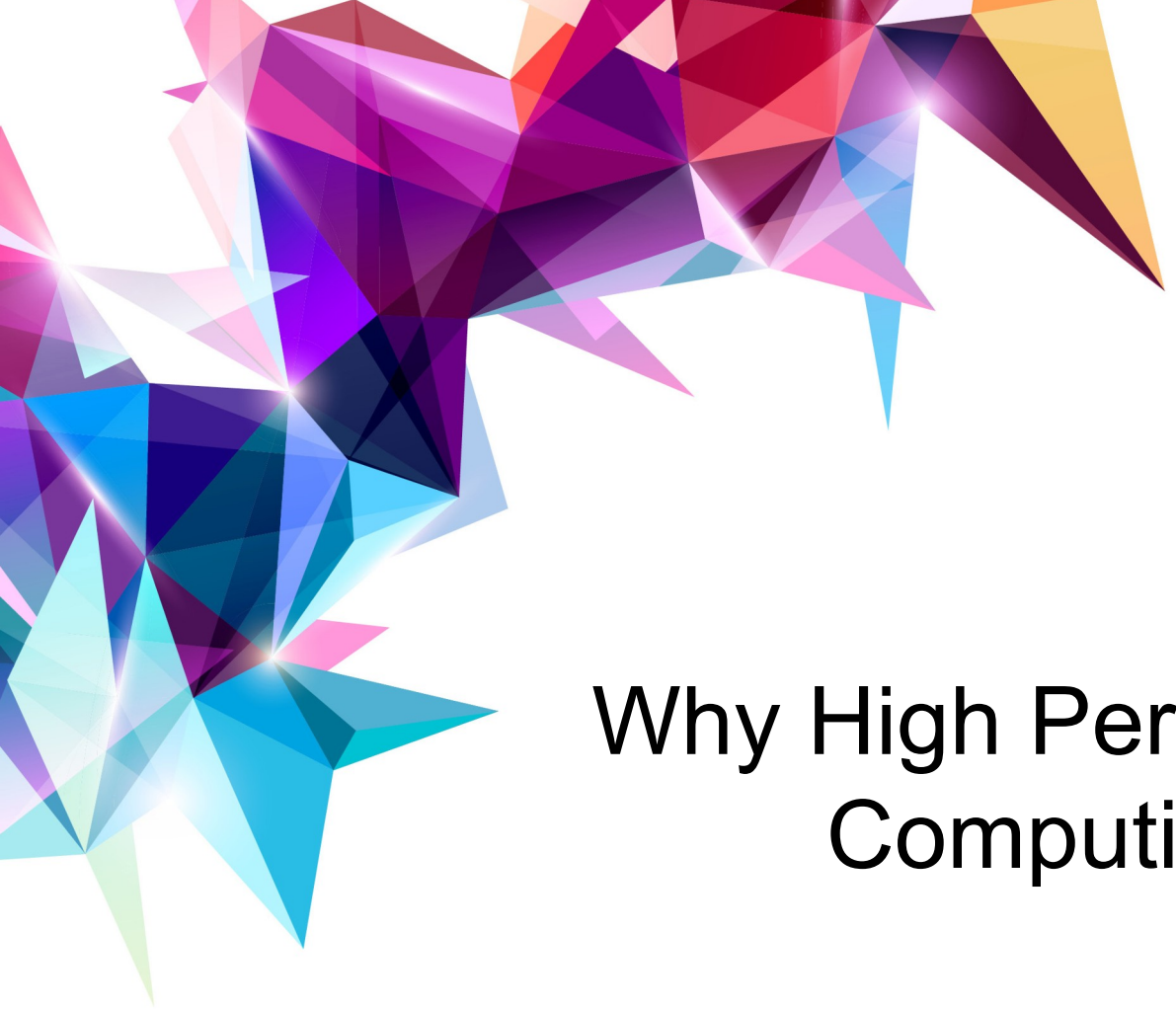
# Getting Started with ARC

**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**: Applications requiring a large amount of compute with high performance inter-processor communication.

