SLURM
JOB SCHEDULER
Slurm is an open source cluster management and job scheduling system for Linux clusters.

1. Keeps track of available resources on the cluster
2. Collects users resources requests for jobs
3. Assign priorities to jobs
4. Run jobs on assigned compute nodes

www.slurm.schedmd.com
## Partitions

Compute nodes are grouped into logical sets called **partitions** depending on their hardware characteristics or function:

<table>
<thead>
<tr>
<th>Partitions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>production</strong> <em>(default)</em></td>
<td>Standard CPU nodes</td>
</tr>
<tr>
<td><strong>debug</strong></td>
<td>Standard CPU nodes for debug <em>(fast allocation times)</em></td>
</tr>
<tr>
<td><strong>maxwell</strong></td>
<td>Nodes with Nvidia Maxwell GPUs</td>
</tr>
<tr>
<td><strong>pascal</strong></td>
<td>Nodes with Nvidia Pascal GPUs</td>
</tr>
<tr>
<td><strong>mic</strong></td>
<td>Nodes with Intel Xeon Phi cards</td>
</tr>
</tbody>
</table>

Ask ACCRE if you would like to get access to specific partitions.
1. **Determine the resources necessary for the specific job**

2. **Create a batch job script**

3. **Submit the job to the scheduler**

4. **Check job status**

5. **Retrieve job information**
**DETERMINE RESOURCES FOR JOB**

**NUMBER OF CPU CORES**
- From 1 to the maximum allowed for your group’s account.
- Default is one CPU core.

**AMOUNT OF MEMORY**
- Up to 246 GB per node.
- Default is 1 GB per core.

<table>
<thead>
<tr>
<th>GB per node</th>
<th># nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>90</td>
</tr>
<tr>
<td>44</td>
<td>45</td>
</tr>
<tr>
<td>58</td>
<td>55</td>
</tr>
<tr>
<td>120</td>
<td>344</td>
</tr>
<tr>
<td>246</td>
<td>44</td>
</tr>
</tbody>
</table>

**TIME**
- Job duration on production can be set up to **14 days**.
- Default is 15 minutes.
- **DEBUG QUEUE**: max 30 minutes

---

**Slurm will immediately kill your job if your process exceeds the requested amount of resources.**

**Slightly overestimate the requested job resources, but do not greatly overestimate to avoid unnecessary long wait times.**
Backfill scheduling will start lower priority jobs if doing so does not delay the expected start time of any higher priority job.

John
12 CPU cores
1 week

Mark
2 CPU cores
5 hours

Lucy
1 CPU core
7 hours
Determine resources for job - Optimization

How to define the right amount of resources for my job?

1. Select a representative job
2. Overestimate resources
3. Run test job
4. Check actual resources utilization
5. Optimize resources request
6. Run production jobs

Optimized requested resources

Lower queue wait time + More research
CREATE A BATCH JOB SCRIPT

A **batch job** consists of a sequence of commands listed in a file with the purpose of being executed by the OS as a single instruction.

**SHEBANG**
- Specify the script interpreter (Bash)
- Must be the first line!

**SLURM DIRECTIVES**
- Start with “#SBATCH”:
  - Parsed by Slurm but ignored by Bash.
  - Can be separated by spaces.
  - Comments between and after directives are allowed.
  - Must be before actual commands!

```
#!/bin/bash

myjob.slurm

# Just a comment
setpkgs -a python
./myprogram
```

```bash
#SBATCH --nodes=1  # Nodes
#SBATCH --ntasks=1
#SBATCH --mem=1G

# Max job duration
#SBATCH --time=1-06:30:00
#SBATCH --job-name=myjob
#SBATCH --output=myjob.out

# Just a comment
setpkgs -a python
./myprogram
```
CREATE A BATCH JOB SCRIPT - THE ESSENTIALS

```
--nodes=N
  • Request N nodes to be allocated. (Default: N=1)

--ntasks=N
  • Request N tasks to be allocated. (Default: N=1)
  • Unless otherwise specified, one task maps to one CPU core.

--mem=NG
  • Request N gigabytes of memory per node. (Default: N=1)

--time=d-hh:mm:ss
  • Request d days, hh hours, mm minutes and ss seconds. (Default: 00:15:00)

--job-name=<string>
  • Specify a name for the job allocation. (Default: batch file name)

--output=<file_name>
  • Write the batch script’s standard output in the specified file.
  • If not specified the output will be saved in the file: slurm-<jobid>.out
```
CREATE A BATCH JOB SCRIPT - EMAIL NOTIFICATION

--mail-user=<address>

- Send email to address.
- It accepts multiple comma separated addresses.

