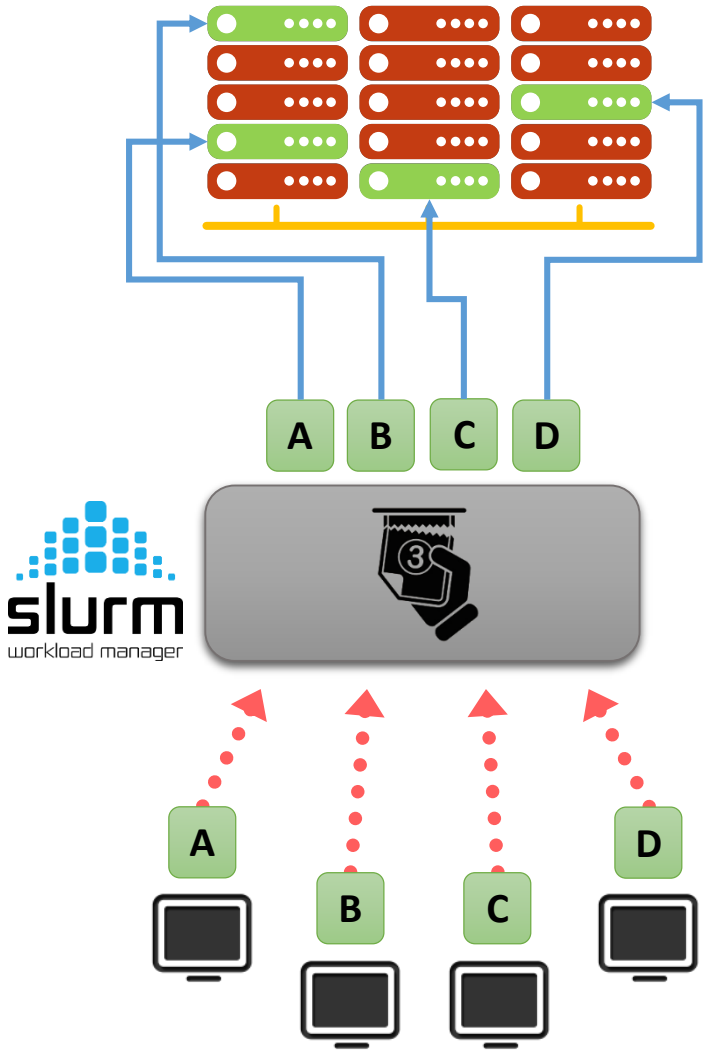
SLURM
JOB SCHEDULER

What is Slurm?

