HPC

You will be using an Azure cluster for the coursework: a virtual machine (VM) service run by Microsoft.

HPC

You will be using an Azure cluster for the coursework: a virtual machine (VM) service run by Microsoft.

Or a “Cloud” in more marketing language

HPC

You will be using an Azure cluster for the coursework: a virtual machine (VM) service run by Microsoft.

Or a “Cloud” in more marketing language

DDAT have some Web pages: you should have a browse through them.
`https://wiki.bath.ac.uk/display/CloudHPC/Cloud+HPC+Teaching+Home`

HPC

You will be using an Azure cluster for the coursework: a virtual machine (VM) service run by Microsoft.

Or a “Cloud” in more marketing language

DDAT have some Web pages: you should have a browse through them.
`https://wiki.bath.ac.uk/display/CloudHPC/Cloud+HPC+Teaching+Home`

We shall only give an outline here

HPC

We have the use of

HPC

We have the use of

- 8 compute nodes, each with 44 shared memory cores, no hyperthreading

HPC

We have the use of

- 8 compute nodes, each with 44 shared memory cores, no hyperthreading
- Each node is two Xeon Platinum 8168 Skylake processors with 24 cores each; running at up to approx 3.4GHz

HPC

We have the use of

- 8 compute nodes, each with 44 shared memory cores, no hyperthreading
- Each node is two Xeon Platinum 8168 Skylake processors with 24 cores each; running at up to approx 3.4GHz
- 2 cores per processor are reserved for use by the VM, so you see $2 \times 22 = 44$ cores

HPC

We have the use of

- 8 compute nodes, each with 44 shared memory cores, no hyperthreading
- Each node is two Xeon Platinum 8168 Skylake processors with 24 cores each; running at up to approx 3.4GHz
- 2 cores per processor are reserved for use by the VM, so you see $2 \times 22 = 44$ cores
- With 8GB per core; 352GB total per node

HPC

The nodes are connected by 100Gb InfiniBand networking

HPC

The nodes are connected by 100Gb InfiniBand networking

As a quick comparison: Ethernet has latencies of the order of 10s of μs, while InfiniBand is sub 1μs

HPC

The computers are physically in Amsterdam

HPC

The computers are physically in Amsterdam

There is a 1Gb link between them and Bath

HPC

They run the Linux operating system

HPC

They run the Linux operating system

Actually Centos, a derivative of Redhat

HPC

They run the Linux operating system

Actually Centos, a derivative of Redhat

See `https://wiki.bath.ac.uk/display/BalenaHPC/Linux+Quick+Reference+Guide` for a brief introduction/reminder on using Linux

HPC

The cluster is a shared resource, so there are controls on how the nodes are allocated and used

HPC

The cluster is a shared resource, so there are controls on how the nodes are allocated and used

In particular, you can’t just use any bunch of nodes you feel like: access is mediated by a *batch submission* system

HPC

The cluster is a shared resource, so there are controls on how the nodes are allocated and used

In particular, you can’t just use any bunch of nodes you feel like: access is mediated by a *batch submission* system

Welcome to the 1960s!

HPC

The cluster is a shared resource, so there are controls on how the nodes are allocated and used

In particular, you can’t just use any bunch of nodes you feel like: access is mediated by a *batch submission* system

Welcome to the 1960s!

The system is named *SLURM* (“Simple Linux Utility for Resource Management”)

HPC

The cluster is a shared resource, so there are controls on how the nodes are allocated and used

In particular, you can’t just use any bunch of nodes you feel like: access is mediated by a *batch submission* system

Welcome to the 1960s!

The system is named *SLURM* (“Simple Linux Utility for Resource Management”)

It is not simple!