**Data Intensive (I/O intensive)**: Applications requiring operating on a large amount of data requiring fast efficient ingress/egress of data. High Performance in the storage hierarchy.

**High Throughput**: Many independent/embarrassingly parallel jobs to be executed, in many cases could be done on a workstation or laptop but need to be done hundreds if not thousands of times

**Memory Intensive**: A single problem/job requiring a large amount of memory to be tractable. Applications are varied but usually require all data to be in local memory (large data sets) or where the outputs used in subsequent calculation will exceed memory available on other resources (Quantum Machine modelling).

ARC provide generalised **High Performance Computing (HPC)** resources that provide good coverage of the above Research Computing categories as well as high performance storage, user and applications' support.

# High Performance Computing

There is no single, all-encompassing definition for HPC.

For our purposes it is:

Computing which cannot be achieved in a reasonable time on a laptop or desktop workstation and may be carried out on multiple processors, in **parallel**.

***HPC is the ability to do work in less time, do more work in the same time, or achieve something that is otherwise impossible.***

HPC does not always mean running code in parallel on a cluster – access to a single large memory “fat” node – is just as valid.

# High Performance Computing

But how fast is fast enough?

- A desktop PC can deliver tens of Gflops (flops = floating point operations per second) tens of millions is a lot of flops but that may or may not be enough…
- An extreme example: short range weather forecast : prediction for the next day **must** be delivered in less than 1 day
- The Met Office models translate into a requirement for ~1 Pflops , that is 1 million times more that the PC, so the simulation has to run on many CPUs, working together on the same model in **parallel**.

The fastest supercomputers achieve Petaflop/Exaflop performance

# Why use an HPC Cluster

- Don't want to tie up your own machine
- Have many long running jobs to run
- Want to run in parallel to get results quicker
  - This may include job parallelism of serial processes
- Need more disk space
- Need more memory
- Want to use software installed on the cluster
- Need to use GPUs
- Want to benefit from ARC support

