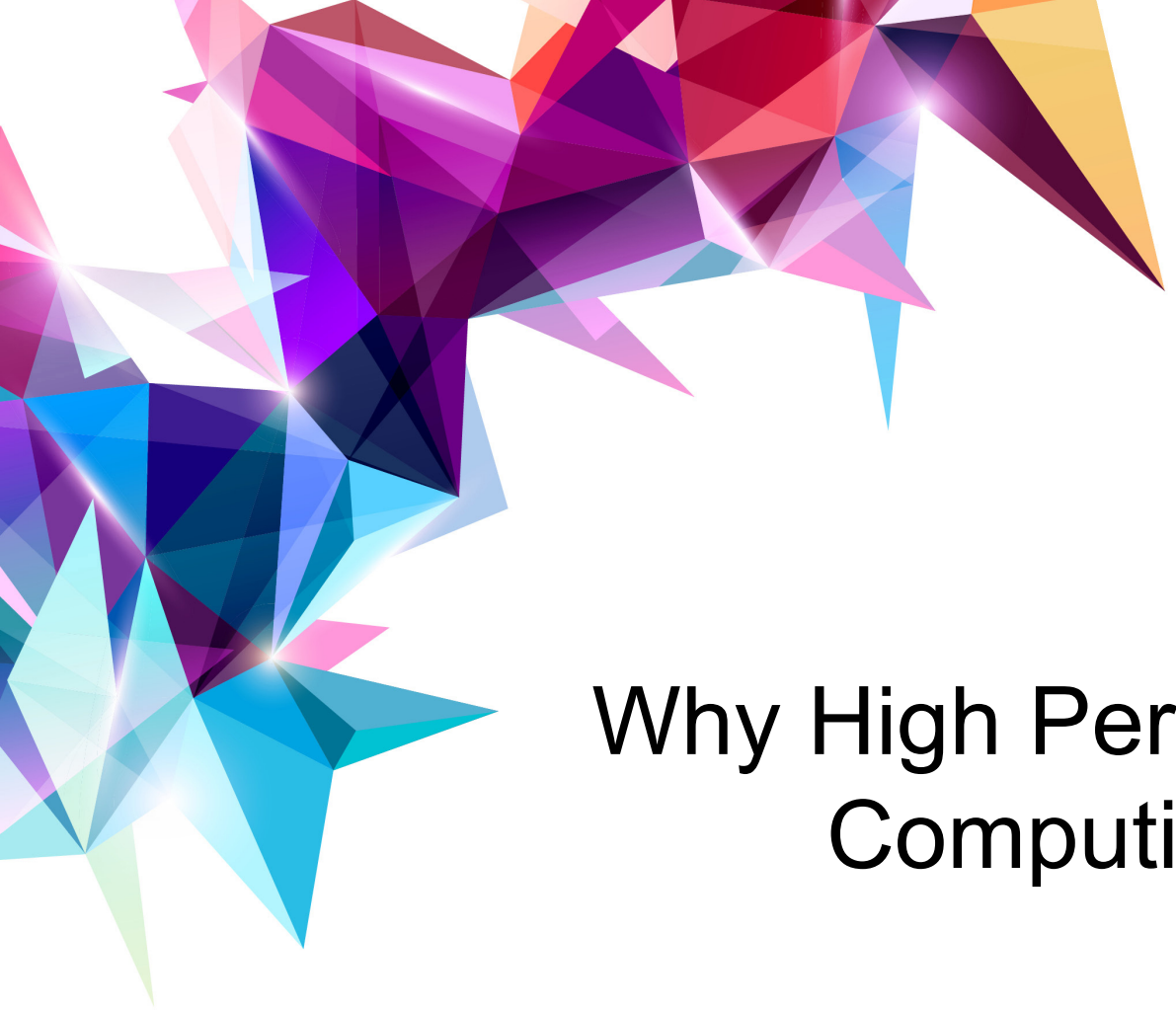
Getting Started with ARC

The ARC Team

www.it.ox.ac.uk

www.arc.ox.ac.uk



Why High Performance Computing?

www.it.ox.ac.uk

www.arc.ox.ac.uk

Advanced Research Computing (ARC)

Research Computing comes in four distinct flavours:

Compute Intensive: These are applications that require a substantial amount of compute power coupled with high-performance inter-processor communication.

Data Intensive (I/O intensive): Such applications operate on large volumes of data and thus demand efficient ingress/egress of data. High performance in the storage hierarchy is crucial for these applications.

High Throughput: This pertains to scenarios where numerous independent or embarrassingly parallel jobs must be executed. While each task can be simple enough to be processed on a workstation or laptop, they need to be performed hundreds, if not thousands, of times.