HPC

To run a program you

HPC

To run a program you

- write, compile and debug your program

HPC

To run a program you

- write, compile and debug your program
- write a job submission script

HPC

To run a program you

- write, compile and debug your program
- write a job submission script
- submit your program to the cluster

HPC

To run a program you

- write, compile and debug your program
- write a job submission script
- submit your program to the cluster
- wait for the results

HPC

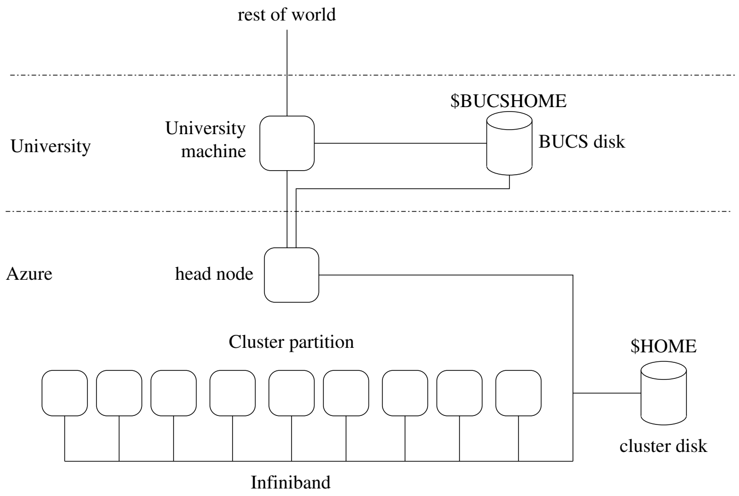
To run a program you

- write, compile and debug your program
- write a job submission script
- submit your program to the cluster
- wait for the results

Normally small jobs have a fast turnaround, so it's not as if you have to wait a week for your results

HPC

To use the cluster you need to log into a *head node*



HPC

Use
`ssh cm30225.hpc.bath.ac.uk`
to login to a head node: there is no direct access to the nodes in the cluster

HPC

Use

```
ssh cm30225.hpc.bath.ac.uk
```

to login to a head node: there is no direct access to the nodes in the cluster

From Windows, you can use Kitty or Putty

HPC

Use

```
ssh cm30225.hpc.bath.ac.uk
```

to login to a head node: there is no direct access to the nodes in the cluster

From Windows, you can use Kitty or Putty

Note that, for security, you can only access the cluster from within the University of Bath

HPC

If you are off-site, you can either `ssh` to the Uni's Linux server:

- `ssh username@linux.bath.ac.uk`
  and then to the cluster by
  `ssh cm30225.hpc.bath.ac.uk`
- or do both in one jump
  `ssh -J username@linux.bath.ac.uk cm30225.hpc.bath.ac.uk`

HPC

If you are off-site, you can either `ssh` to the Uni's Linux server:

- `ssh username@linux.bath.ac.uk`
  and then to the cluster by
  `ssh cm30225.hpc.bath.ac.uk`
- or do both in one jump
  `ssh -J username@linux.bath.ac.uk cm30225.hpc.bath.ac.uk`

Or connect to the Uni's VPN

- `https://www.bath.ac.uk/guides/setting-up-vpn-on-your-device/` where you will appear to be within the Uni and you can then directly `ssh` to the cluster

HPC

For more on Kitty see:

`https://wiki.bath.ac.uk/display/CloudHPC/Getting+Started`

HPC

On the head node you can do lightweight stuff, like submitting jobs — but it is not designed for anything more heavyweight than that

HPC

On the head node you can do lightweight stuff, like submitting jobs — but it is not designed for anything more heavyweight than that

Don’t use the head node for running your code

HPC

On the head node you can do lightweight stuff, like submitting jobs — but it is not designed for anything more heavyweight than that

Don’t use the head node for running your code

If you want to do anything more intensive, you must login to and use an *interactive* node

HPC

On the head node you can do lightweight stuff, like submitting jobs — but it is not designed for anything more heavyweight than that

