Processes and Jobs

In computing, a process is an instance of a computer program that is being executed. It contains the program code and its current activity. A process is normally launched by invoking it by the name of the executable (compiled code) associated with it, either directly at the Unix shell prompt, or within a shell script.

A job in Slurm (the scheduler on Habanero) is an allocation of resources assigned to a user for a specified amount of time. Within a job, process or processes can be launched: for batch jobs that is done in the submit script used for running the job, and for interactive jobs at the shell prompt that appears on running one. All such processes execute on the compute (aka “execute”) nodes, which are dedicated precisely to that purpose.

Restrictions on Login Node Usage

After logging in, you land on a login (aka “head”) node, from which users normally launch their jobs. The login node has some restrictions on the scope of processes that can be run on it. In order to allow for special projects and activities, these restrictions are still quite lenient at this time, and we strongly rely on users to severely limit launching processes on the head node, including those involving extensive transfer of data.

When a user ignores these recommendations and executes processes that are compute- and I/O-intensive, longer than momentary, and especially requiring multiple cores, the login node becomes overloaded, preventing other users from doing their regular work. In such cases, we typically try to terminate the processes and inform the user about it with a request to run the processes within jobs mechanism instead. However, we cannot do so at all hours and that's why users' cooperation becomes crucial. Be aware that the cluster is a shared resource and please limit the computing activity on the head node to absolute minimum.

If you need to run a CPU intensive process for longer than a couple of short minutes or if you need to use more than one core, please launch a batch job, or start an interactive job as described below:

https://wikis.cuit.columbia.edu/confluence/display/rcs/Habanero+-+Submitting+Jobs#Habanero-SubmittingJobs-InteractiveJobs

In addition, all non-trivial data copies and transfers must be performed on our two dedicated transfer nodes, generically named ‘habaxfer’. To log in to one of them, use the following command before initiating the transfer:

`ssh habaxfer.rcs.columbia.edu`

Jobs can request compute resources on a per-core basis or a per-node basis.

Core-Based Jobs

Jobs wishing to use less than a full node should specify the number of cores required. If you're not sure how many cores to request then 1 is most likely to be the correct number to use. The maximum number of cores available on a node is 24, but if that really is the number of cores you need you should probably request an entire node instead (see the Node-Based Jobs section below).

Cores can be requested using either `-c` or `--cpus-per-task` (in this and the following examples, “or” indicates an exactly equivalent alternative syntax).

```
#SBATCH --cpus-per-task 1
```
It is important to also specify your per core memory requirement as this will allow the scheduler to ensure that there will be enough memory available on the node where your job runs - otherwise your job will be allocated only 4,000 MB per core (of course, you may also choose a lower number if appropriate).

Node-Based Jobs

Jobs can also request entire nodes. This is often the most efficient way to run parallel distributed jobs, such as those using MPI. To request an entire node or nodes, specify the --exclusive flag in your submit file.

```bash
#SBATCH --exclusive
```

To specify the number of nodes, use -N or --nodes.

```bash
#SBATCH -N 1
```

or

```bash
#SBATCH --nodes=1
```

You will most likely still need to specify per core memory requirement when requesting exclusive use of a node, especially if you use less than 24 cores and want avail yourself of the bulk of the node's memory.

If your job requires more than the standard 128 GB, you should include a constraint to request one of the cluster's high memory nodes, each of which has 512 GB of memory. The feature to request is "mem512"; see below for an example of its use.

```bash
#SBATCH -C mem512
```

or

```bash
#SBATCH --constraint=mem512
```
Memory Requests

There are two ways to ask for memory and they are mutually exclusive. You can ask either for
1) memory per cpu
or
2) memory per node

If you do not specify the memory requirement, by default you get 4 gb per CPU.

All of Habanero's nodes have at least 128 gb of memory and we have 41 high memory nodes that have 512 GB of memory. Please note that some of it is used for system processes, so you can use 125 or 500 GB.

For example,
`--mem-per-cpu=5gb`
Minimum memory required per allocated CPU. If you request 24 cores (one node) you will get 120gb of memory on both standard node and on high-memory node.

If you specify the real memory required per node.
`--mem = 120gb`
You will get the same.

However, if you specify
`#SBATCH --exclusive`
`#SBATCH -C mem512`
`#SBATCH --mem = 500gb`
You will get 500 gb on high memory node.

Since we only have 41 node with 512gb of memory, perhaps it will take little bit longer for node to be available. If 120gb of memory would work for you than you would have access to all nodes.

Interactive Jobs

Interactive jobs allow user interaction during their execution. They deliver to the user a new shell from which applications can be launched.

To submit an interactive job, run the following, where "<ACCOUNT>" is your group's account name.

```
srun --pty -t 0-01:00 -A <ACCOUNT> /bin/bash
```

The Slurm directives noted below that begin with #SBATCH are available on the command line as well for interactive jobs.

For example, to run an interactive job on a large node, use:

```
srun --pty -t 0-01:00 -C mem512 -A <ACCOUNT> /bin/bash
```

If a node is available, it will be picked for you automatically, and you will see a command line prompt on a shell running on it. If no nodes are available, your current shell will wait.

Test / Debug Queue

Slurm queues, called partitions, are for the most part configured automatically by the scheduler based on your account and resource request (walltime, etc.).
The exception to this is the **test** partition.

The test partition is dedicated to interactive work and testing of code before submitting to batch queues. There are at least 3 standard nodes dedicated to this queue, so if the default partitions are full or you’re waiting a long time for an interactive session in the manner above, you may benefit from using this partition.