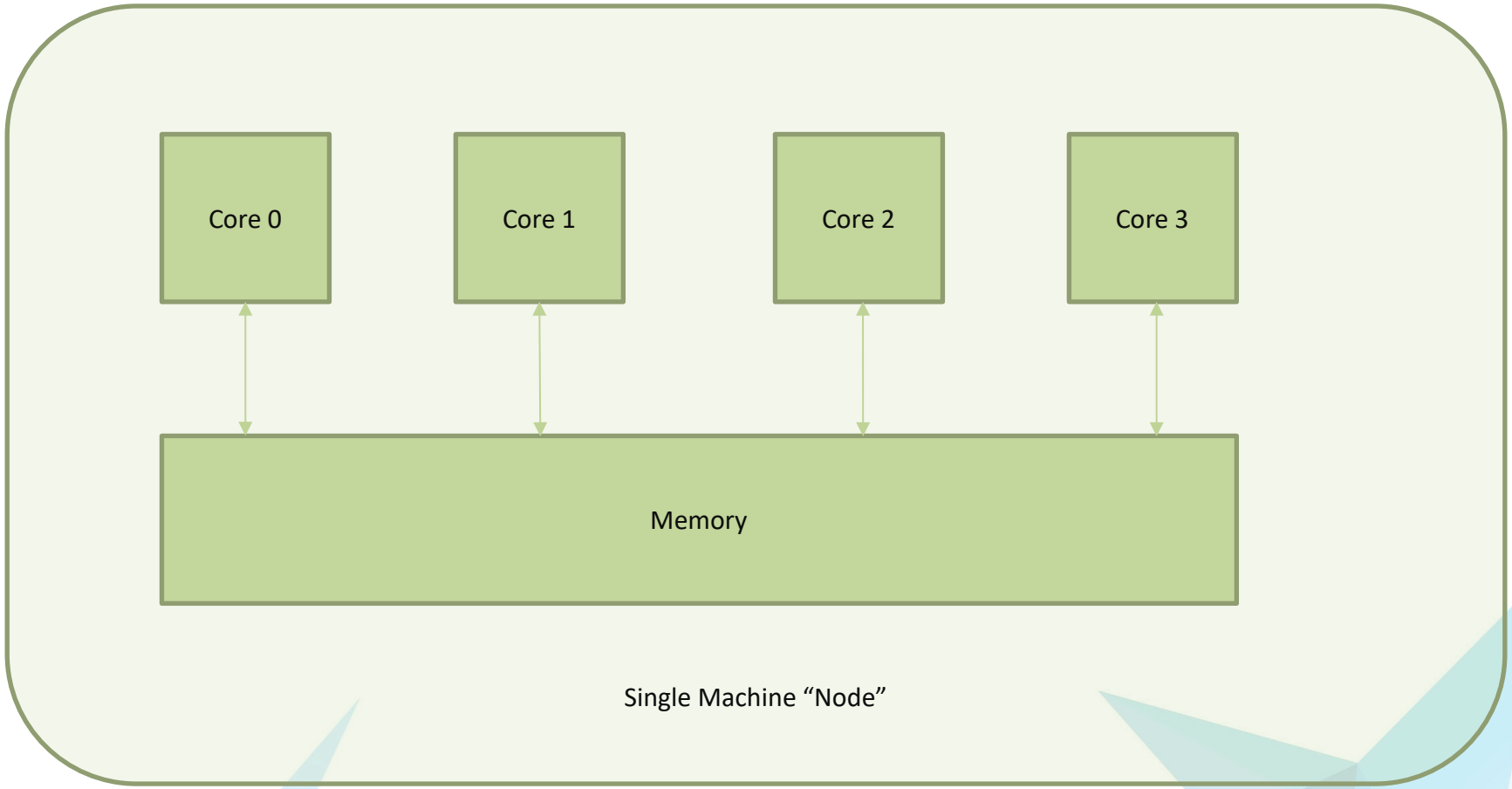
# Before we start…

- Glossary
  - **core** = unit that does the work (sometimes use CPU as a synonym)
  - **processor** = collection of cores in a single package all sharing the same memory
  - **node** = a collection of processors all sharing the same memory
  - **interconnect** = the network in a machine that joins together the separate nodes

