SUPERCOMPUTING WALES
UWCHGYFRIFIADURA CYMRU

Supercomputing Wales

HPC User Guide

Version 2.0

March 2020

Table of Contents

Glossary of terms used in this document

Acronym	Description
API	Application Programming Interface
ARCCA	Advanced Research Computing @ Cardiff
Bash	Bash (the Bourne Again SHell) is the standard GNU command line interpreter.
CFS	Cluster File System
Cluster Management Node	A node providing infrastructural management and administrative support to the cluster, e.g. resource management, job scheduling etc.
CMS	Cluster Management System
Compute Node	A node dedicated to batch computation
Core	One or more cores contained within a processor package
CPU	Central Processing Unit
CUDA	CUDA is a parallel computing platform and application programming interface (API) model created by Nvidia. It allows software developers and software engineers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing — an approach termed GPGPU (General-Purpose computing on Graphics Processing Units). The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels
CYGWIN/X	Cygwin/X is a port of the X Window System to the Cygwin API layer (http://cygwin.com/) for the Microsoft Windows family of operating systems. Cygwin provides a UNIX-like API, thereby minimizing the amount of porting required
DDR4	Double Data Rate Four - the latest generation of DDR (Double Data Rate) memory technology.
DIMM	Dual In-line Memory Module
Extreme Factory	Extreme Factory is an integrated grid or cloud framework from Atos for job execution on distributed and heterogeneous environments
FileZilla	FileZilla is a cross-platform graphical FTP, FTPS and SFTP client with many features, supporting Windows, Linux, Mac OS X and more http://download.cnet.com/FileZilla/3000-2160_4-10308966.html
Gb	Gigabit
GB	Gigabyte
Gbps	Gigabits Per Second
GCC	GNU Compiler Collection, as defined at http://gcc.gnu.org/
GFS	Global File System

Gluster	Red Hat Gluster Storage is a software-defined storage (SDS) platform. It is designed to handle general purpose workloads like backup and archival, as well as analytics. Unlike traditional storage systems, it can be deployed on bare metal, virtual, container, and cloud environments.
GP-GPU	General Purpose computing on Graphics Processing Unit
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HPC	High Performance Computing
HPC Wales	High Performance Computing Wales (HPC Wales) was the £40 million five-year project that preceded Supercomputing Wales. Operating between 2011 – 2015, it provided businesses and universities involved in commercially focussed research across Wales with access to advanced and evolving computing technology.
HPCC	HPC Challenge (Benchmark Suite)
HS	High Speed
HSM	Hierarchical Storage Management
HTC	High Throughput Computing (HTC). HTC systems process independent, sequential jobs that can be individually scheduled on many different computing resources across multiple administrative boundaries
IMB	Intel® MPI Benchmarks, Version 3.2.3
IPMI	Intelligent Platform Management Interface
kW	Kilowatt = 1000 Watts
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
Linux	Any variant of the Unix-type operating system originally created by Linus Torvalds
Login Node	A node providing user access services for the cluster
LSF	Platform Load Sharing Facility (or simply LSF) is a workload management platform, job scheduler, for distributed high performance computing.
Lustre	A software distributed file system, generally used for large scale cluster computing
Memhog	A command that may be used to check memory usage in Linux
Modules	Modules are predefined environmental settings which can be applied and removed dynamically. They are usually used to manage different versions of applications, by modifying shell variables such as PATH and MANPATH
MPI	Message Passing Interface - a protocol which allows many computers to work together on a single, parallel calculation, exchanging data via a network, and is widely used in parallel computing in HPC clusters
MPI-IO	MPI-IO provides a portable, parallel I/O interface to parallel MPI programs

Multi-core CPU	A processor with 8, 12, 16 or more cores per socket
MySCW	The Supercomputing Wales account management interface
NFS	Network File System
NIC	Network Interface Controller
Node	An individual computer unit of the system comprising a chassis, motherboard, processors and all additional components
Nvidia	Nvidia Corporation, more commonly referred to as Nvidia, is a Taiwanese technology company incorporated in Delaware and based in Santa Clara, California. It designs graphics processing units for the gaming and professional markets, as well as system on a chip units for the mobile computing and automotive market.
OpenMP	Open Multi Processing is an Application Program Interface (API) that may be used to explicitly direct *multi-threaded, shared memory parallelism* https://computing.llnl.gov/tutorials/openMP/
OS	Operating System
Pascal P100	Pascal is the codename for a GPU microarchitecture developed by Nvidia, as the successor to the Maxwell architecture. The architecture was first introduced in April 2016 with the release of the Tesla P100 (GP100) on April 5, 2016, and is primarily used in the GeForce 10 series
PCI	Peripheral Component Interconnect
PCM	Platform Cluster Manager www.platform.com
PCI-X	Peripheral Component Interconnect Extended
Portal	A web portal or links page is a web site that functions as a point of access to information in the Web
Processor	A single IC chip package (which may be single-core, dual-core, quad-core, hexa-core, octa-core, .. etc.)
PuTTY	An SSH and telnet client for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers
QDR	Quad Data Rate (QDR) Infiniband (IB) delivers 40 Gbps per port (4x10 Gbps per lane)
RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
RSS	RSS (originally RDF Site Summary is a family of web feed formats used to publish frequently updated works
SAN	Storage Area Network
Sandy Bridge, SNB	Sandy Bridge is the codename for the microarchitecture used in the "second generation" of the Intel Core processors (Core i7, i5, i3), with this microarchitecture the successor to Nehalem microarchitecture
SAS	Serial Attached SCSI

SATA	Serial Advanced Technology Attachment
SCP	Secure copy
Scientific Gateway	Science Gateways enable communities of users sharing a scientific goal to use grid resources through a common interface
SGE	Sun Grid Engine Software
Slurm	Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. https://slurm.schedmd.com/overview.html
SCSI	Small Computer System Interface
SIMD	Single Instruction, Multiple Data
SMP	Symmetric Multiprocessor
SSH	Secure Shell, a remote login program
Sub-System	One of the distributed HPC Clusters comprising the Supercomputing Wales computing Infrastructure
TB	Terabyte
TCP	Transmission Control Protocol
TFlops	TeraFlops = 10^{12} FLOPS (FLoating point Operations Per Second)
UNIX	An operating system conforming to the Single UNIX® Specification as defined by the Open Group at http://www.unix.org/, and will also embrace those operating systems which can be described as Unix-like or Unix-type (or equivalent)
UPS	Uninterruptible Power Supply
VcXsrv	Windows X-server based on the xorg git sources (like xming or cygwin's xwin), but compiled with Visual C++ 2012 Express Edition. https://sourceforge.net/projects/vcxsrv/
Volta or V100	Volta is the codename for a GPU microarchitecture developed by Nvidia, succeeding Pascal. It was NVIDIA's first chip to feature Tensor cores, specially designed cores that have superior deep learning performance over regular CUDA cores. The first graphics card to use it was the datacentre Tesla V100, e.g. as part of the Nvidia DGX-1 system
Westmere	Westmere (formerly Nehalem-C) is the code name given to the 32 nm die shrink of the Intel Nehalem micoarchitecture.
WinSCP	WinSCP is a SFTP client and FTP client for Windows. Its main function is the secure file transfer between a local and a remote computer http://winscp.net/eng/index.php
x86-64	A 64-bit microprocessor architecture
XMING	Xming is an implementation of the X Window System for Microsoft Windows operating systems, including Windows XP, Windows Server 2003, Windows Vista etc.

Xquartz	The XQuartz project is an open-source effort to develop a version of the X.Org X Window System that runs on OS X. Together with supporting libraries and applications, it forms the X11.app that Apple shipped with OS X versions 10.5 through 10.7 (https://www.xquartz.org/)

1 An Introduction to the User Guide

The Supercomputing Wales (SCW) High Performance Computing Service provides a dual Hub parallel computing facility in support of research activity within the Welsh academic user community.

The service comprises two HPC clusters, “Hawk”, housed in the Redwood data centre at Cardiff University, and “Sunbird”, recently moved from the Dylan Thomas Data Centre to the Bay Data Centre in Swansea. Both clusters run the Red Hat Linux operating system. Usage of both systems is described in this User Guide that comprises material from the Supercomputing Wales Portal (https://portal.supercomputing.wales). This portal describes the resources and support for users of the Supercomputing Wales services, and has acted as the main guide following the installation of both Atos clusters in the summer and autumn of 2018.

Note at this stage that Hawk is available for Cardiff and Bangor users of Supercomputing Wales, while Aberystwyth and Swansea users should use the Swansea Sunbird system instead. Guest users from other institutions are allowed access on the basis of the project & institutions they are working with.

The Guide is structured as follows. Following this introduction, section 2 describes the available Hub-based Linux platforms, with a summary of the HPC configurations at both Cardiff and Swansea, along with an outline of the different classes of file system available. Suffice it to say that parts of the final solution e.g., the High Performance Data Analytics platform at Cardiff, remain under development, while other components – Extreme Factory in particular – have never materialised given the repeated failure by Atos to deliver the production quality environment. The present version of the Guide is intended primarily for experienced users who need to understand how the system works and wish to access it using secure shell (SSH) and the ssh command.

Section 3 describes the steps in gaining access to the Supercomputing Wales systems, with details of how to request an account to use the System and how to either create a new project account or request access to an existing project. The Terms and Conditions regarding access are also summarised at this point. This section continues by describing how to gain access to Hawk and Sunbird, from either a Linux or Windows environment. In outlining how to log into both systems, consideration is given to password specification together with the available file transfer mechanisms.

The various aspects of how to use these systems are given in sections **4** through **7**. Section **4.1** introduces features of the user environment, with a detailed description of the use of environment modules and the associated module commands. Sections **4.2** and **4.3** provide the necessary background to be able to run jobs on the clusters, under control of the Slurm Job Scheduler, describing how to submit, monitor and control just how jobs are run on the system, how to specify a project that is running the job, along with how to terminate a job. Consideration is given in section **4.3** to specifying the Job Runtime Environment in Slurm along with the associated #SBATCH directives and environment variables, plus the variety of system queues and available partitions. A variety of example run scripts are presented that cover many of the likely run time requirements of the Supercomputing Wales user community. Using the Nvidia GPUs and the AMD EPYC Rome systems are described in sections **4.4** and **4.5** respectively. Section **4.6** describes the two types of available storage offered by Super Computing Wales – home directories and scratch. The mechanism for checking storage quotas is outlined, together with the possible use of shared storage areas or multiple users to access. The suggested Code of conduct and Best Practice, outlined in section **4.7**, completes this section.

Section 5 considers more advanced topics in using the facilities, covering the common categories of job. Section 5.1 provides examples covering batch submission of parallel jobs using MPI, OpenMP and hybrid MPI/OpenMP jobs. Section **5.2** features the batch submission of serial jobs, while section **5.3**

addresses Job Arrays. Section **5.4** considers the batch submission of multiple serial jobs scheduled for parallel execution, with an alternative approach using Job Arrays for very similar tasks considered in this section. Using the HPC systems interactively via Slurm is described in section **5.5**, while section **5.6** turns to a consideration of X11 GUI forwarding and how to capitalise on the capability of some applications to interact with a graphical user interface (GUI). Although not typical of parallel jobs, many large-memory applications and computationally steered applications can offer such capability.

Section **6** presents an overview of example software jobs and the available application guides. Also included here are references to a variety of "How-To Guides" and Technical Notes.

Much of the detail required in developing, testing and executing end-user applications then follows in section **7**. Available compilers and MPI libraries are outlined, together with the techniques for debugging and analysing parallel software, with a focus on the DDT debugger from Allinea ARM and the analysis software, Performance Report.

Earlier versions of this guide had focused on the two classes of established HPC users who dominated early access to the systems through the provision of a "Rapid Start" introduction. This was directed at both those users of the ARCCA Raven services and the existing HPC Wales User community who had been using the Sandy Bridge and Westmere systems from Fujitsu during the so-called "Interim period" following the formal end of the HPC Wales Project in July 2015. This material is now included in section **8** of this Guide. With Hawk acting as a replacement system for Cardiff's aging supercomputer, Raven, section **8.2** provides a Quick Reference guide in converting job submission scripts from Raven that used the PBS Pro scheduler to use the Slurm scheduler deployed on Hawk, including the commands, job specification and job environment variables. Summary Reference guides for converting scripts that previously used the LSF (section **8.1**) and SGE (section **8.3**) schedulers are also provided.

Following a number of links to useful information in section **9**, section **10** provides details of accessing the User Support facilities, along with a number of Frequently Asked Questions (FAQs).

In addition to the glossary, a number of Appendices are included, with a listing of (i) the most used Intel compiler Flags (Appendix I), and (ii) the most common linux commands (Appendix II).

2 About SCW Systems

Supercomputing Wales operate two Supercomputers and a Data Analytics platform:

2.1 Cardiff HPC System - About Hawk

Portal URL: https://portal.supercomputing.wales/index.php/about-hawk/

Hawk is located in the Redwood Datacentre, Cardiff University and is available for Cardiff and Bangor users of Supercomputing Wales. Aberystwyth and Swansea users should use the Swansea Sunbird system instead.

Guest users from other institutions will be allowed access on the basis of the project & institutions they are working with.

2.1.1 System Specification:

- 280 nodes, totalling 12,736 cores, 68.224 TBytes total memory
  - CPU:
    - ❖ 2 × Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz with 20 cores each
    - ❖ 2 × AMD(R) Epyc 7502 2.5GHz with 32 cores each
  - RAM:
    - ❖ 192 GB on Intel nodes
    - ❖ 256 GB on AMD nodes
    - ❖ 384GB on high memory and GPU nodes
  - GPU:
    - ❖ 26 × nVidia P100 GPUs with 16GB of RAM on 13 nodes
    - ❖ 30 × nVidia V100 GPUs with 16GB of RAM on 15 nodes
- Storage:
    - ❖ 1192TB (usable) scratch space on a Lustre filesystem
    - ❖ 420TB of home directory space over NFS

2.1.2 Partitions:

The available compute hardware is managed by the Slurm job scheduler and organised into 'partitions' of similar type/purpose. Jobs – both batch and interactive – should be targeted at the appropriate one. Limits on the partition are just defaults, we can provide access to different limits, for example to enable array jobs a higher number of submitted jobs may be required. Please contact support if required.

Partition Name	Number of Nodes	Purpose	Maximum running jobs per user	Max. submitted jobs per user
compute	134	Parallel and MPI jobs	15	30

compute_amd	64	Parallel and MPI jobs using AMD EPYC 7502	10	30
highmem	26	Large memory (384GB) jobs	10	20
gpu	13	GPU and Cuda jobs – P100	5	10
gpu_v100	15	GPU (CUDA) jobs - V100	5	10
HTC	26	High Throughput Serial jobs	10	40
dev	2	Testing and development	1	2

2.2 Swansea HPC System - About Sunbird

Portal URL: https://portal.supercomputing.wales/index.php/about-sunbird/

Sunbird is located in Swansea University. It is available for Swansea and Aberystwyth users of Super Computing Wales. Bangor and Cardiff users should use the Cardiff Hawk system.

Guest users from other institutions will be allowed access on the basis of the project & institutions they are working with.