Memory Intensive: A single problem or job requires a considerable amount of memory for feasible processing. Such applications typically need all data to be loaded into local memory, especially when dealing with expansive datasets, or in cases where subsequent calculations based on initial outputs might exceed the memory available on standard resources, such as in quantum machine modelling.

ARC provide generalised **High Performance Computing (HPC)** resources that cater to the diverse needs of the above Research Computing categories as well as high-performance storage solutions, user support, and application support.

High Performance Computing

There is no single, universally agreed-upon definition for High Performance Computing (HPC), for the context of our discussion, we can define it as :
HPC pertains to computational tasks that surpass the capabilities of standard laptops or desktop workstations, often necessitating parallel processing across multiple processors.

The essence of HPC lies in its ability to accomplish tasks faster, manage more tasks concurrently within a given time frame, or tackle challenges that would otherwise be computationally infeasible.

It's important to note that HPC isn't solely about running parallelised code on a cluster. Utilising a single, high-memory "fat" node can also be a legitimate HPC approach.

High Performance Computing

But how fast is fast enough?

- Desktop PC: Tens of Gflops.
- Even tens of billions of flops might not suffice.
- Extreme Example: Next-day weather forecast:

  Met Office needs ~1 Pflops:
  1 million times more than a PC.
  Requires parallel processing on many CPUs.

- Top supercomputers: petaflop to exaflop range.

Why use an HPC Cluster

- **Efficiency**: Don't monopolise your personal machine.
- **Volume**: Handle multiple, lengthy tasks.
- **Speed**: Achieve faster results with parallel processing.
  Supports job parallelism of serial tasks.
- **Resources**:
  Enhanced storage capacity.
  Greater memory allocation.
- **Capabilities:**
  Access specialised software on the cluster.
  Utilise GPUs for accelerated processing.
- **Support:** Benefit from dedicated ARC assistance.

Before we start…

HPC Terminology

Core: Fundamental unit executing tasks. Often referred to as a CPU.

Processor: Assembly of cores, sharing a unified memory.

Node: Independent set of processors with its own memory. Cannot directly access another node's memory.

Interconnect: Network linking individual nodes, enabling communication.

Memory: (or RAM) the temporary area where systems put data before it is processed

Storage: user accessible area for data

Distinction between **Processor vs. Process vs. Thread:**

- **Processor**: Physical hardware component.
- **Process**: Running instance of a program (software).
  Components: Execution instructions & associated data.
  In parallel programming: Multiple instances of the same program.
- **Thread**: Unit of a process.
  A process contains one or more threads of execution.

Parallel Processing Models

Models of parallelism:
Distributed Memory

Distributed Memory Programming Model:

- **System**: Multi-core where each core has its dedicated memory.
- **Memory Accessibility**: A core's memory is private, not directly accessible by other cores.
- **Parallelism Unit**: The process (a program consists of multiple processes).
- **Information Exchange**: Requires explicit message passing between processes.
- **Dominant Programming Standard**: Message Passing Interface **(MPI)**.

Distributed Memory Hardware:

- **Traditional Model**: Conceptually like many PCs linked together (Beowulf cluster).
- **Modern Approach**:
  Multi-core nodes (high-density blades) each with dedicated memory.
  High-bandwidth, low-latency networking.
  Modular, off-the-shelf tech: Premium CPUs, standard storage drives.
- **Significance**: Underpins the most expansive HPC systems.

Distributed Memory ARC systems: the **ARC** cluster (but any machine can be programmed using this model)

Models of parallelism: **Shared Memory**

Shared Memory Programming Model:

- System Type: Multi-core.
- Memory Access: Each core taps into a unified memory space.
- Unit of Parallelism: Thread (with a program encompassing multiple threads).
- Information Exchange: Threads communicate via shared variables.
- Predominant Standard: OpenMP.

Shared Memory Hardware:

- conceptually, a single computer, with a large memory and multiple cores.
- accounts for both small and inexpensive systems (desktops) as well as substantial, high-end configurations featuring premium, high-bandwidth memory access.

Shared Memory ARC Systems: **HTC cluster** and any *single node* of the **ARC** cluster.

Distributed Memory v. Shared Memory

Distributed Memory:

- Scalable to an unlimited number of cores.
- Needs specific tools/libraries (like MPI) for compiling and execution.
- Typically more challenging to program than shared memory.
- Can offer superior performance when optimized properly.
- Fosters effective parallel programming techniques.

