Flexible Slurm Configuration for Large-Scale HPC

Mr Steven Robson
EPCC, The University of Edinburgh
Edinburgh, UK
s.robson@epcc.ed.ac.uk

Mr Kieran Leach
EPCC, The University of Edinburgh
Edinburgh, UK
k.leach@epcc.ed.ac.uk

Dr Stephen Booth
EPCC, The University of Edinburgh
Edinburgh, UK
s.booth@epcc.ed.ac.uk

Mr Greg Blow
EPCC, The University of Edinburgh
Edinburgh, UK
g.blow@epcc.ed.ac.uk

Mr Maciej Hamczyk
EPCC, The University of Edinburgh
Edinburgh, UK
m.hamczyk@epcc.ed.ac.uk

Mr Philip Cass
EPCC, The University of Edinburgh
Edinburgh, UK
p.cass@epcc.ed.ac.uk

***Abstract*—EPCC operates a variety of services including ARCHER2, commissioned by UK Research and Innovation (UKRI) on behalf of the UK Government as the UK's Tier-1 supercomputing service, and Cirrus, a UKRI Tier-2 HPC service.**

EPCC had historically operated services using the PBS Pro scheduler, including HECToR and ARCHER, the previous national Tier-1 services, and Cirrus until mid 2020. In 2020 EPCC began to host ARCHER2, operating with the Slurm scheduler, as well as conducting a rebuild and upgrade of Cirrus, during which Slurm was deployed in place of PBS Pro.

Since 2020, EPCC have developed a number of approaches for management and configuration of the Slurm scheduler which have significantly improved service manageability and user experience. We will describe the key features of these approaches in this paper.

We will present our conclusion from the past several years working with Slurm, considering that:

- **Slurm has proved valuable in its configurability and flexibility as a scheduler;**
- **Scheduler configuration setup can be usefully shared across different services even though they have different requirements and levels of heterogeneity**
- **Automation can considerably reduce the effort required by staff supporting Slurm**

***Index Terms*—scheduling, HPC, service management**

I. Background

EPCC operates a variety of services including ARCHER2, commissioned by UK Research and Innovation (UKRI) on behalf of the UK Government as the UK's Tier-1 national supercomputing service, and Cirrus, a UKRI Tier-2 national HPC service.

EPCC has historically operated services including HECToR and ARCHER, the previous UK national Tier-1 supercomputing services.

A. ARCHER2

The ARCHER2 system is an HPE Cray EX supercomputer with an estimated peak performance of 28 Pflop/s. The machine has 5,860 compute nodes, each with dual AMD EPYC 7742 64-core processors at 2.25GHz, giving 750,080 cores in total connected by a Slingshot-10 interconnect. ARCHER2 is the successor system to ARCHER, a 4,920-node Cray XC30 system which was also operated and supported by EPCC. These systems have been managed and financed by the Engineering and Physical Science Research Council (EPSRC) and UK Research and Innovation (UKRI).

In operating and supporting both these services, EPCC has acted in both the Service Provision (SP) and Computational Science and Engineering (CSE) roles. Under the SP role, EPCC is responsible for system management and administration as well as the operation of the Service Desk. Under the CSE role, EPCC is responsible for deploying application software not included in the programming environment supplied by HPE as well as for assisting users with application software development and management, providing training, administering funding calls for software development projects, and running an outreach programme. These responsibilities are in addition to hosting the ARCHER2 service at EPCC's Advanced Computing Facility (ACF) data centre.

B. Cirrus

The Cirrus system is an HPE SGI 8600 system operated as a UKRI Tier-2 service. Cirrus operates with a mix of 352 CPU nodes (each with 2x Intel Xeon E5-2695 36 core at 2.1 GHz) and 38 GPU nodes (each with 4x Nvidia Tesla V100-SXM2-16GB) connected by a mix of FDR and HDR Infiniband.