2.2.1 System Specification:

- 126 nodes, totalling 5,040 cores, 48.384 TBytes total memory
    - ❖ CPU: 2 × Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz with 20 cores each
    - ❖ RAM: 384GB per node
    - ❖ GPU: 8 × Nvidia V100 GPUs
- Storage:
    - ❖ 808TB (usable) scratch space on a Lustre filesystem
    - ❖ 231TB of home directory space on a Lustre filesystem

2.2.2 Partitions:

The available compute hardware is managed by the Slurm job scheduler and organised into 'partitions' of similar type/purpose. Jobs – both batch and interactive – should be targeted at the appropriate one. Limits on the partition are just defaults, we can provide access to different limits, for example to enable array jobs a higher number of submitted jobs may be required. Please contact support if required.

Partition Name	Number of Nodes	Purpose	Maximum running jobs per user	Maximum submitted jobs per user

compute	122	General use	15	30
GPU	13	GPU / Cuda jobs only	5	10

2.3 Cardiff Data Analytics Platform - About Sparrow (Coming soon)

3 Registration & Access

Portal URL: https://portal.supercomputing.wales/index.php/getting-access/

SCW partner institution users can register to access SCW systems using their local university identity.

3.1 Gaining Access for Users & Projects

3.1.1 Applying for a user account

New Users	Migrating Users (from HPCW and Raven)
The Supercomputing Wales service is available to members of the partner institutions (Aberystwyth, Bangor, Cardiff & Swansea) and their project collaborators. In order to gain access, users should login to the MySCW account interface using their institutional user credentials. On the follow-up details screen, please include a brief statement of your reason for requesting an account on the system (e.g. 'to apply for a new project for abc', 'to join project xyz of PI anon'). Your request will then be seen within our support hours and handled appropriately. Email notifications will be sent as your request progresses. Once your account has been approved, access MySCW again. Your SCW username is now displayed at the top of the **MySCW** page in the **Account Summary** section. Click **Reset SCW Password** and enter a new password for SCW systems. These will be your new SCW systems username & password and are completely independent of all other University authentication systems. You will need to be a member of a new or existing project to login and run jobs, please see below.	Login to MySCW using your institutional account. Migrating users have been pre-added and pre-authorised on the new system. Confirm that all details displayed under **Account Summary** are correct and please notify support if not. Your SCW username is displayed at the top of the **MySCW** page in the **Account Summary** section. Click on **Reset SCW Password** and enter a new password for SCW systems. These will be your new SCW systems username & password and are completely independent of all other University authentication systems. Existing project memberships have also been migrated so no further action should be needed, please login and use the system!

3.1.2 Accessing the systems

Follow the instructions on the Access the System page to login.

3.1.3 Applying for a new project

If you are applying, as the *Technical Lead*, for a new project to have access to the SCW systems, please click the **Create Project Application** item from the **Actions** menu in My SCW. You will then be asked for some details about your project. These will be reviewed by SCW staff and this process may take a few days

3.1.4 Joining an existing project

Without a project membership you will not be able to run a job. If you wish to join an existing project ask the *Technical Lead* of the project to give you the project code. Project codes follow the format *scwXXXX* where XXXX are all numbers. All legacy HPC Wales and Raven projects have been allocated SCW codes, please use these where possible and only revert to HPCW or Raven codes where necessary. The *Technical Lead* can find their project codes on the **Project Memberships** page where they will be listed as **Project Owner**. Users can apply to join via the **Join a Project** item on the **Dashboard**. *Technical Leads* will be notified and then need to approve the request

3.2 Terms & Conditions

Portal URL: https://portal.supercomputing.wales/index.php/terms-conditions/

Access to the Supercomputing Wales systems is subject to the following Terms & Conditions. If you do not accept them, you must not use the systems.

Usage must be in compliance with your host institution's Acceptable Use Policy and the JANET Acceptable Use Policy.

- The JANET document
- Aberystwyth University's policy
- Bangor University's policy
- Cardiff University's policy
- Swansea University's policy

All data stored on the Supercomputing Wales systems is at the owner's own risk.

Unless explicitly advised, backups will not be taken of data by Supercomputing Wales. You are advised to maintain your own backups of important or not easily recreated data.

Supercomputing Wales accepts no liability for any loss of data.

Any user found breaching acceptable use will have their accounts immediately suspended and will be reported to their host institution and be subject to the appropriate disciplinary action.

Any software used on the cluster is under the correct licensing conditions. If a user presents a license for software to be installed on the Supercomputing Wales systems, they are confirming that they have the original software suppliers consent for it to be installed on a 3rd party system.

Unauthorised access to commercial software will result in access being immediately suspended pending further investigation.

3.3 Accessing the Systems from the Internet

Portal URL: https://portal.supercomputing.wales/index.php/accessing-the-system/

Once you have been given an account, and have the credentials (username & password) from My SCW, you can login.

3.3.1 Let's Go!

To access Supercomputing Wales (SCW) systems via the command line, you must use a terminal emulator, which connects your local keyboard and screen to the remote system. This uses a protocol called "Secure Shell", or "SSH"; this is available as standard with a Linux workstation or Mac, but requires a program to be downloaded and installed with Microsoft Windows.

The SCW infrastructure consists of two compute clusters, one based in Cardiff and the other based in Swansea. Each of these clusters has its own 'Login Nodes', which enable you to execute and manage jobs on that particular cluster. Please see details linked from the front page of the User Portal.

3.3.2 SSH Address Details

The details for the two SCW systems are:

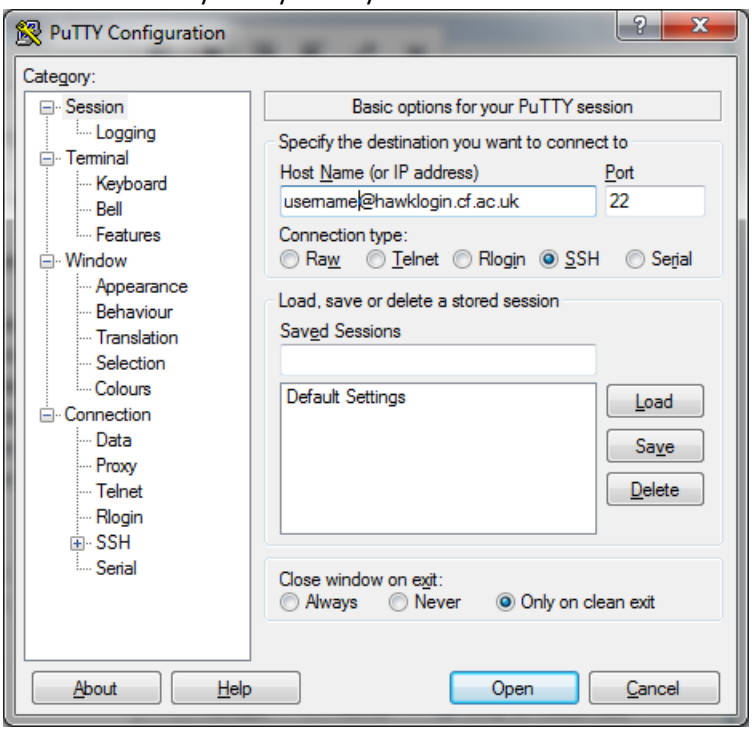
- **Cardiff** – hawklogin.cf.ac.uk
- **Swansea** – sunbird.swansea.ac.uk.

Aberystwyth users should login to the Swansea system.

Bangor users should login to the Cardiff system.

3.3.3 Logging in from Windows

If you are logging in from a Windows system you may need to install an SSH client. The PuTTY software



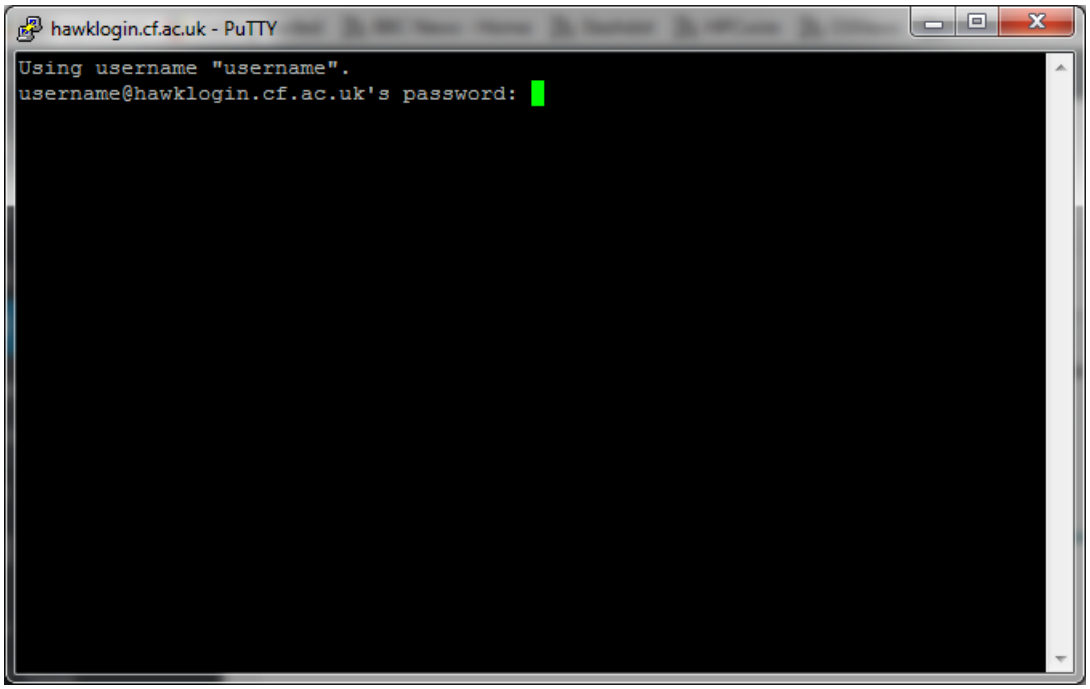
is recommended and can be downloaded from the PuTTY website.

3.3.4 Logging in to the Cardiff System

In order to log in to the SCW Cardiff cluster, enter "<username>@hawklogin.cf.ac.uk" into the "Host Name (or IP address)" box; where "<username>" is your SCW username given to you at registration. You may wish to save this information for reuse, in which case also enter a name in the "Saved Sessions" box, and hit the "Save" button.

After entering the address of the machine you wish to log into ("hawklogin.cf.ac.uk"), click on the "Open" button to connect to the SCW cluster.

The first time you connect you will be asked to accept a security certificate. Accept the certificate, and you will be linked to an SCW login node. A screen will pop up akin to this:



You are now set to log in – now enter your password and you should now have a command prompt.

Please skip the section "Logging in from Linux" and proceed with section "Entering your password".

3.3.5 Logging in to the Swansea System (Sunbird)

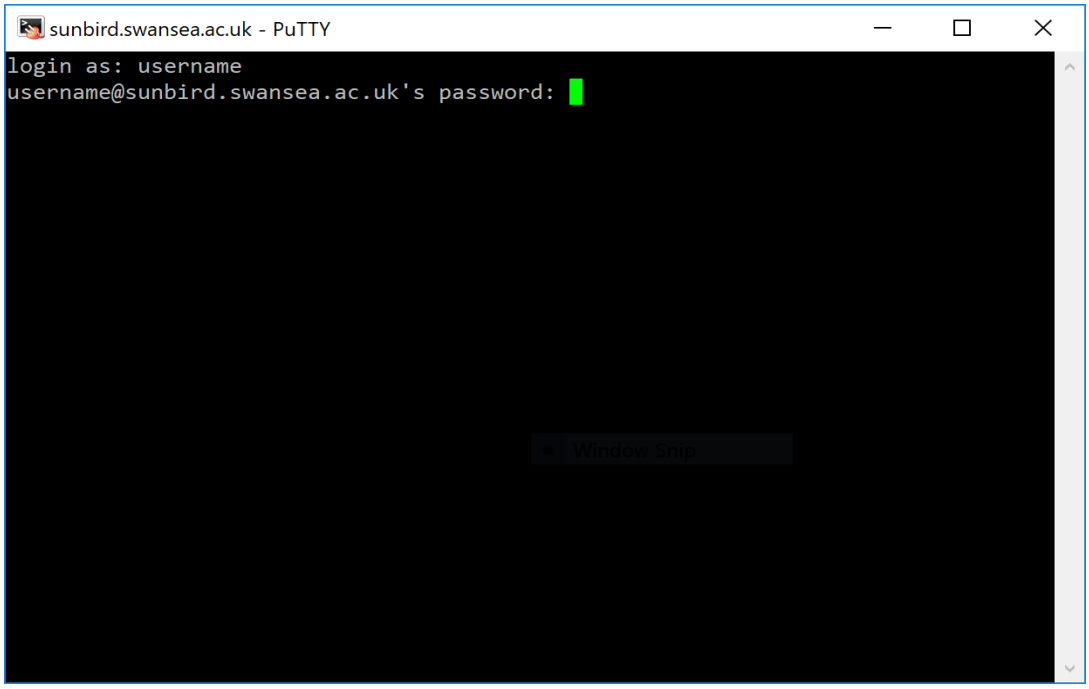
In order to log in to SCW Swansea cluster, enter "<username>@sunbird.swansea.ac.uk" into the "Host Name (or IP address)" box; where "<username>" is your SCW username given to you at registration. You may wish to save this information for reuse, in which case also enter a name in the "Saved Sessions" box, and hit the "Save" button.

After entering the address of the machine you wish to log into ("sunbird.swansea.ac.uk"), click on the "Open" button to connect to the SCW cluster.

The first time you connect you will be asked to accept a security certificate. Accept the certificate, and you will be linked to an SCW login node. A screen will pop up akin to this:

You are now set to log in – now enter your password and you should now have a command prompt.

Please skip the next section “Logging in from Linux” and proceed with section “Entering your password”.



3.3.6 Logging in from Linux/Mac

Linux has SSH (“Secure Shell”) built – in, so you should not need to install ssh manually. To log in to SCW Cardiff cluster, simply start a terminal and enter:

Hawk:

```
ssh username@hawklogin.cf.ac.uk
```

Sunbird:

```
ssh username@sunbirdlogin.swansea.ac.uk
```

Where you should replace “username” with your username. On first logging on from a machine, you will be asked to confirm the secure certificate associated with SCW cluster login node; please confirm “yes”, such as:

```
[~]$ ssh John.Doe@hawklogin.cf.ac.uk
The authenticity of host 'hawklogin.cf.ac.uk (131.251.129.4)' can't be established.
ECDSA key fingerprint is SHA256:P8MxFCLE7+ROcYqIdFRZSZ1WI7CKGIWsJ96o5vjZluo.
ECDSA key fingerprint is MD5:31:fc:cb:35:fa:7c:90:37:ef:4c:f7:3d:55:3c:01:3e.
Are you sure you want to continue connecting (yes/no)?
```

You will then proceed to be asked for your password, and can continue with the next section on logging in (which is common to both Linux and Windows).

3.3.7 Entering your Password

Your password is the one you chose when signing up for a Super Computing Wales account. If you forgot your password and need to reset, login to My SCW with your university credentials and reset it. External users will need to contact the helpdesk.

Once you are logged in successfully, you will see text displayed similar to the following:

```
================================ Supercomputing Wales - Hawk ========================
  This system is for authorised users, if you do not have authorised access
      please disconnect immediately, and contact Technical Support.
 ---------------------------------------------------------------------------
          For user guides, documentation and technical support:
                 Web: http://portal.supercomputing.wales
 ------------------------- Message of the Day -----------------------------
        **** HAWK CARDIFF UNIVERSITY USERS UPDATE 16/01/2020 ****
        Cardiff University Hawk users: please take note of email
      sent to community 14.01.2020 titled "Hawk backup opt-in request"
```

You are now at the command prompt of a SCW cluster login node .