Shared Memory:

- Typically constrained by the number of cores on a node.
- Possible to overpopulate; useful for debugging but detrimental to performance.
- Often requires just an additional compiler flag.
- Generally simpler to program than distributed memory.
- Achieving optimal parallel performance can be challenging.
- Sharing resources isn't always conducive to parallelism.
- Can inadvertently promote lax programming practices.

ARC HPC Services

www.it.ox.ac.uk

www.arc.ox.ac.uk

HPC Clusters

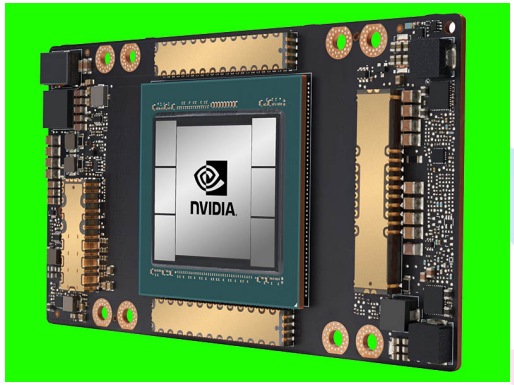
Cluster services :

HTC – high throughput
(Shared Memory)

ARC – large scale parallel
(Distributed Memory)

GPU nodes on HTC include:

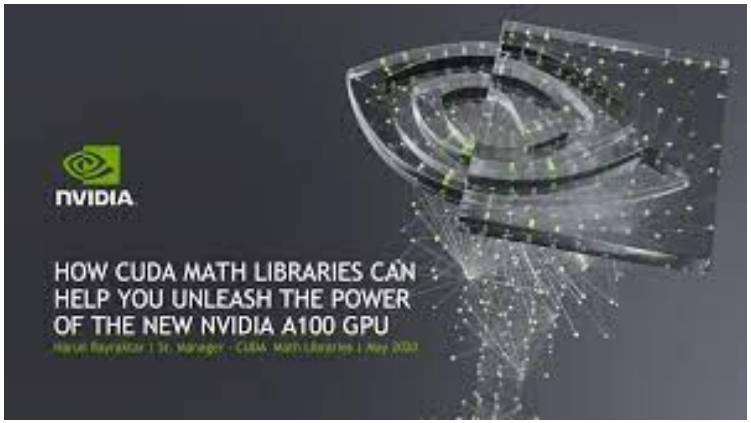
Pascal, Volta, Turing, Ampere,
Grace Hopper



Hosted at Begbroke data centre

HTC Cluster

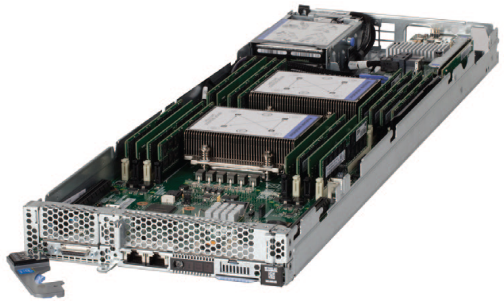
- Minimum job size: 1 core
- CPU on this cluster include:
  High Memory Nodes
  Standard CPU nodes



- GPU nodes include:
  V100, A100, P100, Titan RTX, RTXA6000, V100-LS …

https://arc-user-guide.readthedocs.io/en/latest/arc-systems.html

ARC Cluster



ARC provides a capability cluster comprising of:

- >250 General Compute Nodes; designed for multi-node parallel computation
- OS: CentOS 8.x (two chassis for legacy software CentOS 7.7)
- Scheduler: SLURM

14,000+ CPU cores

All connected with fast low-latency network/interconnect (Infiniband)