C. Slurm

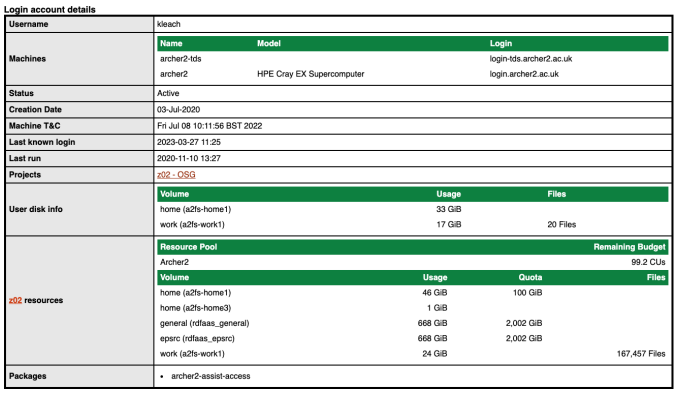
Slurm is a scheduler originally announced by Laurence Livermore National Laboratory in 2003 [1]. Slurm today has a wide range of contributors and is maintained by SchedMD [2]. EPCC first adopted Slurm for a major system with the rebuild of the Cirrus system in 2020.

Cirrus was originally deployed using the PBS Pro scheduler. During a 2020 rebuild and upgrade of the system Cirrus was migrated to Slurm. This was chosen in order to be in line with the anticipated arrival of ARCHER2 using Slurm.

ARCHER2 was delivered by HPE with Slurm in place of PBS Pro which had historically supported Tier-1 national services including HECToR and ARCHER.

D. SAFE

EPCC develops and operates a service management web application known as SAFE [3]. Details of job accounting



Login account details

Username	kleach			
Machines	**Name**	**Model**	**Login**	
	archer2-tds		login-tds.archer2.ac.uk	
	archer2	HPE Cray EX Supercomputer	login.archer2.ac.uk	
Status	Active			
Creation Date	03-Jul-2020			
Machine T&C	Fri Jul 08 10:11:56 BST 2022			
Last known login	2023-03-27 11:25			
Last run	2020-11-10 13:27			
Projects	z02 - OSG			
User disk info	**Volume**	**Usage**	**Files**	
	home (a2fs-home1)	33 GiB		
	work (a2fs-work1)	17 GiB	20 Files	
z02 resources	**Resource Pool**			**Remaining Budget**
	Archer2			99.2 CUs
	Volume	**Usage**	**Quota**	**Files**
	home (a2fs-home1)	46 GiB	100 GiB	
	home (a2fs-home3)	1 GiB		
	general (rdfaas_general)	668 GiB	2,002 GiB	
	epsrc (rdfaas_epsrc)	668 GiB	2,002 GiB	
	work (a2fs-work1)	24 GiB		167,457 Files
Packages	• archer2-assist-access			

Fig. 1. A screenshot of SAFE showing an individual user account on ARCHER2. Of particular note here is the "Remaining Budget" denoted in CUs. 1 CU is equivalent to 1 node hour.

from the Slurm scheduler are uploaded to SAFE. SAFE is also used by users to create and manage their machine accounts, and by project managers to allocate and report on project resources. Resources allocated and managed in this way include disk space and compute node time. The SAFE was originally developed as the "Service Administrative Function" or SAF for the HPCx service which was operated by STFC Daresbury and EPCC starting in December of 2002. SAFE has evolved and supported HPCx, HECToR, ARCHER and now ARCHER2 along with a variety of other services. Throughout this time SAFE has been under constant development. The current production database was originally configured for HECToR and the oldest registered account dates to December of 2006. This provides a significant degree of continuity to users of UKRI national services.

E. Rundeck

Rundeck is an application supporting automation of common system administration tasks [4]. It can offer access to an array of user defined administrative scripts and actions both via a web interface and API access. At EPCC Rundeck is used to automate the vast majority of system administration tasks including, but not limited to:

- User creation and management within Identity Management Systems such as OpenLDAP and FreeIPA;
- directory creation and management;
- file system quota management; and
- management of user definition and budget within schedulers.

F. Motivation

In working to configure and manage Slurm on systems including ARCHER2 and Cirrus, EPCC have developed a number of approaches for management and configuration of the Slurm scheduler which have significantly improved service manageability and the user experience. We describe a number of these approaches below.

II. SAFE Integration and Budget Management

In order to improve efficiency and reduce the required staff effort EPCC has for some time, across both the HECToR and ARCHER services, automated the deployment of budgets and the collection of accounting data from and to the SAFE. Under PBS Pro this was a relatively simple process of budget files being copied into place on a regular basis and accounting files being copied out on the same basis. This provided all the integration that was, at the time, required.

