HPC Scheduling

- HPC Scheduling
  - Scheduling Basics
    - What's the difference between a scheduler and a resource manager?
    - How do I get access to the scheduler?
    - What is a reservation?
  - Priorities
    - Can I change my job's priority?
    - Besides incrementing a job's priority the longer it waits in the queue, how else does the scheduler change priority?
  - Fairshare
  - Backfill
  - Interacting with the Scheduler
    - How do I show a job's state from the scheduler's point of view?
    - There are idle processors so why isn't my job starting?

The CI uses Moab Workload Manager (Moab). It is developed by Adaptive Computing and you can refer to the documentation on their site for more information. Moab is a commercial product. Under the hood Moab has some more advanced features and is updated much more frequently.

This document is to provide you a better understanding about job scheduling and why you see the behavior you do with job flows through the queue. For more information on how to manage your jobs or the resource managers in use at the CI, refer to the job management FAQ. For details of the specific scheduling policy in use on a particular cluster, refer to that cluster's documentation:

- Teraport Scheduling Policy
- PADS Scheduling Policy

Scheduling Basics

What's the difference between a scheduler and a resource manager?

A resource manager is responsible for managing the resources available in the cluster. Resources include hardware such as the node itself, the CPUs, memory, but can also be more abstract like network bandwidth or software available only on certain nodes. The resource manager also defines the queues for the cluster and handles job startup and shutdown, but does not determine which jobs to start or stop. The scheduler is in charge of analyzing the running jobs and jobs wanting to run. The scheduler applies the scheduling policy for the cluster to determine what jobs to start and stop based on the information the resource manager reports back about resource availability. Once the scheduler has determined what job, if any, to run next, it instructs the resource manager to start the job on a specific node or set of nodes. The scheduler monitors running jobs to ensure no job exceeds its allotted time or resources and if needed instructs the resource manager to kill jobs. The scheduler is also in charge of reservations.

In short, the resource manager is in charge of the physical aspects of the cluster and the scheduler is in solely in charge of job administration and the two work hand in hand to ensure proper utilization based on the policy for the cluster. Refer to the job management FAQ for more information on the role of the resource manager and how to interact with it.

How do I get access to the scheduler?

Both Maui and Moab are installed in Softenv and can be accessed with the `+maui` or `+moab` key, respectively.

What is a reservation?

Much like you reserve a hotel room to ensure a room is available for you at the time you checkin, the scheduler creates a reservation on the nodes it's going to assign to your job to ensure that they're available when your job starts. The scheduler does this automatically for each job and this type of reservation is called a job reservation.

There are times where you may require a specific number of nodes available at a specific time for some purpose - say a demonstration for your project's program manager. In these cases the cluster support staff can create what is called an advanced reservation. If you let the support staff know ahead of time how many nodes for what time frame and for what users, they will put a reservation in place to guarantee you have the resources you require. Keep in mind, though, if you let us know an hour before hand and the cluster is full, there's not much we can do so make sure to request the reservation as much in advance as possible.

The last type of reservation is called a standing reservation. Depending on the purpose and utilization of the cluster, the cluster administrators may put in place reservations that happen on a recurring basis. One example might be to reserve all of the cluster every Monday for system maintenance. Another could be to reserve a small set of nodes for specific types of jobs. Whatever the case may be, these types of reservations would be in place on a cluster by cluster basis, so refer to each cluster's documentation on what, if any, standing reservations are in place.
Priorities

Maui and Moab use a job's priority to determine when a job should run. A job with a higher priority will normally run before a job with a lower priority. Everyone submitting to the same queue starts out with the same priority, with some caveats; however, a job's priority can and will change over time based on several factors as will be explained below.

Can I change my job's priority?

You can change your job's priority, but only to a lower priority. However the scheduler will change a job's priority so that no job will ever be starved and will eventually run. The simplest method used is for every minute that a job sits in the queue it's priority is increased by 1, though this can be different depending on the cluster's policy. That means that newly submitted jobs will usually have a lower priority than an older job in the same queue.

There is one method that a user has to control his starting priority and that is to use a different queue as long as his job meets the requirements of that queue.

Refer to the particular cluster's documentation on the exact priority policy.

Besides incrementing a job's priority the longer it waits in the queue, how else does the scheduler change priority?

Fairshare

If we were to depend on a pure first come, first serve (FIFO) basis, it would be possible for a user to starve jobs of resources indefinitely. For example, imagine a user submits 5000 1 hour, 1 node jobs simultaneously. In a strict FIFO policy, any job submitted after those 5000 jobs would have to wait for all 5000 jobs to finish before being able to run. In this case, it would take almost 2 days before any other job could run. Imagine the wait if twice as many jobs that ran 4 times as long were submitted. Obviously it would be ideal to make at least some portion of the cluster available to other users at the same time. Both Maui and Moab have the ability to implement what is called a fairshare policy.

