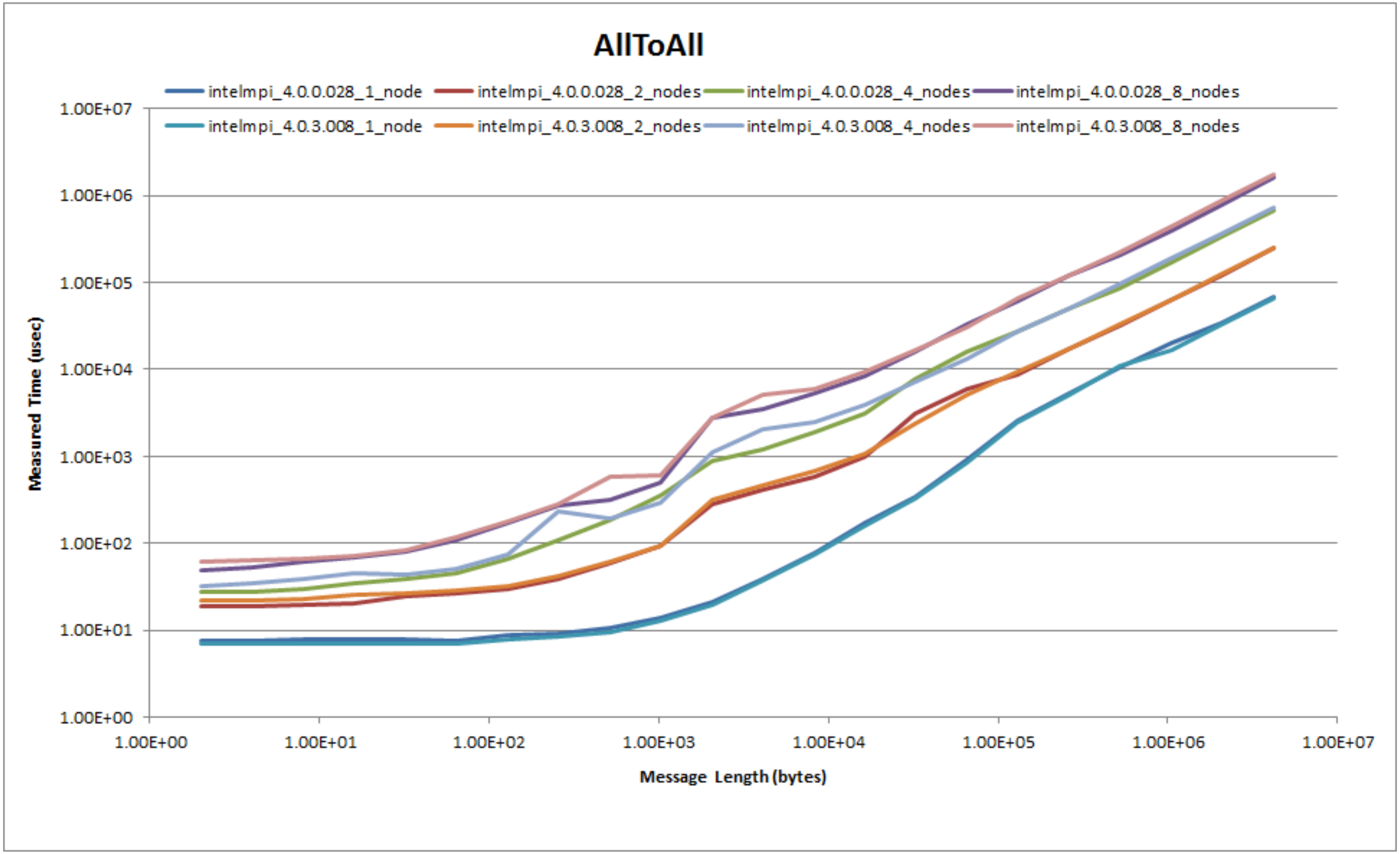
Log.IMB.<>

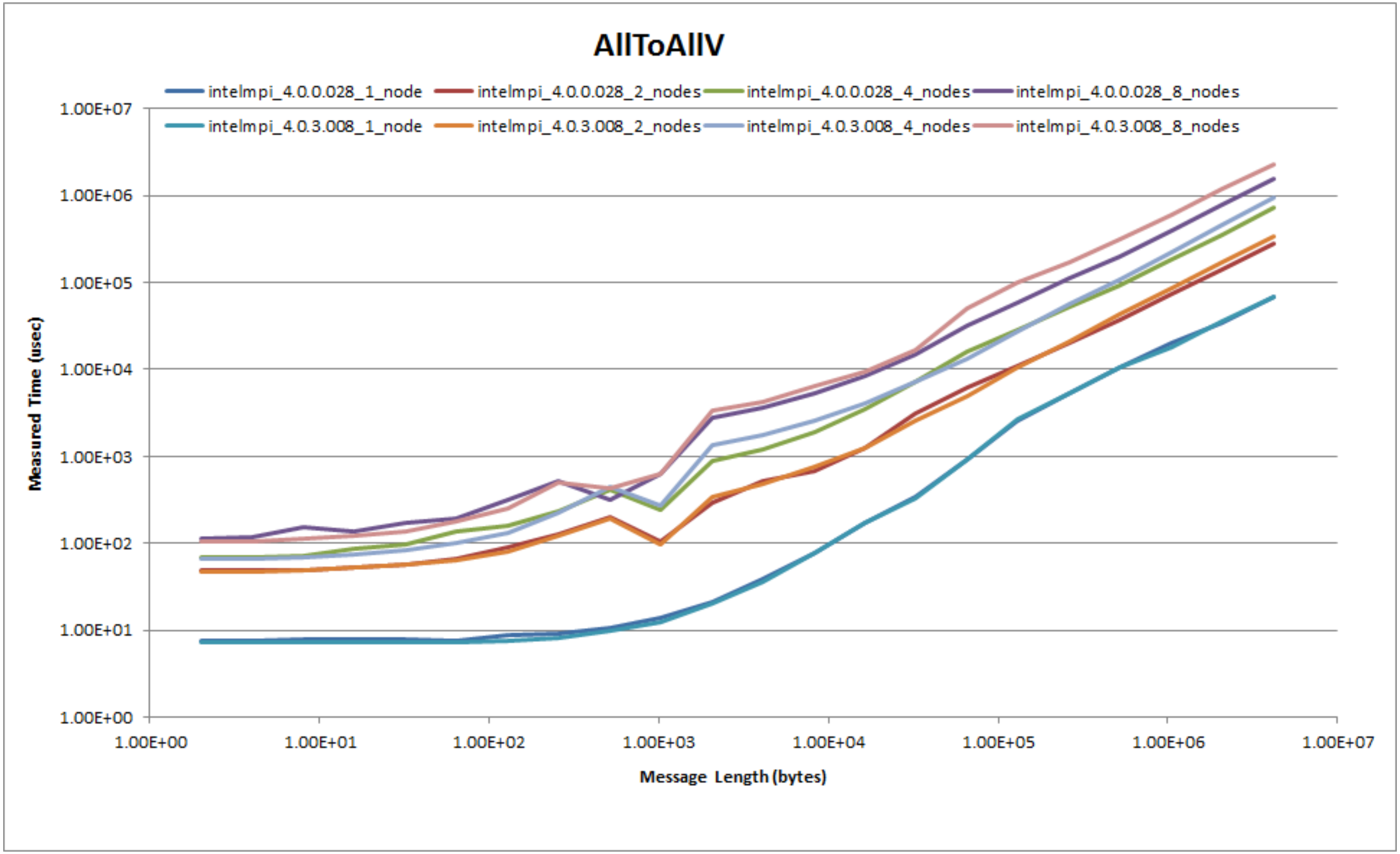
```
#---------------------------------------------------
#    Intel (R) MPI Benchmark Suite V3.2.2, MPI-1 part
#---------------------------------------------------
# Date                : Tue Mar 27 11:26:00 2012
# Machine             : x86_64
# System              : Linux
# Release             : 2.6.18-194.el5
# Version             : #1 SMP Fri Apr 2 14:58:14 EDT 2010
# MPI Version         : 2.1
# MPI Thread Environment: MPI_THREAD_SINGLE
```

Log.IMB.<>

```
#---------------------------------------------------
# Benchmarking Barrier
# #processes = 24
#---------------------------------------------------
 #repetitions  t_min[usec]  t_max[usec]  t_avg[usec]
         1000         5.29         5.29         5.29


# All processes entering MPI_Finalize
```


PingPong
intelmpi_4.0.0.028
intelmpi_4.0.3.008
Bandwidth (mbytes/sec)
1.00E+04
1.00E+03
1.00E+02
1.00E+01
1.00E+00
1.00E+00
1.00E+01
1.00E+02
1.00E+03
1.00E+04
1.00E+05