The move to Slurm has required us to look at this again and develop a very different approach. In order to properly manage users it is now necessary that user accounts be created in Slurm and that these be properly managed. It is also no longer possible to simply copy files in and out to provide and access budget and accounting data. We discuss below how we have approached the automation of user and data management of Slurm via the SAFE for services including ARCHER2 and Cirrus

A. Integration of Slurm and SAFE

During the HECToR and ARCHER services SAFE automation was conducted by:

- Automated copying of files using scp; or
- a perl based script which would access tickets over http and action these. This required the sysadmin team to manually run the script for each ticket or batch of tickets

During early planning for the ARCHER2 service the decision was made to move all automation to the Rundeck application discussed previously. A series of tickets were defined in Rundeck to conduct various actions. All Rundeck tickets for the ARCHER2 service which carry out direct actions, including those interacting with Slurm, are intended to be:

- Atomic: in each case individual actions are created as tickets; and
- idempotent: tickets should be re-runnable without adverse impact or failure.

"Macro" tickets are then defined which call the tickets directly interacting with ARCHER2. These tickets can call several individual "micro" tickets in sequence and handle the communication of data to and from the SAFE.

In order to run tickets the following process takes place:

1) A user change or internal SAFE process occurs which requires a change within the Slurm environment.
2) The SAFE validates all user input.
3) The SAFE connects to Rundeck over HTTP on a private network.
4) Via a webhook notification the SAFE calls the relevant macro ticket, providing whichever variables are required.
5) Rundeck actions the appropriate micro ticket or tickets which first validate the input provided where appropriate.
6) Based upon the node filter Rundeck uses ssh to access to the relevant node - in this case an ARCHER2 login node.

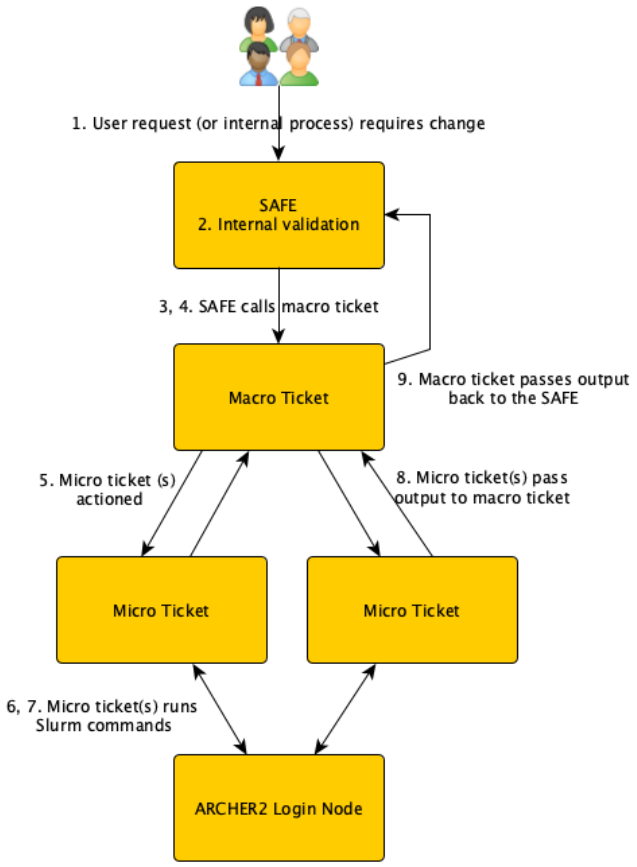


Fig. 2. A diagram showing the process by which Rundeck tickets are run by the SAFE

7) The appropriate commands are then run based upon the script in each micro ticket (the Rundeck user being configured as a Slurm administrator).
8) The micro ticket then either passes back output from the actions conducted or handles any errors appropriately.
9) Output is then passed back to the macro ticket which communicates the result back to the SAFE again over HTTP via a webhook notification.

This process can be seen outlined in figure 2. In this way the following actions for Slurm are managed on ARCHER2:

- Creation and removal of users;
- creation and removal of budgets/accounts;
- the locking and unlocking of budgets/accounts;
- the creation of reservations.

As an example both the macro and micro tickets for the creation of reservations can be found at [5].

