# SLURM Headers

Brinton Eldridge

Graduate Assistant

Research Computing

## Outline

- What is SLURM?
- Key Terminology
- Cluster Structure
  - Partitions
- Building a Simple Job
- SLURM Environment Vars
- Quality of Life Headers
- Commonly Confused/Misused Headers

Slurm Docs
(SchedMD)

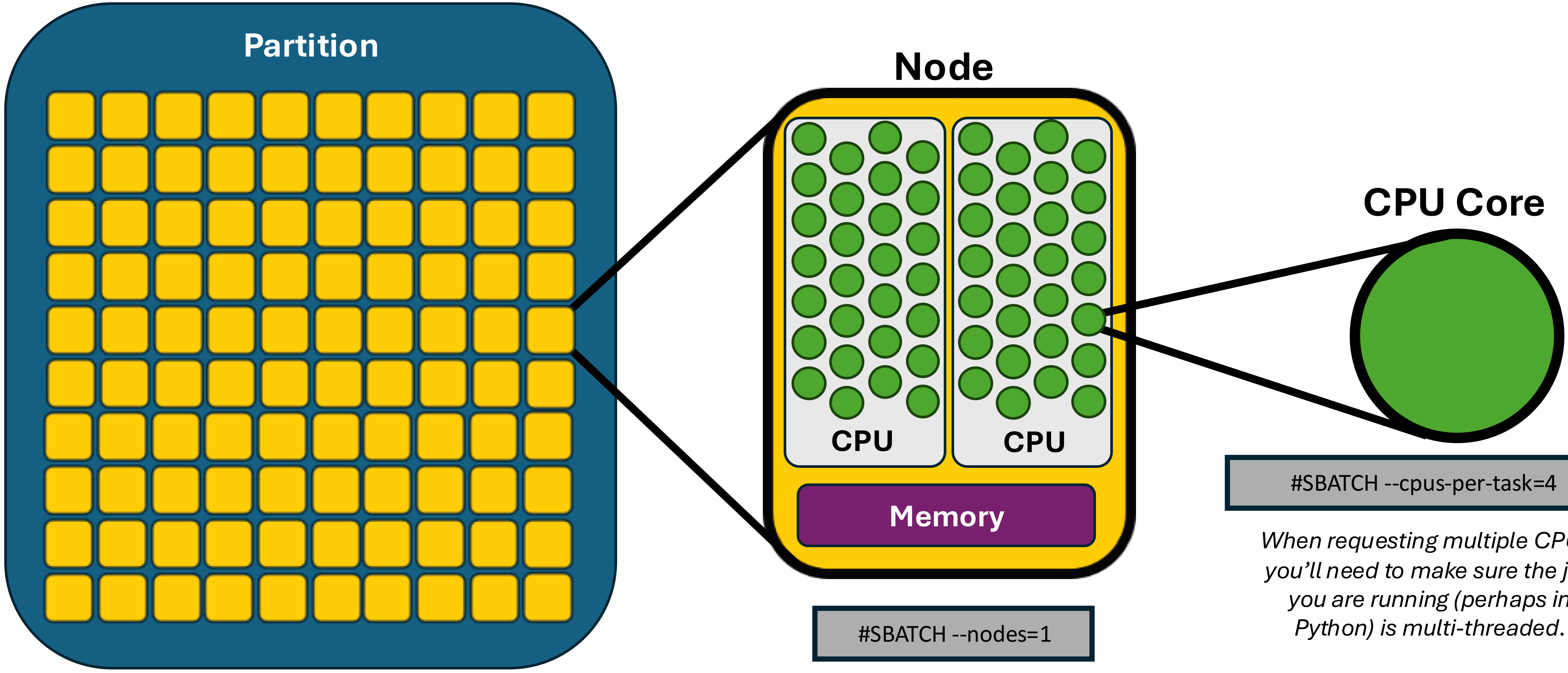
## What is SLURM?