1.00E+06
1.00E+07
Message Length (bytes)


AllToAll
intelmpi_4.0.0.028_1_node
intelmpi_4.0.0.028_2_nodes
intelmpi_4.0.0.028_4_nodes
intelmpi_4.0.0.028_8_nodes
intelmpi_4.0.3.008_1_node
intelmpi_4.0.3.008_2_nodes
intelmpi_4.0.3.008_4_nodes
intelmpi_4.0.3.008_8_nodes
Measured Time (usec)
1.00E+07
1.00E+06
1.00E+05
1.00E+04
1.00E+03
1.00E+02
1.00E+01
1.00E+00
1.00E+00
1.00E+01
1.00E+02
1.00E+03
1.00E+04
1.00E+05
1.00E+06
1.00E+07
Message Length (bytes)


AllToAllV
intelmpi_4.0.0.028_1_node
intelmpi_4.0.0.028_2_nodes
intelmpi_4.0.0.028_4_nodes
intelmpi_4.0.0.028_8_nodes
intelmpi_4.0.3.008_1_node
intelmpi_4.0.3.008_2_nodes
intelmpi_4.0.3.008_4_nodes
intelmpi_4.0.3.008_8_nodes
1.00E+07
1.00E+06
1.00E+05
1.00E+04
1.00E+03
1.00E+02
1.00E+01
1.00E+00
Measured Time (usec)
1.00E+00
1.00E+01
1.00E+02
1.00E+03
1.00E+04
1.00E+05
1.00E+06
1.00E+07
Message Length (bytes)

Questions and Answers

- For more information
  - www.hpcwales.co.uk
- To access our services
  - info@hpcwales.co.uk
- To contact support
  - support@hpcwales.co.uk