http://www.mcs.anl.gov/home/rayl/scheduler/scheduler.html

Argonne SP General Information

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node1.html

SP Scheduler Policy Overview

The goals of the MCS/SP job scheduler are fairness, simplicity, and efficient use of the available resources. These goals are in conflict, but the scheduler is designed to be a compromise of the three. Users will be able to request a set of nodes for any type of use. In order to maintain the quality of machine access, the scheduler will provide a single point of access, `spsubmit`. This program will allow users to queue both interactive and batch access to the SP. When resources are available to a particular user they will be notified by the scheduler and at that time they will have exclusive access to the number of nodes they requested.

While there are currently no limits to the number or size of jobs that can be submitted to the scheduler, the amount of resources requested is used to determine when time is actually provided (more information). There are currently two semi-independent queues Day and Night . The division of jobs among these queues is based on Resource Unit(RU) limits and time limits. The scheduler uses a FIFO with *backfilling* paradigm for putting jobs onto the machine. The intent is to let not the type of jobs drive the queueing policy, but rather the amount of resources requested.

The allocation policy implemented in the scheduler should deter resource hogging. New jobs `spsubmit`ted by CACs that already have scheduled a stated RU limit are placed in a holding pen, thus not allowing any one CAC(Charge Allocation Category) to prevent other users from utilizing machine resources.

Any modifications to this system will be made public, through `spannounce`.

Sat Sep 17 15:56:16 CDT 1994

`http://www.mcs.anl.gov/home/rayl/scheduler/node2.html`

Scheduler Commands

The following list contains the commands which are usefull for using the SP job scheduler. All of them have "-h"options which provide help and any additional functionalities they have.

Click on the command's name to see example output.

getjid Returns the Job ID based on the node you are currently using.

spfree Prints the number of nodes that are currently free.

sphelp Prints this message.

sppause Puts all the users'interactive jobs, in the queue, on 'pause'.

sppolicy Prints out the policy statement for the SP.

spq Lists running jobs and those in the various scheduler Queues. It also lists the job status and how much time a job is requesting or when it will finish if it is currently running.

sprelease Is used to remove jobs from either the queue or the machine, if the job has already started. This returns node(s) back to the system, and stops the charging against the Charge Allocation Category (CAC)

spsubmit Used to submit jobs to the queue.

spstatus Prints out the current status of the SP.

spunpause Changes the status of all the users' interactive jobs, in the queue to active from (P)aused thereby enabling them to be run by the scheduler.

spusage Shows who is using what nodes.

xspusage Same as above except for X windows instead of text based.

spwhat Shows what job would be able to *backfill* if it were submitted.

spwhen Gives an estimate of when a particular job will start based on the current queueing algorithm.

cacReport Reports how many resource units have been used by a specified CAC group.

whatcac Reports which CAC group a particular login ID belongs.

Sat Sep 17 15:56:16 CDT 1994

`http://www.mcs.anl.gov/home/rayl/scheduler/node3.html`

Scheduling SP time with spsubmit

The `spsubmit` command will allow users to submit jobs to the scheduling system. The current scheduler will prompt you for necessary information about your job. NOTE: Future plans include a version of `spsubmit` which supports command-line arguments. The scheduler will assign your job entry a unique numeric job ID. This unique job ID can be used to track how many jobs you have submitted to the scheduler and also to remove your unwanted job entries from the scheduler by using `spq` and `sprelease` respectively.

When the nodes you've requested through the scheduler become available, you will be notified by email, xspusage, and by creating a special file in your `/sphome` directory. The email will inform you of the time the nodes you requested were actually allocated and the names of the nodes that were assigned to you. The file which is created has the form:

`/sphome/<user_name>/SPnodes.<job_ID>`

This file contains a list of the nodes you have been assigned for a particular run.

Nodes allocated to you by the scheduler will be in exclusive access mode. Only your login will be enabled on the nodes that you are allocated. This means you will be the only one that can log into these nodes until you release them with `sprelease`. Once you issue the `sprelease` on the nodes allocated to you, you will not be able to log into those nodes.

Sat Sep 17 15:56:16 CDT 1994

`http://www.mcs.anl.gov/home/rayl/scheduler/node4.html`

Batch submission with `spsubmit`

If you have submitted a program or script for batch execution, it will be automatically executed for you. It does this by issuing an rsh of the program or script to the first node in the group that it allocates to you.

