HPC Job Scheduling

Expand all   Collapse all

Overview

The job scheduler's primary purpose is to reserve resources for users' jobs to ensure jobs run at their highest performance. It keeps a given compute node from being overloaded, and places jobs on hold until resources are available.

The scheduler's secondary purpose is to monitor usage to ensure fair distribution of HPC resources over time. Fair share policies prevent a given user or group from monopolizing the entire cluster for extended periods if other users' jobs are waiting in the queue.

Job Scheduler Software

The HPC uses the Slurm job scheduler, version 17.02. Slurm was selected for reasons including its free software licensing, ability to reserve specialty hardware such as GPUs, strong authentication of multi-node processes, and comprehensive resource accounting.

Job Queues (Partitions)

Using the High Performance Computing for BIOinformatics Software (HPCBIOS) project's HPCBIOS_05-05 as a guide, the HPC contains the following primary job queues:

Queue Name	Description
batch	Default queue
debug	Time/resource limited for debugging jobs
interactive	For interactive jobs
bigmem	For jobs requiring 255000-382000 MB of RAM per node
hugemem	For jobs requiring 383000-890000 MB of RAM per node

The HPC scheduler may route certain jobs to the following secondary job queues, but users should never have to interact with them directly:

Queue Name	Description
gpu	For jobs using GPUs (automatically used when a job in the batch queue requests a GPU)
gpu-debug	Time/resource limited for debugging jobs using GPUs (automatically used when a job in the debug queue requests a GPU)
gpu-interactive	For interactive jobs (automatically used when a job in the interactive queue requests a GPU)
any-debug	Time/resource limited for debugging jobs using GPUs (automatically used when a job in the debug queue requires fewer than 12 cores per node)
any-interactive	For interactive jobs (automatically used when a job in the interactive queue requires fewer than 12 cores per node)

Job Queue Policies

The overall goal of the job queue policies is to balance the speed of job development/debugging against providing resources for debugged, large-scale, long-running jobs. Development/debugging jobs have a higher priority so that they can start quickly, but are limited on time (and sometimes resources). Debugged jobs have a lower priority for start times, but have minimal resource and time limits.

Queue Name	Base Priority	Job Resource Limits	Job Time Limits	Default Job Time

batch	Low	None (can use all non-GPU nodes)	30 days	1 day
debug	Medium	None (can use all non-GPU nodes)	0.5 hours	0.5 hours
interactive	High	Can use 1-4 non-GPU nodes	2 hours	2 hours
bigmem	Low	None (but only 3 nodes have 384 GB of RAM are in queue)	30 days	1 day
hugemem	Low	None (but only 1 node with 896 GB of RAM is in queue)	30 days	1 day
gpu	Low	None (can use all GPU nodes)	30 days	1 day
gpu-debug	Medium	None (can use all GPU nodes)	0.5 hours	0.5 hours
gpu-interactive	High	Can use 1-2 GPU nodes	2 hours	2 hours
any-debug	Medium	None	0.5 hours	0.5 hours
any-interactive	High	Can use 1-4 nodes	2 hours	2 hours

Job Scheduler Policies

Time Reservation Limits

The total time reserved for a given user (for both running and and queued jobs) in all partitions is limited to 1000 CPU-days. Jobs exceeding this total will be blocked until running jobs use up part of their time allocation, or the user cancels some of the running or queued jobs to reduce their total reservation to below 1000 CPU-days. Other users may queue jobs ahead of blocked jobs, helping ensure no single user monopolizes the HPC for extended periods of time when others are waiting to run jobs.