Tip: Note that with the shell command line, you can use the left and right cursor keys (arrow keys) to move the cursor around the current line of text at the command prompt and edit it – to remove any mistakes or change your mind. Similarly, pressing the up or down arrow will bring back the previous command you typed. Pressing the up arrow repeatedly will show older and older commands – whereas the down arrow will bring you back to the last command you typed, or press it again – back to a blank line ready for you to start a new command.

For next steps, please see the following sections of this user guide and/or other pages on the SCW User Portal.

4 Using the Systems

SCW supercomputers are linux-powered and use the Slurm batch scheduler to manage the allocation of computational resources. A full application software stack is provided and supported. For software specific documentation, please see the *Software Training & Guides* section below.

4.1 Command Line Interface & Software Access

Portal URL: https://portal.supercomputing.wales/index.php/command-line-environment/

The command line environment is provided through a shell; the default shell in the Supercomputing Wales (SCW) environment is bash. A shell is the textual interface in which the user issues commands through their keyboard which are evaluated and produce a response. When using the SCW command line remotely, the shell is running on the SCW system and the user's machine is providing the network communications to it.

What is Bash?

Bash (the Bourne Again SHell) is the standard GNU command line interpreter. It combines features from prior shells for both interactive and programmatic use. Bash is available for many operating systems including UNIX variants and Microsoft Windows (via 3rd party packages such as Cygwin), and is the standard shell in most Linux distributions. Many guides are freely available on using and customising bash, and are easily discoverable via a web search engine.

4.1.1 Modules

A consistent modular software environment is available on all SCW clusters. This allows many different software packages and environments to co-exist. This does require you to choose the software package and the version that you wish to use. Environment modules provide an easy way for your shell's environment (PATH, MANPATH, INCLUDE, LD_LIBRARY_PATH, etc.) to be customised on the fly.

The module command is generally self-explanatory. The most common commands are:

- *module avail*
  - This command will list the modules currently available on the system
- *module list*
  - Will list the modules loaded in your session
- *module load*
  - Load a new module into your current session
- *module unload*
  - Remove a module from your current session
- *module purge*
  - Unload all modules
- *module show*
  - Show the effective contents of a particular module

4.1.2 Can I Pre-Load Modules in .myenv?

The simple answer is yes; however, this should be done with caution. If you are using multiple versions of the same software, it is good practise to manually load the modules or add them to your submission scripts. Adding module calls to *.myenv* can lead to odd behaviour with your compilations if you forget they are loaded.

It is recommended that you keep modifications to *.myenv* to a minimum. You should attempt to make modifications in your job submission scripts manually when you login or through a shell script. If you would like further information on this (or how to create a shell script to load modules) please contact the Support Desk.

4.1.3 Customising Bash

On most Linux Systems customising the Bash shell can be achieved through editing of the *.bashrc* file. However, on SCW systems this file may be overwritten by the system. If you wish to make a permanent change to your shell please use *.myenv* (which is located in your home directory). The same syntax and commands can be utilised when customising *.myenv* as would be used in *.bashrc*.

4.1.4 Can I use a Different Shell?

There are many shells available under Linux such as:

- Bourne shell
- Korn shell

While there is no stipulation that you have to use the Bash shell, Bash is the only shell fully supported by SCW. Software Applications and commands in all documentation, user guides and training are carried out and tested using the Bash shell. If you choose to use a different shell you do so at your own risk.

While SCW will do its best to assist you with any technical issues you experience, if you use a different Shell you may be asked to switch back to the Bash shell and replicate the customisations.

4.1.5 Changing your Password

You will be required to change your password on your first login to a SCW system and at regular intervals when prompted. However, should you wish to change your password before the system requests you to, you should issue the following command:

```
$ passwd
```

You will be asked to type your existing password and your new password twice. Your new password will need to contain at least one capital letter and a number, and should be a minimum length of eight characters. Changes to your password are instantly synchronised to all SCW systems.

4.2 Running Jobs with Slurm

4.2.1 Submitting a Job

Running jobs on SCW clusters using Slurm is a simple process. You need a *job script*, which defines the resource requirements for the job and then is a batch script of the commands to run. The resource requirements & other job parameters are specified on lines that start with a *#SBATCH* instruction, thus they are treated as comments by anything other than the Slurm scheduler. The batch of commands to run is just as if you were typing them at the command line.

For example, a simple script might look like:

```
#!/bin/bash --login
###
#job name
#SBATCH --job-name=imb_bench
#job stdout file
#SBATCH --output=bench.out.%J
#job stderr file
#SBATCH --error=bench.err.%J
#maximum job time in D-HH:MM
#SBATCH --time=0-00:20
#number of parallel processes (tasks) you are requesting - maps to MPI processes
#SBATCH --ntasks=80
#memory per process in MB
#SBATCH --mem-per-cpu=4000
#tasks to run per node (change for hybrid OpenMP/MPI)
#SBATCH --ntasks-per-node=40
###


#now run normal batch commands
module load compiler/intel mpi/intel


#run Intel MPI Benchmarks with mpirun - will automatically pick up Slurm parallel
environment
mpirun $MPI_HOME/intel64/bin/IMB-MPI1
```

The directives to Slurm are quite clear and self-descriptive. Of particular note is the memory specification – Slurm is very good at scheduling around and subsequently controlling job memory usage. Too low a memory request can result in a job crashing or being cancelled, but too high a value can result in a job waiting for longer than necessary.

Once this is saved in a file, say called bench.sh, running the job is as simple as:

```
sbatch bench.sh
```

Slurm will return a job number, which can be used to track, account & cancel the job.

4.2.2 Monitoring Jobs

To see your current submitted & running jobs, we use the command *squeue.*

For example:

```
[test.user@cl1 imb]$ squeue
                   JOBID PARTITION     NAME      USER ST TIME NODES NODELIST(REASON)
                   109     compute imb_benc test.user  R  0:49    2 ccs[0121-0122]
                   110     compute imb_benc test.user  R  3:29    8 cst[001-008]
                   113     compute imb_benc test.use  PD  0:00    8 (Resources)
```

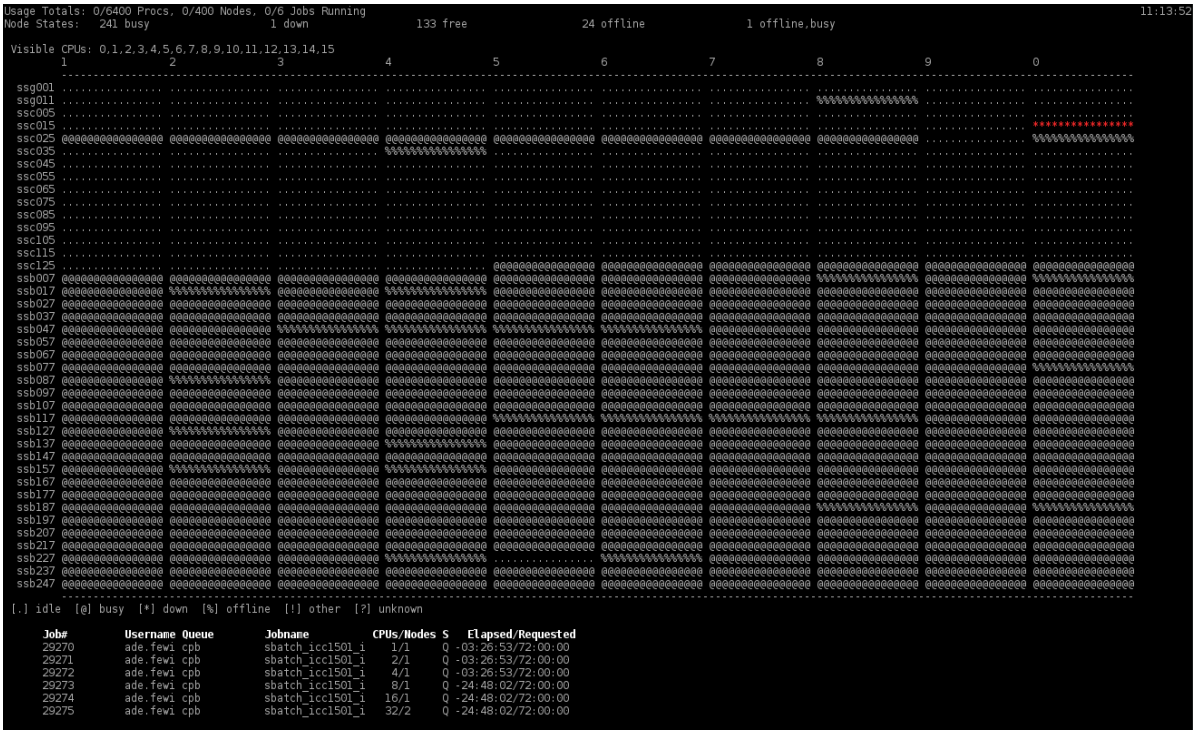
In this case, there are three jobs present, two are running (109 and 110) and one is queued/pending (113) awaiting resources.

When will my job run?

SCW systems deploy a Fair Share scheduling policy to share resources in a fair manner between different users & groups. With the previous LSF based systems it was possible to view all queued and running jobs from all users and thus get a 'feel' as to how busy the systems were and when a job might run. Such visibility of total system state is no longer the case with Slurm as job data is private – hence one cannot get such a view of total system state.

However, Slurm is very good at planning the future state of the system by assessing user fair shares, job time limits and node availability – and so it can predict job start times with a good degree of accuracy. There are a number of ways to view this information, but we have installed an additional tool called 'slurmtop', which will present a view of your currently queued and running jobs, including a countdown to start for queued jobs.

Example 'slurmtop' output:



We see in the 'slurmtop' output:

- the first line shows the current status of our usage of the cluster.
- the second line shows the overall status of the compute nodes of the cluster.
- the grid shows – with one character per processor – the compute nodes of the cluster, and will locate our jobs when they are running.
- the job list shows our queued and running jobs, plus any that have completed within the last five minutes. Note that 'Elapsed' time column showing a negative 'count-down' to start-time for queued jobs.

It's worth noting that the prediction of future runs is somewhat fluid due to the use of fair share and because jobs typically do not run to their maximum configured duration, however this is a good indication of system and job status.

Viewing Running Jobs

To get more detailed information on a running job, one can use *sstat <jobid>.*

By default this gives a verbose set of data. A more succinct output targeting memory usage can be obtained using some simple output formatting arguments:

```
sstat --format JobID,NTasks,nodelist,MaxRSS,MaxVMSize,AveRSS,AveVMSize &lt;jobid&gt;
```

Example output:

```
[test.user@cstl001 imb]$ sstat --format
JobID,NTasks,nodelist,MaxRSS,MaxVMSize,AveRSS,AveVMSize 113
       JobID   NTasks             Nodelist     MaxRSS  MaxVMSize     AveRSS  AveVMSize
------------ -------- -------------------- ---------- ---------- ---------- ----------
113.0               8         cst[001-008]    464196K    982928K    300810K    851119K
```

Many different formatting options can be specified, see the man page for details.

Slurm writes standard error and standard out files in fairly real time. Thus, you can see job progress by looking at the job script specified *stdout* and *stderr* files at runtime.

4.2.3 Killing a Job

If you have a job running that you wish to cancel for some reason, it is very easy to terminate using the job id that is returned at submission and can be seen in *squeue* output. Slurm is particularly robust at removing running jobs.

```
[test.user@cstl001 imb]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               122   compute imb_benc test.use PD       0:00      8 ccs[001-008]
               120   compute imb_benc test.use  R       0:17      8 ccs[001-008]
               121   compute imb_benc test.use  R       0:17      8 ccs[001-008]
[test.user@cstl001 imb]$ scancel 122
[test.user@cstl001 imb]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               120   compute imb_benc test.use  R       0:17      8 ccs[001-008]
               121   compute imb_benc test.use  R       0:17      8 ccs[001-008]
```

If you wish to cancel all your running and queued jobs, then use:

```
scancel -u username
```

4.2.4 Completed Jobs

Once a job has completed – it is no longer visible in the output from *squeue* and the output files are completed – we can use a different command to get job statistics:

```
[test.user@cstl001 imb]$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
104           imb_bench    compute    scw1000         32  COMPLETED      0:0
104.batch         batch               scw1000         32  COMPLETED      0:0
```

```
104.0          pmi_proxy               scw1000          2  COMPLETED      0:0
105            imb_bench    compute    scw1000         32    TIMEOUT      1:0
105.batch          batch               scw1000         32  CANCELLED     0:15
105.0          pmi_proxy               scw1000          8 CANCELLED+      0:9
106            imb_bench    compute    scw1000         32 CANCELLED+      0:0
106.batch          batch               scw1000         32  CANCELLED     0:15
106.0          pmi_proxy               scw1000          8  COMPLETED      0:0
```

In this case, we see three separate complete jobs. Job 104 completed successfully. Job 105 ran over its time limit. Job 106 was cancelled by the user.

We also see that one submitted job has resulted in three accounted task steps different parts executed by the job. If a single job were to call *mpirun* multiple times, for example in dividing a job allocation in two or running one parallel task after another, then we would see multiple parallel task steps. This is because MPI directly interacts with Slurm to take advantage of faster task launching.

We can also format the output of *sacct* in a very similar way to *sstat:*

```
[test.user@cstl001 imb]$ sacct --format JobID,jobname,NTasks,AllocCPUS,CPUTime,Start,End

       JobID    JobName   NTasks  AllocCPUS    CPUTime               Start                 End
------------ ---------- -------- ---------- ---------- ------------------- -------------------
104            imb_bench                  32   02:18:40 2015-07-21T11:03:14 2015-07-21T11:07:34
104.batch          batch        1         32   02:18:40 2015-07-21T11:03:14 2015-07-21T11:07:34
104.0          pmi_proxy        2          2   00:08:40 2015-07-21T11:03:14 2015-07-21T11:07:34
105            imb_bench                  32   10:51:12 2015-07-21T11:15:44 2015-07-21T11:36:05
105.batch          batch        1         32   10:51:12 2015-07-21T11:15:44 2015-07-21T11:36:05
105.0          pmi_proxy        8          8   01:00:00 2015-07-21T11:15:45 2015-07-21T11:23:15
106            imb_bench                  32   00:38:24 2015-07-21T11:40:53 2015-07-21T11:42:05
106.batch          batch        1         32   00:39:28 2015-07-21T11:40:53 2015-07-21T11:42:07
106.0          pmi_proxy        8          8   00:09:52 2015-07-21T11:40:54 2015-07-21T11:42:08
```

Again, the man pages for the Slurm commands should be referenced for a full set of possible output fields.

4.2.5 Specifying which project is running the job

If you are a member of multiple projects use the -A option to sbatch to choose which project is running the job. This will help ensure that accounting statistics are correct for each project.

If you are only in one project then you don't have to do this.

```
sbatch -A scw1000 bench.sh
```

You can find a list of your project codes on the "Project Memberships" page on MySCW.

4.2.6 Example Jobs

Please see here for further information on the training tarball that provides a wide variety of example jobs.

4.3 Slurm Job Directives, Variables & Partitions

Portal URL: https://portal.supercomputing.wales/index.php/index/slurm/migrating-jobs/

4.3.1 Job Runtime Environment in Slurm

When a job is submitted, Slurm will store all environment variables in-place at submission time and replicate that environment on the first allocated node where the batch script actually runs.