Don’t use the head node for running your code

If you want to do anything more intensive, you must login to and use an *interactive* node

```
sint -p iteaching
```

takes you to an interactive session on the `iteaching` partition

HPC

On the head node you can do lightweight stuff, like submitting jobs — but it is not designed for anything more heavyweight than that

Don't use the head node for running your code

If you want to do anything more intensive, you must login to and use an *interactive* node

```
sint -p iteaching
```

takes you to an interactive session on the `iteaching` partition

A *partition* is just a bunch of nodes reserved for a particular purpose, interactive development in this case

HPC

On `iteaching` you can edit, compile, run small tests, debug and generally develop code

HPC

On `iteaching` you can edit, compile, run small tests, debug and generally develop code

Note: it may take a minute or three to login to `iteaching`

HPC

On `iteaching` you can edit, compile, run small tests, debug and generally develop code

Note: it may take a minute or three to login to `iteaching`

This is because initialising (“spinning up”) a VM takes some time, and the first user of the day will have to wait

HPC

On `iteaching` you can edit, compile, run small tests, debug and generally develop code

Note: it may take a minute or three to login to `iteaching`

This is because initialising (“spinning up”) a VM takes some time, and the first user of the day will have to wait

The VM stays running for a while, so subsequent logins are faster

HPC

You can have at most **one** interactive session active at a time: if you get an error (“QOSMaxSubmitJobPerUserLimit”) you are already logged in somewhere else: see `scancel` below to kill a session

HPC

You can have at most **one** interactive session active at a time: if you get an error (“QOSMaxSubmitJobPerUserLimit”) you are already logged in somewhere else: see `scancel` below to kill a session

You are limited to 6 hours in a single session on `iteaching` (to ensure everyone gets a go; and for your health!)

HPC

You can have at most **one** interactive session active at a time: if you get an error (“QOSMaxSubmitJobPerUserLimit”) you are already logged in somewhere else: see `scancel` below to kill a session

You are limited to 6 hours in a single session on `iteaching` (to ensure everyone gets a go; and for your health!)

For serious runs and timings of your code, you will be using the compute partition `teaching` (see below)

HPC

The cluster has disk space separate from your other DDAT space

HPC

The cluster has disk space separate from your other DDAT space

This disk space is not visible outside the cluster

HPC

The cluster has disk space separate from your other DDAT space

This disk space is not visible outside the cluster

Your usual DDAT filespace (H drive) is also mounted at a mountpoint given by `$BUCSHOME`

HPC

The cluster has disk space separate from your other DDAT space

This disk space is not visible outside the cluster

Your usual DDAT filespace (H drive) is also mounted at a mountpoint given by `$BUCSHOME`

So you can copy data back and forth simply as

```
cp $BUCSHOME/path/to/prog.c dir/on/cluster
```

when on the headnode

HPC

You can use `scp`, `rsync` or `sftp` to copy to the cluster when on `linux.bath.ac.uk`; or `FileZilla` or `WinSCP` when using a Windows machine

Note that the DDAT filespace is *not* mounted on the compute nodes or the interactive nodes: only on the head node

HPC

You can use `scp`, `rsync` or `sftp` to copy to the cluster when on `linux.bath.ac.uk`; or `FileZilla` or `WinSCP` when using a Windows machine

Note that the DDAT filespace is *not* mounted on the compute nodes or the interactive nodes: only on the head node

In particular, when you run jobs, their code and data will need to be on the cluster disk

HPC

You could either:

- keep your programs and data on the cluster: write and compile your program on the cluster, copy the results back to main DDAT when you need to; or
- keep your programs and data on DDAT, edit on DDAT, copy to the cluster, compile and run your program on the cluster, copy the results back to main DDAT

HPC

The cluster disk is backed up once a day, and backups are kept for 30 days. If you need anything older than that, you are out of luck

HPC

The cluster disk is backed up once a day, and backups are kept for 30 days. If you need anything older than that, you are out of luck

So it makes sense to archive your stuff yourself, too (e.g., keep a copy on DDAT)

HPC

The cluster disk is backed up once a day, and backups are kept for 30 days. If you need anything older than that, you are out of luck

So it makes sense to archive your stuff yourself, too (e.g., keep a copy on DDAT)

You have a quota of 1Gb disk on the cluster. This will be plenty of space: if you need more, you are doing something wrong!