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

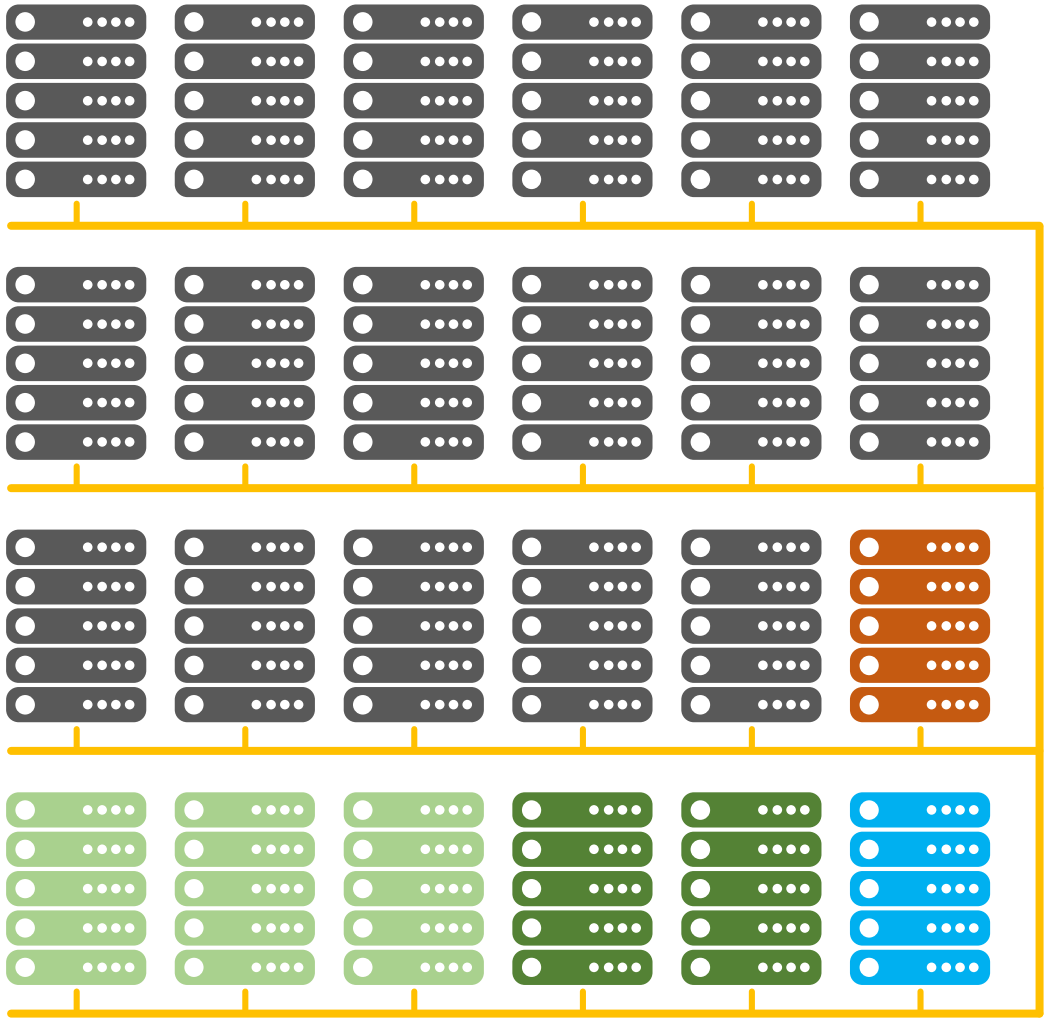
www.slurm.schedmd.com

Partitions

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

	production *(default)*	Standard CPU nodes
	debug	Standard CPU nodes for debug (*fast allocation times*)
	maxwell	Nodes with Nvidia Maxwell GPUs
	pascal	Nodes with Nvidia Pascal GPUs
	mic	Nodes with Intel Xeon Phi cards

Ask ACCRE if you would like to get access to specific partitions.



Job execution workflow

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

ACCRE
Advanced Computing Center for Research & Education

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.