- **S**imple **L**inux **U**tility for **R**esource **M**anagement
- It is a workload management software
  - Responsible for scheduling, running, and limiting the resources of jobs.
- This is the main way research starts on the cluster
- Two methods:
  1. `sbatch` ompanied by a submission script
  2. `srun` accompanied by several flags

Submitting a Job
(uofmwiki)

## Key Terminology

- Partition
  - A group of nodes with specific constraints (i.e., *acomputeq* or *igpuq*).
- Node
  - An individual compute machine in the cluster. Comprised of several CPU-cores.
- CPU and CPU-core
  - Technically 2 CPUs per each node, but each one is split into multiple cores.
  - Intel nodes have 20 CPU-cores per CPU (40 for the whole node).
  - AMD nodes have 96 CPU-cores per CPU (192 for the whole node).
- Task
  - A process or unit of execution in a job, which can be distributed across nodes or cores.



```
#SBATCH --partition=acomputeq
```

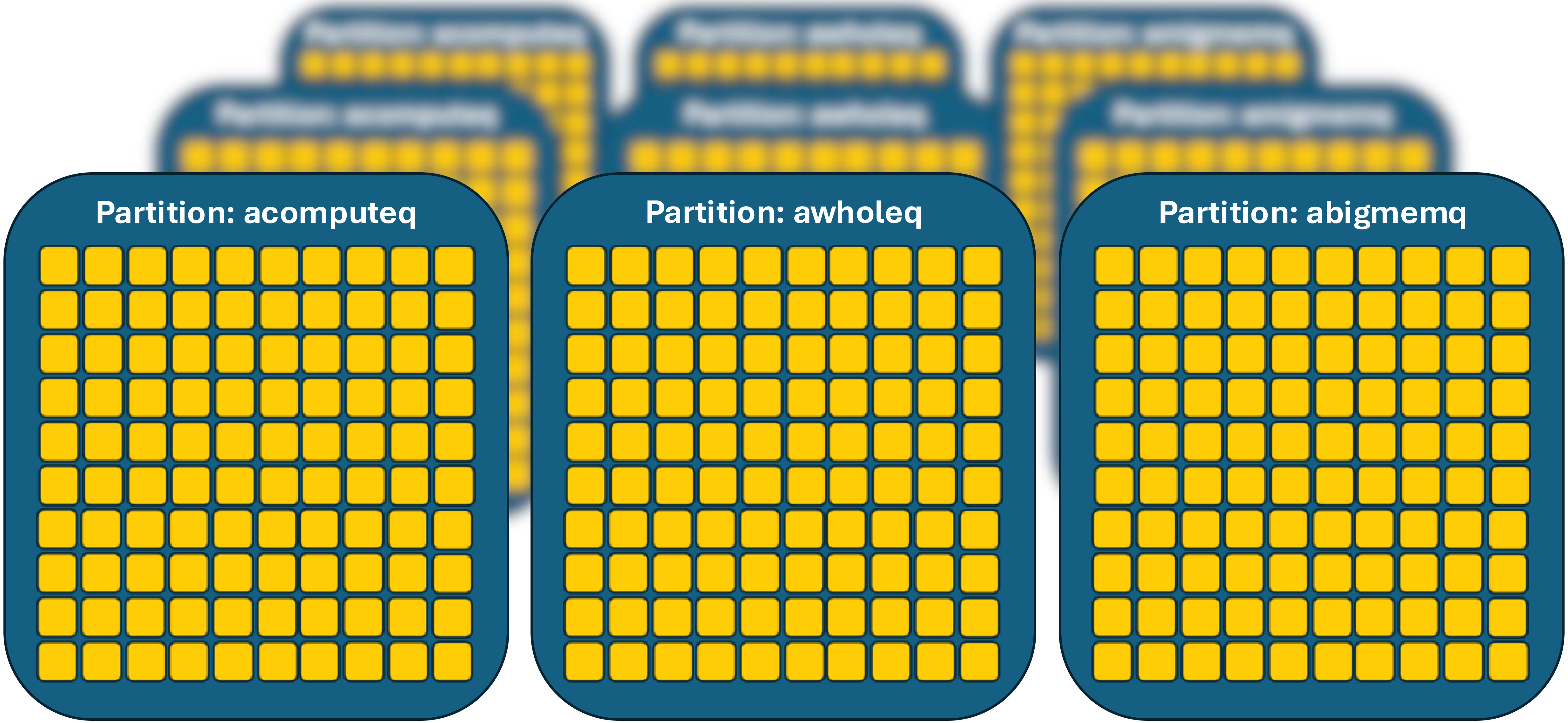
*Adding more than one partition as a comma-separated list will enable your job to queue to whichever one becomes available first.*