HPC

The cluster disk is backed up once a day, and backups are kept for 30 days. If you need anything older than that, you are out of luck

So it makes sense to archive your stuff yourself, too (e.g., keep a copy on DDAT)

You have a quota of 1Gb disk on the cluster. This will be plenty of space: if you need more, you are doing something wrong!

Note: do not install any of your own software on the cluster disk (e.g., development environments) as these use CPU and disk, which costs us real money

HPC

There is a common directory at `$CM30225_DIR` where useful stuff will be kept

HPC

There is a common directory at `$CM30225_DIR` where useful stuff will be kept

You can read and copy stuff from here, but not modify anything

HPC

There is a lot of software available to run on the cluster, including several variants of compilers and parallel libraries

HPC

There is a lot of software available to run on the cluster, including several variants of compilers and parallel libraries

So there is a simple *module* system that aids in selecting the right combination of bits of software, e.g., the GCC compiler and a version of MPI that is compatible with it

HPC

- `module avail` to see the list of available modules;
- `module list` lists the currently loaded modules;
- `module load` to load a module;
- `module unload` to unload a module;
- `module purge` to unload all modules

HPC

- `module avail` to see the list of available modules;
- `module list` lists the currently loaded modules;
- `module load` to load a module;
- `module unload` to unload a module;
- `module purge` to unload all modules

Fortunately, the modules for this Unit are pre-loaded for you (`gcc-9.2.0` and `mpi/openmpi`)

HPC

When you have a working program, you need to submit a batch job for the cluster to run it

HPC

When you have a working program, you need to submit a batch job for the cluster to run it

This entails writing a batch submission script

HPC

A simple single processor job. Lines starting with `#SBATCH` are options for the `sbatch` command:

```
#!/bin/sh
# Account & partition (must have these)
#SBATCH --account=cm30225
#SBATCH --partition=teaching
# Name of job (optional)
#SBATCH --job-name=Test_Serial
# one node
#SBATCH --nodes=1
# any normal shell stuff
pwd

# Run the program
./helloworld
```

HPC

This is a shell script, a **plain text** file, called, e.g., `runhello.batch` or `runhello.slm` or anything you like

HPC

This is a shell script, a **plain text** file, called, e.g., `runhello.batch` or `runhello.slm` or anything you like

Submit the job **on the head node**:

```
sbatch runhello.batch
```

HPC

This is a shell script, a **plain text** file, called, e.g., `runhello.batch` or `runhello.slm` or anything you like

Submit the job **on the head node**:

```
sbatch runhello.batch
```

This will be submitted to a *run queue*, where it will sit until resources become available to run the job

HPC

This is a shell script, a **plain text** file, called, e.g., `runhello.batch` or `runhello.slm` or anything you like

Submit the job **on the head node**:
```
sbatch runhello.batch
```

This will be submitted to a *run queue*, where it will sit until resources become available to run the job

When it runs, output to `stdout` will end up in file
`slurm-<jobnumber>.out`
and `stderr` in
`slurm-<jobnumber>.err`

HPC

To repeat: `sbatch` only works when used on the head node

HPC

To repeat: `sbatch` only works when used on the head node

It may appear to work when used on `iteaching`, but it produces a zombie process that never starts

HPC

There are several queues, the `cm30225` queue is the one which you will be using

HPC

There are several queues, the `cm30225` queue is the one which you will be using

The partition indicates the subset of the nodes to be used: this unit has a compute partition of 8 nodes reserved for our use

HPC

There are several queues, the `cm30225` queue is the one which you will be using

The partition indicates the subset of the nodes to be used: this unit has a compute partition of 8 nodes reserved for our use