Additionally, at run time, Slurm will set a number of shell environment variables that relate to the job itself and can be used in the job run. The Slurm documentation's manpage on *sbatch* provides an exhaustive guide, but we highlight some useful ones here.

4.3.2 #SBATCH Directives

In line with most batch schedulers, Slurm uses directives in submission scripts to specify job requirements and parameters for a job – the #SBATCH directives. Thus for an MPI task we might typically have:

```
#SBATCH -p cpb
#SBATCH -o runout.%J
#SBATCH -e runerr.$J
#SBATCH --job-name=mpijob
#SBATCH -n 128
#SBATCH --tasks-per-node=16
#SBATCH --exclusive
#SBATCH -t 0-12:00
#SBATCH --mem-per-cpu=4000
```

Walking through these:

Slurm #SBATCH directive	Description
#SBATCH --partition=compute or #SBATCH -p compute	In Slurm, jobs are submitted to 'partitions'. Despite the naming difference, the concept is the same.
#SBATCH --output=runout.%J or #SBATCH -o runout.%J	File for STDOUT from the job run to be stored in. The '%J' to Slurm is replaced with the job number.
#SBATCH --error=runerr.%J or #SBATCH -e runerr.%J	File for STDERR from the job run to be stored in. The '%J' to Slurm is replaced with the job number.
#SBATCH --job-name=mpijob	Job name, useful for monitoring and setting up inter-job dependency.
#SBATCH --ntasks=128 or #SBATCH -n 128	Number of processors required for job.
#SBATCH --tasks-per-node=16	The number of processors (tasks) to run per node.
#SBATCH --exclusive	Exclusive job allocation - i.e. no other users on allocated nodes.

Slurm #SBATCH directive	Description
#SBATCH --time=0-12:00 or #SBATCH -t 0-12:00	Maximum runtime of job. Note that it is beneficial to specify this and not leave it at the maximum as it will improve the chances of the scheduler 'back-filling' the job and running it earlier.
#SBATCH --mem-per-cpu=4000	Memory requirements of job. Slurm's memory-based scheduling is more powerful than many schedulers.

4.3.3 Environment Variables

Once an allocation has been scheduled and a job script is started (on the first node of the allocation), Slurm sets a number of shell environment variables that can be used in the script at runtime. Below is a summary of some of the most useful:

Slurm	Description
$SLURM_JOBID	Job ID.
$SLURM_JOB_NODELIST	Nodes allocated to the job i.e. with at least once task on.
$SLURM_ARRAY_TASK_ID	If an array job, then the task index.
$SLURM_JOB_NAME	Job name.
$SLURM_JOB_PARTITION	Partition that the job was submitted to.
$SLURM_JOB_NUM_NODES	Number of nodes allocated to this job.
$SLURM_NTASKS	Number of tasks (processes) allocated to this job.
$SLURM_NTASKS_PER_NODE (Only set if the --ntasks-per-node option is specified)	Number of tasks (processes) per node.
$SLURM_SUBMIT_DIR	Directory in which job was submitted.
$SLURM_SUBMIT_HOST	Host on which job was submitted.
$SLURM_PROC_ID	The process (task) ID within the job. This will start from zero and go up to $SLURM_NTASKS-1.

4.3.4 System Queues and Partitions in Slurm

Please use the *sinfo* command to see the names of the partitions to use in your job scripts. If not specified, the default partition will be used for job submissions. *sinfo -s* will give a more succinct partition list. Please see the Hawk (section 2.1.2) and Sunbird (section 2.2.2) sections for a list of partitions and their descriptions on each system.

4.4 Using GPUs

Portal URL: https://portal.supercomputing.wales/index.php/using-gpus/

4.4.1 GPU Usage

Slurm controls access to the GPUs on a node such that access is only granted when the resource is requested specifically. Slurm models GPUs as a Generic Resource (GRES), which is requested at job submission time via the following additional directive:

```
#SBATCH --gres=gpu:2
```

This directive requires Slurm to allocate two GPUs per allocated node, to not use nodes without GPUs and to grant access. SCW GPU nodes have two GPUs each. Jobs must also be submitted to the GPU-enabled nodes queue:

```
#SBATCH -p gpu
```

It is then possible to use CUDA enabled applications or the CUDA toolkit modules themselves, modular environment examples being:

```
module load cuda/cuda-9.1
```

```
module load gromacs/2018.2-single-gpu
```

4.4.2 CUDA Versions & Hardware Differences

Multiple versions of the CUDA libraries are installed on SCW systems, as can always be seen by:

```
[b.iss03c@cl1 ~]$ module avail CUDA

---- /apps/modules/libraries ----

CUDA/10.0 CUDA/10.1 CUDA/8.0 CUDA/9.0 CUDA/9.1 CUDA/9.2
```

The GPU nodes always run the latest nvidia driver to support the latest installed version of CUDA and also offer backwards-compatabilitity with prior versions.

However, **Hawk** contains *Pascal* generation nVidia Tesla cards which are supported in all installed versions of CUDA, but **Sunbird** contains *Volta* generation nVidia Tesla cards which are only supported in CUDA 9+. Codes that require CUDA 8, such as Amber 16, will not run on the *Volta* cards available on Sunbird.

4.4.3 GPU Compute Modes

nVidia GPU cards can be operated in a number of Compute Modes. In short the difference is whether multiple processes (and, theoretically, users) can access (share) a GPU or if a GPU is exclusively bound to a single process. It is typically application-specific whether one or the other mode is needed, so please pay particular attention to example job scripts. GPUs on SCW systems default to ‘shared’ mode.

Users are able to set the Compute Mode of GPUs allocated to their job through a pair of helper scripts that should be called in a job script in the following manner:

To set exclusive mode:

```
clush -w $SLURM_NODELIST "sudo /apps/slurm/gpuset_3_exclusive"
```

And to set shared mode (although this is the default at the start of any job):

```
clush -w $SLURM_NODELIST "sudo /apps/slurm/gpuset_0_shared"
```

To query the Compute Mode:

```
clush -w $SLURM_NODELIST "nvidia-smi -q|grep Compute"
```

In all cases above, sensible output will appear in the job output file.

Additionally, as Slurm models the GPUs as a consumable resource that must be requested in their own right (i.e. not implicitly with processor/node count), the default of the scheduler would be to *not* allocate the same GPU to multiple users or jobs – it would take some manual work to achieve this.

4.5 Using the AMD EPYC Systems

4.5.1 What are the AMD EPYC nodes?

As part of an extension to Hawk called Phase 2, 4096 compute cores have been added in the form of 64 nodes each containing two 32-core AMD EPYC (Rome) processor chips and 256GB memory (4 GB per core).

Comparing these processors to the Intel Xeon (Skylake) processors used in Hawk's Phase 1 is not trivial. The clock rate is a fraction higher (2.5GHz vs 2.4GHz) but both the chip architecture and processing pipeline are considerably different. A further significant difference is that Xeon Skylake supports the AVX512 instruction set, but EPYC Rome does not. Codes that make good use of AVX512 should run better on Xeon Skylake; codes that do not will generally see very similar performance on a core-to-core comparison basis, but benefit much more on a node-to-node comparison as Skylake offers 40 cores per node where Rome offers 64 cores per node.

4.5.2 Using the AMD EPYC nodes

Job submission to the AMD nodes is only permitted from a dedicated AMD login node called cla1.

To get to from outside, ssh to 'hawkloginamd.cf.ac.uk'. If you're already logged in to a Skylake login node – cl1 or cl2 – just ssh to cla1:

```
[c.sismfg@cl1 ~]$ ssh cla1
Last login: Wed Feb 19 01:50:17 2020 from cl2
======================== Supercomputing Wales - Hawk ==========================
   This system is for authorised users, if you do not have authorised access
       please disconnect immediately, and contact Technical Support.
 ------------------------------------------------------------------------------
           For user guides, documentation and technical support:
                  Web: http://portal.supercomputing.wales
--------------------------- Message Of The Day --------------------------
==============================================================================
[c.sismfg@cla1 ~]$
```

Jobs should then be submitted to the 'compute_amd' queue/partition and updated to use the 64 cores per node where appropriate. In a job script:

```
#!/bin/bash --login
#SBATCH --nodes=2
```

```
#SBATCH --ntasks=128
#SBATCH --threads-per-core=1
#SBATCH --ntasks-per-node=64
#SBATCH -p compute_amd
#SBATCH -A scwXYZ
# SBATCH -J MyJob
#SBATCH --time=2:00:00
#SBATCH --exclusive
```

4.5.3 Optimisation Options in the Compiler

If you are compiling your own code, we recommend doing this on the AMD login node itself. This enables easier use of the compiler optimisation flags in the same way as on Skylake, i.e:

With the Intel Compilers:

```
icc -xHOST -O3
```

This corresponds to 'icc -march=core-avx2 -O3' when run on the AMD login node

With the GNU Compilers:

```
gcc -march=native -O3
```

This correspond to 'gcc -march=znver2 -O3' when run on the AMD login node

4.5.4 Use of the Intel Math Kernel Library (MKL)

Many codes deployed on SCW systems make use of the Intel MKL to accelerate certain mathematical compute. MKL is able to be used on the AMD chips through the setting of two environment variables prior to task execution. One can either set them manually or load an environment module that will set them where appropriate. Centrally provided software will be modified to do this automatically where necessary for MKL use.

```
export MKL_DEBUG_CPU_TYPE=5
export MKL_CBWR=COMPATIBLE
```

Or, more recommended as it will pick up future changes if necessary:

```
module mkl_env
```

4.5.5 OpenMP

We recommend specifying OpenMP options where code supports it. OMP_NUM_THREADS should be set as appropriate and OMP_PROC_BIND causes process-core binding to occur, which is typically of a performance benefit due to cache effects.

```
export OMP_NUM_THREADS=1

export OMP_PROC_BIND=true
```

4.5.6 Finding the Incompatibles

If you do submit a code that has been compiled for Intel Skylake but contains illegal instructions when used on the AMD Rome nodes, you will see an error in one of a few ways.

1. Meaningful message:

```
  $ /apps/materials/QuantumEspresso/6.1/el7/AVX2/intel-2018/intel-2018/bin/pw.x

Please verify that both the operating system and the processor support Intel(R)
X87, CMOV, MMX, FXSAVE, SSE, SSE2, SSE3, SSSE3, SSE4_1, SSE4_2, MOVBE, POPCNT,
F16C, AVX, FMA, BMI, LZCNT and AVX2 instructions.
```

2. Immediate exit with instruction error:

```
$ /apps/chemistry/gromacs/2018.2-single/el7/AVX512/intel-2018/intel-
2018/bin/mdrun_mpi<br> Illegal instruction (core dumped)
```

In any such case, the code needs recompiling as per above advice. If this occurs with a centrally provided software module, please contact the helpdesk.

4.6 Data Storage

Portal URL: https://portal.supercomputing.wales/index.php/storage-quotas/

4.6.1 Types of Storage

Super Computing Wales offers two types of storage: home directories and scratch.

The scratch disks are large, have no quota, are on fast storage but only intended for short term storage of temporary data. Please don't leave data on the scratch disk once you are finished with it. If the disk fills up it will negatively impact on other users' ability to run jobs.

Home directories have a default quota of 50GB per user, they are slower to access and are intended for longer term storage of data.

4.6.2 Checking Storage Quotas

Quotas are enabled on Supercomputing Wales home directories.

These limit the amount of storage space and number of files that a user or group can store in their respective home directory location.

To view your allocation, use the 'myquota' command, like this:

```
[ade.fewings@csc001 ~]$ myquota
Disk quotas for group ade.fewings (gid 16778693):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
cfsfs001-s05:/nfsshare/exports/space05
                   402G    500G    525G            192k   1000k   1050k
```

In this example, we see six items of data:

- The first set of three items represent storage space:
  - 'blocks' is the amount of storage space utilised, in this case 402GB
  - 'quota' is the 'soft limit' assigned, in this case 500GB
  - 'limit' is the 'hard limit' assigned, in this case 525GB
- The second set of three items represent file count:
  - 'files' is the number of files present, in this case 192000
  - 'quota' is the 'soft limit' assigned, in this case 1000000

- ‘limit’ is the ‘hard limit’ assigned, in this case 1050000

A ‘soft limit’ can be exceeded for a period of maximum 7 days, after which utilisation must be reduced below the threshold or error will result. In this case, the ‘grace’ item in the ‘myquota’ output will display the number of days remaining in this state.

A ‘hard limit’ cannot be exceeded and attempts to do so will result in error.

Increases in quota allocations are possible on a case-by-case basis, please contact the Support Desk to request. However, it is worth noting that quotas are implemented to reduce the unfair use of the system by a few users and thus provide capacity for new users.

4.6.3 Shared Areas

We are able to create two types of shared area for multiple users to access:

Project Shared Areas are a way for the users of a particular project to share data. They are separate from user home directories and quotas as such.

Collaboration Shared Areas are also possible that cross project and institutional boundaries as separate areas from home directories for data sharing among multiple users.

Please contact Support to discuss either of these if this would be useful.

4.7 Conduct & Best Practice

Portal URL: https://portal.supercomputing.wales/index.php/best-practice/

- Be kind to other users, don’t run too many jobs at once or fill up the scratch drive.
- Don’t run jobs on the login node, submit them to the cluster with Slurm instead.
- Don’t run serial jobs in the compute partition on Hawk, use the ‘htc’ (high throughput) partition instead – the system will attempt to auto-redirect such jobs if you do submit them, but please try to submit straight to the most appropriate place.
  - Sunbird has no HTC partition and its compute partition is for both serial and parallel jobs.
- Only run jobs which need a GPU on the GPU partition. Only run jobs that need high memory on the ‘highmem’ partition.
- Use the “dev” partition for testing short runs of your software.
- If you are a member of multiple projects, use the -A option to sbatch to ensure your job is accounted against the right project.
- Try to match your job specification to its actual needs. Don’t request more memory, CPU cores or time than you need.
- If you request exclusive use of a node, make sure you are using all 40 cores. We have relatively few nodes, each with a lot of cores.
- Make sure that jobs last at least a few minutes. Small jobs add a lot of overhead to the scheduler.
- Use GNU Parallel to combine multiple small jobs into a single larger one.
- Don’t store data on scratch, it may be deleted after 60 days of non-access.
- Don’t use SCW as a long term data store, home directories aren’t backed up. If you need a long term data store ask your institution.

- Publish your data and somebody else will store it for you!
    - ❖ Data sets up to 50 gigabytes can be published via Zenodo.
    - ❖ Data, code and protocols can be submitted for peer review and publication via GigaScience.
- Credit SCW in your publications.
- Tell SCW staff if you get a paper published from work which used the system or if you get a grant that will use it funded.

5 Advanced Use

Use of the supercomputers is largely straightforward, but some types of more advanced use are covered here.

5.1 MPI and/or OpenMP Jobs

Portal: https://portal.supercomputing.wales/index.php/index/slurm/batch-submission-of-mpi-and-openmp/

There are many different ways to specify to Slurm the requirements of a job, through arguments such as the number of nodes, the number of tasks, memory requirements per task, etc.

We provide here some examples covering the mixes of MPI and OpenMP. This requires knowledge of the underlying hardware, remember SCW standard compute nodes have 40 cores per node, previous systems had less than this, e.g. 16 cores per node on Sandy Bridge systems.