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

*You likely will not need to worry about requesting multiple nodes unless you are running in the awholeq/iwholeq.*

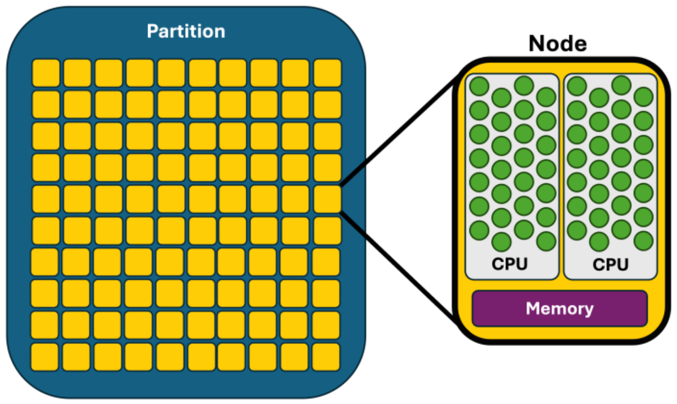
```
#SBATCH --cpus-per-task=4
```

*When requesting multiple CPUs, you'll need to make sure the job you are running (perhaps in Python) is multi-threaded.*


Partition: acomputeq
Partition: awholeq
Partition: abigmemq

```
scontrol show partition
```

| Partition | Nodes | Memory per Node | CPU Cores per Node | Additional Notes |
|---|---|---|---|---|
| acomputeq | 16 | 768 GB | 192 | |
| awholeq | 8 | 768 GB | 192 | Alloc cores in units of 192 |
| abigmemq | 4 | 1.5 TB | 192 | |
| agpuq | 4 | 768 GB | 192 | 2 A100 GPUs |
| icomputeq | 40 | 192 GB | 40 | |
| iwholeq | 38 | 192 GB | 40 | Alloc cores in units of 40 |
| ibigmemq | 4 | 1.5 TB | 40 | |
| igpuq | 6 | 192 GB | 40 | 2 V100 GPUs |



Partitions (uofmwiki)