You can use up to 4 nodes in a single job (for coursework 2; coursework 1 only needs one node at a time)

HPC

SLURM options:

- `#SBATCH --time=`*`hh:mm:ss`*
  Limit the amount of time the program can take. The program will be killed automatically if the time given is exceeded. `time` is real elapsed time

You can set this to something reasonable (and small) to help SLURM determine how to schedule your job

HPC

SLURM options:

- `#SBATCH --time=`*`hh:mm:ss`*
  Limit the amount of time the program can take. The program will be killed automatically if the time given is exceeded. `time` is real elapsed time

You can set this to something reasonable (and small) to help SLURM determine how to schedule your job

Setting this to the smallest value you need may encourage SLURM to run your job sooner

HPC

SLURM options:

- `#SBATCH --time=`*`hh:mm:ss`*
  Limit the amount of time the program can take. The program will be killed automatically if the time given is exceeded. `time` is real elapsed time

You can set this to something reasonable (and small) to help SLURM determine how to schedule your job

Setting this to the smallest value you need may encourage SLURM to run your job sooner

But beware of the spin-up time when setting this value

HPC

CM30225 jobs have a time limit of 20 minutes: this starts ticking from the point SLURM starts running your job

HPC

CM30225 jobs have a time limit of 20 minutes: this starts ticking from the point SLURM starts running your job

But, again, the job VM can take up to 10 minutes to spin up, so there is an effective limit of 10 minutes for your code to run

HPC

CM30225 jobs have a time limit of 20 minutes: this starts ticking from the point SLURM starts running your job

But, again, the job VM can take up to 10 minutes to spin up, so there is an effective limit of 10 minutes for your code to run

The longest jobs you should run (both for Assignment 1 and Assignment 2) should be about 10 minutes

HPC

$$\begin{tikzcd}
\text{submit} \arrow[d] & & \\
\text{queue} \arrow[r] & \text{dispatch} \arrow[r] & \text{spinup} \arrow[dd] \\
 & \text{SLURM clock} & \\
 & \text{starts} & \text{runs}
\end{tikzcd}$$

The VM is kept alive for a while after the end of the job, so if you run another job soon enough the next spinup will be fast

HPC

- `#SBATCH --mail-type=[events]`
  If you want to be notified when a job has finished, SLURM can send you an email. Use `END` to request an email for normal exit; `BEGIN` for when the job starts; `FAIL` for when the job fails. Mostly used for long running jobs

HPC

- `#SBATCH --mail-type=[events]`
  If you want to be notified when a job has finished, SLURM can send you an email. Use `END` to request an email for normal exit; `BEGIN` for when the job starts; `FAIL` for when the job fails. Mostly used for long running jobs
- `#SBATCH --mail-user=[user]` to specify an email address

HPC

- `#SBATCH --job-name=[jobname]`
  A name for this job, mostly for human benefit. Also gives the default file names for the where the output from your program goes (see above)

HPC

- `#SBATCH --job-name=[jobname]`
  A name for this job, mostly for human benefit. Also gives the default file names for the where the output from your program goes (see above)
- `#SBATCH --output=[filename]`
  Send program output to a named file

HPC

- `#SBATCH --job-name=[jobname]`
  A name for this job, mostly for human benefit. Also gives the default file names for the where the output from your program goes (see above)
- `#SBATCH --output=[filename]`
  Send program output to a named file
- `#SBATCH --error=[filename]`
  Send program error output to a named file

HPC

- `#SBATCH --nodes=[n]`
  Specify the number of nodes your job wants

HPC

- `#SBATCH --nodes=[n]`
  Specify the number of nodes your job wants
- `#SBATCH --ntasks-per-node=[n]`
  Specify the number of MPI processes on a node (see later)

HPC

- `#SBATCH --nodes=[n]`
  Specify the number of nodes your job wants
- `#SBATCH --ntasks-per-node=[n]`
  Specify the number of MPI processes on a node (see later)

And more to control maximum memory used, etc.