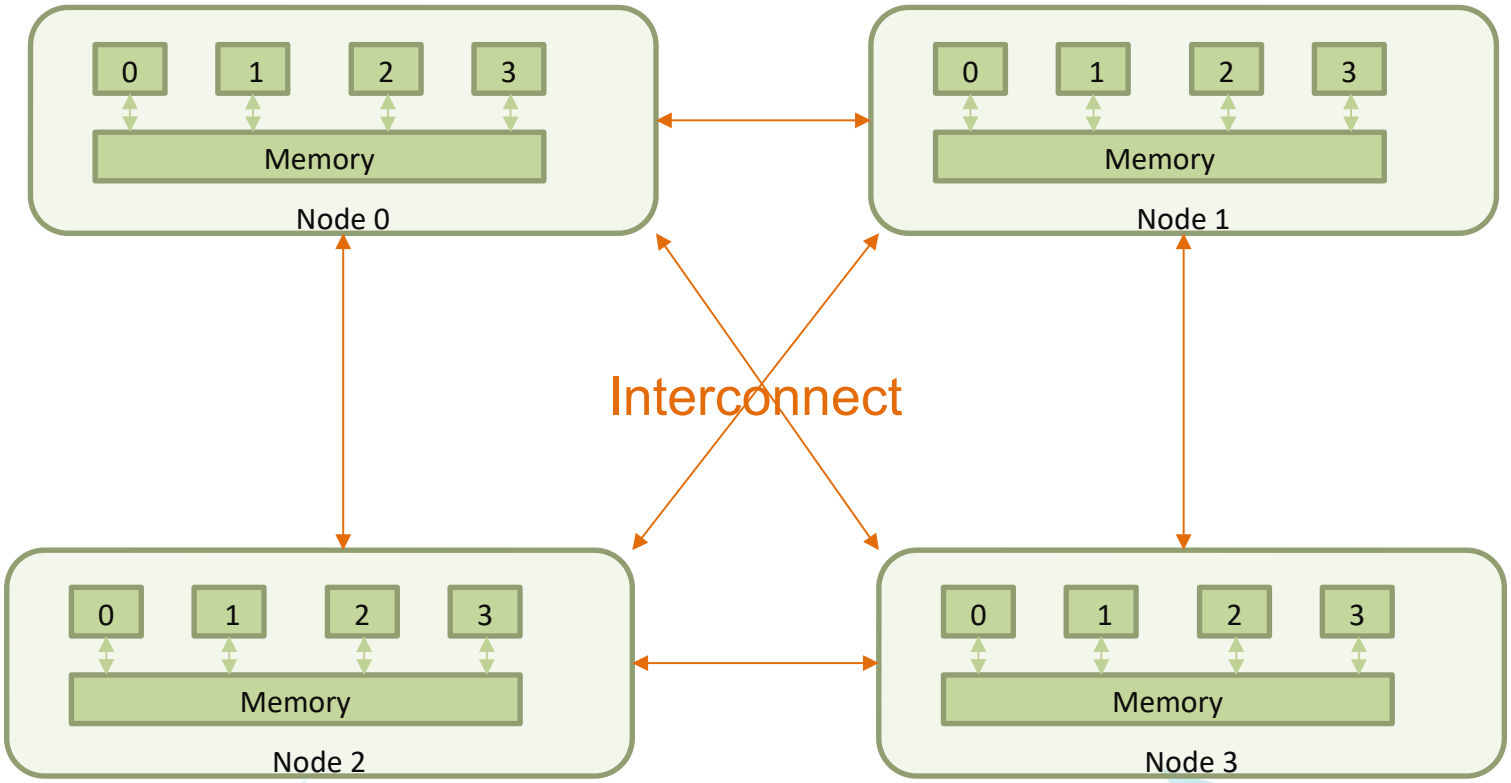
Note: each node has its own memory and cannot directly "see" another node's memory.

- Distinction between processor, process and thread
  - **processor** = a physical piece of **hardware**
  - **process** = an instance of a running program (**software**)
    - essentially it has two components: instructions to execute and associated data
    - in parallel programming we often have multiple instances (processes) of the same program…
  - a process always consists of one or more **threads** of execution

# This could be your laptop...



# ...or a simple cluster.




Login Node
( Editing, Compilation, Job Submission etc.)