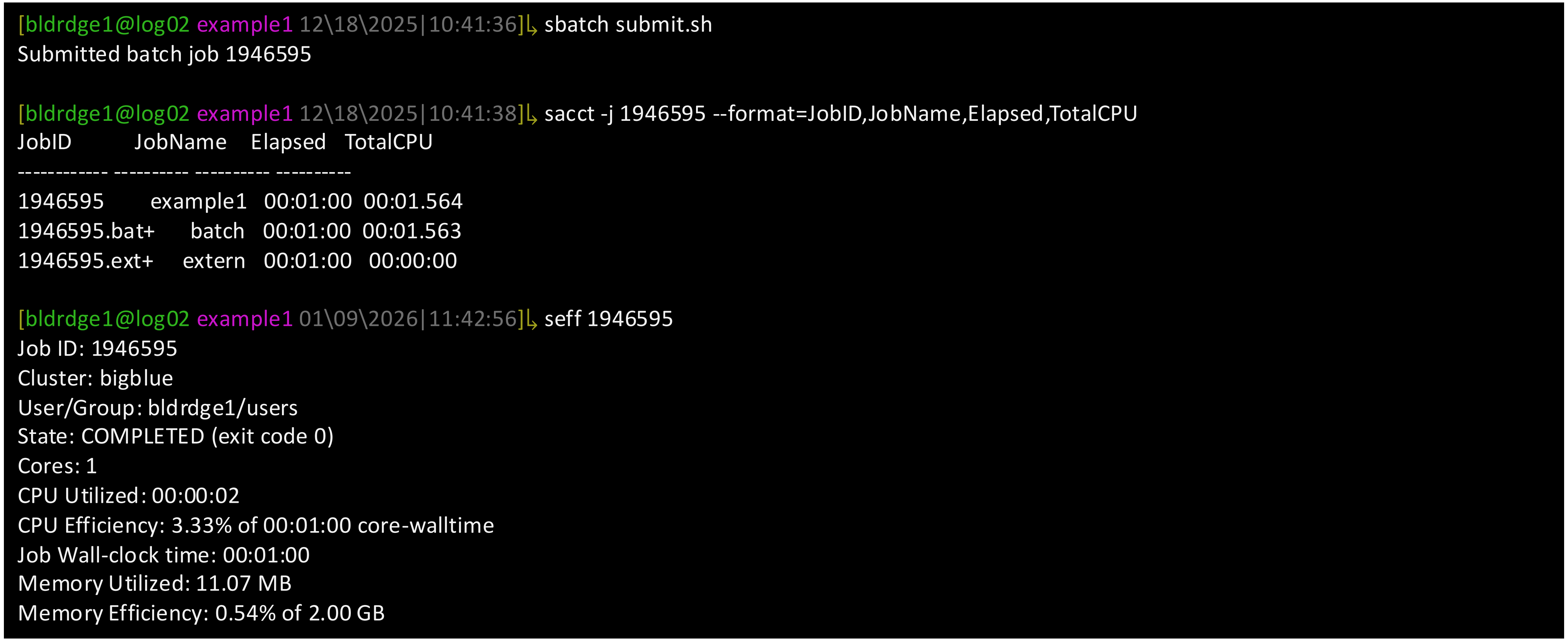
# Let’s Build a Job

```
#!/bin/bash
#SBATCH --job-name=example1
#SBATCH --time=00:15:00
#SBATCH --mem=2G
#SBATCH --partition=acomputeq



module load spack
module load bwa
wget https://tinyurl.com/2z9y433r
mv 2z9y433r 2z9y433r.gz
gunzip 2z9y433r.gz
bwa index 2z9y433r
```

Necessary*

Typical

Slurm Example
(uofmwiki)



```
[bldrdge1@log02 example1 12\18\2025|10:41:36]↳ sbatch submit.sh
Submitted batch job 1946595

[bldrdge1@log02 example1 12\18\2025|10:41:38]↳ sacct -j 1946595 --format=JobID,JobName,Elapsed,TotalCPU
JobID           JobName    Elapsed   TotalCPU
------------ ---------- ---------- ----------
1946595        example1   00:01:00  00:01.564
1946595.bat+      batch   00:01:00  00:01.563
1946595.ext+     extern   00:01:00   00:00:00

[bldrdge1@log02 example1 01\09\2026|11:42:56]↳ seff 1946595
Job ID: 1946595
Cluster: bigblue
User/Group: bldrdge1/users
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 00:00:02
CPU Efficiency: 3.33% of 00:01:00 core-walltime
Job Wall-clock time: 00:01:00
Memory Utilized: 11.07 MB
Memory Efficiency: 0.54% of 2.00 GB
```