HPC

- `#SBATCH --nodes=[n]`
  Specify the number of nodes your job wants
- `#SBATCH --ntasks-per-node=[n]`
  Specify the number of MPI processes on a node (see later)

And more to control maximum memory used, etc.

The `teaching` partition allows you to use up to a maximum of 4 nodes at a time, so up to 176 distributed memory cpu cores (coursework 2)

HPC

The maximum number of cores in a shared memory configuration is 44 (coursework 1)

HPC

The maximum number of cores in a shared memory configuration is 44 (coursework 1)

It is pointless asking for more than that on a node and SLURM will reject such a request

HPC

Note on `ntasks-per-node`: you will be allocated exclusive use of whole nodes, so you will always have access to 44 cores on a node regardless of this value

HPC

Note on `ntasks-per-node`: you will be allocated exclusive use of whole nodes, so you will always have access to 44 cores on a node regardless of this value

So in a single-node, shared memory program its value is effectively ignored

HPC

Note on `ntasks-per-node`: you will be allocated exclusive use of whole nodes, so you will always have access to 44 cores on a node regardless of this value

So in a single-node, shared memory program its value is effectively ignored

But when running MPI the values of `nodes` and `ntasks-per-node` are needed and used in the initialisation and connection of the separate MPI processes

HPC

To repeat: jobs are given exclusive use of nodes and no other user's jobs will be scheduled on your cores while your job is running

HPC

To repeat: jobs are given exclusive use of nodes and no other user’s jobs will be scheduled on your cores while your job is running

There may be a few system processes, but otherwise the allocated cores are occupied 100% running your job

HPC

To repeat: jobs are given exclusive use of nodes and no other user’s jobs will be scheduled on your cores while your job is running

There may be a few system processes, but otherwise the allocated cores are occupied 100% running your job

If your program is running slowly, it’s not because someone else is sharing your cores!

HPC

When you submit a job

`sbatch` *jobfile*

will reply with a *job id* unique to that job

HPC

When you submit a job
`sbatch` *`jobfile`*
will reply with a *job id* unique to that job

To check on the progress of a job use
`squeue`
or
`squeue -u` *`username`*
or
`squeue -p teaching`
for all jobs on the `teaching` partition

HPC

When you submit a job

`sbatch` *`jobfile`*

will reply with a *job id* unique to that job

To check on the progress of a job use

```
squeue
```

or

`squeue -u` *`username`*

or

```
squeue -p teaching
```

for all jobs on the `teaching` partition

To kill a job use

`scancel` *`jobid`*

where you can get `jobid` from `squeue` or from the original `sbatch`

HPC

More detailed information about a particular job:

`scontrol show job` *jobid*

HPC

More detailed information about a particular job:

```
scontrol show job jobid
```

More detailed information about a particular node:

```
scontrol show node nodename
```

HPC

More detailed information about a particular job:
`scontrol show job` *`jobid`*

More detailed information about a particular node:
`scontrol show node` *`nodename`*

Use `sinfo` for general status information

HPC

More detailed information about a particular job:
`scontrol show job` *`jobid`*

More detailed information about a particular node:
`scontrol show node` *`nodename`*

Use `sinfo` for general status information

`sshare` to see which accounts you have access to, and how much you have used them

HPC

More detailed information about a particular job:
`scontrol show job` *`jobid`*

More detailed information about a particular node:
`scontrol show node` *`nodename`*

Use `sinfo` for general status information

`sshare` to see which accounts you have access to, and how much you have used them

`sprio -u` *`username`* to see the priority of your jobs

HPC

More detailed information about a particular job:
`scontrol show job` *`jobid`*

More detailed information about a particular node:
`scontrol show node` *`nodename`*