5.1.1 MPI

A job of 160 parallel processes, distributed as 40 processes per node, hence occupying 4 nodes:

```
#SBATCH --ntasks=160
#SBATCH --tasks-per-node=40
```

5.1.2 OpenMP

A job of 8 OpenMP thread, defaulting to 1 node, hence occupying 8 processors of one node:

```
#SBATCH --cpus-per-task=8
```

5.1.3 MPI + OpenMP

A hybrid MPI + OpenMP job, with 4 MPI tasks, each of 10 OpenMP thread, and distributed one MPI task per node:

```
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=10
#SBATCH --tasks-per-node=1
```

5.2 Batch Submission of Serial Tasks

Portal: https://portal.supercomputing.wales/index.php/index/slurm/interactive-use-job-arrays/batch-submission-of-serial-jobs-for-parallel-execution/

Large numbers of serial jobs can become incredibly inefficient and troublesome on mixed-mode HPC systems. The SCW Slurm deployment limits the number of running & submitted jobs any single user may have.

However, there are ways to submit multiple jobs:

1. Background jobs using shell process control and wait for processes to finish on a single node.
2. Combining *GNU Parallel* and Slurm's *srun* command allows us to handle such situations in a more controlled and efficient way than in the past. Using this method, a single job is submitted that requests an allocation of X cores, and the GNU *parallel*command enables us to utilise all of those cores by launching the serial tasks using the *srun* command.
3. Using Job Arrays for very similar tasks. See section **5.3**.

5.2.1 Shell process control

Here is an example of submitting 2 processes on a single node:

```
#!/bin/bash
#SBATCH --ntasks=32
#SBATCH --ntasks-per-node=32
#SBATCH -o example.log.%J
#SBATCH -e example.err.%J
#SBATCH -J example
#SBATCH --time=1:00:00
#SBATCH --exclusive

time my_exec &lt; input1.csv &gt; input1.log.$SLURM_JOBID &amp;
time my_exec &lt; input2.csv &gt; input2.log.$SLURM_JOBID &amp;
# important to make sure the batch job won't exit before all the
# simultaneous runs are completed.
wait
```

The *my_exec* commands in this case would be multithreaded to use 32 cores between them.

5.2.2 GNU Parallel and Slurm's srun command

Here is the example, commented, job submission file *serial_batch.sh*:

```
#!/bin/bash --login
#SBATCH -n 40                    #Number of processors in our pool
#SBATCH -o output.%J             #Job output
#SBATCH -t 12:00:00              #Max wall time for entire job
#change the partition to compute if running in Swansea
#SBATCH -p htc         #Use the High Throughput partition which is intended for serial jobs

module purge
module load hpcw
module load parallel

# Define srun arguments:
srun="srun -n1 -N1 --exclusive"
# --exclusive     ensures srun uses distinct CPUs for each job step
# -N1 -n1         allocates a single core to each task

# Define parallel arguments:
parallel="parallel -N 1 --delay .2 -j $SLURM_NTASKS --joblog parallel_joblog --resume"
# -N 1              is number of arguments to pass to each job
# --delay .2        prevents overloading the controlling node on short jobs
# -j $SLURM_NTASKS  is the number of concurrent tasks parallel runs, so number of CPUs
allocated
# --joblog name     parallel's log file of tasks it has run
# --resume          parallel can use a joblog and this to continue an interrupted run (job
resubmitted)
```

```
# Run the tasks:
$parallel "$srun ./runtask arg1:{1}" ::: {1..64}
# in this case, we are running a script named runtask, and passing it a single argument
# {1} is the first argument
# parallel uses ::: to separate options. Here {1..64} is a shell expansion defining the
values for
#    the first argument, but could be any shell command
#
# so parallel will run the runtask script for the numbers 1 through 64, with a max of 40
running
#    at any one time
#
# as an example, the first job will be run like this:
#    srun -N1 -n1 --exclusive ./runtask arg1:1
```

So, in the above we are requesting an allocation from Slurm of 12 processors, but we have 32 tasks to run. Parallel will execute the jobs as soon as space on our allocation becomes available (i.e. tasks finish). As this does not have the overhead of setting up a new full job, it is more efficient.

A simple 'runtask' script that demonstrates the principal by logging helpful text is included here, courtesy of the University of Chicago Research Computing Centre:

```
#!/bin/sh

# this script echoes some useful output so we can see what parallel
# and srun are doing

sleepsecs=$[($RANDOM % 10) + 10]s

# $1 is arg1:{1} from parallel.
# $PARALLEL_SEQ is a special variable from parallel. It the actual sequence
# number of the job regardless of the arguments given
# We output the sleep time, hostname, and date for more info&gt;
echo task $1 seq:$PARALLEL_SEQ sleep:$sleepsecs host:$(hostname) date:$(date)

# sleep a random amount of time
sleep $sleepsecs
```

So, one would simply submit the job script as per normal:

```
$ sbatch serial_batch.sh
```

And we then see output in the Slurm job output file like this:

```
...
task arg1:34 seq:34 sleep:11s host:ccs0132 date:Fri 29 Jun 09:37:26 BST 2018
srun: Exclusive so allocate job details
task arg1:38 seq:38 sleep:12s host:ccs0132 date:Fri 29 Jun 09:37:27 BST 2018
srun: Exclusive so allocate job details
task arg1:45 seq:45 sleep:11s host:ccs0132 date:Fri 29 Jun 09:37:29 BST 2018
srun: Exclusive so allocate job details
task arg1:41 seq:41 sleep:12s host:ccs0132 date:Fri 29 Jun 09:37:28 BST 2018
```

```
srun: Exclusive so allocate job details
task arg1:47 seq:47 sleep:11s host:ccs0132 date:Fri 29 Jun 09:37:29 BST 2018
srun: Exclusive so allocate job details
...
```

Also the parallel job log records completed tasks:

```
Seq Host Starttime JobRuntime Send Receive Exitval Signal Command
8 : 1530261102.040 11.088 0 75 0 0 srun -n1 -N1 --exclusive ./runtask arg1:8
9 : 1530261102.248 11.088 0 75 0 0 srun -n1 -N1 --exclusive ./runtask arg1:9
5 : 1530261101.385 12.088 0 75 0 0 srun -n1 -N1 --exclusive ./runtask arg1:5
12 : 1530261102.897 12.105 0 77 0 0 srun -n1 -N1 --exclusive ./runtask arg1:12
1 : 1530261100.475 17.082 0 75 0 0 srun -n1 -N1 --exclusive ./runtask arg1:1
2 : 1530261100.695 17.091 0 75 0 0 srun -n1 -N1 --exclusive ./runtask arg1:2
3 : 1530261100.926 17.088 0 75 0 0 srun -n1 -N1 --exclusive ./runtask arg1:3
10 : 1530261102.450 16.088 0 77 0 0 srun -n1 -N1 --exclusive ./runtask arg1:10
6 : 1530261101.589 17.082 0 75 0 0 srun -n1 -N1 --exclusive ./runtask arg1:6
...
```

So, by tweaking a few simple commands in the job script and having a 'runtask' script that does something useful, we can accomplish a neat, efficient serial batch system.

5.2.3 Multi-Threaded Tasks

It is trivially possible to use the above technique and scripts, with a very small modification, to run multi-threaded or otherwise intra-node parallel tasks. We achieve this by changing the SBATCH directive specifying processor requirement (#SBATCH -n …) in the submission script to the following form:

```
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=3
#SBATCH --cpus-per-task=4
```

In this case, parallel will launch across 3 nodes, and will run 3 tasks of 4 processors each per node.

5.3 Job Arrays

5.3.1 Submission

Job arrays operate in Slurm much as they do in other batch systems. They enable a potentially huge number of similar jobs to be launched very quickly and simply, with the value of a runtime-assigned *array id* then being used to cause each particular job iteration to vary slightly what it does. Array jobs are declared using the *–array* argument to *sbatch*, which can (as with all arguments to *sbatch*) be inside as job script as an *#SBATCH* declaration or passed as a direct argument to *sbatch*. There are a number of ways to declare:

```
[test.user@cl1 hello_world]$ sbatch --array=0-64 sbatch_sub.sh
```

…declares an array with iteration indexes from 0 to 64.

```
[test.user@cl1 hello_world]$ sbatch --array=0,4,8,12 sbatch_sub.sh
```

…declares an array with iteration indexes specifically identified as 0, 4, 8 and 12.

```
[test.user@cl1 hello_world]$ sbatch --array=0-12:3 sbatch_sub.sh
```

…declares an array with iteration indexes from 0 to 12 with a stepping of 3, i.e. 0,3,6,9,12

5.3.2 Monitoring

When a job array is running, the output of squeue shows the parent task and the currently running iteration indexes:

```
[test.user@cl1 hello_world]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
        143_[6-64]       all    hello test.use PD       0:00      4 (Resources)
             143_4       all    hello test.use  R       0:00      4 ccs[005-008]
             143_5       all    hello test.use  R       0:00      4 ccs[005-008]
             143_0       all    hello test.use  R       0:03      4 ccs[001-004]
             143_1       all    hello test.use  R       0:03      4 ccs[001-004]
[test.user@cl1 hello_world]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
       143_[15-64]       all    hello test.use PD       0:00      4 (Resources)
            143_14       all    hello test.use  R       0:00      4 ccs[001-004]
            143_10       all    hello test.use  R       0:02      4 ccs[005-008]
            143_11       all    hello test.use  R       0:02      4 ccs[005-008]
             143_1       all    hello test.use  R       0:07      4 ccs[001-004]
```

IDs and Variables

Each iteration in an array assumes its own job ID in Slurm. However, Slurm creates two new environment variables that can be used in the script in addition to *SLURM_JOB_ID* storing the particular iteration's job ID.

SLURM_ARRAY_JOB_ID stores the value of the parent job submission – i.e. the ID reported in the output from *sbatch* when submitted.

SLURM_ARRAY_TASK_ID stores the value of the array index.

Additionally, when specifying a job's STDOUT and STDERR files using the *-o* and *-e* directives to *sbatch*, the reference *%A* will take on the parent job ID and the reference *%a* will take on the iteration index. In summary:

BASH Environment Variable	SBATCH Field Code	Description
$SLURM_JOB_ID	%J	Job identifier
$SLURM_ARRAY_JOB_ID	%A	Array parent job identifier
$SLURM_ARRAY_TASK_ID	%a	Array job iteration index

And so, with this example script:

```
#!/bin/bash
#SBATCH -J arraytest
#SBATCH --array=0-4
#SBATCH -o output-%A_%a-%J.o
#SBATCH -n 1
echo SLURM_JOB_ID $SLURM_JOB_ID
```

```
echo SLURM_ARRAY_JOB_ID $SLURM_ARRAY_JOB_ID
echo SLURM_ARRAY_TASK_ID $SLURM_ARRAY_TASK_ID
```

We can submit the script:

```
[test.user@cl1 sbatch]$ sbatch array.sh

Submitted batch job 231
```

Resulting in the following output files:

```
output-231_0-232.o

output-231_1-233.o

output-231_2-234.o

output-231_3-235.o

output-231_4-231.o
```

Each iteration of which contained variables as follows:

```
output-231_0-232.o:

SLURM_JOB_ID 232

SLURM_ARRAY_JOB_ID 231

SLURM_ARRAY_TASK_ID 0

output-231_1-233.o:

SLURM_JOB_ID 233

SLURM_ARRAY_JOB_ID 231

SLURM_ARRAY_TASK_ID 1

output-231_2-234.o:

SLURM_JOB_ID 234

SLURM_ARRAY_JOB_ID 231

SLURM_ARRAY_TASK_ID 2

output-231_3-235.o:

SLURM_JOB_ID 235
```

```
SLURM_ARRAY_JOB_ID 231

SLURM_ARRAY_TASK_ID 3
```

output-231_4-231.o:

```
SLURM_JOB_ID 231

SLURM_ARRAY_JOB_ID 231

SLURM_ARRAY_TASK_ID 4
```

More advanced job array information is available in the Slurm documentation here.

5.4 Custom Job Geometry

Portal: https://portal.supercomputing.wales/index.php/slurm/interactive-use-job-arrays/custom-parallel-task-geometry/

If you need to run an MPI (/OpenMP) task that requires a custom task geometry, perhaps because one task requires a larger amount of memory than the others, then this can easily be achieved with Slurm.

To do this, rather than specifying the number of processors required, one can specify the number of nodes (#SBATCH –nodes=X) plus the number of tasks per node (#SBATCH –tasks-per-node=X). The geometry can then be defined to the SLURM_TASKS_PER_NODE environment variable at runtime. As long as there are enough nodes to match the geometry, then Slurm will allocate parallel tasks to the MPI runtime to follow the geometry specification.

For example:

```
#!/bin/bash --login

#SBATCH --job-name geom_test
#SBATCH --nodes 4
#SBATCH --ntasks-per-node 16
#SBATCH --time 00:10:00
#SBATCH --output geom_test.%J.out

module purge
module load mpi/intel

export SLURM_TASKS_PER_NODE='1,16(x2),6'
mpirun ./mpi_test
```

In this case, we are requesting 4 nodes and 16 processors on those 4 nodes. Therefore, a maximum job size of 64 parallel tasks (to match the number of allocated processors) would apply. However, we override the SLURM_TASKS_PER_NODE environment variable to be just a single task on the first node, then fill the next two allocated nodes, and then place just six parallel tasks on the final allocated node. So, in this case, a total of 1+16+16+6=39 parallel processes. ‘mpirun’ will automatically pick this up from the Slurm allocated runtime environment.

5.5 Interactive Compute Use

Portal URL: https://portal.supercomputing.wales/index.php/slurm/interactive-use-job-arrays/interactive-use/

In order to use an HPC system interactively – i.e. whilst sat in front of the terminal interacting live with allocated resources – there is a simple two stage process in Slurm.

Firstly, we must create an allocation – that is, an allocation of a certain amount of resources that we specify we need. This is done using the *salloc* command, like this:

```
[test.user@cl1 imb]$ salloc -n 8 --ntasks-per-node=1
 salloc: Granted job allocation 134
```

Now that an allocation has been granted, we have access to those specified resources. Note that the resource specification we made in this case is exactly as the parameters passed for batch use was – so in this case we have asked for 8 tasks (processes) with them distributed at one per node.

Now that we are 'inside' an allocation, we can use the *srun* command to execute against the allocated resources, for example:

```
[test.user@cl1 imb]$ srun hostname
ccs0129
ccs0130
ccs0126
ccs0133
ccs0125
ccs0121
ccs0134
ccs0122
```

The above output shows how, by default, *srun* executes a command on all allocated processors. Arguments can be passed to *srun* to operate differently, for example:

We could also launch an MPI job here if we wished. We would load the software *modules* as we do in a batch script and call *mpirun* in the same way. This can be useful during code debugging.

It is also possible to use *srun* to launch an interactive shell process for some heavy processing on a compute node, for example:

```
srun -n 2 --pty bash
```

This would move us to a shell on a compute node.

5.6 X11 Graphical Use

Portal URL: https://portal.supercomputing.wales/index.php/slurm/interactive-use-job-arrays/x11-gui-forwarding/

Some applications provide the capability to interact with a graphical user interface (GUI). It is not typical of parallel jobs, but large-memory applications and computationally steered applications can offer such capability.