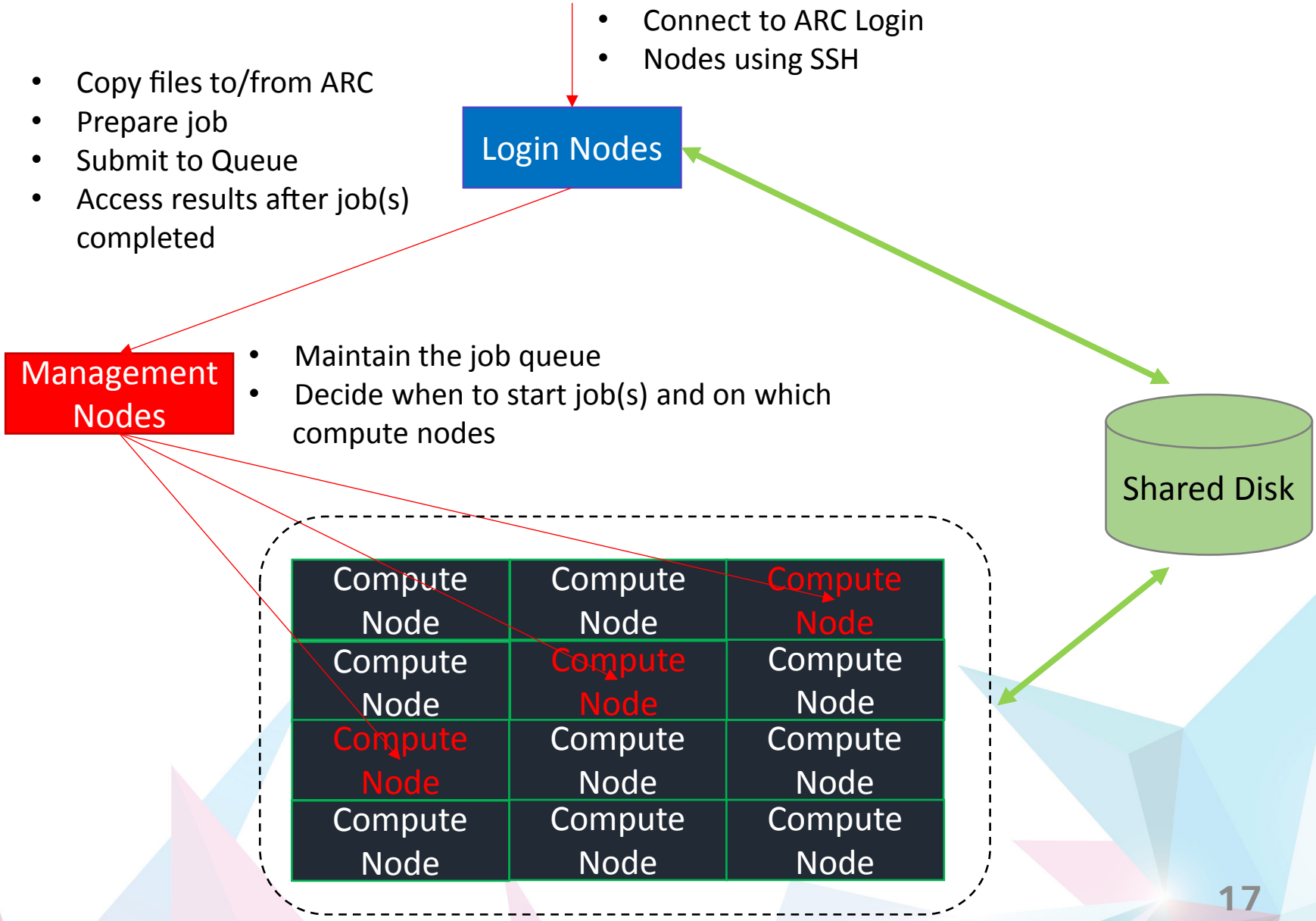
https://arc-user-guide.readthedocs.io/en/latest/arc-systems.html

Cluster Workflow

Cluster of compute nodes

- Connect to ARC Login Nodes using SSH

- Copy files to/from ARC
- Prepare job
- Submit to Queue
- Access results after job(s) completed



Software Environment

Linux Operating System

All ARC systems operate on the Linux Operating System, rather than on platforms like Windows or MacOS.

Why Linux is the Preferred Choice for HPC:

Cost-Efficiency
Dependability
Optimal Performance

To use ARC, a basic knowledge of Linux is essential. Numerous online resources are available to help you acquire this understanding.

See the ARC training pages for more information:
https://www.arc.ox.ac.uk/training

Environment Modules

The "module" utility in Linux is designed to manage the working environment, particularly in preparation for executing applications installed on the ARC systems.

When you load a module for a specific application, the associated environment variables for that application are automatically set or adjusted.

To find a specific application on the system, you can use the "module spider" command. For instance:

module spider matlab
Returns the choice of:

MATLAB/R2019b
MATLAB/R2020a
MATLAB/R2020b
MATLAB/R2021b
MATLAB/R2022a
MATLAB/R2023a

To then load the most recent version of MATLAB, you would input:
module load MATLAB/R2023a

ARC Software Environment

The software environment of ARC/HTC encompasses a blend of commercial applications, tools constructed with the EasyBuild framework, see: https://easybuild.io/

To integrate applications into your ARC/HTC environment, utilize the environment modules system through the module command.

Given that the EasyBuild framework introduces numerous module components, the most efficient method to locate a desired application on the system is by employing the module spider command.