What a fairshare policy does is take into account a user's past usage of the cluster relative to the cluster's utilization for the same time frame. It uses this past usage history to either decrease or increase a user's priority of current and future jobs. So if a user uses the cluster heavily while the cluster as a whole is being heavily used, the user's job priorities will be decreased to allow for other users to use the resource. Subsequently, if a user has not used the cluster during peak times, that user's job priority may be increased faster.

Each cluster may have a different fairshare policy so please refer to that cluster's policy for specifics. For more information on fairshare, please refer to the Adaptive Computing's documentation.

Backfill

Under normal circumstances, a higher priority job will run before a lower priority job will run. There are cases where this is not desirable and results in cluster resources being squandered. Imagine that the highest priority job is an 8 node 12 hour job, however only 7 nodes are currently free and the eighth node won't be free for another 10 hours. After the top priority job there are 7 1 node 9 hour jobs. Under the normal top-priority-job-first model those 7 nodes would sit idle until the eighth node freed up at which point the top priority 8 node job would run. The next 7 9 hour jobs wouldn't run until after that job completes in 12 hours even though nodes were available. Backfill allows lower priority jobs to sneak ahead and run on nodes that are otherwise being reserved for another job. In our example, backfill would create a reservation for the top priority 8 node job to guarantee that its start time is no later (10 hours from now when the eighth node will be free) than it already would have been, but then allows the next 7 jobs to run since they will run and complete before the top priority job's eighth node is free. Backfill allows the scheduler to keep the number of idle nodes to a minimum; however, it does depend on users supplying relatively accurate walltimes for their jobs. More detailed information can be obtained from Adaptive Computing's documentation.

Interacting with the Scheduler

How do I show a job's state from the scheduler's point of view?

Use the `check job` command. The following shows the output of a running job:
$ checkjob 854176-1

checking job 854176-1

State: Running
Creds: user:leggett group:ci-users class:extended qos:threeday
WallTime: 00:05:56 of 1:00:00
SubmitTime: Thu Feb 12 10:01:19
  (Time Queued Total: 00:00:02 Eligible: 00:00:02)
StartTime: Thu Feb 12 10:01:21
Total Tasks: 1

Req[0] TaskCount: 1 Partition: DEFAULT
Network: [NONE] Memory >= 0 Disk >= 0 Swap >= 0
Opsys: [NONE] Arch: [NONE] Features: [NONE]
NodeCount: 1
Allocated Nodes:
[tp-c119:1]

IWD: [NONE] Executable: [NONE]
Bypass: 0 StartCount: 1
PartitionMask: [ALL]
Flags:       RESTARTABLE

Reservation '854176-1' (--00:03:49 -> 00:56:11 Duration: 1:00:00)
PE: 1.00 StartPriority: 100

A few pieces of information that are useful from this are:

- How long your job was in the queue before it was started
- How long your job has been running
- How many and what nodes were allocated to your job
- The name of the reservation this job is running under
- The priority of the job when it started

Here's the output from a job waiting in the queue:

$ checkjob 854176-500

checking job 854176-500

State: Idle
Creds: user:leggett group:ci-users class:extended qos:threeday
WallTime: 00:00:00 of 1:00:00
SubmitTime: Thu Feb 12 10:01:20
  (Time Queued Total: 00:03:50 Eligible: 00:03:50)
Total Tasks: 1

Req[0] TaskCount: 1 Partition: ALL
Network: [NONE] Memory >= 0 Disk >= 0 Swap >= 0
Opsys: [NONE] Arch: [NONE] Features: [NONE]
NodeCount: 1

IWD: [NONE] Executable: [NONE]
Bypass: 0 StartCount: 0
PartitionMask: [ALL]
Flags:       RESTARTABLE

PE: 1.00 StartPriority: 103
job cannot run in partition DEFAULT (idle procs do not meet requirements : 0 of 1 procs found)
idle procs: 36 feasible procs: 0

Rejection Reasons: [State  : 103][ReserveTime : 1]

A few pieces of information that are useful from this are:

- The state of your job
- When your job was submitted and how long it has been waiting
- The current priority of your job
- Why your job cannot run
There are idle processors so why isn't my job starting?

Any number of reasons might be the root cause. There could be an advanced reservation in place. If the cluster's scheduling policy will not co-schedule different users on the same machine, there could be a user running on only one processor causing the other processors to sit idle but un-schedulable. Or there could simply be a problem with the resource manager or scheduler and the support staff needs to take a look. Your first step should be to see what the scheduler thinks is the problem. If that provides no help, let the cluster support staff know.