5.6.1 Setting up X Forwarding

First we must login to SCW with X Forwarding enabled.

```
$ ssh -X username@hawklogin.cf.ac.uk
```

Windows users will need to go to the Connection->SSH->X11 options and enable "Enable X11 Forwarding". You will also need an X server such as VcXsrv or XMing.

Mac users will need to download Xquartz.

There are two ways to run graphical jobs on the system:

5.6.2 SSH into the node

Alternatively you can SSH directly into a node you've been allocated via *salloc*. By adding the *-X* option to this the X session will be forwarded from the compute node back to the login node. This is shown in the example below.

```
[username@cl2 ~]$ salloc -n 1
salloc: Granted job allocation 30937
salloc: Waiting for resource configuration
salloc: Nodes ccs0046 are ready for job
[username@cl2 ~]$ ssh -X ccs0046 xterm
```

5.6.3 Use srun

Note: this method is not currently working

With Slurm, once a resource allocation is granted for an interactive session (or a batch job when the submitting terminal if left logged in), we can use *srun* to provide X11 graphical forwarding all the way from the compute nodes to our desktop using *srun -x11 <application>.*

```
[username@cl2 ~]$ salloc -n 1
salloc: Granted job allocation 30937
salloc: Waiting for resource configuration
salloc: Nodes ccs0046 are ready for job

[username@cl2 ~]$ srun -x11 xterm
```

Note that the user must have X11 forwarded to the login node for this to work – this can be checked by running *xclock* at the command line.

Additionally, the *-x11* argument can be augmented in this fashion *–x11=[batch|first|last|all]* to the following effects:

- *-x11=first* This is the default, and provides X11 forwarding to the first compute hosts allocated.
- *-x11=last* This provides X11 forwarding to the last of the compute hosts allocated.
- *-x11=all* This provides X11 forwarding from all allocated compute hosts, which can be quite resource heavy and is an extremely rare use-case.
- *-x11=batch* This supports use in a batch job submission, and will provide X11 forwarding to the first node allocated to a batch job. The user must leave open the X11 forwarded login node session where they submitted the job.

5.7 VNC Guide (Linux/OS X) - Sunbird Only

6 Software Training & Guides

A number of example jobs an application guides are available, please see:

6.1 Training Software Examples

As part of training to use the system, two *tarballs* (compressed archives) of a large number and variety of application example runs are available at:

```
/apps/Hawk_hpcw_training_workshop.2018.tar.gz

/apps/Hawk_raven_training_workshop.2018.tar.gz
```

As apparent from the naming scheme, these focus on the software stacks migrated from HPCW/Raven, but will migrate to new optimised software builds over time.

The training tarballs are self-documenting, so please start by viewing the associated README files:

```
/apps/README.training_workshop.HPCWUsers.2018

/apps/README.training_workshop.RavenUsers.2018
```

6.2 Application Guides & References

Portal URL: https://portal.supercomputing.wales/index.php/index/application-guides/

6.2.1 Hawk Application Guides

Each of the guides in the Table below provides instructions on how to run a standard test case for the application in question on the Supercomputing Wales system.

APPLICATION	NOTES
ABySS	ABySS (Assembly By Short Sequences) is a de novo, parallel, paired-end sequence assembler that is designed for short reads. The single-processor version is useful for assembling genomes up to 100 Mbases in size. The parallel version is implemented using MPI and is capable of assembling larger genomes. ABySS is described in detail at http://genome.cshlp.org/content/19/6/1117.long.
BLAST	The Basic Local Alignment Search Tool (BLAST) finds regions of local similarity between sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches. BLAST can be used to infer functional and evolutionary relationships between sequences as well as help identify members of gene families.
ClustalW	ClustalW is a general purpose multiple sequence alignment program for DNA or proteins. The home page for ClustalW is http://www.clustal.org/clustal2/and the installed version 2 is described in the article available at http://bioinformatics.oxfordjournals.org/content/23/21/2947.long
CP2K	CP2K is a freely available (GPL) program written in Fortran 95 to perform atomistic and molecular simulations of solid state, liquid, molecular and biological systems. It provides a general framework for different methods: density functional theory (DFT) using a mixed Gaussian and plane waves approach (GPW), classical pair and many-body potentials, semiempirical (AM1, PM3, MNDO, MNDOd, PM6)

	Hamiltonians, and Quantum Mechanics/ Molecular Mechanics (QM/MM) hybrid schemes relying on the Gaussian Expansion of the Electrostatic Potential (GEEP).
DL_POLY Classic	DL_POLY Classic is a general purpose (parallel and serial) molecular dynamics simulation package derived from the package formerly known as DL_POLY_2 (Version 20) which was originally written by W. Smith and T.R. Forester at Daresbury Laboratory to support the CCP5 project `Computer Simulation of Condensed Phases'. Note that DL_POLY Classic is not the same as DL_POLY 4 – for small to medium sized systems on moderate core counts, DL_POLY Classic will be competitive and the instructions for using the two codes are very similar, only the performance is different.
DL_POLY 4	DL_POLY is a general purpose classical molecular dynamics (MD) simulation software developed at Daresbury Laboratory by I.T. Todorov and W. Smith. DL_POLY_4's general design provides scalable performance from a single processor workstation to a high performance parallel computer. DL_POLY_4 offers fully parallel I/O as well as a netCDF alternative (HDF5 library dependence) to the default ASCII trajectory file.
GAMESS (UK-GA)	The General Atomic and Molecular Electronic Structure System (GAMESS) is a general purpose ab initio quantum chemistry package. The original code split in 1981 into GAMESS (US) and GAMESS (UK) variants, which now differ significantly. GAMESS (UK) can perform a number of general computational chemistry calculations, including Hartree–Fock, Møller–Plesset perturbation theory (MP2 & MP3), coupled cluster (CCSD & CCSD(T)), density functional theory (DFT), configuration interaction (CI), and other advanced electronic structure methods. Calculation of valence bond wave functions is possible by the TURTLE code, due to J. H. van Lenthe. Note there are two versions of the parallel code, one based on the Global Array (GA) tools from Pacific Northwest National Laboratory, and the second based on MPI and ScaLAPACK. The current note is restricted to a consideration of the GA-based code.
GAMESS (UK-MPI)	The General Atomic and Molecular Electronic Structure System (GAMESS) is a general purpose ab initio quantum chemistry package. The current note is restricted to a consideration of the MPI/ScaLAPACK code that focuses on the performance of efficient, large-scale DFT calculations.
GAMESS (US)	GAMESS (US) is a program for ab initio molecular quantum chemistry. Briefly, GAMESS can compute SCF wavefunctions ranging from RHF, ROHF, UHF, GVB, and MCSCF. Correlation corrections to these SCF wavefunctions include CI, 2nd order perturbation theory and coupled-cluster (CC) approaches, as well as the DFT approximation. Excited states can be computed by CI, EOM, or TD-DFT procedures. Nuclear gradients are available for automatic geometry optimisation, transition state searches, or reaction path following. Computation of the energy Hessian permits prediction of vibrational frequencies, with IR or Raman intensities. Solvent effects may be modelled by the discrete effective fragment potentials, or continuum models such as the Polarizable Continuum Model. Numerous relativistic computations are available. The FMO method permits use of many of these sophisticated treatments to be used on very large systems, by

	dividing the computation into small fragments. Nuclear wavefunctions can also be computed, in VSCF, or with explicit treatment of nuclear orbitals by the NEO code
GROMACS	GROMACS (GROningen MAchine for Chemical Simulations) is a molecular dynamics package primarily designed for simulations of proteins, lipids and nucleic acids. It was originally developed in the Biophysical Chemistry department of the University of Groningen, and is now maintained by contributors in universities and research centers across the world. GROMACS is one of the fastest and most popular molecular dynamics software packages available, and can run on GPUs as well as CPUs. It is free, open source released under the GNU General Public License. Starting from version 4.6, GROMACS is released under the GNU Lesser General Public License.
JAGS	JAGS is Just Another Gibbs Sampler. It is a program for analysis of Bayesian hierarchical models using Markov Chain Monte Carlo (MCMC) simulation. It is similar to the BUGS program (http://www.mrc-bsu.cam.ac.uk/software/bugs/) developed by the UK Medical Research Council. JAGS is licensed under the GNU General Public License. You may freely modify and redistribute it under certain conditions.
LAMMPS	LAMMPS is a freely available (GPL) classical molecular dynamics code and an acronym for Large-scale Atomic/Molecular Massively Parallel Simulator. LAMMPS has potentials for solid-state materials (metals, semiconductors) and soft matter (biomolecules, polymers) and coarse-grained or mesoscopic systems. It can be used to model atoms or, more generically, as a parallel particle simulator at the atomic, meso, or continuum scale. LAMMPS runs on single processors or in parallel using message-passing techniques and a spatial decomposition of the simulation domain. The code is designed to be easy to modify or extend with new functionality. It is distributed by Sandia National Laboratories, a US Department of Energy laboratory.
NWChem	NWChem is a general-purpose computational chemistry code specifically designed to run on distributed memory parallel computers. The core functionality of the code focuses on Hartree–Fock theory, DFT methods for both plane-wave basis sets as well as Gaussian basis sets, tensor contraction engine-based coupled cluster capabilities, combined QM/MM descriptions and MD. Scalable implementations of these methods rely on a global address space library, the Global Array toolkit. NWChem has been developed by the Molecular Sciences Software group of the Theory, Modeling & Simulation program of the Environmental Molecular Science Laboratory (EMSL) at the Pacific Northwest National Laboratory. The early implementation was funded by the EMSL Construction Project.
Quantum ESPRESSO	Quantum ESPRESSO (QE) is an integrated suite of open-source computer codes for electronic structure calculations and materials modelling at the nanoscale. It is based on DFT, plane waves, and pseudopotentials. QE has evolved into a distribution of independent and inter-operable codes in the spirit of an open-source project. The QE distribution consists of a “historical” core set of components, and a set of plug-ins that perform more advanced tasks, plus a number of third-party packages designed to be inter-operable with the core components. Researchers active in the field of electronic-structure calculations

	are encouraged to participate in the project by contributing their own codes or by implementing their own ideas into existing codes.
SWAN	SWAN is a wave model to compute random, short-crested wind-generated waves in coastal regions and inland waters (https://www.cesdb.com/swan.html)
TELEMAC	TELEMAC is a collection of modelling tools to simulate free-surface flows. (https://www.researchgate.net/publication/248017312_TELEMAC_modelling_system_an_overview)
VASP	The Vienna Ab initio Simulation Package (VASP) is a computer program for atomic scale materials modelling, e.g. electronic structure calculations and quantum-mechanical molecular dynamics, from first principles. VASP computes an approximate solution to the many-body Schrödinger equation, either within density functional theory (DFT), solving the Kohn-Sham equations, or within the Hartree-Fock (HF) approximation, solving the Roothaan equations. Hybrid functionals that mix the Hartree-Fock approach with density functional theory are implemented as well. Furthermore, Green's functions methods (GW quasiparticles, and ACFDT-RPA) and many-body perturbation theory (2nd-order Møller-Plesset) are available. (https://www.vasp.at/index.php/about-vasp/59-about-vasp). Please note that only the parallel version of VASP is installed on Supercomputing Wales. Supercomputing Wales can make this application available to selected users, as a VASP licence is required. Please contact the Supercomputing Wales helpdesk to request access
WRF	The Weather Research and Forecasting Model (WRF) is a state-of-the-art atmospheric modelling system (https://www.mmm.ucar.edu/weather-research-and-forecasting-model) that is applicable for both meteorological research and numerical weather prediction. It offers a host of options for atmospheric physical processes and is suitable for a broad range of applications across scales ranging from tens of metres to the global, including: (i) Meteorological investigations, (ii) Real-time numerical weather prediction, (iii) Idealised atmospheric simulations, (iv) Data assimilation studies and development, (v) Coupling with other earth system models, and (vi) Modelling and model use instruction and training.

6.2.2 Sunbird User Guides

Using IDL

6.2.3 HPC Wales User Guides

ABySS

BLAST

ClustalW

CP2K

DL_POLY

DL_POLY_Classic

Fluidity

GAMESS_UK

GAMESS_UK_MPI

GAMESS_US

Gerris

Gromacs

JAGS

LAMMPS

NW_Chem

OpenFOAM

QuantumESPRESSO

SWAN

VASP

TELEMAC

WRF

6.3 How-To Guides (Archive)

Using Linux (Old Version)

Using Vi

Using Emacs

Using nano

Using Valgrind

Using Scalasca

Using OSS

Using IPM

FileZilla_setup (Please note: change the connect host to either Hawk or Sunbird addresses)

Using Singularity Containers

Installing Python Modules

Installing R Packages

6.4 Technical Notes

TERRA Report

I-TASSER Report

FLITE Report

GROMACS Report

Computational Rheological Modelling Report

7 Software Development

7.1 Development environment

Portal URL: https://portal.supercomputing.wales/index.php/development-environment/

The development environment on SCW is designed to maximise the usage of the system. We have a number of compilers and development libraries available.

7.1.1 Compilers

Compilers are available in the module system and can be accessed like any other module.

```
$ module load compiler/intel/2018
```

This will load the Intel compilers ifort, icc, icpc.

Suggestions for compiler flags are:

```
-xHost : maximise optimisation for CPU being compiled on

-xCORE-AVX512 : tune just for AVX-512 processors such as Skylake on SCW.

-axCORE-AVX2,CORE-AVX512 : compile for multiple architectures and tune for each processor.
```

Other compilers are available, please check using

```
$ module av compiler
```

For certain features there is a dependency between Intel Compiler and the currently available GCC (GNU Compiler Collection) compiler in the environment. For example, this is could be due to Intel C++ compiler using the GCC C++ library to maintain compatibility between Intel compiled C++ code and GCC. Therefore, if newer features in C++ are required (maybe an updated standard) then you need to load a compatible GCC version with the required C++ standard supported. At time of writing Intel Compilers support GCC between version 4.3 and 6.3.

To load a GCC compiler to use with Intel compiler just load the GCC module first.

```
$ module load compiler/gnu/5

$ module load compiler/intel/2018
```

7.1.2 MPI

MPI libraries can be loaded using for example

```
$ module load mpi/intel/2018
```

This will provide the mpif90, mpicc and mpic++ (and Intel versions mpiifort, mpiicc, mpiicpc)

Be careful about using OpenMP and MPI together. Recent versions of Intel MPI library now use the multi-threaded libraries by default but depending on MPI implementation you might need to check what you want. Threading behaviour changes the response from MPI_Init_Thread.

Other MPI libraries are available, please check using

```
$ module av mpi
```

7.1.3 CUDA

To maximise the use of the NVIDIA GPU nodes available in SCW we can use the CUDA library. This can be loaded using

```
$ module load CUDA
```

The NVIDIA compiler is then added to your PATH and you can compile using nvcc.

7.1.4 Other tools

There are a number of tools available and others planned to follow. Please get in touch if specific help is required.