If you require that data be staged to nodes or other job setup, you should submit a job script which performs the appropriate setup and then runs your program. Be sure when submitting programs or scripts that you provide the full path to them. ie

(/sphome/<user_name>/<program>)

You should also use full paths to any files or programs referenced inside your programs or job scripts.

NOTE: even though a job is running as a batch job, the user who submitted it will have access to the nodes it is executing on to check its progress if necessary.

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node5.html

Interactive submission with `spsubmit`

It may be useful to request interactive time on the SP until you have thoroughly tested your programs and/or scripts. This will allow you to closely monitor their performance until you are confident that they will run correctly in batch mode.

Be sure if you schedule interactive time and it gets queued because the system is currently loaded, that you will be available to use the resources when they actually become available. For example, if the system is heavily loaded and the interactive time you requested becomes available at midnight you will need to be able to use those resources at midnight because they will be allocated to you irregardless and thus billed against your Charge Allocation Category (CAC) until they are released with `sprelease`. You may wish to remove interactive jobs or pause them with `sppause` from the queues when leaving work for the day.

When using `spsubmit` to request interactive time it still asks you for information about your job even though you can use the nodes any way you wish interactively. The scheduler is just a means of collecting this information to study how people are using the machine.

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node6.html

Spsubmit Example 1: An MPL Script

Here is a submission example where the user wants to submit a script which runs an Message Passing Library(MPL) program in "user space" mode over the IBM switch.

```
bonnie.mcs.anl.gov% spsubmit
Charge Allocation Category: [default lifka]
Maximum Wall-clock Run-Time (minutes): (1-???) 15
Number of Nodes Required: (1 - 128) 4
CAC: "lifka" *now* has 28114 RUs available, after committing.
(I)nteractive (B)atch: b
Job Classification (M)PL,(T)ask Farm, (P)4: m
IP over the switch [y/n]: n
Full path to Shell Script/Program: /sphome/lifka/MPL/RunMe
Command Line arguments for your job:

Username: lifka
Charge Allocation Group: lifka
Job Type: B
```

```
Job Classification: mpl
IP over the switch: no
Number of Nodes: 4
Maximum Wall-clock Run-Time: 15
Program/Job Script: /sphome/lifka/MPL/RunMe
Command Line Arguments:

(C)ommit or (A)bort: c

Your JID for this job is: 12132612
bonnie.mcs.anl.gov%
```

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node7.html

Spsubmit Example 2: A P4 Program

This submmission example was for a P4 program. Notice that the p4 debugging options and output redirection were entered as desired command-line arguments.

```
bonnie.mcs.anl.gov% spsubmit
Charge Allocation Category: [default lifka]
Maximum Wall-clock Run-Time (minutes): (1-???) 15
Number of Nodes Required: (1 - 128) 16
CAC: "lifka" *now* has 27814 RUs available, after committing.
(I)nteractive (B)atch: b
Job Classification (M)PL, (T)ask Farm, (P)4: p
IP over the switch [y/n]: y
Full path to Shell Script/Program: /sphome/lifka/P4/UniTree/Utest
Command Line arguments for your job:-p4dbg 10 -p4rdbg 10 >
/sphome/lifka/run.out

Username: lifka
Charge Allocation Group: lifka
Job Type: B
Job Classification: p4
IP over the switch: yes
Number of Nodes: 16
Maximum Wall-clock Run-Time (minutes): 15
Program/Job Script: /sphome/lifka/P4/UniTree/Utest
Command Line Arguments: -p4dbg 10 -p4rdbg 10 > /sphome/lifka/run.out
(C)ommit or (A)bort: c
Your JID for this job is: 12135506
bonnie.mcs.anl.gov%
```

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node8.html

Spsubmit Example 3: A Task Farm Script

```
bonnie.mcs.anl.gov% spsubmit
Charge Allocation Category: [default lifka]
Maximum Wall-clock Run-Time (minutes): (1-???) 15
Number of Nodes Required: (1 - 128) 32
CAC: "lifka" *now* has 26526 RUs available, after committing.
(I)nteractive (B)atch: b
Job Classification (M)PL, (T)ask Farm, (P)4: t
IP over the switch [y/n]: n
Full path to Shell Script/Program: /sphome/lifka/RunMe
Command Line arguments for your job:4 16

Username: lifka
Charge Allocation Group: lifka
Job Type: B
Job Classification: task
IP over the switch: no
Number of Nodes: 32
Maximum Wall-clock Run-Time (minutes): 15
Program/Job Script: /sphome/lifka/RunMe
Command Line Arguments: 4 16

(C)ommit or (A)bort: c
Your JID for this job is: 12135557
bonnie.mcs.anl.gov%
```