# Parallel Processing Models

# Models of parallelism: **Distributed Memory**

**Distributed Memory Programming Model**:

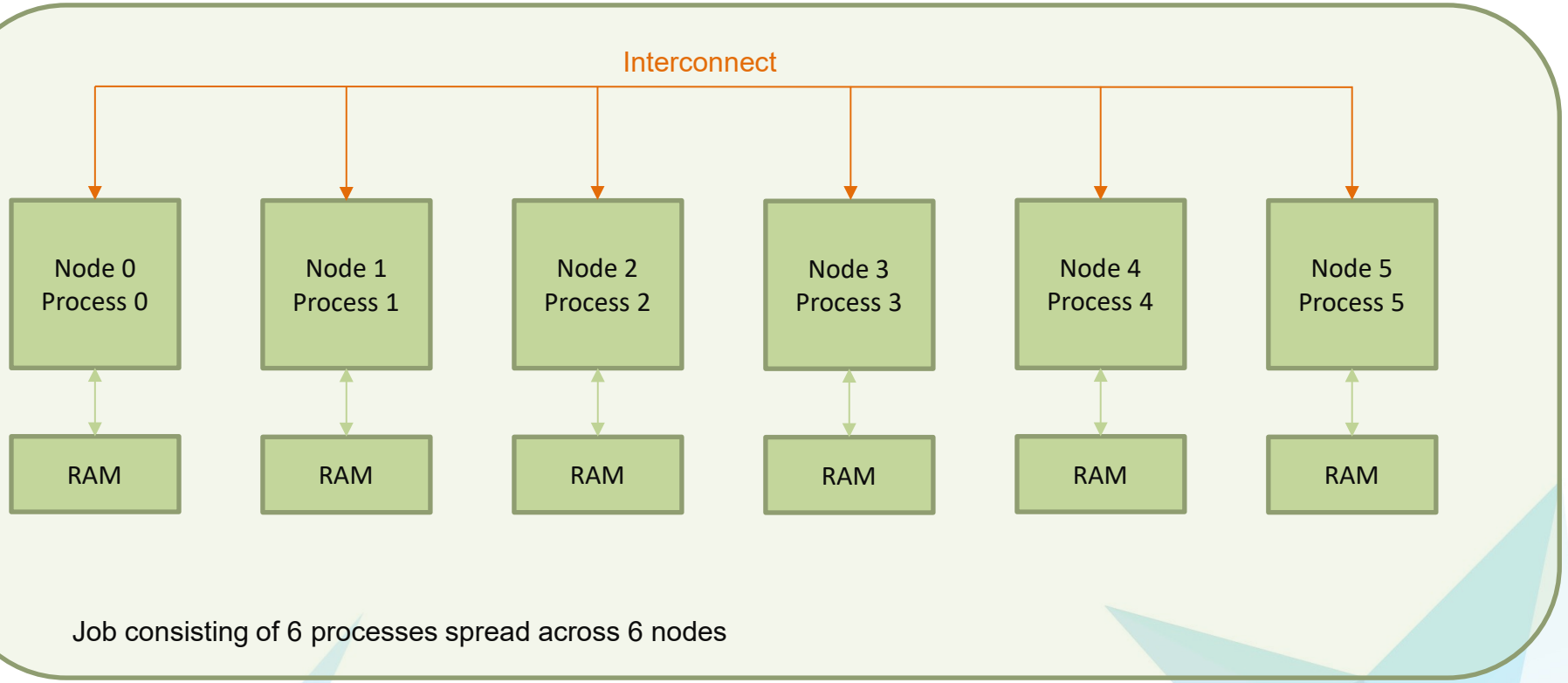
- multi-core system, each core has its own private memory
- local core memory is invisible to all other processors
- agent of parallelism: the **process** (program = collection of processes)
- exchanging information between processes requires **explicit** message passing
- the dominant programming standard: **MPI**

**Distributed Memory Hardware**:

- conceptually, many PCs connected together (traditional Beowulf cluster)
- current approach:
  - multi-core computer nodes (high-density blades) with own memory
  - high-bandwidth, low-latency network connection
  - off-the shelf modular technology (high-end CPUs, standard hard disk)
  - accounts for the largest HPC systems