7.2 Using GPUs

Portal URL: https://portal.supercomputing.wales/index.php/using-gpus/

7.2.1 GPU Usage

Slurm controls access to the GPUs on a node such that access is only granted when the resource is requested specifically. Slurm models GPUs as a Generic Resource (GRES), which is requested at job submission time via the following additional directive:

```
#SBATCH --gres=gpu:2
```

This directive requires Slurm to allocate two GPUs per allocated node, to not use nodes without GPUs and to grant access. SCW GPU nodes have two GPUs each. Jobs must also be submitted to the GPU-enabled nodes queue:

```
#SBATCH -p gpu
```

It is then possible to use CUDA enabled applications or the CUDA toolkit modules themselves, modular environment examples being:

```
module load cuda/cuda-9.1
```

```
module load gromacs/2018.2-single-gpu
```

7.2.2 CUDA Versions & Hardware Differences

Multiple versions of the CUDA libraries are installed on SCW systems, as can always be seen by:

```
[b.iss03c@cl1 ~]$ module avail CUDA

---- /apps/modules/libraries ----

CUDA/10.0 CUDA/10.1 CUDA/8.0 CUDA/9.0 CUDA/9.1 CUDA/9.2
```

The GPU nodes always run the latest nvidia driver to support the latest installed version of CUDA and also offer backwards-compatabilitity with prior versions.

However, **Hawk** contains *Pascal* generation nVidia Tesla cards which are supported in all installed versions of CUDA, but **Sunbird** contains *Volta* generation nVidia Tesla cards which are only supported in CUDA 9+. Codes that require CUDA 8, such as Amber 16, will not run on the *Volta* cards available on Sunbird.

7.2.3 GPU Compute Modes

nVidia GPU cards can be operated in a number of Compute Modes. In short the difference is whether multiple processes (and, theoretically, users) can access (share) a GPU or if a GPU is exclusively bound to a single process. It is typically application-specific whether one or the other mode is needed, so please pay particular attention to example job scripts. GPUs on SCW systems default to 'shared' mode.

Users are able to set the Compute Mode of GPUs allocated to their job through a pair of helper scripts that should be called in a job script in the following manner:

To set exclusive mode:

```
clush -w $SLURM_NODELIST "sudo /apps/slurm/gpuset_3_exclusive"
```

And to set shared mode (although this is the default at the start of any job):

```
clush -w $SLURM_NODELIST "sudo /apps/slurm/gpuset_0_shared"
```

To query the Compute Mode:

```
clush -w $SLURM_NODELIST "nvidia-smi -q|grep Compute"
```

In all cases above, sensible output will appear in the job output file.

Additionally, as Slurm models the GPUs as a consumable resource that must be requested in their own right (i.e. not implicitly with processor/node count), the default of the scheduler would be to *not* allocate the same GPU to multiple users or jobs – it would take some manual work to achieve this.

7.3 Debugging

7.4 Profiling

8 Migrating from other HPC systems

For users migrating from HPC Wales and/or Raven:

8.1 Rapid Start for HPCW Users

Portal URL: https://portal.supercomputing.wales/index.php/quick-start-for-hpc-wales-users-migrating-to-scw/

For users of the HPC Wales services who are migrating to the SCW service, please be aware of these important notes in order to get up-and-running as quickly & effectively as possible.

Hawk is the new supercomputer at Cardiff and is for the use of Cardiff & Bangor led projects.

Sunbird is the new supercomputer at Swansea and is for the use of Swansea & Aberystwyth led projects.

8.1.1 User Credentials

- These have changed to be linked to your institutional accounts.
- The method to access your new credentials and get started with Hawk is covered on the Getting Access page – please follow it closely.
    - ❖ Contact Support if you have any issues.
- Existing project memberships have also been migrated.

8.1.2 Data

- Old home directories from the three HPCW sites have been cloned to a read-only area on Hawk for quick access.
    - ❖ These will remain for approximately 6 months only and then be removed.
    - ❖ These cloned home directories are available on the Hawk login nodes at:
        - Cardiff home directories
            - ❖ ***/migrate/hpcw-cf/HPCW_USERID***
        - Swansea home directories
            - ❖ ***/migrate/hpcw-sw/HPCW_USERID***
        - Bangor home directories
            - ❖ ***/migrate/hpcw-ba/HPCW_USERID***
- New home directory quotas for all are as follows:
    - ❖ 50GB per user and 100GB per project share on Hawk
    - ❖ 100GB per user and 200GB per project share on Sunbird
    - ❖ extensions are quickly available on justified request.
- *Project Shared Areas* are a way for users of a particular project to share data. They are separate from user home directories.

- ❖ If a Project Shared area does not exist for your project and one would be useful, please request from Support.

- We can also create *Collaboration Shared Areas* that cross project and institutional boundaries as separate areas from home directories for data sharing among multiple users.
  - ❖ Please contact Support to discuss if this would be useful.
- As with HPC Wales, no backup of data in home directories is taken, please keep necessary data backed up elsewhere.
- The scratch file system will see automatic cleanup of data that is not accessed for 60 days.
  - ❖ Exceptions available by justified request to Support.

8.1.3 Jobs & Applications

- Once successfully migrated, we request that you no longer submit jobs on the HPC Wales systems as they are increasingly contended whilst we await the launch of the other new system.
- At this stage, a slowly-increasing number of the most utilised application codes have been re-built in an optimized form on the new systems.
  - ❖ These can be seen and accessed from the default output of ***module avail*.**
- For other codes that have _not_ been re-built in optimal form at launch, the HPCW software stack is made available by first performing ***module load hpcw***.
  - ❖ Once the ***hpcw*** module is loaded, the HPCW software stack is displayed by ***module avail*** and modules are loaded in the usual way using ***module load <modulename>***.
  - ❖ The large majority of the old HPCW software stack will function on Hawk just fine.
    - A few things won't. If you find one, please highlight to Support and a new solution will be prioritised.
- We will monitor usage of old codes in order to prioritise those to be re-built in optimal form for the new system, and also those which will at some point be decommissioned due to zero usage.
- Old job submission scripts should be modified to make the most of the new system's greater processor per node count and also target the right partition (queue).
  - ❖ Where previous systems had 12 or 16 processor cores per node, Hawk has ***40***.
    - It is therefore more important to use Hawk's nodes efficiently as proportionally more resource can be easily wasted.
    - Be aware to update the Slurm directive **#SBATCH –tasks-per-node** in migrated submission scripts.
  - ❖ To better reflect the wide diversity of jobs and improve the user experience, partition naming and use has been re-worked. Please see the hardware descriptions for Hawk & Sunbird, output of ***sinfo*** and the below.
    - Standard parallel jobs to the default **compute** partition (Hawk & Sunbird).
    - GPU accelerated tasks to the **gpu** partition (Hawk & Sunbird).
    - High-memory tasks to the **highmem** partition (Hawk only).
    - Serial / high-throughput tasks to the **htc** partition (Hawk only).

- Small and short development tasks (up to 80 processor cores, 60 minutes runtime) to the **dev** partition (Hawk only).

A newly refreshed training tarball, full of example jobs across a variety of applications is available. Please see here.

8.2 Rapid Start for Raven Users

Portal URL: https://portal.supercomputing.wales/index.php/rapid-start-for-raven-users-migrating-to-scw-hawk/

For users of the ARCCA Raven services who are migrating to the SCW service, please be aware of these important notes in order to get up-and-running as quickly & effectively as possible.

8.2.1 User Credentials

- These have changed to be linked to your institutional account.
- The method to access your new credentials and get started with Hawk is covered on the Getting Access page – please follow it closely.
- Contact Support if you have any issues.

8.2.2 Projects

- Existing projects and their memberships from Raven have been migrated to Hawk.
- You can apply for new projects through the MySCW system, see Getting Access.
- It will soon become necessary to specify which of your project memberships to account a compute job against.

8.2.3 Data

- Home directories from Raven have been cloned to a read-only area on Hawk for quick access.
  - ❖ These will remain for approximately 6 months only and then be removed.
  - ❖ They are being re-synchronised from Raven every day.
  - ❖ These cloned home directories are available on the Hawk login nodes at:***/migrate/raven/RAVEN_USERID***
- New home directory quotas for all are 50GB per user and 100GB per project share – extensions are quickly available on justified request.
- *Project Shared Areas* are a way for users of a particular project to share data. They are separate from user home directories.
  - ❖ If a Project Shared area does not exist for your project and one would be useful, please request from Support.
- We can also create *Collaboration Shared Areas* that cross project and institutional boundaries as separate areas from home directories for data sharing among multiple users.
  - ❖ Please contact Support to discuss if this would be useful.
- The scratch file system will see automatic cleanup of data that is not accessed for 60 days.
  - ❖ Exceptions available by justified request to Support.
- Raven is still an active system.
- Cardiff User home directories on Hawk are currently being cloned to a backup system, but there are no historical archives kept thereof. This backup is done on a best efforts basis.

8.2.4 Gluster Storage

- Gluster storage will be available as it was on Raven via mountpoints on the login nodes.
- Access to Gluster storage is via membership of relevant groups on Hawk. These groups are mapped from the Cardiff university groups used for Gluster storage.
- To simplify administration and keep things better organised all mounts are under /gluster.
- N.B. /gluster/neurocluster contains its own group of mountpoints.
- Users are expected to copy data to scratch for processing – Gluster mountpoints will not be accessible to cluster nodes due to network routing.

8.2.5 Jobs & Applications

- Once migrated and you have validated the operation & correctness of software you use and all is doing as you would expect, we request that you no longer submit jobs on Raven.
- At this stage, a slowly-increasing number of the most utilised application codes have been re-built in an optimized form on the new systems.
  - ❖ These can be seen and accessed from the default output of ***module avail***.
- For other codes that have _not_ been re-built in optimal form at launch, the Raven software stack is made available by first performing ***module load raven***.
  - ❖ Once the **raven** module is loaded, the Raven software stack is displayed by **module avail** and modules are loaded in the usual way using **module load <modulename>**.
  - ❖ The large majority of the Raven software stack will function on Hawk just fine.
  - ❖ A few things won't. If you find one, please highlight to Support and a new solution will be prioritised.
- We will monitor usage of old codes in order to prioritise those to be re-built in optimal form for the new system, and also those which will at some point be decommissioned due to zero usage.
- Job submission scripts from Raven will need to be modified to use the Slurm scheduler deployed on Hawk.
  - ❖ The PBS Pro to Slurm Migration Reference (see section 17) highlights the normal changes needed as part of this.
  - ❖ For more complex job scripts, please see the other documentation on this site regarding interactive use, array jobs, etc.
- To make the most of the new system's greater processor per node count and also target the right partition (queue).
  - Where previous systems had 12 or 16 processor cores per node, Hawk has ***40***.
    - ❖ It is therefore more important to use Hawk's nodes efficiently as proportionally more resource can be easily wasted.
    - ❖ Be aware to correctly populate the Slurm directive ***#SBATCH –tasks-per-node*** in migrated submission scripts.
  - To better reflect the wide diversity of jobs and improve the user experience, partition naming and use has been re-worked. Please see the output of ***sinfo*** and the below.
    - ❖ Standard parallel jobs to the default ***compute*** partition.
    - ❖ High-memory tasks to the ***highmem*** partition.
    - ❖ GPU accelerated tasks to the ***gpu*** partition.

- ❖ Serial / high-throughput tasks to the ***htc*** partition.
- ❖ Small and short development tasks (up to 40 processor cores, 30 minutes runtime) to the ***dev*** partition.

- A newly refreshed training tarball, full of example jobs across a variety of applications is available. Please see here.
- Please don't hesitate to contact us with any questions, issues or comments.

8.3 Rapid Start – Other HPC Systems

If you're familiar with and migrating from other HPC systems, these quick reference documents will help you.

8.4 PBS Pro Migration Quick Ref

Portal URL: https://portal.supercomputing.wales/index.php/index/slurm/pbs-pro-to-slurm-very-quick-reference/

8.4.1 Commands

PBS	Slurm	Description
qsub script_file	sbatch script_file	Submit a job from script_file
qsub -I	salloc [options]	Request an interactive job
qdel 123	scancel 123	Cancel job 123
qstat -u [username]	squeue	List user's pending and running jobs
qstat -f 123 qstat -fx 123	scontrol show job 123	Show job details -x in PBS to show finished job
qstat queue_name	sinfo -s	Cluster status with partition (queue) list With '-s' a summarised partition list, which is shorter and simpler to interpret.

8.4.2 Job Specification

PBS	Slurm	Description
#QSUB	#SBATCH	Scheduler directive
-q queue_name	-p queue_name	Queue to 'queue_name'

PBS	Slurm	Description
-l select=4:ncpus=16 (request 4 nodes and request that each node has the property of 16 cpus)	-n 64	Processor count of 64
-l walltime=h:mm:ss	-t [minutes] or -t [days-hh:mm:ss]	Max wall run time
-o file_name	-o file_name	STDOUT output file
-e file_name	-e file_name	STDERR output file
-N job_name	--job-name=job_name	Job name
-l place=excl	--exclusive	Exclusive node usage for this job - i.e. no other jobs on same nodes
-l mem=1gb	--mem-per-cpu=128M or --mem-per-cpu=1G	Memory requirement
-l nodes=x:ppn=16	--tasks-per-node=16	Processes per node
-P proj_code	--account=proj_code	Project account to charge job to
-t 1-10	--array=array_spec	Job array declaration

8.4.3 Job Environment Variables

PBS	Slurm	Description
$PBS_JOBID	$SLURM_JOBID	Job ID
$PBS_O_WORKDIR	$SLURM_SUBMIT_DIR	Submit directory
	$SLURM_ARRAY_JOB_ID	Job Array Parent
$PBS_ARRAYID	$SLURM_ARRAY_TASK_ID	Job Array Index
$PBS_O_HOST	$SLURM_SUBMIT_HOST	Submission Host
$PBS_NODEFILE	$SLURM_JOB_NODELIST	Allocated compute nodes

PBS	Slurm	Description
	$SLURM_NTASKS (mpirun can automatically pick this up from Slurm, it does not need to be specified)	Number of processors allocated
$PBS_QUEUE	$SLURM_JOB_PARTITION	Queue job is running in

Much more detail available in the Slurm documentation.

8.5 LSF Migration Quick Ref

8.5.1 Commands

LSF	Slurm	Description
bsub < script_file	sbatch script_file	Submit a job from script_file
bkill 123	scancel 123	Cancel job 123
bjobs	squeue	List user's pending and running jobs
bqueues	sinfo sinfo -s	Cluster status with partition (queue) list With '-s' a summarised partition list, which is shorter and simpler to interpret.

8.5.2 Job Specification

LSF	Slurm	Description
#BSUB	#SBATCH	Scheduler directive
-q queue_name	-p queue_name	Queue to 'queue_name'
-n 64	-n 64	Processor count of 64
-W [hh:mm:ss]	-t [minutes] or -t [days-hh:mm:ss]	Max wall run time
-o file_name	-o file_name	STDOUT output file
-e file_name	-e file_name	STDERR output file
-J job_name	--job-name=job_name	Job name

LSF	Slurm	Description
-x	--exclusive	Exclusive node usage for this job - i.e. no other jobs on same nodes
-M 128	--mem-per-cpu=128M or --mem-per-cpu=1G	Memory requirement
-R "span[ptile=16]"	--tasks-per-node=16	Processes per node
-P proj_code	--account=proj_code	Project account to charge job to
-J "job_name[array_spec]"	--array=array_spec	Job array declaration

8.5.3 Job Environment Variables

LSF	Slurm	Description
$LSB_JOBID	$SLURM_JOBID	Job ID
$LSB_SUBCWD	$SLURM_SUBMIT_DIR	Submit directory
$LSB_JOBID	$SLURM_ARRAY_JOB_ID	Job Array Parent
$LSB_JOBINDEX	$SLURM_ARRAY_TASK_ID	Job Array Index
$LSB_SUB_HOST	$SLURM_SUBMIT_HOST	Submission Host
$LSB_HOSTS $LSB_MCPU_HOST	$SLURM_JOB_NODELIST	Allocated compute nodes
$LSB_DJOB_NUMPROC	$SLURM_NTASKS (mpirun can automatically pick this up from Slurm, it does not need to be specified)	Number of processors allocated
	$SLURM_JOB_PARTITION	Queue

Much more detail available in the Slurm documentation.

