Getting Started with ARC

Hilary 2025

The ARC Team

https://www.arc.ox.ac.uk

https://www.it.ox.ac.uk

Why High Performance Computing?

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 timeframe, 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:
  MetOffice needs ~1 Pflops
  1 milion times more that a PC
  Requires parallel Processing on many CPUs;
- Top supercompters: petaflops to exaflops 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.

Parallel Processing Models

Models of Parallelism

Distributed Memory

Distributed Memory Programming Model:

- **System Type:** Multi-core where each core has its dedicated memory;
- **Memory Access:** A core's memory is private and cannot be accessed directly by other cores;
- **Parallelism Unit:** The process, where a programme consists of multiple processes;
- **Information Exchange:** Communication between processes requires explicit message passing;
- **Dominant Programming Standard:** Message Passing Interface (**MPI**) is the predominant framework for implementing this model.

Distributed Memory Hardware:

- **Traditional Model:** Conceptually like many PCs linked together (resembling a Beowulf cluster);
- **Modern Approach:** Incorporates multi-core nodes, often in high-density blades, each equipped with its own dedicated memory, using high bandwidth, low-latency networking for efficient communication; Modular, off-the-shelf components such as 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;
- **Parallelism Unit:** Threads, where a single program consists of multiple threads running concurrently;
- **Information Exchange:** Threads communicate via shared variables;
- **Dominant Programming Standard:** OpenMP.

Shared Memory Hardware:

- conceptually, this resembles a single computer, with a large memory pool and multiple processing cores;
- includes both small and cost effective systems like desktops, as well as high-end configurations featuring premium, high-bandwidth memory access.

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

Distributed vs 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 optimised properly;
- Fosters effective parallel programming techniques.

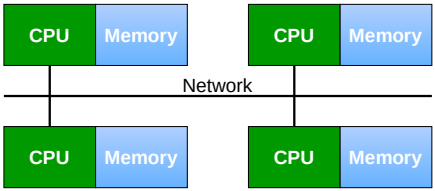


Figure: Distributed Memory

Distributed vs Shared Memory

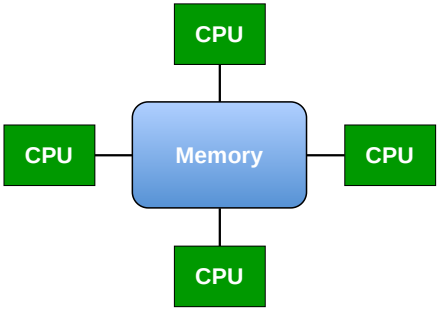


Figure: Shared Memory

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.

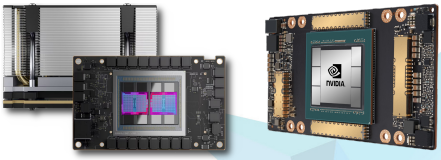
ARC HPC Services

HPC Clusters

Cluster services:

- HTC – high throughput (Shared Memory)
- ARC – large scape parallel (Distributed Memory)

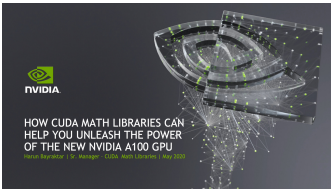
GPU nodes on HTC include:
Pascal, Volta, Turing, Ampere, Grace Hopper, MI250X



Hosted at Begbroke Data Centre

HTC Cluster

- Minimum job size: 1 core
- CPU and GPU node on this cluster include:
  - High Memory Nodes
  - Single Precision GPU Compute Nodes
  - Double Precision GPU Compute Nodes
- GPU nodes include:
  V100, A100, P100, Titan RTX, RTXA6000, V100-LS, GH200, MI250X, ...

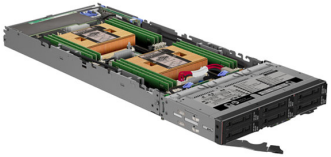


https://arc-user-guide.readthedocs.io/en/latest/arc-systems.html

ARC Cluster

ARC provides a capability cluster comprising of:

- >250 general compute nodes; designed for multi-node parallel computation
- OS: CentOS 8.x (two nodes for legacy software running CentOS 7.7)
- Scheduler: SLURM