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node9.html

Supported Job Types

With the new SP Scheduler, nodes can be used both for batch and interactive computing. Batch jobs are run via the scheduler by a script the user indicates upon `spsubmit`ting. For more information, see Batch submission with `spsubmit`. Interactive jobs are run directly by the user. After the user is notified that the nodes are now theirs, the user logs in directly to the nodes and runs the program. For more information, see Interactive submission with `spsubmit`.

WARNING: If you submit an interactive job, especially a small job that is likely to *backfill*, be sure to use the `sppause` feature. If your interactive job is put on the machine at night and you are not there to use the resources, you will burn your scheduled time, and RUs. Future plans include a “I'm here.” acknowledgement for interactive jobs. Jobs that do not get the “I'm here.” after 15 minutes will be killed, and the user charged for a full 60 minutes. However, this is not implemented at this time.

The nodes are configured to run parallel programs which use the IBM switch or ethernet for communication. Jobs which use MPL, MPI, or p4 for interprocess communication fall into this category. MPL is ideal for I/O bound parallel programs using lots of message passing between nodes. P4 uses ethernet connections for communication between nodes. P4 is ideal for compute bound parallel programs with few messages passed between the nodes. P4 uses the SP as a cluster of workstations.

Task Farming does not use communication between the nodes. Task Farmed jobs utilize the SP like a room full unconnected of RS6000 workstations where different programs (or the same program using different data sets) can be run on each node.

If you are not sure how to classify a program please consult the SP User's Guide. If you are still having trouble contact `spsupport@mcs.anl.gov`.

If there is a job type or execution method which is not supported in some fashion by the job-scheduler, please contact `spsupport@mcs.anl.gov`.

Sat Sep 17 15:56:16 CDT 1994

`http://www.mcs.anl.gov/home/rayl/scheduler/node10.html`

MPL

MPL is the new IBM switch software which replaces EUIH. It provides much better performance than EUIH plus a friendlier user interface. To use MPL you must use `mpcc` and `mpxlf` instead of `xlc` and `xlf` to compile and link your programs. For more details on compiling and linking MPL code see the SP User's Guide. To run an MPL code there are several environment variables which must be set.

MP_HOSTFILE Points to the file containing the list of nodes that you have been assigned to run on by the scheduler.

MP_INFOLEVEL Provides an overwhelming amount of debugging information, most of which is difficult to interpret. The default level is 1 but should be 0 which is not information. We have suggested this to IBM.

MP_EUILIB This tells MPL to either run IP over the switch if set to "ip" or direct mode over the switch if set to "us".

MP_PROCS The number of processors the job runs on (should correspond to the number of nodes in the hostfile.)

These variables can be set interactively if you are scheduling interactive time or within a script if you are scheduling batch time. It is crucial that you set the environment variable `MP_HOSTFILE` to the file containing the list of nodes the scheduler has allocated to you otherwise MPL will bomb. Here are some examples of a csh batch scripts which will run a MPL programs.

MPL Batch Script Example for IP Over the IBM Switch
MPL Batch Script Example for Direct Mode Over the IBM Switch

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node11.html

MPL Batch Script Example for IP Over the IBM Switch

```
#!/usr/local/bin/tcsh
# get the job id for this job
set JID = `/usr/local/bin/getjid`
# Run IP over the switch
setenv MP_EUILIB ip
# convert the Scheduler generated SPnodes file to an SWnodes file for
# IP over the switch
cat /sphome/<username>/SPnodes.$JID | sed 's/sp/sw/' >
/sphome/<username>/SWnodes.$JID
# set SWnodes file the hostlist for this job
setenv MP_HOSTFILE /sphome/<username>/SWnodes.$JID
# set the number of processes that you wish to run on
setenv MP_PROCS 64
# set the debugging level (0 is generally recommended)
setenv MP_INFOLEVEL 0
# execute the program with the output redirected to a file
poe /sphome/<username/<program_name> > /sphome/<username>/job.output
# release the nodes for the next user
sprelease $JID
```