8.6 SGE Migration Quick Ref

8.6.1 Commands

SGE	Slurm	Description
qsub script_file	sbatch script_file	Submit a job from script_file
qdel 123	scancel 123	Cancel job 123
qstat	squeue	List user's pending and running jobs
qhost -q	sinfo sinfo -s	Cluster status with partition (queue) list With '-s' a summarised partition list, which is shorter and simpler to interpret.

8.6.2 Job Specification

SGE	Slurm	Description
#$	#SBATCH	Scheduler directive
-q queue_name	-p queue_name	Queue to 'queue_name'
-pe 64	-n 64	Processor count of 64
-l h_rt=[ss]	-t [minutes] or -t [days-hh:mm:ss]	Max wall run time
-o file_name	-o file_name	STDOUT output file
-e file_name	-e file_name	STDERR output file
-J job_name	--job-name=job_name	Job name
-l exclusive	--exclusive	Exclusive node usage for this job - i.e. no other jobs on same nodes
-l mem_free=128M or -l mem_free=1G	--mem-per-cpu=128M or --mem-per-cpu=1G	Memory requirement
Set by parallel environment config	--tasks-per-node=16	Processes per node

SGE	Slurm	Description
-P proj_code	--account=proj_code	Project account to charge job to
-t "[array_spec]"	--array=array_spec	Job array declaration

8.6.3 Job Environment Variables

SGE	Slurm	Description
$JOBID	$SLURM_JOBID	Job ID
$SGE_O_WORKDIR	$SLURM_SUBMIT_DIR	Submit directory
$JOBID	$SLURM_ARRAY_JOB_ID	Job Array Parent
$SGE_TASK_ID	$SLURM_ARRAY_TASK_ID	Job Array Index
$SGE_O_HOST	$SLURM_SUBMIT_HOST	Submission Host
cat $PE_HOSTLIST (this is a file)	$SLURM_JOB_NODELIST	Allocated compute nodes
$NSLOTS	$SLURM_NTASKS (mpirun can automatically pick this up from Slurm, it does not need to be specified)	Number of processors allocated
$QUEUE	$SLURM_JOB_PARTITION	Queue

Much more detail available in the Slurm documentation.

9 External Links

Here we provide links to interesting documentation and sites that are not ours - so, we're not responsible for them:

Official Slurm Docs

ARM Performance Reports User Guide

ARM Forge User Guide (MAP and DDT)

10 User Support and Contact

We describe below the process for contacting us through Submission of a Support Ticket, along with a summary of our FAQ page for the answers to some frequently asked questions.

10.1 Submit Support Ticket

Portal URL: https://portal.supercomputing.wales/index.php/submit-support-ticket/

1. For Cardiff University users of the system, please email arcca-help@cardiff.ac.uk to contact the ARCCA helpdesk.
2. For existing Supercomputing Wales users, please email support@supercomputingwales.ac.uk and include a description of the issue or fill in the following submission form.

Please Note: *If you are having trouble resetting your password at the command-line, please use the MySCW interface to do so at http://my.supercomputing.wales*

10.1.1 Ticket Submission Form

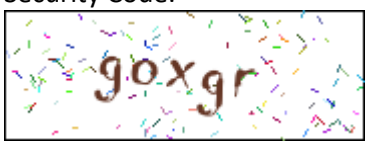
User name*

Contact Email*

Summary*

Description*

Security Code:



Please enter the security code:

Submit

10.2 Frequently Asked Questions

Portal URL: https://portal.supercomputing.wales/index.php/index/slurm/slurm-faq/

Below you will find frequently asked questions about Slurm on Supercomputing Wales. We will be continually updating this page.

Q1. How do I create a machines file for mpirun?

Intel MPI and OpenMPI will automatically pick up the numbers of processors and the allocated hosts when *mpirun* is called within a running job script allocation. This is performed through Slurm setting environment variables appropriately that *mpirun* picks up. However, if you require to use an MPI that does not support this mechanism, then an allocated machines file is easily created by using the *srun* command inside the job script. *srun* is a Slurm command that runs a specified shell command against all or a selection of allocated nodes. When specified without any arguments it will run on all allocated processors, hence we can create a machines file easily:

```
srun hostname &gt;&gt; machines_file
```

srun also supports a large selection of specification semantics for limiting the target nodes. See the Slurm documentation for further details.

Q2. How do get a list of processors/nodes that have been allocated to my job run?

See the answer to Q1.

Q3. How do I perform graphical interface X11 forwarding in Slurm jobs?

See section 5.6.1.

Q4. Can I access allocated nodes interactively during job runs for debugging?

You are able to *ssh* to allocated machines whilst your allocation remains in-place (i.e. your job is running there). Users cannot *ssh* to machines they are not allocated to and will be forcibly logged out of any such sessions when their jobs finish.

Q5. What queues are there now? 'sinfo' is confusing

'sinfo' is a little verbose as it reports on the node status within the queues (partitions) in addition to the queue presence itself. Try to run 'sinfo -s' for a shorter & more succinct report, for example:

```
sinfo -s
PARTITION                AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
compute*                    up 3-00:00:00      133/1/0/134  ccs[0001-0134]
xcompute                    up 3-00:00:00      133/1/0/134  ccs[0001-0134]
```

```
compute_amd                  up 3-00:00:00      18/46/0/64  cca[0001-0064]
xcompute_amd                 up 3-00:00:00      18/46/0/64  cca[0001-0064]
highmem                      up 3-00:00:00       22/3/1/26  ccs[1001-1026]
xhighmem                     up 3-00:00:00       22/3/1/26  ccs[1001-1026]
gpu                          up 2-00:00:00        7/6/0/13  ccs[2001-2013]
xgpu                         up 2-00:00:00        7/6/0/13  ccs[2001-2013]
gpu_v100                   down 2-00:00:00       0/15/0/15  ccs[2101-2115]
xgpu_v100                    up 2-00:00:00       0/15/0/15  ccs[2101-2115]
htc                          up 3-00:00:00       64/1/0/65  ccs[0120-0134,1009-1026,2008-
2013,3001-3026]
xhtc                         up 3-00:00:00       26/0/0/26  ccs[3001-3026]
dev                          up    1:00:00         0/2/0/2  ccs[0135-0136]
xdev                         up    6:00:00         0/2/0/2  ccs[0135-0136]
c_compute_chemy1             up 3-00:00:00         0/1/0/1  ccs9001
c_compute_chemy_webmo        up 3-00:00:00         0/1/0/1  ccs9001
c_compute_mdi1               up 3-00:00:00         0/2/0/2  ccs[9002-9003]
c_gpu_comsc1                 up 3-00:00:00         0/1/0/1  ccs9201
c_gpu_diri1                  up 3-00:00:00         0/1/0/1  ccs9202
c_compute_ligo1            down 3-00:00:00       0/35/0/35  ccs[8001-8035]
c_gpu_ligo1                down 3-00:00:00         0/1/0/1  ccs8201
c_vhighmem_dri1              up 10-00:00:0         0/2/0/2  ccs[9010-9011]
c_highmem_dri1               up 10-00:00:0         0/9/0/9  ccs[9012-9020]
c_compute_dri1               up 10-00:00:0         8/1/0/9  ccs[9021-9029]
c_compute_neuro1             up 14-00:00:0         1/7/0/8  ccs[9030-9037]
c_compute_neuro1_long        up 14-00:00:0         1/3/0/4  ccs[9030-9033]
c_compute_neuro1_smrtlink    up 14-00:00:0         0/4/0/4  ccs[9034-9037]
c_compute_chemy2             up 3-00:00:00         2/4/0/6  ccs[9004-9009]
c_compute_ligo2            down 3-00:00:00      0/41/20/61  ccr[0000-0060]
c_compute_huygens            up 3-00:00:00        8/4/0/12  ccr[0061-0072]
c_compute_wgp                up 3-00:00:00        6/6/0/12  ccr[0073-0084]
c_compute_wgp1               up 3-00:00:00        6/6/0/12  ccr[0073-0084]
```

Q6. When will my job(s) run?

Please see this section in the Running & Monitoring Guide.

Q7. Why do I see non-editable Bash functions in my ksh environment when running under Slurm?

Ksh is sometimes required for some software to run (e.g. The Unified Model) and with the recent Bash update to fix the Shellshock vulnerability has created a situation where ksh inherits the environment variable beginning with BASH_FUNC when ksh is started from within a bash shell but it is unable to clear or modify the variable. This causes problems if you read the environment and try and recreate it elsewhere (such as within a mpirun wrapper script). This is a known bug and is reported in https://bugzilla.redhat.com/show_bug.cgi?id=1147645.

Q8. I used to run many hundreds or thousands of small jobs on LSF, but Slurm limits the number I can have. What can I do?

The previous scheduler made it too easy for users to abuse the system and/or cause to operate inefficiently by submitting many many jobs. Slurm enforces a Quality of Service settings that limits the number of jobs any single user can have running and the number they can have submitted. Attempts to submit further jobs will result in errors. Through the use of GNU parallel and Slurm's srun command, we can now create (with a little bit of script) a far more efficient configuration for handling such use-cases that protects the service and also will result in quicker results being obtained (due to less job scheduling overhead). Please see Batch Submission of Serial Jobs for Parallel Execution.

Q9. When submitting a job, I receive an error about inefficient tasks-per-node specification. What do I do?

The old HPCW and Raven systems each had 12 or 16 processor cores per node. The new SCW systems have a similar overall processor count, but consisting of nodes with 40 processor cores per node. Hence, the overall node count is significantly lower, which raises the risk that partially-populated nodes (i.e. those on which all processor cores are not allocated to jobs) are utilised exclusively (through the Slurm exclusive flag) and that large amounts of processor cores are unallocated as a result and thus overall system occupancy & efficiency would be impacted. Many scripts migrating from the previous systems will specify tasks-per-node as 12 or 16, so this is a significant potential cause of this. This submission-time error alerts users to this, so you should follow its instruction – i.e. either update your jobs or, if you do know what you are doing, then set an appropriate environment variable to enable override of this protection mechanism.

Q10. How do I acknowledge SCW in publications?

It is invaluable to the SCW project that this happens, so thank you and please use this text:

We acknowledge the support of the Supercomputing Wales project, which is part-funded by the European Regional Development Fund (ERDF) via Welsh Government.

11 Appendix I. Intel Compiler Flags

Code Optimisations	
Compiler FLAG	
	Unless specified, -O2 is assumed.
-O0	Disables all optimizations
-O1	Enables optimizations for speed and disables some optimizations that increase code size and affect speed.
-O2	Enables optimizations for speed. This is the generally recommended optimization level. Vectorization is enabled at O2 and higher levels.
-O3	Performs O2 optimizations and enables more aggressive loop transformations such as Fusion, Block-Unroll-and-Jam, and collapsing IF statements.
-fast	Fast is a collection of compiler flags which, for the latest version of the compiler, expands to "-ipo, -mdynamic-no-pic, -O3, -no-prec-div, -fp-model fast=2".
-no-prec-div	This option enables optimizations that give slightly less precise results than full IEEE division i.e. division is transformed into multiplication by the reciprocal of the denominator.
	Processor Optimisations
	Unless specified, SSE level 2 is assumed.
-xhost	Produces code optimised to run on the processor on which it was compiled.
-axCOMMON-AVX512	May generate Intel(R) Advanced Vector Extensions 512 (Intel(R) AVX-512) Foundation instructions, Intel(R) AVX-512 Conflict Detection instructions, as well as the instructions enabled with CORE-AVX2.
-axCORE-AVX512	May generate Intel(R) Advanced Vector Extensions 512 (Intel(R) AVX-512) Foundation instructions, Intel(R) AVX-512 Conflict Detection instructions, Intel(R) AVX-512 Doubleword and Quadword instructions, Intel(R) AVX-512 Byte and Word instructions and Intel(R) AVX-512 Vector Length extensions, as well as the instructions enabled with CORE-AVX2.
-axCORE-AVX2	May generate Intel(R) Advanced Vector Extensions 2 (Intel(R) AVX2), Intel(R) AVX, SSE4.2, SSE4.1, SSE3, SSE2, SSE, and SSSE3 instructions for Intel(R) processors.
-axAVX	May generate Intel(R) Advanced Vector Extensions (Intel(R) AVX), Intel(R) SSE4.2, SSE4.1, SSE3, SSE2, SSE, and SSSE3 instructions for Intel(R) processors.
–axsse4.2	Produces code optimised to run on SSE level 4.2 capable processors.
	Inter Procedural Optimisation
	Unless specified, IPO is disabled.

	Code Optimisations
-ipo	This option enables interprocedural optimization between files. When you specify this option, the compiler performs inline function expansion for calls to functions defined in separate files.
	Profile Guided Optimisation
	Unless specified, PGO is disabled.
	PGO consists of three phases: Phase one is to build an instrumented executable; Phase two is to run that instrumented executable one or more times (on a range of typical data sets, or on a range of typical parameters) to generate one or more typical execution profiles; Phase three is to build an optimised executable based on information from the generated execution profiles.
-prof-gen	This option creates an instrumented executable (phase one).
-prof-use	This option creates an optimised executable (phase three).
	Other
-static	This option prevents linking with shared libraries, causing executables to link all libraries statically.

12 Appendix II. Common Linux Commands

The most common Linux commands are shown in the table below:

Command	Description
cat [filename]	Display file's contents to the standard output device (usually your monitor).
cd /directorypath	Change to directory.
chmod [options] mode filename	Change a file's permissions.
chown [options] filename	Change who owns a file.
clear	Clear a command line screen/window for a fresh start.
cp [options] source destination	Copy files and directories.
date [options]	Display or set the system date and time.
df [options]	Display used and available disk space.
du [options]	Show how much space each file takes up.
file [options] filename	Determine what type of data is within a file.
find [pathname] [expression]	Search for files matching a provided pattern.
grep [options] pattern [filesname]	Search files or output for a particular pattern.
kill [options] pid	Stop a process. If the process refuses to stop, use kill -9 pid.
less [options] [filename]	View the contents of a file one page at a time.
ln [options] source [destination]	Create a shortcut.
locate filename	Search a copy of your filesystem for the specified filename.
lpr [options]	Send a print job.
ls [options]	List directory contents.
man [command]	Display the help information for the specified command.
mkdir [options] directory	Create a new directory.
mv [options] source destination	Rename or move file(s) or directories.
passwd [name [password]]	Change the password or allow (for the system administrator) to change any password.
ps [options]	Display a snapshot of the currently running processes.
pwd	Display the pathname for the current directory.
rm [options] directory	Remove (delete) file(s) and/or directories.
rmdir [options] directory	Delete empty directories.
ssh [options] user@machine	Remotely log in to another Linux machine, over the network. Leave an ssh session by typing **exit**.
su [options] [user [arguments]]	Switch to another user account.
tail [options] [filename]	Display the last *n* lines of a file (the default is 10).
tar [options] filename	Store and extract files from a tarfile (.tar) or tarball (.tar.gz or .tgz).
top	Displays the resources being used on your system. Press q to exit.

Command	Description
touch filename	Create an empty file with the specified name.
who [options]	Display who is logged on.