For those interested in exploring a comprehensive list of ARC modules online, it's available here: https://arc-module-list.readthedocs.io/en/latest/

module load example

Loading MATLAB via the Linux **`module load`** command on a terminal :

```
[ouit0554@arc-c304 ~]$
[ouit0554@arc-c304 ~]$ matlab
-bash: matlab: command not found
[ouit0554@arc-c304 ~]$ module load MATLAB/R2022a
[ouit0554@arc-c304 ~]$ matlab -nojvm -nosplash
MATLAB is selecting SOFTWARE OPENGL rendering.

                            < M A T L A B (R) >
                  Copyright 1984-2022 The MathWorks, Inc.
                R2022a Update 3 (9.12.0.1975300) 64-bit (glnxa64)
                                   June 2, 2022


For online documentation, see https://www.mathworks.com/support
For product information, visit www.mathworks.com.

>>
```

Accessing Installed Software Applications
https://arc-software-guide.readthedocs.io/en/latest/arc-modules.html

Using Python Anaconda on ARC clusters

The ARC team maintain central Python Anaconda installations for Anaconda 2 and Anaconda 3.

For example:
`module load Anaconda3`

Or use
`module spider Anaconda`
And use the specific Anaconda version you need.
For example:
`module load Anaconda3/2023.09-0`

If a python module you require is not available on the central Anaconda installations we suggest you follow our instructions to set up your personal Anaconda virtual environment :
https://arc-software-guide.readthedocs.io/en/latest/python/anaconda_venv.html

Using R on ARC clusters

`module spider R`

The base install has many popular R packages installed. Additionally there are modules available with the **`-ARC`** suffix e.g. **`R/4.1.2-foss-2021b-ARC`**

To install packages in your own R Library follow the instructions on our software pages:

https://arc-software-guide.readthedocs.io/en/latest/R/arc_r_intro.html

Software Containers on ARC

ARC offers support for containerized applications through Singularity, now known as **Apptainer**.

In addition to executing its proprietary containers, **Singularity/Apptainer** also has the capability to run **Docker** containers.

Since Singularity is integrated into the OS, there's no need to execute a module load command.

See for example:
https://sylabs.io/guides/2.6/user-guide/singularity_and_docker.html

The Scheduler

Scheduler

ARC uses the SLURM scheduler. SLURM stands for:

Simple **L**inux **U**tility for **R**esource **M**anagement

Job and Node Management with SLURM

- SLURM oversees the job queue, dictating the start time, sequence, and allocation of nodes for each job.
- It handles the administration of compute nodes.
- SLURM assigns tasks to available compute nodes.
- It also supports "accelerator cards," including those from Nvidia like GPU nodes.
- The ARC and HTC clusters both utilise the SLURM scheduler.

Connecting to ARC systems - SSH

The SSH protocol is used for all remote user connections to our systems. Windows users can use well-known SSH clients ”MobaXterm” or ”Putty” or the built-in openssh available in Windows 10 and later versions.

Linux and Mac users can use the Linux terminal and run the built-in ssh client.

Open a terminal and from the prompt enter your ARC username and password :

```
ssh <userid>@arc-login.arc.ox.ac.uk
```

Or if you want X to forward graphics applications:

```
ssh -X <userid>@arc-login.arc.ox.ac.uk
```

For more details see
https://arc-user-guide.readthedocs.io/en/latest/connecting-to-arc.html

Job submission

Creating a Submission Script for SLURM

- A submission script specifies the resources you want SLURM to allocate for your task.
- It also includes the commands to run your desired application(s) and any necessary setup for those applications.
- Use a Linux text editor, like nano or vi, to create your script.
  `nano submit.sh`
- Important: For optimal functionality, we advise crafting and modifying submission scripts directly on the cluster. Editing them on a Windows machine can introduce issues.

Please note: We recommend creating and editing submission scripts on the cluster rather than editing them on a Windows machine, as this can introduce issues.

Job submission (Example)

Here is a simple Linux shell script (simple text file) with instructions to SLURM

The SLURM instructions, or directives, (the lines starting with **#SBATCH**) request cluster resources. The other shell commands say what to do in job.

Example (MPI or Message Passing Interface job)

```
#! /bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=48
#SBATCH --time=08:00:00
#SBATCH --partition=short
#SBATCH --clusters=arc

module purge
module load mpitest
mpirun mpihello
```

Partitions on ARC clusters

The clusters have the following time-based scheduling partitions available:

- **short** (default run time 1hr, maximum run time 12hrs)
- **medium** (default run time 12hrs, maximum run time 48hrs)
- **long** (default run time 24hrs, no run time limit)
- **devel** (maximum run time 10 minutes - for batch job testing only)
- **interactive** (maximum run time 24hrs, can oversubscribe, for pre/post-processing and building software)

Jobs in the short and medium partitions are scheduled with higher priority than those in the long partition; however, they will not be able to run for longer than the time allowed on those partitions.