B. Data Extraction from Slurm to SAFE

In addition to the on-demand processes above, the SAFE and Rundeck also interact in order to extract accounting data from Slurm on a regular basis. In order to achieve this Rundeck activates tickets on a daily basis which ssh to the ARCHER2 login node, collect job and subjob data and post this back to the SAFE via HTTPS to a data ingest endpoint.

This allows the SAFE to store Slurm accounting persistently and to support accurate reporting to users of accounting data. This data includes details of jobs run and energy used and is available both to individual users and the principal investigators of each project. This accounting data is also used to manage budgets as discussed below.

C. Automated Implementation of Budgeting Controls

On the ARCHER2 and Cirrus services projects are assigned allocations for compute time along with storage allocations. Time allocations run for a finite period of time, and a non-overlapping sequence of allocations can be configured for a project. The currently active allocations control the state of budgets implemented in Slurm. These budgets are assigned to the project as a whole and then designated project managers may optionally assign sub-budgets to individual users or groups of users, and move time between sub-budgets.

For each budget and sub-budget defined in this way an account is created in Slurm with the appropriate users associated to said account. Users then run work against the relevant account and this is reported to the SAFE through the data extraction process discussed above.

When the SAFE identifies, through the data uploaded, that the budget for an account has been exceeded, a Rundeck ticket is actioned to lock the relevant account. Should further compute time be added (or a new allocation become active) an equivalent ticket is actioned to unlock the account in question.

Locking and unlocking are achieved by setting the account's MAXTresMins attribute which represents the number of minutes for which a user can use a given TRES resource for each job they submit. This is set to be zero when locking an account and -1 (infinite) when an account is unlocked. This can be managed independently for both CPU and GPU resources, as shown below.

```
#Lock the CPU account for budget01
sacctmgr -i modify account where cluster=
    archer2 Account=budget01 set maxtresmins=
    cpu=0
#Lock the GPU account for budget01
sacctmgr -i modify account where cluster=
    archer2 Account=budget01 set maxtresmins=
    gres/gpu=0
#Unlock the CPU account for budget01
sacctmgr -i modify account where cluster=
    archer2 Account=budget01 set maxtresmins=
    cpu=-1
#Unlock the GPU account for budget01
sacctmgr -i modify account where cluster=
    archer2 Account=budget01 set maxtresmins=
    gres/gpu=-1
```

III. Approaches for System Manageability

A. Preconfigured QoS

For much of the early life of the ARCHER2 service it was time consuming, and required a service outage, to make any changes to the Slurm configuration files. This restriction meant that the definition of partitions and QoS permitted to access partitions could not be updated in a responsive manner.

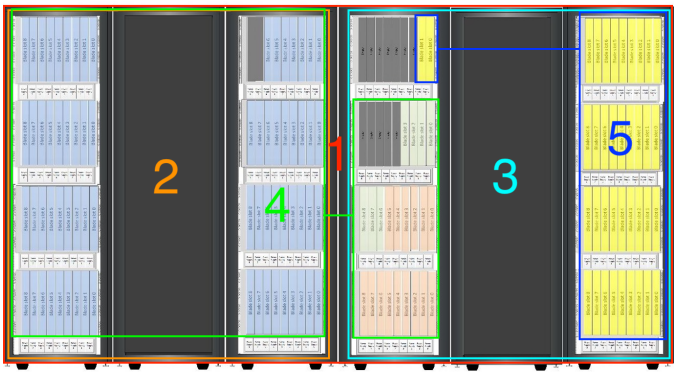


Fig. 3. A diagram showing the layout of maintenance partitions on Cirrus

In order to support flexibility during the early life of the service, when modifications to the scheduler were anticipated to support evolving user requirements, a number of QoS names were defined against the standard node partition. The standard node partition was configured [6] with the additional QoS names highpriority, reservation, testing, weekend and profiling, These were included to support potential future requirements. Whilst not currently defined in Slurm, if QoS matching these descriptions are needed in future they can be created with a simple and immediate sacctmgr command, rather than any change being required to the Slurm configuration files.

B. Maintenance partitions

In order to support the ability of both EPCC and contractor staff to conduct maintenance on our services and sections of those services EPCC has adopted the practice of deploying maintenance partitions to all Slurm services. Partitions are defined for each sensible section of the system in question - for example a physical rack, or GPU nodes.