# SLURM Environment Variables

```
#!/bin/bash
#SBATCH --job-name=example2
#SBATCH --time=00:05:00
#SBATCH --mem=2G
#SBATCH --partition=acomputeq
#SBATCH --cpus-per-task=2

python multi_sum.py $SLURM_CPUS_PER_TASK
```

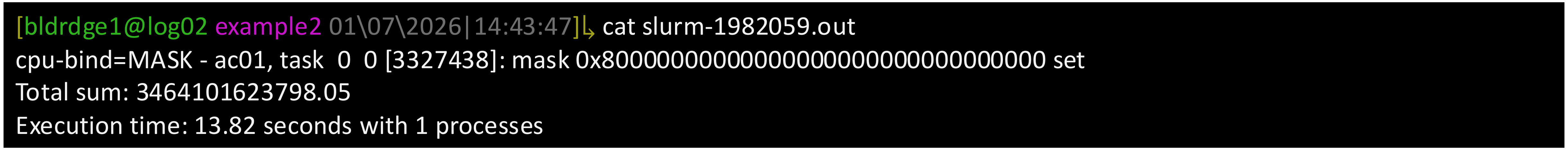
Necessary*

Typical

Important for Example

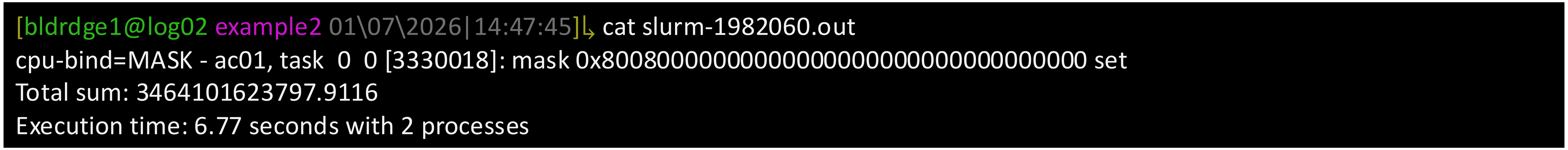
| Variable Name | Description |
|---|---|
| `SLURM_JOB_ID` | The ID of the job allocation. |
| `SLURM_JOB_NAME` | The name of the job (from `--job-name`). |
| `SLURM_JOB_NODELIST` | List of nodes allocated to the job. |
| `SLURM_JOB_NUM_NODES` | Total number of nodes in the job allocation. |
| `SLURM_JOB_CPUS_PER_NODE` | Number of CPUs available per node (format: count[xnodes] for multiples). |
| **`SLURM_CPUS_PER_TASK`** | **Number of CPUs requested per task (from `--cpus-per-task`).** |
| `SLURM_NTASKS` | Total number of tasks in the job (from `--ntasks`). |
| `SLURM_TASKS_PER_NODE` | Number of tasks to be launched per node. |
| `SLURM_MEM_PER_NODE` | Memory allocated per node (from `--mem`). |
| `SLURM_MEM_PER_CPU` | Memory allocated per CPU (from `--mem-per-cpu`). |
| `SLURM_GPUS_ON_NODE` | Number of GPUs allocated on the current node. |
| `SLURM_SUBMIT_DIR` | The directory from which `sbatch` was invoked. |
| `SLURM_ARRAY_TASK_ID` | Job array index (if using job arrays). |
| `SLURM_CLUSTER_NAME` | Name of the cluster running the job. |
| `SLURM_PROCID` | The MPI rank or relative process ID for the current process. |
| `SLURM_NODEID` | The relative ID of the current node in the allocation (0-based). |

#SBATCH --cpus-per-task=1



```
[bldrdge1@log02 example2 01\07\2026|14:43:47]↳ cat slurm-1982059.out
cpu-bind=MASK - ac01, task  0  0 [3327438]: mask 0x80000000000000000000000000000000 set
Total sum: 3464101623798.05
Execution time: 13.82 seconds with 1 processes
```