Note: QOS (Quality of Service) for co-investment nodes overrides partition time limits. In other words you may see jobs running for more than 12 hours on the short partition, these jobs belong to members of Research groups who have purchased the co-invested nodes. All other users will have a time limit of 12 hours on the short queues.

The **devel** partition

The **devel** partitions should be used to test your submission script

on ARC (for CPU usage) **devel** partition has two nodes , 48 cores each

On HTC the **devel** GPU partition has one GPU

To use the **devel** partition add the following line to your submission script:

```
#SBATCH --partition=devel
```

Note , maximum time limit on this partition is 10 minutes, so you must also adjust your time requirement accordingly:

```
#SBATCH --time=00:10:00
```

Submission to interactive partition

An interactive job logs you in to a compute node and gives you a shell.

This allows users to interact with the node it in real time, much like one would interact with a desktop PC, or the login nodes.

You **must** use interactive jobs in order to run pre/post processing and software build activities.

To start an interactive session, you need to use the **`srun`** command

```
srun -p interactive --pty /bin/bash
```

or for a session that allows graphical interfaces (via X forwarding):

```
srun -p interactive --x11 --pty /bin/bash
```

This would allocate 1 core on one interactive node and log you in to the system (giving you a shell on the system). Multiple cores, memory, or other resources can be requested the same way as for **`sbatch`**.

Exiting the the shell ends the job. It will also be aborted once it exceeds the time limit.

GPU submission

GPUs are only available on compute nodes which are part of the HTC cluster.

The most basic way you can access a GPU is by requesting a GPU device using the **`gres`** option in your submission script:

```
#SBATCH --gres=gpu:1
```

The above will request 1 single GPU device (of any type)

Note that - as with CPUs and memory - you will only be able to see the number of GPUs you requested.

You may also request a specific type of GPU device, for example:

```
#SBATCH --gres=gpu:v100:1
```

To request one V100 device

GPU submission (continued...)

Available GPU devices are P100, V100, RTX (Titan RTX), RTX8000, and A100.

Alternatively you can request a GPU (**`--gres=gpu:1`**) and specify the type via a constraint on the GPU SKU, GPU generation, or GPU compute capability:

```
#SBATCH --gres=gpu:1 --constraint='gpu_sku:V100'

#SBATCH --gres=gpu:1 --constraint='gpu_gen:Pascal'

#SBATCH --gres=gpu:1 --constraint='gpu_cc:3.7'

#SBATCH --gres=gpu:1 --constraint='gpu_mem:32GB'

#SBATCH --gres=gpu:1 –constraint='nvlink:2.0'
```

Current list of GPUs on HTC cluster can be found on:
https://arc-user-guide.readthedocs.io/en/latest/arc-systems.html#gpu-resources

High memory nodes

On HTC there are two generally available high memory nodes:

You can use the high-memory nodes by adding a value between 400G and 3000G in the **`--mem`** option:

e.g.

```
#SBATCH --mem=1500G
```

to request 1.5TB

Email support@arc.ox.ac.uk for more details

ARC graphical nodes

You can access the Graphical nodes via a web browser or the client software

You can connect directly via web browser to **`nx.arc.ox.ac.uk`** via the web-based client connection (which is lower quality in terms of visual display).

To access the Graphical nodes via the client software
Download the NoMachine Enterprise Client and install this on your local machine.

More details here:
https://arc-user-guide.readthedocs.io/en/latest/connecting-to-arc.html#connecting-using-arc-graphical-nodes

Job Submission Demonstration

www.it.ox.ac.uk

www.arc.ox.ac.uk

Submission script example

For this demonstration we will connect to the ARC cluster and we will ask for the following resources:

- 2 compute nodes;
- 48 processes per node (using MPI);
- with one CPU per task (the default);
- and a 10 minutes wall time on the ‘devel’ partition.