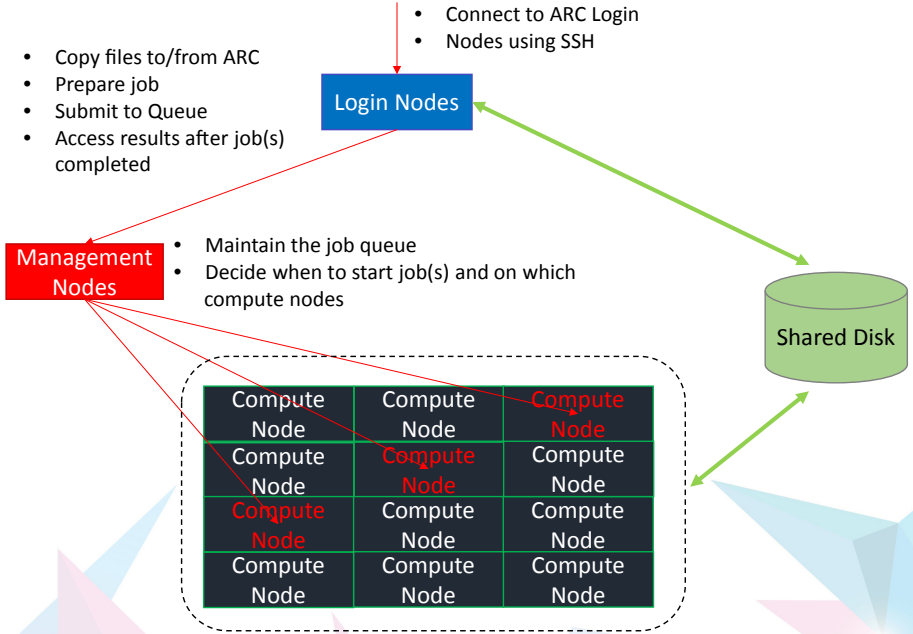
>14 000 CPU cores

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



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 and the command line interface 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`

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/

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 the 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:

```
[user@arc ~]$ module spider matlab
----------------------------------------------------------------------------
  MATLAB:
----------------------------------------------------------------------------
     Versions:
        MATLAB/R2019b
        MATLAB/R2020a
        MATLAB/R2020b
        MATLAB/R2021b
        MATLAB/R2022a
        MATLAB/R2022b
        MATLAB/R2023a
        MATLAB/R2023b
----------------------------------------------------------------------------
```

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

```
[user@arc ~]$ module load MATLAB/R2023b
```

Module load example

Loading MATLAB via the linux `module load` command on the terminal:

```
[user@arc ~]$
[user@arc ~]$ matlab
-bash: matlab: command not found
[user@arc ~]$ module load MATLAB/R2023b
[user@arc ~]$ matlab -nojvm -nosplash
MATLAB is selecting SOFTWARE OPENGL rendering.

                            < M A T L A B (R) >
                  Copyright 1984-2023 The MathWorks, Inc.
                R2023b Update 7 (23.2.0.2515942) 64-bit (glnxa64)
                            January 30, 2024


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:

```
[user@arc ~]$ module load Anaconda3
```

Or use

```
[user@arc ~]$ module spider Anaconda
```

And use the specific Anaconda version you need.
For example:

```
[user@arc ~]$ module load Anaconda3/2024.02-1
```

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

Adding Python Anaconda modules

Follow the steps below, on an interactive node:

```
[user@arc ~]$ srun -p interactive --pty /bin/bash
[user@arc ~]$ module load Anaconda3
[user@arc ~]$ export CONPREFIX=$DATA/myenv
[user@arc ~]$ conda create --prefix $CONPREFIX
[user@arc ~]$ source activate $CONPREFIX
```

You can now install the modules you need:

```
[user@arc ~]$ conda install <modulename>
```

For a GPU version same commands on an interactive node on the htc cluster, adding:

```
[user@arc ~]$ module spider CUDA
[user@arc ~]$ module load CUDA/<version>
```

Using R on ARC cluster

```
[user@arc ~]$ module spider R
[user@arc ~]$ module load R/4.4.0-gfbf-2023a
```

The base install has many popular R packages installed.

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:

```
[user@host ~]$ ssh <userid>@arc-login.arc.ox.ac.uk
```

Or if you want X to forward graphics applications:

```
[user@host ~]$ 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, maximum run time 30 days)
- **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:

```
[user@arc ~]$ srun -p interactive --pty /bin/bash
```

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

```
[user@arc ~]$ 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 1.5 TB

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 `https://nx.arc.ox.ac.uk` via the webbased 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

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:

```
[user@arc ~]$ sbatch <name of submission script>
```

e.g.