These partitions can then be used to submit test jobs or equivalent work against relevant sections of the system whilst the primary, user accessible, partitions are disabled. This has in the past been used to demonstrate proper operation of the system after maintenance on specific sections of the system, to diagnose problems on sections of a system and to conduct benchmarking work.

An example can be seen in figure 3 which shows the layout of maintenance partitions on Cirrus. These maintenance partitions are:

- 1. maintenance-all: A partition encompassing all compute nodes
- 2. maintenance-r1: A partition covering all nodes in the e-cell r1
- 3. maintenance-r2: A partition covering all nodes in the e-cell r2
- 4. maintenance-cpu: A partition covering all CPU nodes
- 5. maintenance-gpu: A partition covering all GPU nodes

C. Jinja-2 templating modifications

In the deployment of Slurm configuration to the various nodes of ARCHER2 the Cray EX software stack takes the input of a Jinja-2 template. Jinja-2 is a templating engine which allows python code to be inserted within the text of a template file.

This was, at delivery, used to define items within the slurm.conf file such as the definition of the various compute nodes and the IP and hostname of the Slurm controller node. Two further modifications have been made since this point which we consider worthy of note.

Firstly, in order to use a single slurm.conf template between the main ARCHER2 system and the Test and Development System which supports it we have included logic to manage the differences between the two services. As can be seen below the configuration for energy data gathering is defined based upon the name of the cluster:

```
{% if cluster_name == 'archer2' %}
AcctGatherEnergyType=acct_gather_energy/
    pm_counters
{% elif cluster_name == 'tds' %}
AcctGatherEnergyType=acct_gather_energy/ipmi
{% endif %}
```

Secondly, in cooperation with HPE, configuration has been deployed to weight compute nodes as either HighMem or StandardMem based upon the information recorded in the system's hardware database. This supports us in weighting high memory nodes such that they are used by standard jobs only when all standard nodes are already in use, in order to as far as possible keep them available for users with a high memory requirement. This has also allowed us to define partitions composed of the sets of high and standard memory nodes. The configuration items for these can be seen below, and the full configuration template file can be found at [6].

```
{{ ',HighMem Weight=1000' if node.RealMemory >
     262144 }}

{{ ',StandardMem Weight=500' if node.
    RealMemory < 262145 }}
```

IV. Priority Management

When transitioning to the ARCHER2 service a primary concern was ensuring that users migrating from ARCHER had a comfortable and, as far as possible, familiar experience. To enable this we set out to create a scheduling environment which was reminiscent of that on ARCHER whilst also taking advantage of features available in Slurm which had not been available in PBS Pro.

On the ARCHER and ARCHER2 systems we organise and have organised jobs based on a number of factors. The first of these is assigned queue or QoS priority. Priority is typically either low, standard or high.

The vast majority of work falls into the standard priority band – this work should always run ahead of low priority work but behind high priority work. Low priority work is typically uncharged or charged at a reduced rate but will only run when there is no suitable standard or high priority work to run in its place. High priority is typically limited to specific use cases

– COVID-19 research has been the primary use case of high priority capability on ARCHER2 to date.

The second factor considered in scheduling work is time. This may include the amount of time a job has been queueing for, the length of a job or both. Finally, the size of a job will be taken into consideration - expressed as a node count or proportion of the whole system.

A. Priority on ARCHER

$$queuepriority + \frac{nodecount+walltime(hours)+(eligibletime(hours)*100)}{10} + adminpriority$$

Shown above is the PBS Pro priority formula used over the final years of the ARCHER service. In this formula only queue and admin priority are fully weighted – the other factors considered are divided by ten before being added to the queue priority and admin priority.

As discussed above the queue priority is set into bands of high, standard and low. Admin priority allows system managers to increase or decrease the priority of individual jobs however this facility was not actively used on the ARCHER service.

The other factors considered in this formula are node count, wall time – representing the planned length of a job - and eligible time, representing the length of time a job has been queueing. Node count is the number of nodes requested by the job being submitted. Wall time is then included at a rate equivalent to one additional node per hour of wall time. Eligible time is then at a rate equivalent to one hundred additional nodes per hour.