Submission script for arc cluster

For this the following submission script would be used:

```
#! /bin/bash

#SBATCH --nodes=2
#SBATCH --ntasks-per-node=48
#SBATCH --time=00:10:00
#SBATCH --partition=devel
#SBATCH --job-name=Primes

module load mpitest/1.0

for i in {1..4}
do
        mpirun mpiprimes 1000000
        sleep 5
done
```

If you would like to experiment with this script it may be found at:

```
/apps/common/examples/training/2022/mpi_submit/submit.sh
```

Submitting the job

A SLURM submission script is submitted using the **`sbatch`** command:

```
$ sbatch <name of submission script>
```

e.g.

```
$ sbatch submit.sh
```

SLURM will respond with an output that looks like this:

`submitted batch job <jobid>` (e.g. 273812)

`squeue` monitor the queue
`scancel` cancel a job (made a mistake?)
`seff` view job efficiency

`ls -l` to see the output from the job (must be run from the same directory you submitted the job from)

Managing Jobs

www.it.ox.ac.uk

www.arc.ox.ac.uk

More on submission of jobs

To re-queue your jobs: `sbatch [--requeue --no-requeue]`

Job dependencies: `sbatch -d afterok:<jobid>`

Job arrays: `sbatch -a 1-20`

Requesting GPUs: `sbatch --gres=gpu:1`

Submission queue

Use the command **squeue** to see jobs currently running on the ARC clusters

squeue -u <userid> (list of jobs on the current cluster, arc or htc)
squeue -u <userid> --all (list of jobs on the current cluster, arc and htc)

```
             24021     short      ffs ptch0431  R       6:57      1 arc-c235
             24022     short      ffs ptch0431  R       6:57      1 arc-c067
             24023     short      ffs ptch0431  R       6:57      1 arc-c071
             24024     short      ffs ptch0431  R       6:57      1 arc-c086
             24025     short      ffs ptch0431  R       6:57      1 arc-c142
             24014     short      ffs ptch0431  R      40:33      1 arc-c054
             24015     short      ffs ptch0431  R      40:33      1 arc-c062
             24011     short      ffs ptch0431  R      40:36      1 arc-c269
             24012     short      ffs ptch0431  R      40:36      1 arc-c190
             24013     short      ffs ptch0431  R      40:36      1 arc-c202
             23957     short      ffs ptch0431  R    2:31:47      1 arc-c109
             23958     short      ffs ptch0431  R    2:31:47      1 arc-c241
             23956     short      ffs ptch0431  R    2:31:53      1 arc-c028
             23948     short      ffs ptch0431  R    2:32:11      1 arc-c030
             23568    medium AZBTrsMI univ3182  R 1-03:03:58      4 arc-c[244-245
,254-255]
             23569    medium Ir3.8228 univ3182  R 1-03:03:58      4 arc-c[050,053
,259,263]
             24009 interacti     bash ouit0578  R      49:42      1 arc-c304
             23747    medium    AuTop mans3954  R    8:53:37      2 arc-c[187-188
]
             22872      long impMinte scat7362  R 2-18:04:04      4 arc-c[072,139
,239,256]
```

Information about the cluster

`sinfo`

This command reports the state of partitions and nodes managed by SLURM on the cluster ARC or HTC :

```
sinfo
PARTITION   AVAIL  TIMELIMIT  NODES  STATE NODELIST
short          up   12:00:00      2  drain arc-c[001,125]
short          up   12:00:00      4    mix arc-c[063,133,135,293]
short          up   12:00:00    257  alloc arc-c[002-015,028,046-062,064-124,126-132,134,
136-158,160-292]
short          up   12:00:00     30   idle arc-c[016-027,029-045,159]
medium         up 2-00:00:00      1  drain arc-c125
medium         up 2-00:00:00      4    mix arc-c[063,133,135,293]
medium         up 2-00:00:00    242  alloc arc-c[046-062,064-124,126-132,134,136-158,160-
292]
medium         up 2-00:00:00      1   idle arc-c159
long*          up   infinite      1  drain arc-c125
long*          up   infinite      4    mix arc-c[063,133,135,293]
long*          up   infinite    242  alloc arc-c[046-062,064-124,126-132,134,136-158,160-
292]
long*          up   infinite      1   idle arc-c159
devel          up      10:00      2   idle arc-c[302-303]
interactive    up 1-00:00:00      1    mix arc-c304
interactive    up 1-00:00:00      1   idle arc-c305
[ouit0578@arc-login01 ~]$
```