```
[user@arc ~]$ 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

More on submission of Jobs

- To re-queue your jobs:

```
[user@arc ~]$ sbatch [--requeue | --no-requeue]
```

- Jobs dependencies:

```
[user@arc ~]$ sbatch -d afterok:<jobid>
```

- Job arrays:

```
[user@arc ~]$ sbatch -a 1-20
```

- Requestion GPUs:

```
[user@arc ~]$ sbatch --gres=gpu:1
```

Submission queue

Command: squeue

```
           7120226     short FC2_KH2P mert3235  R    1:04:12      1 arc-c032
           7120227     short FC2_KH2P mert3235  R    1:04:12      1 arc-c032
           7120293     short FC2_KH2P mert3235  R    1:04:12      1 arc-c033
           7120302     short FC2_KH2P mert3235  R    1:04:12      1 arc-c033
           7120303     short FC2_KH2P mert3235  R    1:04:12      1 arc-c033
           7120304     short FC2_KH2P mert3235  R    1:04:12      1 arc-c033
           7120305     short FC2_KH2P mert3235  R    1:04:12      1 arc-c033
           7120366     short FC4_KH2P mert3235  R    1:04:12      1 arc-c034
           7120367     short FC4_KH2P mert3235  R    1:04:12      1 arc-c034
           7120368     short FC4_KH2P mert3235  R    1:04:12      1 arc-c034
           7119188     short FC4_Na3P mert3235  R    3:18:48      1 arc-c033
           7119189     short FC4_Na3P mert3235  R    3:18:48      1 arc-c034
           7119190     short FC4_Na3P mert3235  R    3:18:48      1 arc-c035
           7119191     short FC4_Na3P mert3235  R    3:18:48      1 arc-c036
           7119192     short FC4_Na2C mert3235  R    3:18:48      1 arc-c038
           7119186     short FC4_Na3P mert3235  R    3:26:23      1 arc-c039
           7119185     short FC4_Na3P mert3235  R    3:30:01      1 arc-c096
           7119166     short FC2_K2C2 mert3235  R    3:30:01      1 arc-c288
           6992125     short  r3dep_5 hert5155  R    6:16:08      8 arc-c[019-020,029,041,043-045,307]
           7119094     short     cwx3 phys2414  R    1:04:12      1 arc-c035
           7119212     short   qising phys2414  R    1:04:12      1 arc-c036
           7120682     short pz_EoS_M magd4268  R      18:15      1 arc-c056
     7106802_1235     short       SnS ball5018  R      10:19      1 arc-c242
     7106802_1236     short       SnS ball5018  R    1:11:56      1 arc-c055
     7106802_1237     short       SnS ball5018  R    1:11:56      1 arc-c057
     7106802_1238     short       SnS ball5018  R    1:11:56      1 arc-c058
     7106802_1239     short       SnS ball5018  R    1:11:56      1 arc-c067
     7106802_1240     short       SnS ball5018  R    1:11:56      1 arc-c069
     7106802_1241     short       SnS ball5018  R    1:11:56      1 arc-c071
           7119431     short Droplets exet5317  R    1:11:56      1 arc-c072
           7119432     short Droplets exet5317  R    1:11:56      1 arc-c078
     7119184_1251     short       SnS ball5018  R    1:11:56      1 arc-c079
     7119184_1252     short       SnS ball5018  R    1:11:56      1 arc-c080
     7119184_1253     short       SnS ball5018  R    1:11:56      1 arc-c083
     7119184_1254     short       SnS ball5018  R    1:11:56      1 arc-c084
     7119184_1255     short       SnS ball5018  R    1:11:56      1 arc-c085
           7119370     short      MACE ball5018  R      10:18      1 arc-c301
[ouit0578@arc-login03 ~]$
```

Information about the Cluster

`sinfo` reports the state of partitions and nodes managed by SLURM on the clusters ARC or HTC:

```
[ouit0578@htc-login03 ~]$ sinfo
PARTITION   AVAIL  TIMELIMIT  NODES  STATE NODELIST
short          up   12:00:00      4 drain* htc-c034,htc-g[020-021,025]
short          up   12:00:00      2  down* htc-g[028,033]
short          up   12:00:00      2   comp htc-g[035,048]
short          up   12:00:00      3   drng htc-g[004,045-046]
short          up   12:00:00      4  drain htc-g[030-032,034]
short          up   12:00:00      8   resv htc-g[022-024,026-027,029,043,052]
short          up   12:00:00     29    mix htc-c[005-007,010-013,033],htc-g[001-003,005-006
,010-011,013-018,036-038,044,047,049-051]
short          up   12:00:00     37  alloc htc-c[008-009,014-032,035-046],htc-g[009,012,041
-042]
medium         up 2-00:00:00      1 drain* htc-c034
medium         up 2-00:00:00      1   comp htc-g048
medium         up 2-00:00:00      2   drng htc-g[045-046]
medium         up 2-00:00:00     18    mix htc-c[006-007,010-013,033],htc-g[010-011,013-018
,044,047,049]
medium         up 2-00:00:00     35  alloc htc-c[008-009,014-032,035-046],htc-g[009,012]
long*          up   infinite      1 drain* htc-c034
long*          up   infinite      1   comp htc-g048
long*          up   infinite      2   drng htc-g[045-046]
long*          up   infinite     18    mix htc-c[006-007,010-013,033],htc-g[010-011,013-018
,044,047,049]
long*          up   infinite     35  alloc htc-c[008-009,014-032,035-046],htc-g[009,012]
devel          up      10:00      1   idle htc-g039
interactive    up   12:00:00      1    mix htc-g040
[ouit0578@htc-login03 ~]$
```