#SBATCH --cpus-per-task=2



```
[bldrdge1@log02 example2 01\07\2026|14:47:45]↳ cat slurm-1982060.out
cpu-bind=MASK - ac01, task  0  0 [3330018]: mask 0x80080000000000000000000000000000000 set
Total sum: 3464101623797.9116
Execution time: 6.77 seconds with 2 processes
```

## Quality of Life

```
#!/bin/bash
#SBATCH --job-name=example3
#SBATCH --time=00:05:00
#SBATCH --mem=2G
#SBATCH --partition=acomputeq
#SBATCH --cpus-per-task=2
#SBATCH --output=%x-%j.out
#SBATCH --mail-type=END,FAIL
#SBATCH --mail-user=bldrdge1@memphis.edu

python multi_sum.py $SLURM_CPUS_PER_TASK
```

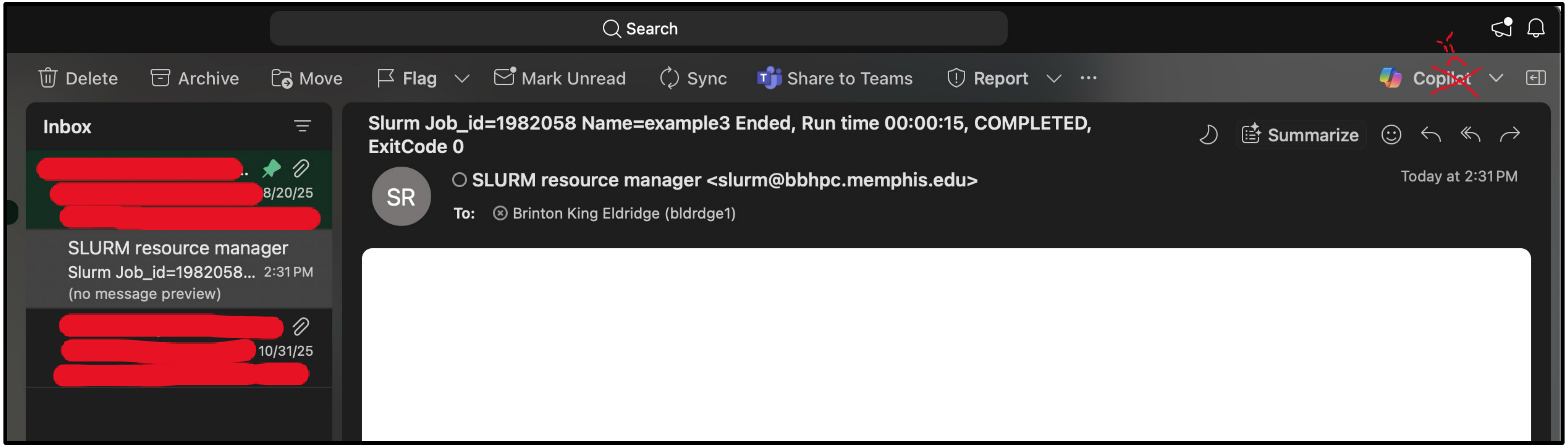
Necessary*

Typical

Important for Example

QoL

| Pattern | Description |
|---|---|
| %A | Master job allocation ID (for job arrays). |
| %a | Job array task ID (index). |
| %J | Job ID followed by step ID (format: jobid.stepid). |
| **%j** | **Job ID only (e.g., the numeric ID of the job).** |
| %N | Short hostname of the first node (creates separate files per node if used). |
| %n | Node relative ID (0-based index within the job's nodes). |
| %s | Step ID of the job step. |
| %t | Task ID (rank) within the job (creates separate files per task). |
| %u | Username of the submitting user. |
| **%x** | **Job name (from `--job-name` or the script name if unspecified).** |



Inbox

[redacted] 8/20/25

SLURM resource manager
Slurm Job_id=1982058... 2:31 PM
(no message preview)