Given that ARCHER had 4920 nodes the maximum benefit in priority terms that a large job could receive was around two days – that is to say that a newly submitted job attempting to use the full system would be considered equivalent in priority to a theoretical zero node job which had been submitted 49 hours ago.

The weighting given to eligible time was originally lower on the ARCHER service, with a much greater effective priority given to larger jobs. The original job sort formula applied no multiplier to eligible time meaning that each hour in the queue was equivalent to only one compute node. Even a moderately size job of 256 nodes would immediately be given priority over a ten day old theoretical zero node job – a job using half the system would be given priority over a hundred day old job.

As demand on ARCHER increased during 2016 wait times for some users became particularly long – the users in question being those with smaller jobs. After reviewing available options a multiplier was added to the eligible time – taking the equivalent node count per hour to ten. This was then increased so that the equivalent node count per hour reached 100, as above.

With this change in place wait times on the system became more acceptable – at a review the following year it was identified that all but 0.5% of jobs were running within three days and that 75% of jobs were waiting less than 6.5 hours to run.

By the time of the transition to ARCHER2 the ARCHER priority formula was considered successful and the original aim for ARCHER2 was to replicate this as closely as possible.

B. Priority on ARCHER2

Whilst PBS Pro takes a formula as input for scheduling priority, as shown above, under Slurm priority is defined as a number of variables set in the slurm.conf file. Under ARCHER2 the relevant variables were originally planned as:

```
PriorityType = priority/multifactor
PriorityWeightQOS=10000
PriorityWeightAge=500
PriorityMaxAge = 14-0
PriorityWeightJobSize=100
```

This configuration was designed to result in the equivalent configuration to the priority formula shown below.

$$(10,000 * queuepriority(0-1)) + (500 * jobage(0-1)) + (100 * jobsize(0-1))$$

Each of the factors considered is calculated by Slurm as a number between zero and one – this is then multiplied by the weight assigned to the item. No priority is assigned for any factor for which no weight has been set.

The first factor, QoS priority, is equivalent to queue priority for ARCHER and is based upon the QoS a user has submitted to. The QoS priority is very heavily weighted (10,000) compared to other factors – for jobs in the high priority QoS this will result in a priority of 10,000, for the standard QoS a priority of 1,000 and for the low QoS a priority of 2.

The second factor used here is the age of a job – the length of time for which it has been queueing. Whilst not directly technically equivalent, job age is similar in intent and function to eligible time in PBS Pro. A maximum age to be considered has been set which limits consideration at 14 days. Given the weighting of this factor at 500, priority would be assigned on a scale from 0 to 500 where a brand new job would be assigned zero and a job which was submitted 14 or more days ago would be assigned 500.

The next factor is job size. Given the weighting at 100, priority will be assigned on a scale from 0 to 100 where 0 represents a (theoretical) 0 node job and 100 represents a full-system job.

Given that the maximum available priority for the size of a job is 100 and the maximum available priority for a job's age is 500 for 14 days or more a job can gain up to three days worth of priority from its size – the advantage in hours and days at various sizes is shown in table I.

As discussed above the intent in these settings is to broadly replicate the setup which was in place on the ARCHER system as this was considered to be effective and fair to users. The maximum advantage available to a job based on size differs somewhat but is broadly comparable – 67 hours for 5,860 nodes as compared to 49 hours for 4,920 nodes.

Job size	Advantage over zero-size job
512	6 hours
1024	12 hours
2048	1 day
4096	2 days
5860	3 days

TABLE I
ADVANTAGE TIMES FOR VARIOUSLY SIZED JOBS OVER A THEORETICAL ZERO SIZED JOB

In the same way that we did for the ARCHER service, we monitor scheduling and engage with users to ensure that scheduling works for all users and workloads on ARCHER2. As part of this work it became clear early in the service that some users were able to submit large amounts of work and effectively fill the scheduler for significant amounts of time. As such it was decided to implement the "fairshare" facility provided by Slurm, which had been used to support scheduling on the 1,024 node ARCHER2 pioneer system.