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

# Distributed Memory View
## Multiple nodes




Job consisting of 6 processes spread across 6 nodes

# Models of parallelism: **Shared Memory**

**Shared Memory Programming Model**:

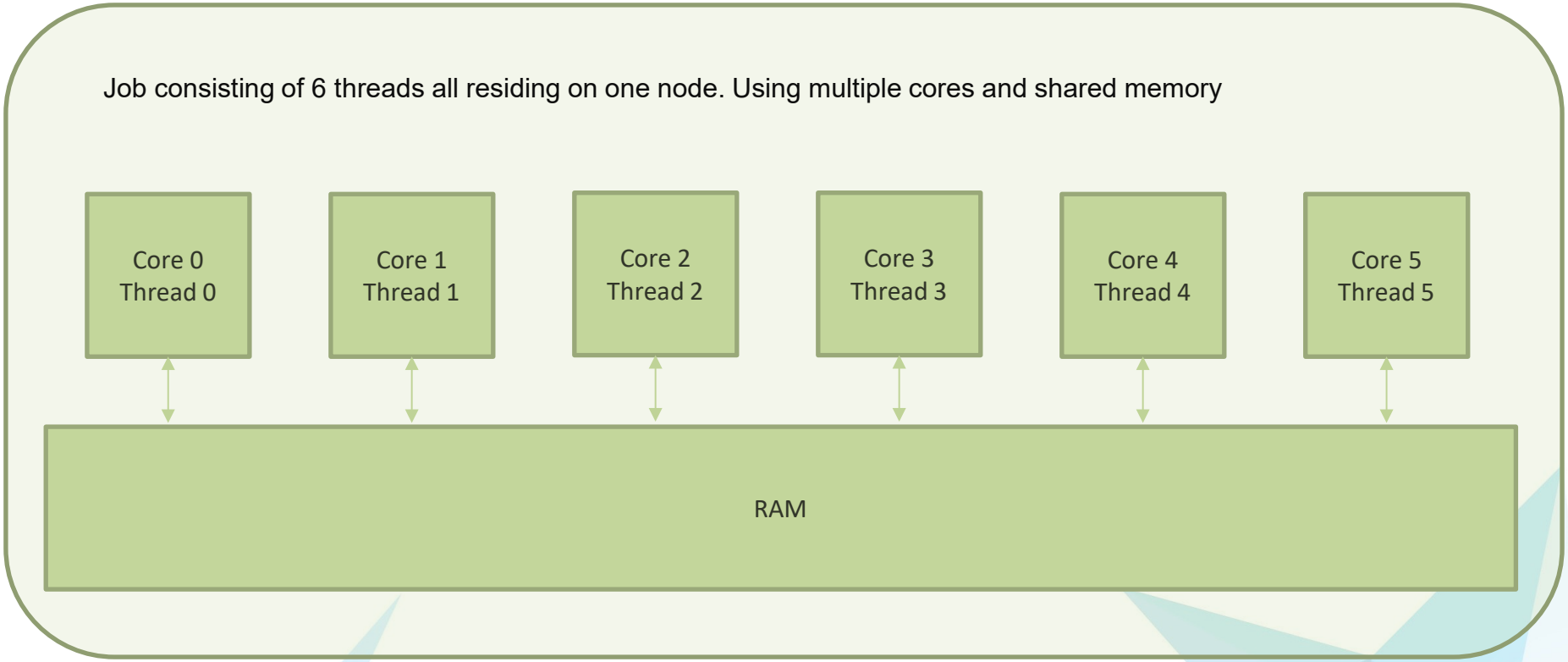
- multi-core system
- each core has access to a shared memory space
- agent of parallelism: the **thread** (program = collection of threads)
- threads exchange information **implicitly** by reading/writing shared variables
- the dominant programming standard: **OpenMP**

**Shared Memory Hardware**:

- conceptually, a single PC, with a large memory and many cores
- accounts for both small and inexpensive systems (desktops) and very large and expensive system (with very expensive high bandwidth memory access)

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

# Shared Memory View
## Single node



# Distributed Memory v. Shared Memory

- Distributed Memory:
  - **Can scale to any number of cores**
  - Requires special tools/libraries (e.g. MPI) to compile and run the code
  - Can be harder to program that shared memory
  - But will generally perform better if done well
  - And it teaches good parallel programming “habits”
- Shared memory
  - **Is usually limited to the number of cores in a node**
    - Can overpopulate, good for debug, bad idea for performance
  - Generally just requires an extra flag on the compiler
  - Can be easier to program than distributed memory
  - It is often hard to get good parallel performance
    - Sharing things is not good for parallelism …
  - Can easily let people be a bit sloppy when programming …

# 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)

Specialised architecture services :

(Nvidia DGX1V, MaxQ)