[redacted] 10/31/25

Slurm Job_id=1982058 Name=example3 Ended, Run time 00:00:15, COMPLETED, ExitCode 0

SR SLURM resource manager <slurm@bbhpc.memphis.edu>
To: Brinton King Eldridge (bldrdge1)
Today at 2:31 PM

Typical:

```
[bldrdge1@log02 example2 01\07\2026|14:47:48]↳ ls
multi_sum.py  slurm-1982059.out  submit.sh
```

#SBATCH --output=%x-%j.out

```
[bldrdge1@log02 example3 01\07\2026|14:42:29]↳ ls
example3-1982058.out  multi_sum.py  submit.sh
```

## Commonly Confused/Misused Headers

```
#!/bin/bash
#SBATCH --job-name=example4
#SBATCH --time=00:05:00
#SBATCH --mem=2G
#
#       or
#
#SBATCH --mem-per-cpu=2G
#SBATCH --partition=acomputeq
#SBATCH --cpus-per-task=2


python multi_sum.py $SLURM_CPUS_PER_TASK
```

2GB of memory requested over the whole node, no matter how many CPUs the job accesses.

2GB of memory requested for each CPU accessed.
--mem = --mem-per-cpu * --cpus-per-task

though this is not the only way to request CPUs

## Commonly Confused/Misused Headers

```
#!/bin/bash
#SBATCH --job-name=example4
#SBATCH --time=00:05:00
#SBATCH --mem-per-cpu=2G
#SBATCH --partition=acomputeq
#SBATCH --cpus-per-task=16
#
#       or
#
#SBATCH --ntasks=4
#SBATCH --cpus=4

module load openmpi/4.1.6/gcc.8.5.0/mt
# python multi_sum.py $SLURM_CPUS_PER_TASK
mpirun -n $SLURM_NTASKS python multi_sum.py $SLURM_CPUS_PER_TASK
```

16 CPUs requested for a single task (because default tasks=1).
Task must be designed to accept multiple cores.

4 CPUs requested per task, 4 tasks running at once.
Job must be designed to utilize MPI (or similar) to make parallel tasks.
Tasks must be designed to accept multiple cores.

## Commonly Confused/Misused Headers

```
#!/bin/bash
#SBATCH --job-name=example4
#SBATCH --time=00:05:00
#SBATCH --mem-per-cpu=2G
#SBATCH --partition=acomputeq
#SBATCH --cpus-per-task=16
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#
#       or
#
#SBATCH --ntasks=4
#SBATCH --ntasks-per-node=2

...
```

2 different nodes requested, each planning on running two separate tasks.

4 total tasks being requested to be distributed across nodes, with two tasks per node, thus requesting 2 nodes.

# Best Practices for Optimization

- **Test your Script**: Submitting your submission file using `sbatch --test-only submit.sh`
- **Match to Workload Type**:
  - For serial jobs, use 1 task/CPU/node.
  - For parallel jobs, request multiple tasks to handle each process.
  - For multithreaded processes, request multiple CPUs to be used in the process.
- **Add Buffers Wisely**: 10-20% extra for memory/time to handle spikes, but not more—over-buffering blocks others.
- **Combine Parameters**:
  - `--ntasks=8` `--cpus-per-task=2` `--nodes=2` for 8 tasks across 2 nodes, each with 2 CPUs.
  - Avoid conflicts like using `--mem` with `--mem-per-cpu`.
- **Batch Short Jobs**: Group small tasks into arrays (`--array`) or longer scripts to reduce scheduler load.
- **Common Pitfalls**: Ignoring defaults (leads to surprises), not re-testing after code changes, or over-constraining. `scontrol show partition` `scontrol show config`

# slurm
# cheat sheet

## Daemons

| | |
|---|---|
| slurmctld | Executes on cluster's "head" node to manage workload. |
| slurmd | Executes on each compute node to locally manage resources. |
| slurmdbd | Manages database of resources limits, licenses, and archives accounting records. |
| slurmrestd | Interface to Slurm via REST API. |