Note, the test partition does not currently include any GPU nodes.

<table>
<thead>
<tr>
<th>Partition</th>
<th>Max Cores Per Job</th>
<th>GB per node</th>
<th>Time Limit</th>
<th>Max Jobs/User</th>
<th>Max Submitted Jobs/User</th>
</tr>
</thead>
<tbody>
<tr>
<td>test</td>
<td>24</td>
<td>128 GB</td>
<td>4 hours</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

To submit to the test partition, add `-p test` to your srun command as shown below:

```
$ srun -p test --pty -t 0-01:00 -A <ACCOUNT> /bin/bash
```

**Basic Job Directives**

The following table lists the most common (in our estimation) directives used in Slurm submit scripts. Each should be preceded by `#SBATCH` when used in a submit script. Many directives have a short alternate name and these are listed where available. The examples sometimes use the long version of a given directive and sometimes the short version; in either case no preference is implied.

<table>
<thead>
<tr>
<th>Directive</th>
<th>Short Version</th>
<th>Description</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--account=&lt;account&gt;</code></td>
<td><code>-A &lt;account&gt;</code></td>
<td>Account.</td>
<td><code>#SBATCH --account=stats</code></td>
<td></td>
</tr>
<tr>
<td><code>--job-name=&lt;job name&gt;</code></td>
<td><code>-J &lt;job name&gt;</code></td>
<td>Job name.</td>
<td><code>#SBATCH -J DiscoProject</code></td>
<td></td>
</tr>
<tr>
<td><code>--time=&lt;time&gt;</code></td>
<td><code>-t &lt;time&gt;</code></td>
<td>Time required.</td>
<td><code>#SBATCH --time=10:00:00</code></td>
<td>The maximum time allowed is five days.</td>
</tr>
<tr>
<td><code>--mem=&lt;memory&gt;</code></td>
<td>Memory required per node.</td>
<td><code>#SBATCH --mem=16gb</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--constraint=mem512</code></td>
<td>Specifying a large (512 GB RAM) node.</td>
<td><code>#SBATCH -C mem512</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--cpus-per-task=&lt;cpus&gt;</code></td>
<td>CPU cores per task.</td>
<td><code>#SBATCH -c 1</code></td>
<td>Nodes have a maximum of 24 cores.</td>
<td></td>
</tr>
<tr>
<td><code>--nodes=&lt;nodes&gt;</code></td>
<td>Nodes required for the job.</td>
<td><code>#SBATCH -N 4</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--array=&lt;indexes&gt;</code></td>
<td>Submit a job array.</td>
<td><code>#SBATCH -a 1-4</code></td>
<td>See below for discussion of job arrays.</td>
<td></td>
</tr>
<tr>
<td><code>--mail-type=&lt;ALL,BEGIN,END,FAIL, NONE&gt;</code></td>
<td>Send email job notifications</td>
<td><code>#SBATCH --mail-type=ALL</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--mail-user=&lt;email_address&gt;</code></td>
<td>Email address</td>
<td><code>#SBATCH --mail-user=me@email.com</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Walltime**

The walltime is specified with `-t` flag. For example:

```
#SBATCH -t 10:00:00
```

That is walltime format that translates to 10 hours (00 minutes and 00 seconds). If you want to request just 1 hour walltime, you should request 1:00:00.

Acceptable time formats in Slurm scheduler are: "minutes", "minutes:seconds", "hours:minutes:seconds", "days-hours", "days-hours:minutes" and "days-hours:minutes:seconds".
The maximum time allowed is five days (or 6 hours for free group users).

Job arrays

Multiple copies of a job can be submitted by using a job array. The --array option can be used to specify the job indexes Slurm should apply.

An existing submit file can be used to submit a job array by adding the flag to the sbatch command line.

```bash
$ sbatch -a 1-5 helloworld.sh
Submitted batch job 629249
```

In this example the job IDs will be the number 629249 followed by _1, _2, etc. so the first job in the array can be accessed using the job ID 629249_1 and the last 629249_5.

```bash
$ scontrol show job 629249_1
```

Note: There is a limit of 1,001 max job elements in an job array. If you try to submit more than 1,001 elements, the scheduler issues the following:

"Batch job submission failed: Invalid job array specification".

The environment variable $SLURM_ARRAY_TASK_ID indicates the index of the array element (i.e. job) in the job array, and is accessible from within that job.

Job scheduling basics

The walltime limit on the cluster is 5 days (120 hours).

You have priority access to the nodes owned by your group. That implies that the maximum time a job from your group will need to wait for one of those nodes is 12 hours (and that occurs only in one of those rare case when a job by someone from another group starts running on such a node just before you or someone else in your group launch yours).

In order to guarantee that, users are allowed to run on other groups' nodes only for maximum of 12 hours.

Obviously, public nodes are not subject to such restrictions and everyone competes for those with the walltime limit of 5 days.

Job limits

Note: There is a limit of 1,005 max jobs running per user. Any jobs that exceed this limit will remain in queue until (some of) the user's other running jobs complete.

Additionally, there is a limit of 5,000 max submitted jobs per user simultaneously. If you try to submit more than 5,000 jobs simultaneously, the schedule spits out the following:

sbatch: error: Batch job submission failed: Job violates accounting/QOS policy (job submit limit, user's size and/or time limits
See the Slurm Quick Start Guide for a more in-depth introduction on using the Slurm scheduler.