Fairshare is intended to ensure that all users have equivalent access to shared resource, prioritising work to be run by users who have run less work recently. The variables "PriorityDecayHalfLife" and "PriorityWeightFairshare" were added to the priority configuration in slurm.conf as below:

```
PriorityType = priority/multifactor
PriorityWeightQOS=10000
PriorityWeightAge=500
PriorityMaxAge = 14-0
PriorityWeightJobSize=100
PriorityDecayHalfLife = 2-0
PriorityWeightFairshare=300
```

This resulted in the equivalent priority formula shown below:

$$(10,000 * queuepriority(0-1)) + (500 * jobage(0-1)) + (100 * jobsize(0-1)) + (300 * fairshare(0-1))$$

The configuration added gives us a fairshare factor. This is calculated based on the previously submitted and completed work of different groups on the system with groups and users who have run less work recently given greater priority. Consideration of work run is discounted at a rate with a half-life of 2 days - a piece of work run 2 days ago would have half the impact on the priority of a new job than a job run yesterday would. The half life ensures that fairshare consideration is only given to recent work. As the fairshare factor is weighted at 300, previous work can be counted as the equivalent of up to 8 days of queuing time.

With the fairshare adjustment in place we have found that users are generally satisfied with the scheduling on ARCHER2 which we consider offers fair access to our diverse user community.

V. INTEGRATION OF LUA SCRIPTING FOR IMPROVED USER EXPERIENCE

During the HECToR and ARCHER services a number of Python hook scripts were deployed at various points in the job life-cycle to manage job submission and user experience. Under Slurm on ARCHER2 and Cirrus these have been replaced by a single Lua script which runs at the time of job submission.

This script supports job management and user experience in a number of ways as discussed below. This discussion is based around the ARCHER2 job submit script [7].

A. Logging and general tests

A number of details of the job, as submitted, are logged. This includes details of the account the job is submitted under and the features requested. This supports the system support team when it is necessary to investigate problems with jobs on the system and has improved the ability of the team to do so.

A number of options which the users may not have specified are then set explicitly to a default:

- Minimum memory per node and per cpu (min_mem_per_node, min_mem_per_node) is set to 0 and
- the number of tasks and the number of tasks per node (num_tasks, ntasks_per_node) is set to 1.

This ensures standard and predictable behaviour, including in partition tests which follow, for users and the support team. Where this action is taken this is also logged.

B. Partition tests

The next part of the script conducts a series of partition tests to confirm that the job submitted is appropriate for the partition it has been submitted to. Actions taken by this section of the script include:

- Rejecting jobs which request memory - as we budget based upon core, node or GPU hours this is not permitted;
- setting jobs in certain partitions (standard and highmem on ARCHER2) to be node exclusive;
- rejecting jobs which specify the GPU partition and which fail to specify a number of GPUs, or which specify a number of cores - as we budget GPU jobs on based upon GPU hours this is not permitted;
- specify the number of allocated cores for GPU jobs scaled to on the number of GPUs requested;
- reject jobs submitted to the serial partition which request a number of nodes or exclusivity;
- reject jobs where no partition has been specified and
- reject jobs specifying or excluding particular nodes, except where this is conducted under a maintenance partition.

As previously all tests and actions are logged.

C. QoS and reservation tests

A number of tests are then run which ensure that reservation and QoS requirements are met. The first action taken is to specify the correct reservation for any job submitted under the "short" QoS. This reservation and the associated QoS keeps nodes available for users to run short (20 minute or less) jobs as part of their development process.

A number of tests are then run which reject jobs which:

- Attempt to use the short QoS reservation without using the short QoS;
- attempt to access another reservation without using the "reservation" QoS;
- have been submitted to the long QoS but request a walltime which would fit into the standard QoS and
- have been submitted without specifying a QoS.

As previously all tests and actions are logged.

D. Time and working directory tests

The final two sets of tests confirm the correctness of the time and working directory attributes of the job. Firstly where a user has not set a time limit for their job either:

- This is specified as 20 minutes where a user has submitted their work to the short QoS or
- a warning message is provided to the user stating that as no time limit has been specified the default will be applied, and that this default is short (1 hour).

Finally the working directory provided is validated to ensure that the relevant file system is available on the partition selected. Where this is not the case a warning message is provided to the user. Again these tests and actions are logged to support investigation where necessary.