## Job Submission

**sbatch** - Submit a batch script for later execution.

**srun** - Obtain a job allocation (as needed) and execute an application.

**salloc** - Obtain a job allocation.

| | |
|---|---|
| --array=<indexes> (e.g. "--array=1-10") | Job array specification. |
| --account=<name> | Account to be charged for resources used. |
| --begin=<time>(e.g. "--begin=18:00:00") | Initiate job after specified time. |
| --clusters=<name> | Cluster(s) to run the job. (sbatch command only) |
| --constraint=<features> | Required node features. |
| --cpus-per-task=<count> | Number of CPUs required per task. |
| --dependency=<state:jobid> | Defer job until specified job(s) reach specified state. |
| --error=<filename> | File in which to store job error messages. |
| --exclusive[=user] | Allocated nodes can not be shared with other jobs/users. |
| --export=<name[=value]> | Export identified environment variables. |
| --gres=<name[:count]> | Generic resources required per node. |
| --job-name=<name> | Job name. |
| --label | Prepend task ID to output. (srun command only) |
| --licenses=<name[:count] | License resources required for entire job. |
| --mem=<MB> | Memory required per node. |
| --mem-per-cpu=<MB> | Memory required per allocated CPU. |
| -N<minnodes[-maxnodes]> | Node count required for the job. |
| -n<count> | Number of tasks to be launched. |
| --nodelist=<names> | Specific host names to include in job allocation. |
| --output=<name> | File in which to store job output. |
| --partition=<names> | Partition/queue in which to run the job. |
| --qos=<name> | Quality Of Service. |
| --signal=[B:]<num>[@time] | Signal job when approaching time limit. |
| --time=<time> | Wall clock time limit. |
| --wrap=<command_string> | Wrap specified command in a simple "sh" shell. (sbatch command only) |

## Accounting

**sacct** - Display accounting data.

| | |
|---|---|
| --allusers | Displays all users' jobs. |
| --accounts=<name> | Displays jobs with specified accounts. |
| --endtime=<time> | End of reporting period. |
| --format=<spec> | Specify the fields shown in the output. |
| --partition=<names> | Comma separated list of partitions to select jobs and job steps from. |
| --state=<state_list> | Display jobs with specified states. |
| --starttime=<time> | Start of reporting period. |
| --long | Provides output with more fields listed. . |

**sacctmgr** - View and modify account information.

Options:

| | |
|---|---|
| --immediate | Commit changes immediately. |
| --parsable | Output delimited by '\|' |

Commands:

| | |
|---|---|
| add <ENTITY> <SPECS> create <ENTITY> <SPECS> | Add an entity. Identical to the create command. |
| delete <ENTITY> where <SPECS> | Delete the specified entities. |
| show <ENTITY> [<SPECS>] | Display information about the specific entity. |
| modify <ENTITY> where <SPECS> set <SPECS> | Modify an entity. |

Entities:

| | |
|---|---|
| account | Account used to group users. |
| association | Entity on cluster, consisting of cluster, account, user and (optionally) partition. |
| cluster | ClusterName parameter in the slurm.conf. |
| qos | Quality of Service. |

**sreport** - Report job usage and cluster utilization.

| | |
|---|---|
| --allusers | Displays all users' jobs. |
| --accounts=<name> | Displays jobs with specified accounts. |

Report Types:

| | |
|---|---|
| cluster | AccountUtilizationByUser, UserUtilizationByAccount, UserUtilizationByWckey, Utilization, WCKeyUtilizationByUser |
| job | SizesByAccount, SizesByAccountAndWckey, SizesByWckey |
| reservation | Utilization |
| user | TopUsage |

Slurm Cheat Sheet
(SchedMD)

SCHEDMD
The Slurm Company