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node12.html

MPL Batch Script Example for Direct Mode Over the IBM Switch

```
#!/usr/local/bin/tcsh
# get the job id for this job
set JID = `/usr/local/bin/getjid`
# Run IP over the switch
setenv MP_EUILIB us
# make the scheduler generated SPnodes file the hostlist for this job
setenv MP_HOSTFILE /sphome/<username>/SPnodes.$JID
# set the number of processes that you wish to run on
setenv MP_PROCS 64
# set the debugging level (0 is generally recommended)
setenv MP_INFOLEVEL 0
# execute the program with the output redirected to a file
```

```
poe /sphome/<username/<program_name> > /sphome/<username>/job.output
# release the nodes for the next user
sprelease $JID
```

Note to Interactive MPL users: If you are going to use your interactive time for MPL jobs you will have to use a MPL hostlist file so that your MPL job will run on the nodes you have been allocated. If you do not, MPL may try to run your jobs on nodes which you do not have access to, causing the job to fail.

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node17.html

Using Temporary Storage on the SP nodes with the Scheduler

If your program requires that data be moved into and/or out of the `/tmp` directory on the nodes it will be important that your program or script perform these moves before and after execution. This is because you cannot be guaranteed access to specific SP nodes for data staging before the nodes are actually allocated to you or after you have released the nodes using the `sprelease` command. If you request interactive time you will be able to log into the nodes allocated to you and do your data staging before and after running your program interactively. NOTE: Once a node is released all user data in `/tmp` is removed so that the next user has the same amount of `/tmp` space available for their job.

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node18.html

Releasing Resources when finished

When you are finished with the SP resources that you have been allocated you will have to release them so that they can be re-scheduled by the system. It is extremely important that you release resources when you are finished with them so that the idle node time is not charged against you. The scheduler will maintain accounting records based on wall-clock time and the number of allocated nodes that will be charged against the number of node-hours allocated to your Charge Allocation Category (CAC). Full-node hours will be charged against your account for the resources you request whether you use them or not. This is because while the nodes assigned to you no-one else will have access to them. To return nodes back to the scheduler once you are done, you will have to use the command `sprelease` on each of your allocated nodes or `sprelease <job_id>` to release all nodes associated with that particular Job ID. `sprelease` will log your release of the node, remove your access from the node, kill all current processes on the node owned by you, and clear `/tmp` for the next user. Because `/tmp` is cleared it will be necessary to move any important data off the `/tmp` disk to your `/sphome` directory before releasing the node.

The `sprelease` command can be run interactively by the user, called at the end of a shell script, or from a system call in your programs. The advantage of embedding it at the end of your scripts or programs is that you will not have to monitor you batch runs to ensure that your resources were released at the correct time. The `sprelease` command offers a big improvement over the previous scheduler because nodes can be made available and used again as soon as they finished being used. In the past large time-blocks were reserved and held irregardless of how long they were actually used.

Sat Sep 17 15:56:16 CDT 1994

http://www.mcs.anl.gov/home/rayl/scheduler/node19.html

Current Queueing Algorithm

Jobs are currently run in an optimized first-in-first-out, FIFO, fashion, but with back-filling/packing. The optimization works by first seeing if the next job in a particular queue has enough nodes available to start. If so it is allocated the nodes and started. If there are not enough nodes available for the next job in the queue, the time that job is blocked for is determined by looking at jobs currently on the system and determining how long it will be before enough of them finish so that the next job can start. The scheduler tries to fill this time gap with another job in the queue which will not run longer than this time gap and does not require more nodes than are currently available. The following example illustrates the *backfilling* procedure:

Assuming a a 64 node machine, and the following jobs in the queue:

```
Job  #Nodes  MaxTime
A      30      4hrs
B      20      2hrs
C      32      5hrs
D      10      3hrs
E      10      1hrs
F      20      3hrs
G      4       3hrs
```

Jobs A and B will get on the machine as it has all 64 nodes to give out, and A only needs 30 nodes and B needs 20, which leaves 14 nodes free. The scheduler then figures out what time, if all jobs (A&B) ran to their respective MaxTimes, would enough resources be free to run the "NextJob", or job C. In this example it turns out that in 2 hours, when B releases, there will be enough nodes for C to run. We call C the "NextJob" as it's the next job to be run, and the job that other *backfill* jobs can't block. Using *Backfilling* the scheduler would figure out what jobs could start, that won't block the "NextJob" for any longer period of time. So in this case, job D would be checked, and while it would fit, there are 14 nodes free and D only needs 10, running D would mean that there wouldn't be enough nodes for C to use when it trys to start up in two hours, because D would still be running. The scheduler continues down the queue looking for jobs that will fit in the "space" and run within the "time". Job E will fit both space and duration. (10nodes < 14nodes and 1hr < 2hrs). So job E would start. And while job E is running before job C and D, running job E will not delay job C or D from starting.