GB per node	# nodes
20	90
44	45
58	55
120	344
246	44

Time

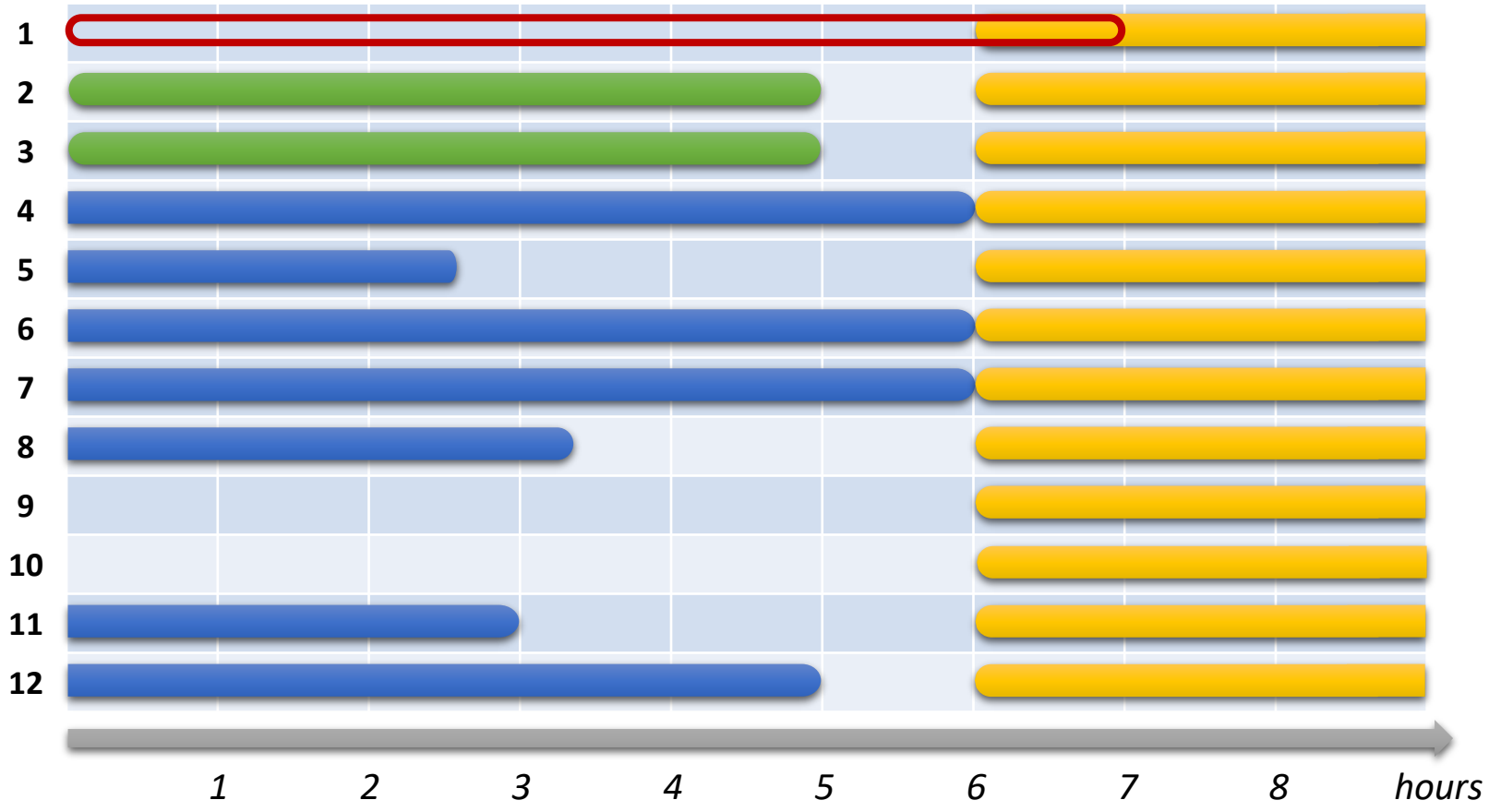
- Job duration on production can be set up to **14 days**.
- Default is 15 minutes.
- **Debug queue**: max 30 minutes

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.

Determine resources for job - Backfill

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?

Select a representative job

↓

Overestimate resources

↓

Run test job

↓

Check actual resources utilization

↓

Optimize resources request

↓

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!

Script commands

- Commands you want to execute on the compute nodes.

myjob.slurm

```
#!/bin/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:

BEGIN	Job begins
END	Job ends
FAIL	Job fails
ALL	BEGIN+END+FAIL
TIME_LIMIT_50	Elapsed time reaches 50% of allocated time
TIME_LIMIT_80	Elapsed time reaches 80% of allocated time
TIME_LIMIT_90	Elapsed time reaches 90% of allocated time

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

How do I remove a job from the queue?

`scancel` *`jobid`*

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

Submit job to the scheduler

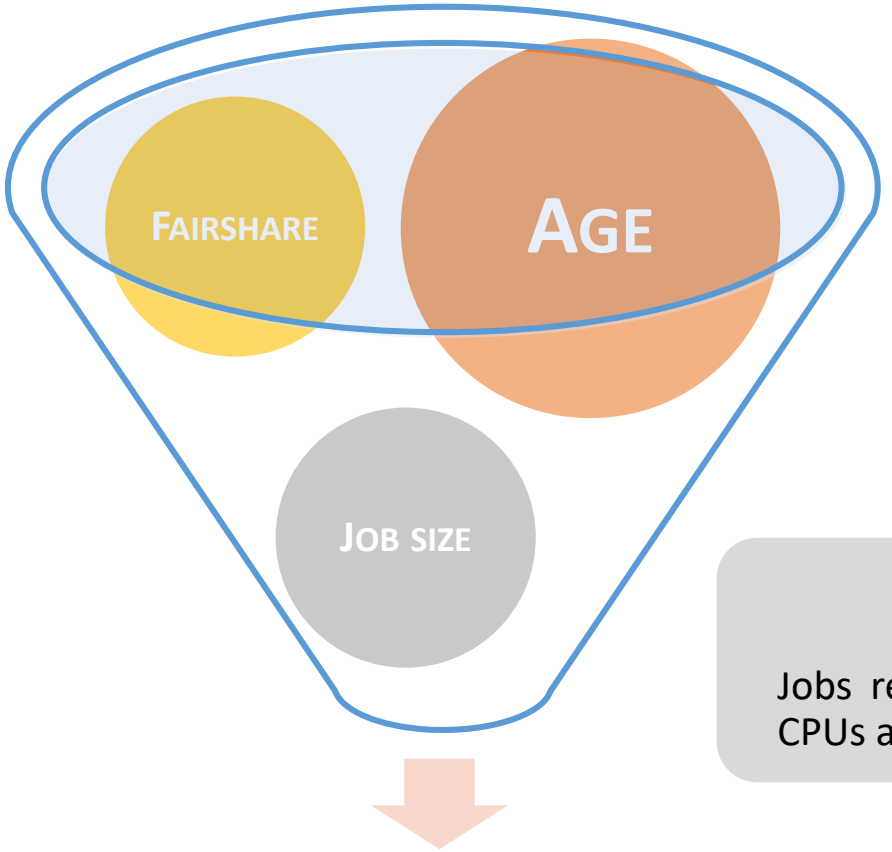
How is my job's priority calculated?

Fairshare

Prioritizes jobs belonging to under-serviced 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.

Check job status

squeue -u *vunetid*

- Show the queued jobs for user *vunetid*.