The current job limits allow a user to submit any of the following combinations of jobs:

- 33 jobs using a single CPU core for 30 days each (33 jobs × 1 CPU/job × 30 days = 990 CPU-days)
- 1 job using all 28 CPU cores in 35 nodes for 1 day (1 job × 28 CPUs/node × 35 nodes/job × 1 day = 980 CPU-days)
- 35 jobs using all 28 CPU cores in separate nodes for 1 day each (35 jobs × 28 CPUs/node × 1 node/job × 1 day = 980 CPU-days)
- 1 job using all 28 CPU cores and both GPUs in 3 GPU nodes for 11 days (1 job × 28 CPUs/node × 3 nodes/job × 11 days = 924 CPU-days)
- 3 jobs using all 28 CPU cores and both GPUs in separate nodes for 11 days (3 jobs × 28 CPUs/node × 1 node/job × 11 days = 924 CPU-days)

Fair Share

Each user submitting jobs is allocated an equal share of compute resources averaged over an extended period of time. If a given user's jobs have exceeded their share the last 2 weeks, their default job priority will be decreased relative to a user who hasn't consumed their full share for that period. Heavy usage further in the past is also considered, but has less impact on the job priority (older jobs and usage levels have exponentially less impact than recent ones).

Additionally, as GPUs are a scarce resource, each user is limited to to 8 GPU jobs either running or queued. Further GPU jobs are blocked until running GPU jobs complete, allowing other users to queue jobs ahead of the blocked jobs.

Node Sharing

In order to maximize efficiency of the overall cluster, jobs requiring fewer than 28 cores per compute node may share a node with other jobs. This can be overridden with the `--exclusive` flag to `sbatch`, but consider that many jobs won't see a performance benefit from exclusive access when using fewer than all 28 cores in a compute node.

Additionally, the cluster nodes use Slurm's support for Linux Control Groups (cgroups) to ensure that a job does not consume more CPU resources than it allocates. For example, if an MPI or similar parallel job allocates 8 cores on a node, but attempts to use more than 8 cores, that job's processes will be limited to no more than 8 cores. This prevents a misconfigured job from affecting other users' performance on the shared node.

Memory

When a job using any queue other than bigmem or hugemem reserves a CPU core, the scheduler reserves 2000 MB of RAM. As a given compute node has at least 64 GB of RAM and 28 CPU cores, this ensures that a given node won't be overloaded on memory requirements in most cases. If your job requires substantially more or less RAM per process, feel free to modify your memory reservation with the `--mem=X` or `--mem-per-cpu=X` flags to `sbatc` or in your job script (where the units of memory are in megabytes by default). Jobs in the bigmem queue reserve 12000 MB of RAM per CPU core, and jobs in the hugemem queue reserve 28000 MB of RAM per CPU core.

Memory available in various nodes and queues:

Node(s)	Physical RAM (GB)	RAM Available to Jobs (MB)	Available Queues
node001-node022	64	62000	batch, debug, interactive
node023-node034	128	126000	batch, debug, interactive
node035-node040	256	254000	batch, debug, interactive
gpunode001-gpunode003	384	382000 (318000 available to jobs in bigmem queue)	gpu, gpu-debug, gpu-interactive, bigmem
gpunode004	896	894000 (830000 available to jobs in hugemem queue)	gpu, gpu-debug, gpu-interactive, hugemem

Job Qualities of Service (QOSes)

QOSes can be used to exceed the default scheduler limits. All users have access to a `normal` QOS that uses the limits in the Job Scheduler Policies section. If a user has blocked jobs due to exceeding their time reservation limit, HPC administrators have the option to move blocked jobs to an `expedited` QOS with looser limits. This will be done if **all** of the following conditions are met:

1. The user requests HPC adminstrators to move the job to the `expedited` QOS,
2. The overall HPC load is low (as determined by the HPC administrators),
3. The jobs will finish in a short enough time to run little risk of excessively delaying jobs that could be submitted by other users in the near future (as determined by the HPC administrators).

Other QOSes may be defined in the future, for example, a QOS available to all users with looser limits, but where jobs in that QOS could be suspended by jobs in the `normal` QOS, and resumed after those jobs are complete.

How helpful was this information?

Your Rating: ☆☆☆☆☆ Results: ★★★★☆ 66 rates