More info about your job

`scontrol show JobID=<jobid>`

gives more information about jobs's starttime and endtime, nodes allocated, ...

```
[ouit0578@arc-login03 ~]$  scontrol show JobID=6992125
JobId=6992125 JobName=r3dep_5
   UserId=hert5155(5869) GroupId=internal(1001) MCS_label=N/A
   Priority=1583 Nice=0 Account=chem-amais QOS=standard
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=06:16:46 TimeLimit=12:00:00 TimeMin=N/A
   SubmitTime=2024-01-08T01:31:01 EligibleTime=2024-01-29T20:08:49
   AccrueTime=2024-01-29T20:08:49
   StartTime=2024-02-01T08:28:38 EndTime=2024-02-01T20:28:38 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-02-01T08:28:38
   Partition=short AllocNode:Sid=arc-slurm:1145388
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=arc-c[019-020,029,041,043-045,307]
   BatchHost=arc-c019
   NumNodes=8 NumCPUs=384 NumTasks=384 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=384,mem=3000G,node=8,billing=384
   Socks/Node=* NtasksPerN:B:S:C=48:0:*:* CoreSpec=*
   MinCPUsNode=48 MinMemoryCPU=8000M MinTmpDiskNode=0
   Features=cpu DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=../mlp_lammps_varMQ_11.sh -c u16_m1.5_fw0.1_PQ1k_G18_scratch.mtp -s /home/hert5155/scripts/lammps/sc1M.dat
a
   WorkDir=/data/chem-amais/hert5155/GAP_to_MTP/varMQs/1M_u16_r3
   Comment=[ "Department": "Inorganic Chemistry (DH)", "Division": "MPLS" ]
   StdErr=/data/chem-amais/hert5155/GAP_to_MTP/varMQs/1M_u16_r3/slurm-6992125.out
   StdIn=/dev/null
   StdOut=/data/chem-amais/hert5155/GAP_to_MTP/varMQs/1M_u16_r3/slurm-6992125.out
   Power=
   NtasksPerTRES:0
```

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 JobID=<num>` command.

For example, if the job ID is 12345:

```
[user@arc ~]$ scontrol show JobID=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:

```
[user@arc ~]$ scontrol release 12345
```

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.

```
[ouit0578@arc-login03 ~]$ mybalance
Please wait: Calculating balance ...
You are a member on the following project(s): system,system-priority,system-basic,engs-unified-model,engs-unified-mod
el-priority,engs-unified-model-basic
and your current balance is: 812601212 credits (~225722 core hours)

Detailed account balance:
Id    Name                          Amount    Reserved Balance   CreditLimit Available
----- ----------------------------- --------- -------- --------- ----------- ---------
51    system                        580134892        0 580134892           0 580134892
5723  system-priority                89993921        0  89993921           0  89993921
5725  system-basic                   89999752        0  89999752           0  89999752
93490 engs-unified-model             52472647        0  52472647           0  52472647
93491 engs-unified-model-priority           0        0         0           0         0
93492 engs-unified-model-basic              0        0         0           0         0
[ouit0578@arc-login03 ~]$
```

Managing Data & Storage

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. (15 GiB)

`$DATA` `/data/<projectname>/<username>`
Large quota shared between project members (5 TiB 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 files 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…

ARC User Documentation

- Main ARC website:
  https://www.arc.ox.ac.uk/
- ARC User Guide:
  https://arc-user-guide.readthedocs.io/
- ARC Software Guide:
  https://arc-software-guide.readthedocs.io/
- Website includes policy documents and information on costs for purchasing priority compute credits
  https://www.arc.ox.ac.uk/arc-accounting/
  https://www.arc.ox.ac.uk/arc-service-level-agreements/

Thank You!

Any Questions?