```
[vanzod@vmps10 ~]$ squeue -u vanzod
       JOBID     PARTITION        NAME       USER     ST          TIME   NODES   NODELIST(REASON)
     9528424    production     mdrun_1     vanzod      R   1-03:53:33       1   vmp825
     9528421    production     mdrun_2     vanzod     PD         0:00       2   (Priority)
     9528398    production     mdrun_3     vanzod     PD         0:00       3   (AssocGrpCpuLimit)
```

Status

R = Running

PD = Pending

CA = Cancelled

Nodelist (Reason)

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

Priority	Other jobs in queue have higher priority.
Resources	Insufficient resources available on the cluster.
AssocGrpCpuLimit	Reached maximum number of allocated CPUs by all jobs belonging to the user's account.
AssocGrpMemLimit	Reached maximum amount of allocated memory by all jobs belonging to the user's account.
AssocGrpTimeLimit	Reached maximum amount of allocated time by all jobs belonging to the user's account.

Retrieve job information

```
rtracejob jobid
```

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

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

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.

Job arrays

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.

```
#!/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	Output file
`my_program file_1`	`job_1234567_task_1.out`
`my_program file_2`	`job_1234567_task_2.out`
`my_program file_3`	`job_1234567_task_3.out`
`my_program file_4`	`job_1234567_task_4.out`