--mail-type=<event>

- Define the events for which you want to be notified:

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td>Job begins</td>
</tr>
<tr>
<td>END</td>
<td>Job ends</td>
</tr>
<tr>
<td>FAIL</td>
<td>Job fails</td>
</tr>
<tr>
<td>ALL</td>
<td>BEGIN+END+FAIL</td>
</tr>
<tr>
<td>TIME_LIMIT_50</td>
<td>Elapsed time reaches 50% of allocated time</td>
</tr>
<tr>
<td>TIME_LIMIT_80</td>
<td>Elapsed time reaches 80% of allocated time</td>
</tr>
<tr>
<td>TIME_LIMIT_90</td>
<td>Elapsed time reaches 90% of allocated time</td>
</tr>
</tbody>
</table>
**Submit Job to the Scheduler**

- **sbatch** *batch_file*
  - Submit *batch_file* to Slurm.
  - If successful, it returns the job ID of the submitted job.

---

**Submission**

- Job is added to the queue

**Priority**

- A priority value is assigned to the job.

**Wait**

- Job waits in queue until:
  1. Resources are available
  2. There are no jobs with higher priority in queue

**Allocation and Execution**

- **scancel** *jobid*
  - Cancel the job corresponding to the given *jobid* from the queue.

---

**How do I remove a job from the queue?**
How is my job’s priority calculated?

**FAIRSHARE**
Prioritizes jobs belonging to underserviced accounts.
It reflects:
1. The share of resources contributed by your research group.
2. The historical amount of computing resources consumed by your account.

**AGE**
The longer the job waits in queue, the larger its age factor becomes.

**JOB SIZE**
Jobs requesting more CPUs are favored.

**PRIORITY**
# Check Job Status

Use the command `squeue -u vunetid` to show the queued jobs for user `vunetid`.

```
$ squeue -u vanzod

<table>
<thead>
<tr>
<th>JOBDID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9528424</td>
<td>production</td>
<td>mdrun_1</td>
<td>vanzod</td>
<td>R</td>
<td>1-03:53:33</td>
<td>1</td>
<td>vmp825 (Priority)</td>
</tr>
<tr>
<td>9528421</td>
<td>production</td>
<td>mdrun_2</td>
<td>vanzod</td>
<td>PD</td>
<td>0:00</td>
<td>2</td>
<td>(AssocGrpCpuLimit)</td>
</tr>
<tr>
<td>9528398</td>
<td>production</td>
<td>mdrun_3</td>
<td>vanzod</td>
<td>PD</td>
<td>0:00</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
```

## Status

- **R** = Running
- **PD** = Pending
- **CA** = Cancelled

## NodeList (Reason)

- For running jobs shows the allocated nodes.
- For pending jobs shows the wait reason:

<table>
<thead>
<tr>
<th>Reason</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Priority</strong></td>
<td>Other jobs in queue have higher priority.</td>
</tr>
<tr>
<td><strong>Resources</strong></td>
<td>Insufficient resources available on the cluster.</td>
</tr>
<tr>
<td><strong>AssocGrpCpuLimit</strong></td>
<td>Reached maximum number of allocated CPUs by all jobs belonging to the user’s account.</td>
</tr>
<tr>
<td><strong>AssocGrpMemLimit</strong></td>
<td>Reached maximum amount of allocated memory by all jobs belonging to the user’s account.</td>
</tr>
<tr>
<td><strong>AssocGrpTimeLimit</strong></td>
<td>Reached maximum amount of allocated time by all jobs belonging to the user’s account.</td>
</tr>
</tbody>
</table>
**Retrieve Job Information**

---

*rtracejob jobid*

- Print requested and utilized resources (and more) for the given *jobid*.