Use `sinfo` for general status information

`sshare` to see which accounts you have access to, and how much you have used them

`sprio -u` *`username`* to see the priority of your jobs

And more, like suspending a job, moving jobs between queues and so on

HPC

If you submit a large number of jobs your priority will decrease to give other people's jobs a chance

HPC

If you submit a large number of jobs your priority will decrease to give other people's jobs a chance

Remember that the `teaching` partition is shared amongst the entire class of CM30225

HPC

If you submit a large number of jobs your priority will decrease to give other people's jobs a chance

Remember that the `teaching` partition is shared amongst the entire class of CM30225

Expect the queue to get longer as you approach hand-in date!

HPC
Using MPI

MPI is supported on the cluster

HPC

Using MPI

MPI is supported on the cluster

OpenMPI is the simplest to use, but there are others if you feel brave

HPC

Using MPI

MPI is supported on the cluster

OpenMPI is the simplest to use, but there are others if you feel brave

The OpenMPI module is pre-loaded for you

HPC

Using MPI

Compile your MPI code using `mpicc`

HPC

Using MPI

Compile your MPI code using `mpicc`

```
mpicc -Wall -Wextra -Wconversion -o mpiprog
mpiprog.c
```

HPC

Using MPI

Compile your MPI code using `mpicc`

```
mpicc -Wall -Wextra -Wconversion -o mpiprog
mpiprog.c
```

This links in the necessary MPI libraries

HPC

Using MPI

Compile your MPI code using `mpicc`

```
mpicc -Wall -Wextra -Wconversion -o mpiprog
mpiprog.c
```

This links in the necessary MPI libraries

The modules loaded make sure the right libraries are linked in for your chosen version of MPI

HPC

Using MPI

The number of processors used is specified in the SLURM script by `nodes` and `ntasks_per_node`

```
#SBATCH --ntasks-per-node=[n]
```

Specify the number of MPI processes on a node

HPC
Using MPI

The number of processors used is specified in the SLURM script by `nodes` and `ntasks_per_node`

```
#SBATCH --ntasks-per-node=[n]
```
Specify the number of MPI processes on a node

To start an MPI program use (in the SLURM script)
```
mpirun ./mpiprog
```

HPC
Using MPI

The number of processors used is specified in the SLURM script by `nodes` and `ntasks_per_node`

```
#SBATCH --ntasks-per-node=[n]
```
Specify the number of MPI processes on a node

To start an MPI program use (in the SLURM script)
```
mpirun ./mpiprog
```

You can use a maximum of 4 nodes $\times$ 44 cores per node $=$ 176 cores in an MPI job. This limit is to ensure everyone can get to run their jobs

HPC
Using MPI

`mpirun` does all the hard work of setting up the processes on each processor and connecting them over the network or via shared memory, as appropriate

HPC
Using MPI

`mpirun` also

HPC

Using MPI

`mpirun` also

- connects the standard I/O streams from the remote processes to the master process (rank 0), so, for example, a `printf` anywhere will appear in the output file

HPC

Using MPI

`mpirun` also

- connects the standard I/O streams from the remote processes to the master process (rank 0), so, for example, a `printf` anywhere will appear in the output file
- sets several environment variables.e.g., `SLURM_SUBMIT_DIR`, which names the directory the SLURM script was submitted from

HPC

Using MPI

`mpirun` also

- connects the standard I/O streams from the remote processes to the master process (rank 0), so, for example, a `printf` anywhere will appear in the output file
- sets several environment variables.e.g., `SLURM_SUBMIT_DIR`, which names the directory the SLURM script was submitted from
- kills off all the processes and tidies up at the end; also if there is an error

HPC

Debugging

A variety of debugging tools are available, depending on your level of sophistication

HPC
Debugging

A variety of debugging tools are available, depending on your level of sophistication

- Using print statements in your program is more effective than you might think!