This system should work well when the job size and time are varied. We expect our job mix to change over time, but we hope to have the machine in full use while keeping things fair. Yes we know that:

"Nowhere in the world is there a sign that says, 'Life is fair'. There is a reason for this."

but we feel someone has to try. :-)

To see resource unit limits that the scheduler uses to classify jobs use the spq -h command. During the time periods when jobs requiring large numbers of resources are started, jobs requiring few numbers of resources will be started if there are no large jobs queued to run. Also notice that jobs requiring very large numbers of resources are considered to be extraordinary events and will be treated as such by the scheduling system.

Sat Sep 17 15:56:16 CDT 1994

http://www.mcs.anl.gov/home/rayl/scheduler/node20.html

SP Allocation Policy Overview

**** UNDER CONSTRUCTION ****

Charge Allocation Category (CAC): the grouping of userids which allocations get changed against. If the allocation is a credit card (a good way to think of it), then the CAC are the people authorized to use the card. The "money" is in terms of RUs instead of dollars.

Resource Units (RUs) are what a CAC's allocations are in terms of. Currently they are based on node minutes - the number of nodes times the maxtime, in minutes.

Previously all jobs submitted against a CAC were totaled and compared with a Hold Limit. Jobs submitted after the Hold Limit was reached were held aside from the queue until the earlier submitted jobs left the queue. Now, the Day queue CAC totals and Night queue CAC totals will be considered separately. Thus, you may submit multiple long-running jobs for the Night queue, but still do development during the day.

Sat Sep 17 15:56:16 CDT 1994

---

http://www.mcs.anl.gov/home/rayl/scheduler/node21.html

Miscellaneous Information and Terminology

Note: STDERR and STDOUT are redirected to email.

MaxTime: jobs are killed off after this user entered, wall-clock time. Used to determine the scheduled number of RUs. Set in `spsubmit`. Longer/Larger estimates are safer, however, they are less likely to get scheduler or *backfilled* early.

NextJob: The job in the queue that is at the top. Any jobs that back fill can not block this job from starting any later than if no additional jobs were started.

ShadowSpace: The number of free nodes, and the time that they could be used *without* impacting the start time of the NextJob. This is based on the most conservative case where all the running jobs ran till their MaxTime had elapsed.

`spannounce`: used for group distribution of messages to all SP users. Being a member of `spannounce` is a requirement for retaining access to the SP.

SP Queues - There are current two SP queues that are distinct but interact together: the "Day Queue" and "Night Queue". The Night queue run from 6:00pm to 9:00am Monday-Thursday. Note that "Day" class jobs will *backfill* any time on nights and weekends. [NOTE: We have eliminated the "Special" job classification. From Friday at 6:00pm to 9:00am on Monday, all "Night" classified jobs will be able to run at any time. Former "Special" class jobs will now be considered "Night" class. However, if they require more than 15 hours of wallclock time they will only be able to start during the weekend. These jobs will be designated in "spq" as NightWE, so that it is clear that they will not run during the weekday nights.]

Job Statuses -Note: spq -h contains current status flags. Status Flags include:

- R - job is currently running on the machine.
- W - job is currently waiting to be run.
- P - job has been paused by user. Note, this job will not block waiting jobs.

- H - job has been placed on hold status due to that CAC has requested more resources than the scheduling system will allow to be scheduled at any particular time. Their scheduled RU values exceed specified limits. (Guilty of resource hogging.) Job will remained in hold status until the number of outstanding resources requested is reduced either by some of the jobs running or by removing some of the queued requests.
- h - an interactive paused job that has been held. If a job with the status of "h" is unpaused it will change to "H" until the number of outstanding resources requested is reduced either by some of the jobs running or by removing some of the queued requests.

Sat Sep 17 15:56:16 CDT 1994