---

<table>
<thead>
<tr>
<th>User: vanzod</th>
<th>JobID: 9837216</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>accre</td>
</tr>
<tr>
<td>Job Name</td>
<td>test_job</td>
</tr>
<tr>
<td>State</td>
<td>Completed</td>
</tr>
<tr>
<td>Exit Code</td>
<td>0:0</td>
</tr>
<tr>
<td>Wall Time</td>
<td>3:00:00:00</td>
</tr>
<tr>
<td><strong>Requested Memory</strong></td>
<td>40Gn</td>
</tr>
<tr>
<td><strong>Memory Used</strong></td>
<td>40333256K</td>
</tr>
<tr>
<td>CPUs Requested</td>
<td>8</td>
</tr>
<tr>
<td>CPUs Used</td>
<td>8</td>
</tr>
<tr>
<td>Nodes</td>
<td>1</td>
</tr>
<tr>
<td>Node List</td>
<td>vmp372</td>
</tr>
<tr>
<td>Wait Time</td>
<td>5.2 minutes</td>
</tr>
<tr>
<td><strong>Run Time</strong></td>
<td>452.0</td>
</tr>
<tr>
<td>Submit Time</td>
<td>Mon Aug 8 09:14:53 2016</td>
</tr>
<tr>
<td>Start Time</td>
<td>Mon Aug 8 09:14:55 2016</td>
</tr>
<tr>
<td>End Time</td>
<td>Mon Aug 8 16:46:56 2016</td>
</tr>
<tr>
<td><strong>Today's Date</strong></td>
<td>Mon Aug 8 16:51:13 2016</td>
</tr>
</tbody>
</table>

---

⚠️ The used memory may not be an exact value. Take it with reservations.
JOB ARRAYS

Submit multiple similar jobs with a single job batch script. To each job within the array is assigned a unique task ID.

--array=start-end[:step][%limit]

- Define task ID interval from start to end as unsigned integer values.
- The step between successive values can be set after colon sign.
- Set the limit to the number of simultaneously running jobs with “%”.
- Individual task IDs can be specified as a comma separated values list.

- --array=0-7 → 0, 1, 2, 3, 4, 5, 6, 7
- --array=1-13:3 → 1, 4, 7, 10, 13
- --array=2,3,6,15 → 2, 3, 6, 15

All jobs in a job array must have the same resource requirements.
The maximum array size is 30,000 jobs.

Significantly shorter submission times than submitting jobs individually.
How to select different input/output for each job in the array?

Use Slurm environment variable: `SLURM_ARRAY_TASK_ID`

The task ID for the specific job in the array.

```bash
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --time=00:05:00
#SBATCH --job-name=job_array
#SBATCH --array=1-4
#SBATCH --output=job_%A_task_%a.out

my_program file_${SLURM_ARRAY_TASK_ID}
```

**Executed command**

- my_program file_1
- my_program file_2
- my_program file_3
- my_program file_4

**Output file**

- job_1234567_task_1.out
- job_1234567_task_2.out
- job_1234567_task_3.out
- job_1234567_task_4.out
What if my input files do not have a numerical index?

```bash
#!/bin/bash
#SBATCH ...

myfile=$( ls DataDir | awk -v line=${SLURM_ARRAY_TASK_ID} '{if (NR==line) print $0}' )

my_program ${myfile}
```

1. Get the list of files names in the data directory in alphabetical order
2. Send the list to awk
3. Pass the value of the bash variable SLURM_ARRAY_TASK_ID to the awk variable “line”
4. Print only the NRth line in the list of files names for which NR corresponds to the job task ID
5. Pass the file name in the myfile variable to the main program
MULTITHREADED JOBS

OpenMP

POSIX THREADS

- Single task with multiple concurrent execution threads.
- Each thread uses a single CPU core.
- All threads share the same allocated memory.

Single node only!

1 node
1 task
8 CPUs per task

1 node
2 tasks
4 CPUs per task
MULTITHREADED JOBS

--cpus-per-task=N

- Request $N$ CPU cores to be allocated for each task.

With OpenMP in your batch script don’t forget to set:

```
export OMP_NUM_THREADS = $SLURM_CPUS_PER_TASK
```
Distributed Memory Jobs

**Message Passing Interface (MPI)**

- Multiple tasks with private memory allocations.
- Tasks exchange data through communications.
- Tasks can reside on the same node or on multiple nodes.

**Single or multiple nodes**

- 2 nodes
- 8 tasks per node
- 1 CPU per task
DISTRIBUTED MEMORY JOBS

In the batch script, run the MPI program with:

```
srun ./program_name
```

- Run MPI program called `program_name`.

--nodes=N

- Request $N$ nodes to be allocated.

--tasks-per-node=N

- Request $N$ tasks per node.
- Unless otherwise specified, one task maps to one CPU core.

Do not use `mpirun` or `mpiexec`! `srun` will use the correct launcher for the MPI library you selected via `setpkgs`.