HPC
Debugging

A variety of debugging tools are available, depending on your level of sophistication

- Using print statements in your program is more effective than you might think!
- `gdb`

HPC
Debugging

A variety of debugging tools are available, depending on your level of sophistication

- Using print statements in your program is more effective than you might think!
- `gdb`
- `valgrind`

HPC

Debugging

A variety of debugging tools are available, depending on your level of sophistication

- Using print statements in your program is more effective than you might think!
- `gdb`
- `valgrind`
- You may be able to install OpenMPI and run `mpicc` and `mpirun` on your own PC

HPC

Summary

- Programs and data should be on the cluster filesystem so they are visible to the compute and interactive nodes

HPC

Summary

- Programs and data should be on the cluster filesystem so they are visible to the compute and interactive nodes
- For a shared memory program on one node, use `--nodes=1`; then you can use up to 44 cores

HPC

Summary

- Programs and data should be on the cluster filesystem so they are visible to the compute and interactive nodes
- For a shared memory program on one node, use `--nodes=1`; then you can use up to 44 cores
- For an MPI program, compile using `mpicc` and run using `mpirun`

HPC

Summary

- Programs and data should be on the cluster filesystem so they are visible to the compute and interactive nodes
- For a shared memory program on one node, use `--nodes=1`; then you can use up to 44 cores
- For an MPI program, compile using `mpicc` and run using `mpirun`
- You can use a maximum of 4 nodes in a job

HPC

Summary

- Programs and data should be on the cluster filesystem so they are visible to the compute and interactive nodes
- For a shared memory program on one node, use `--nodes=1`; then you can use up to 44 cores
- For an MPI program, compile using `mpicc` and run using `mpirun`
- You can use a maximum of 4 nodes in a job
- Your jobs have a time limit of 20 minutes

HPC

Summary

- Programs and data should be on the cluster filesystem so they are visible to the compute and interactive nodes
- For a shared memory program on one node, use `--nodes=1`; then you can use up to 44 cores
- For an MPI program, compile using `mpicc` and run using `mpirun`
- You can use a maximum of 4 nodes in a job
- Your jobs have a time limit of 20 minutes
- There is a lot of documentation online: `https://wiki.bath.ac.uk/display/CloudHPC/Cloud+HPC+Home`

HPC

Summary

- Don’t run anything other than the assignment code on the cluster: this will be noticed and bad things may happen to you

HPC

Summary

- Don’t run anything other than the assignment code on the cluster: this will be noticed and bad things may happen to you
- Using Azure costs the Department real money, so don’t waste CPU time

HPC

Summary

- Don’t run anything other than the assignment code on the cluster: this will be noticed and bad things may happen to you
- Using Azure costs the Department real money, so don’t waste CPU time
- Don’t run compute code on the head node

HPC

Summary

- Don’t run anything other than the assignment code on the cluster: this will be noticed and bad things may happen to you
- Using Azure costs the Department real money, so don’t waste CPU time
- Don’t run compute code on the head node
- You can run sequential code directly on interactive nodes, but you must run parallel code on the compute nodes (interactive nodes have the same hardware configuration as compute nodes)

HPC

Summary

- Don’t run anything other than the assignment code on the cluster: this will be noticed and bad things may happen to you
- Using Azure costs the Department real money, so don’t waste CPU time
- Don’t run compute code on the head node
- You can run sequential code directly on interactive nodes, but you must run parallel code on the compute nodes (interactive nodes have the same hardware configuration as compute nodes)
- Submit parallel jobs from the head node

HPC

Summary

- Don’t run anything other than the assignment code on the cluster: this will be noticed and bad things may happen to you
- Using Azure costs the Department real money, so don’t waste CPU time
- Don’t run compute code on the head node
- You can run sequential code directly on interactive nodes, but you must run parallel code on the compute nodes (interactive nodes have the same hardware configuration as compute nodes)
- Submit parallel jobs from the head node
- The cluster is a shared resource, so be kind to your classmates!