More information about your job

scontrol show <jobid>

This command gives more information about job StartTime and End Time,nodes allocated …

```
[ouit0578@arc-login02 ~]$ scontrol show JobID=23776
JobId=23776 JobName=wthiehip
   UserId=lina3518(5913) GroupId=internal(1001) MCS_label=N/A
   Priority=1592 Nice=0 Account=chem-jlpbmd QOS=normal
   JobState=PENDING Reason=Priority Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=4-04:00:00 TimeMin=N/A
   SubmitTime=2021-04-21T01:03:18 EligibleTime=2021-04-21T01:03:18
   AccrueTime=2021-04-21T01:03:18
   StartTime=Unknown EndTime=Unknown Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2021-04-21T14:14:05
   Partition=long AllocNode:Sid=arc-slurm:2863937
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=(null)
   NumNodes=16-16 NumCPUs=768 NumTasks=32 CPUs/Task=24 ReqB:S:C:T=0:0:*:*
   TRES=cpu=768,mem=6T,node=16,billing=768
   Socks/Node=* NtasksPerN:B:S:C=2:0:*:* CoreSpec=*
   MinCPUsNode=48 MinMemoryCPU=8G MinTmpDiskNode=0
   Features=cpu DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/data/chem-jlpbmd/lina3518/Z-Wang/final_complex/wildtype/HIE_HIP/04.s
h
   WorkDir=/data/chem-jlpbmd/lina3518/Z-Wang/final_complex/wildtype/HIE_HIP
   StdErr=/data/chem-jlpbmd/lina3518/Z-Wang/final_complex/wildtype/HIE_HIP/slurm
-23776.out
   StdIn=/dev/null
   StdOut=/data/chem-jlpbmd/lina3518/Z-Wang/final_complex/wildtype/HIE_HIP/slurm
-23776.out
```

Why does my queued job not start ?

Jobs may be queued for various reasons. A job may be waiting for resources to become available. Or you might have hit a limit for the maximum number of jobs that can be running on the system. One way to determine why a job is queuing is to use the **`scontrol show job`** command.

For example, if the job ID is 12345:

```
scontrol show job 12345
```

If the **`Reason`** value of the job state is **`JobHeldUser`**:

This means your job is held because your ARC project has run our out of compute "credit". Please contact support@arc.ox.ac.uk for a top up.

You can release user held jobs using the command:

```
scontrol release <JobID>
```

More information on non-running jobs can be found here:

https://arc-user-guide.readthedocs.io/en/latest/slurm-faq.html

Checking credit balance

Users are given a credit allocation, usually shared with other users of the same project. You can check the number of credits at any point using the command **mybalance**

The command shows the existing number of credits and the number of credits reserved from jobs for all users sharing the same project.

```
$ mybalance

Please wait: Calculating balance ...

You are a member on the following project(s): system

system-priority,system-basic and your current balance is: 1077842827 credits ( 299400 hours)
Detailed account balance:
Id        Name                Amount    Reserved Balance   CreditLimit Available
51        system              897848728 0        897848728 0           897848728
5723      system-priority     89994288  0        89994288  0           89994288
```

Managing Data & Storage

www.it.ox.ac.uk

www.arc.ox.ac.uk

User and project ARC Storage

ARC provides users with a number of storage areas on the high performance file-system.

`$HOME  /home/username`
Small quota. Used during login. (15GB per user)

`$DATA  /data/project/username`
Large quota shared between project members (5TB per project)

You should also note **`$TMPDIR`** is local to a compute node, **`$SCRATCH`** is on a shared file-system and available to all nodes in a job, if a job spans multiple nodes

Use the **`myquota`** command to check your **`$HOME`** and **`$DATA`** storage use.

https://arc-user-guide.readthedocs.io/en/latest/arc-storage.html

Data integrity and backup

ARC makes best effort to ensure the integrity of data stored on our facilities. However, we are under no obligation to guarantee the integrity or availability of data - **This is the responsibility of the individual user.**

NO BACKUPS

Limited snapshots of **`$HOME`** are taken. However, ARC does not accept any liability, financial or otherwise for loss of data.

We recommend that users employ standard industry practice for their important data and store it at sites other than ARC, for example, on their department servers.

Transferring data to/from ARC

Copying to the ARC systems

Make sure you know the full path to the destination directory on ARC - The best way to to this is to log in to ARC, change to that directory and run the command **`pwd`** this will show you the full path to the directory.

See details on our website:

https://arc-user-guide.readthedocs.io/en/latest/arc-copying-data.html

More Information...

www.it.ox.ac.uk

www.arc.ox.ac.uk

ARC User documentation

Main ARC website is: www.arc.ox.ac.uk

ARC User Guide:
https://arc-user-guide.readthedocs.io/en/latest/

ARC Software Guide:
https://arc-software-guide.readthedocs.io/en/latest/

Web site includes policy documents and information on costs for purchasing priority compute credit.

https://www.arc.ox.ac.uk/arc-accounting
https://www.arc.ox.ac.uk/arc-service-level-agreement

Questions...

www.it.ox.ac.uk

www.arc.ox.ac.uk