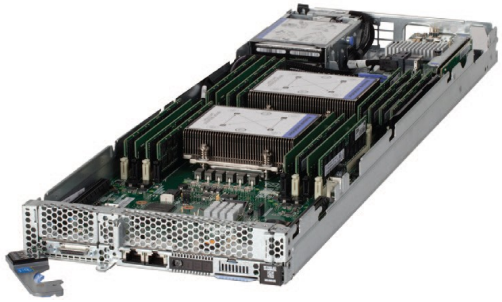
Hosted at Begbroke data centre

# HTC Cluster


- Minimum job size: 1 core
- CPU and GPU nodes on this cluster include:
- High Memory Nodes
- Double Precision GPU Compute Nodes
- Single Precision GPU Compute Nodes

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

Login Nodes

Management Nodes

- Maintain the job queue
- Decide when to start job(s) and on which compute nodes

Shared Disk

| | | |
|---|---|---|
| Compute Node | Compute Node | Compute Node |
| Compute Node | Compute Node | Compute Node |
| Compute Node | Compute Node | Compute Node |
| Compute Node | Compute Node | Compute Node |

# Software Environment

www.it.ox.ac.uk

www.arc.ox.ac.uk

# Linux Operating System

All ARC systems run the Linux Operating System (as opposed to Windows or MacOS).

Reasons why Linux is commonly used in HPC :

- **Cost**
- **Reliability**
- **Performance**

You will need a basic understanding of Linux to use ARC, there are online resources that can provide this information.

See the ARC training pages for more information:

http://www.arc.ox.ac.uk/training

# Environment Modules

`module` is a Linux utility, which is used to manage of working environment in preparation for running the applications installed on the ARC systems.

By loading the module for a certain installed application, the environment variables that are relevant for that application are automatically defined or modified.

The best way to search for an application you require on the system is by using the `module spider` command, for example:

```
module spider matlab
```

Returns the choice of:

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

You could then use: `module load MATLAB/R2022a` to load the latest MATLAB version.

# ARC Software Environment

The ARC/HTC software environment comprises a mixture of commercial applications, software built using the EasyBuild framework (http://easybuild.io) and software built using our own local build recipes.

You should use the environment modules system (via the `module` command) to load applications into your environment on ARC/HTC.

Because the EasyBuild framework adds many new module components into the module list - the best way to search for an application you require on the system is by using the `module spider` command.

If you wish to browse a current list ARC modules online, you can find it here:

ARC Module List:
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.

To see the current versions use :

```
module spider anaconda3
```

or

```
module spider anaconda2
```

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

ARC supports **Singularity (Apptainer)** for containerised applications.

As well as running its own native containers **Singularity** is able to run **Docker** containers.

**Singularity** is available as part of the OS so no `module load` is required.

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:

## **S**imple **L**inux **U**tility for **R**esource **M**anagement

- Manages the job queue , determines when jobs start, what order and on which nodes.
- Manages the compute nodes
- SLURM schedules work on compute nodes that are free
- SLURM has support for “accelerator cards” such as Nvidia GPU nodes
- Same scheduler on ARC and HTC clusters

# Job submission

A submission script describes the resources you need for SLURM to allocate to your job.

It also contains the commands needed to execute the application(s) you wish to run, including any set-up the application(s) may require.

The script needs to be created using a Linux text editor such as `nano` or `vi`

e.g.

```
nano submit.sh
```

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

# Job submission (Example)

Here is a simple Linux shell script (simple text file) with instructions to SLURM
SLURM instructions or directives (`#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

# Connecting to ARC

www.it.ox.ac.uk

www.arc.ox.ac.uk

# 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

# Changing your password

www.it.ox.ac.uk

www.arc.ox.ac.uk

# User Password management

Passwords can be changed on cluster login nodes.

To change your password run the **`passwd`** command:

```
$ passwd
Enter login(LDAP) password:
Enter new password:
Re-enter new password:
```

If your password has expired email support@arc.ox.ac.uk

Forwarding email addresses can be changed, please ask the ARC team.

# 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/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