Job arrays

What if my input files do not have a numerical index?

```
#!/bin/bash
#SBATCH
…
                    1          2                    3                                    4
myfile=$( ls DataDir | awk -v line=${SLURM_ARRAY_TASK_ID} '{if (NR==line) print $0}' )

5 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

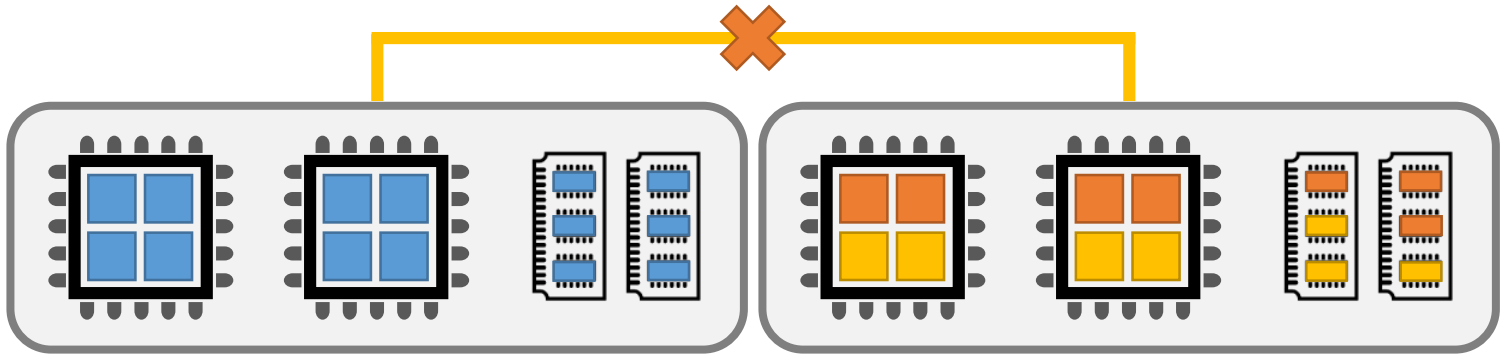
MULTITHREADED JOBS

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:

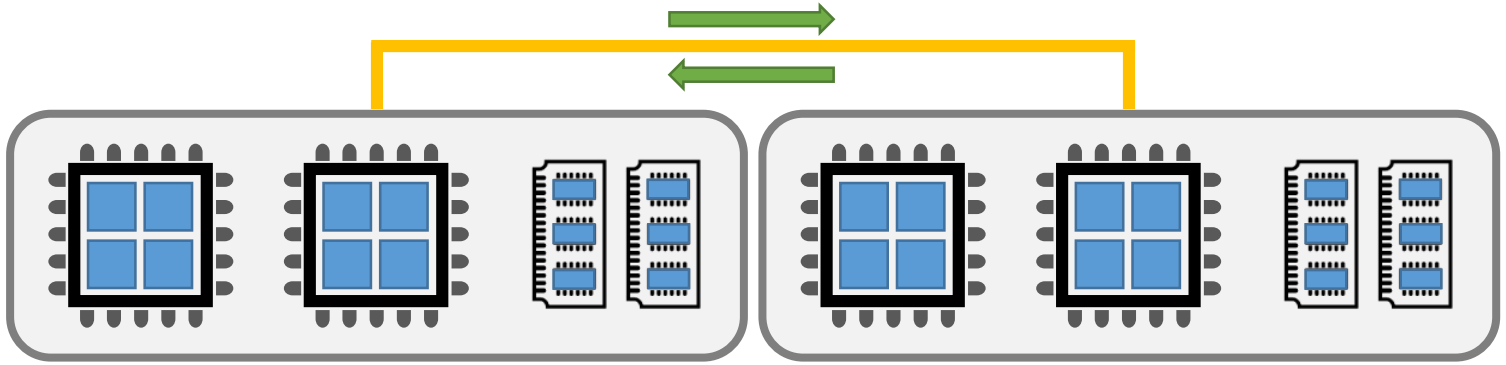
```
export OMP_NUM_THREADS = $SLURM_CPUS_PER_TASK
```

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

`--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.

In the batch script, run the MPI program with:

`srun ./program_name`

- Run MPI program called *program_name*.

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`.

Gateway

Compute node

```
[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 you 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

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

Standard Nodes Info:      554 of   567 nodes active                      ( 97.71%)
                         4664 of  5912 processors in use by local jobs    ( 78.89%)
                          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
```

TROUBLESHOOTING

Why is my job still pending?

Check overall cluster utilization

Check you 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                   148          203           156           156
            chenl11           3           10             0             0
            colbrall          1            1             0             0
            fishae            3            3             0             0
            sivleyrm        125          125           156           156
            zhanj10          16           64             0             0
-----------------------------------------------------------------------------------
```

TROUBLESHOOTING

Why is my job still pending?

Check overall cluster utilization

Check you account's resources use

Check your account limits

showLimits -g *group*

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

```
[vanzod@vmps09 ~]$ showLimits -g capra_lab
ACCOUNT          GROUP    FAIRSHARE    MAXCPUS    MAXMEM(GB)   MAXCPUTIME(HRS)
-----------------------------------------------------------------------------------
capra_lab_account            16         272        2720            26112
                 capra_lab    1           -          -                -
-----------------------------------------------------------------------------------
```

Users in the same group share the same amount of resources.

TROUBLESHOOTING

Why did my job fail?

1

Check with **rtracejob**:

```
| State          | Failed
| Exit Code      | 11:0
```

A non-zero exit code means your application failed.

2

Check the job's output file for error messages.

3

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