For OpenMPI only, add the following flag to `srun`:

```
srun --mpi=pmi2 ./program_name
```
**Interactive Shell Job**

**salloc options**
- Obtain job allocation with shell access.
- Accepts all the same *options* previously seen for `sbatch`.

```
vanzod@vmps10 ~]$ salloc --nodes=1 --ntasks=4 --mem=16G --time=1:00:00
```

Recommended for debugging and benchmarking sessions.
**Troubleshooting**

Why is my job still pending?

- **Check overall cluster utilization**
- **Check your account’s resources use**
- **Check your account limits**

---

**SlurmActive -m mem**

- Show the overall cluster utilization.
- Count as available cores only the ones with at least mem amount of memory (in GB). Default: 1GB

```bash
[vanzod@vmps08 ~]$ SlurmActive -m 10
```

**Standard Nodes Info:**
- 554 of 567 nodes active (97.71%)
- 4664 of 5912 processors in use by local jobs (78.80%)
- **945 of 5912 processors are memory-starved** (15.98%)
- **303 of 5912 available processors** (5.13%)

**GPU Nodes Info:**
- Fermi: 40 of 52 GPUs in use (76.92%)
- Maxwell: 9 of 48 GPUs in use (18.75%)

**Phi Nodes Info:**
- 0 of 2 nodes active (0.00%)
- 0 of 32 processors in use by local jobs (0.00%)
- 8 of 32 processors are memory-starved (25.00%)

**ACCRE Cluster Totals:**
- 567 of 594 nodes active (95.45%)
- 4769 of 6192 processors in use by local jobs (77.02%)
- 966 of 6192 processors are memory-starved (15.60%)
- 457 of 6192 available processors (7.38%)

3041 running jobs, 2069 pending jobs, 1 jobs in unrecognized state
Why is my job still pending?

**Troubleshooting**

- Check overall cluster utilization
- Check your account’s resources use
- Check your account limits

---

**qSummary -g group**

- Show the total number of jobs and CPU cores allocated or waiting for allocation for the selected *group*.

```
[vanzod@vmps09 ~]$ qSummary -g capra_lab
GROUP    USER     ACTIVE_JOBS ACTIVE_CORES PENDING_JOBS PENDING_CORES
---------- ---------- ---------- ---------- ------------ -----------
capra_lab  chenll1  148        203         156          156
          colbrall  1          1           0            0
          fishae    3          3           0            0
          sivleyrm  125        125         156          156
          zhanjl0   16         64           0            0
```

---

**Summary**

- The qSummary command with the `-g` option shows the total number of jobs and CPU cores allocated or waiting for allocation for the selected group.

---

**Why is my job still pending?**

- Check your account’s resources use
- Check overall cluster utilization
- Check your account limits

---
**Troubleshooting**

Why is my job still pending?

**Check overall cluster utilization**

**Check your account’s resources use**

**Check your account limits**

---

**showLimits** -g `group`

- Show the cluster resources limits for a specific `group`.

---

![Command Example](command_example.png)

```
[vanzod@vm09 ~]$ showLimits -g capra_lab
```

<table>
<thead>
<tr>
<th>ACCOUNT</th>
<th>GROUP</th>
<th>FAIRSHARE</th>
<th>MAXCPUS</th>
<th>MAXMEM(GB)</th>
<th>MAXCPUTIME(HRS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>capra_lab_account</td>
<td>16</td>
<td></td>
<td>272</td>
<td>2720</td>
<td>26112</td>
</tr>
<tr>
<td>capra_lab</td>
<td>1</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

---

**Users in the same group share the same amount of resources.**
Why did my job fail?

Check with `rtracejob`:

<table>
<thead>
<tr>
<th>State</th>
<th>Exit Code</th>
<th>Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>11:0</td>
</tr>
</tbody>
</table>

A non-zero exit code means your application failed.

1. Check the job’s output file for error messages.

2. Check your Slurm batch job script for syntax or logic errors.

3. www.accre.vanderbilt.edu/slurm
NEED MORE HELP?

Check our Frequently Asked Question webpage:
www.accre.vanderbilt.edu/faq

Submit a ticket from the helpdesk:
www.accre.vanderbilt.edu/help

Open a ticket to request an appointment with an ACCRE specialist.

DO NOT submit tickets in “Rush cluster”!
Rush tickets are for cluster-wide issues only.