VI. CPU and GPU Frequency Management via Slurm

A. CPU Frequency Management

As part of efforts to improve energy efficiency across services at EPCC an investigation [8] was made of the performance of ARCHER2 at a variety of CPU frequencies. From this investigation it was identified that a number of software could operate at lower CPU frequency, and therefore power draw, with only a limited impact to performance. For some software there was a distinct energy saving - on the order of 10-15% - with only a moderate impact to performance of 1-5%. Based upon this information and in the climate of potential energy cuts due to high demand over Winter 2022/2023, a decision was made to move the default CPU frequency from 2.25 GHz to 2 GHz to reduce the power draw of the ARCHER2 service. The default for certain software, where this would be expected to have an out-sized impact on performance, was kept at 2.25 GHz. Management of the CPU frequency was achieved through the use of a Slurm configuration option. The CPU frequency can be set in Slurm in one of two ways - via an environment variable or by a direct option call when submitting your job [9]. For ARCHER2 the application software module environment was modified such that the following environment variable is set in all user sessions by default:

```
export SLURM_CPU_FREQ_REQ=2000000
```

For those applications where this frequency is not preferable the frequency is overridden in the relevant software module. In this way CPU frequency can be assigned on a per-application basis. Since deployment of this frequency management power draw savings on ARCHER2 have been on the order of 600 kW or 20% of system power draw.

B. GPU Frequency Management

In order to manage energy use on GPU systems on site investigations have also been made into managing the frequency of GPUs. Investigations of this on another system on site identified that Slurm on the system was not built with support for GPUs, and that it was not possible in the short term to replace the Slurm version.

In order to provide GPU frequency management logic was therefore added to the Slurm prolog script [10]. This first parses the Slurm job parameters which are obtained via an scontrol command. The desired frequency is then validated to confirm that it is valid.

In order to set the GPU frequency the Nvidia System Management Interface (nvidia-smi) is used. Once the desired frequency has been identified and validated this frequency is passed to the GPU as follows:

```
/usr/bin/nvidia-smi -ac $gpuFreqMemory,
    $gpuFreq
```

If no value is provided then a default is passed to nvidia-smi. The GPU memory frequency for the GPUs in question is fixed, hence this is not user specifiable.

VII. Conclusions

We have found that Slurm has allowed us to offer effective and efficient scheduling across a variety of systems at EPCC, adapting well to varying system types and diverse user bases. We consider that this has been enabled by the configurability and flexibility of Slurm.

We have found that even with a diverse range of systems with different requirements, levels of heterogeneity and user bases we have been able to share a significant degree of configuration between systems. This has enabled improvements deployed to one system to be easily deployed to another.

Finally we have found that automation can considerably reduce the effort required by systems staff to support Slurm. Integration with SAFE and Rundeck allows all day-to-day actions to be automated from the point of user request, without requiring any intervention by staff. This has offered a considerable saving in staff workload.

We finally note that Cirrus has been operating successfully with Slurm since 2020 and that ARCHER2 has been doing so since 2021. We have been able to maintain high levels of user satisfaction on both these services and we consider that the benefits of Slurm described above have made a significant contribution to this satisfaction.

VIII. Future Work

We are keen to further develop our Slurm deployments going forward. Items of interest include:

- Job pre-emption for urgent computing needs
- Deployment of burst-buffer file system access via Slurm

- Automated management of QoS membership and access on a per-project or per-user basis

We would of course be pleased to engage with other sites who are using or are interested in using Slurm with HPE systems.

REFERENCES

[1] https://slurm.schedmd.com/slurm_design.pdf
[2] https://slurm.schedmd.com/team.html
[3] https://www.epcc.ed.ac.uk/hpc-services/safe
[4] https://resources.rundeck.com/learning/an-overview-of-rundeck/
[5] https://github.com/EPCCed/Flexible-Slurm-Configuration-for-Large-Scale-HPC/tree/main/Reservation_Tickets
[6] https://github.com/EPCCed/Flexible-Slurm-Configuration-for-Large-Scale-HPC/blob/main/archer2-slurm-template.conf
[7] https://github.com/EPCCed/Flexible-Slurm-Configuration-for-Large-Scale-HPC/blob/main/job_submit.lua
[8] https://www.archer2.ac.uk/news/2022/12/12/CPUFreq.html
[9] https://docs.archer2.ac.uk/user-guide/energy/
[10] https://github.com/EPCCed/Flexible-Slurm-Configuration-for-Large-Scale-HPC/blob/main/gpu-frequency-prolog.sh