Agave User Environment

Gil Speyer speyer@asu.edu

MAE 598/ APM 525

August 29, 2018

ASU Knowledge Enterprise Development

ARIZONA STATE UNIVERSITY

Outline

- Get an account
- System information
- Initial login
- Transferring files
- Modules
- Compiling
- Batch System
- Job Monitoring
- Interactive mode
- Good citizenship

Anti-outline

- Shell scripting
- Installing software
- Debugging (parallel or otherwise)

For any assistance with these – please contact us at any time:

support@hpchelp.asu.edu

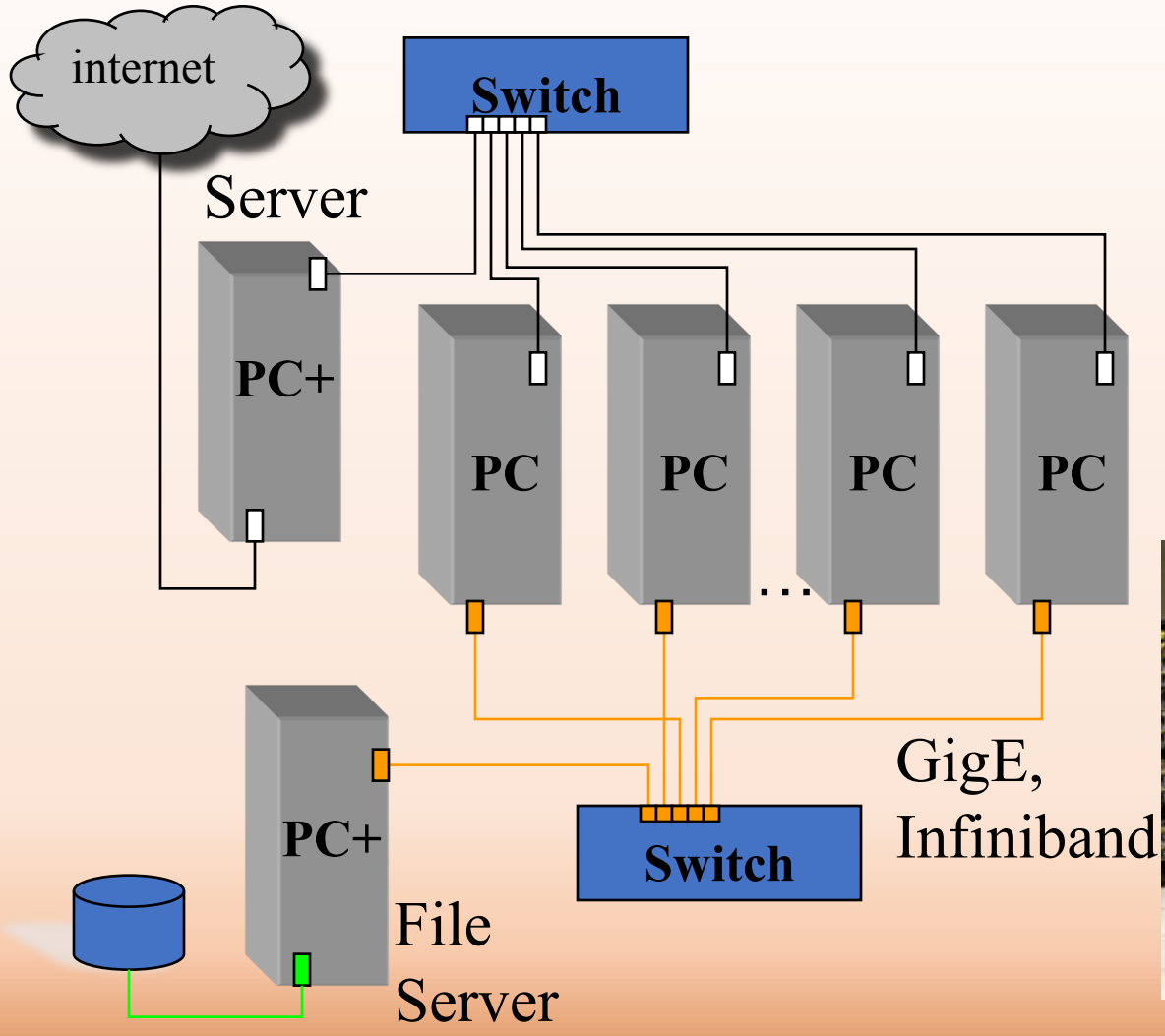
Office hours: GWC546 1-4pm Tues, Wed (during academic year)

Get an account

- researchcomputing.asu.edu - "Create an account"
- Faculty and students get 25K hours every month. Students link to faculty (PI) account
- For assistance installing/using software on the cluster: support@hpchelp.asu.edu
- Login via ssh, putty (Windows).

```
ssh userid@agave.asu.edu
```

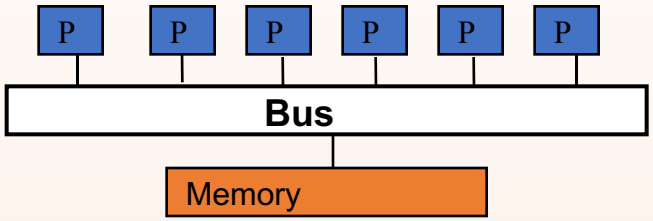
Generic Cluster Architecture

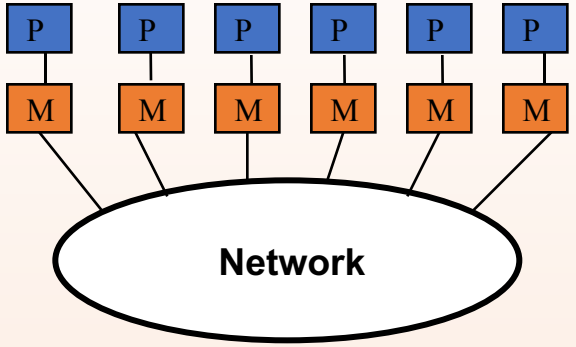


(Adv. HPC System)

lonestar.tacc.utexas.edu

Shared and Distributed Memory





Shared memory (SMP): single address space. All processors have access to a pool of shared memory. Agave normal and phi nodes have 28 and 64 cores per node, respectively.

Methods of memory access :
- Bus
- Crossbar

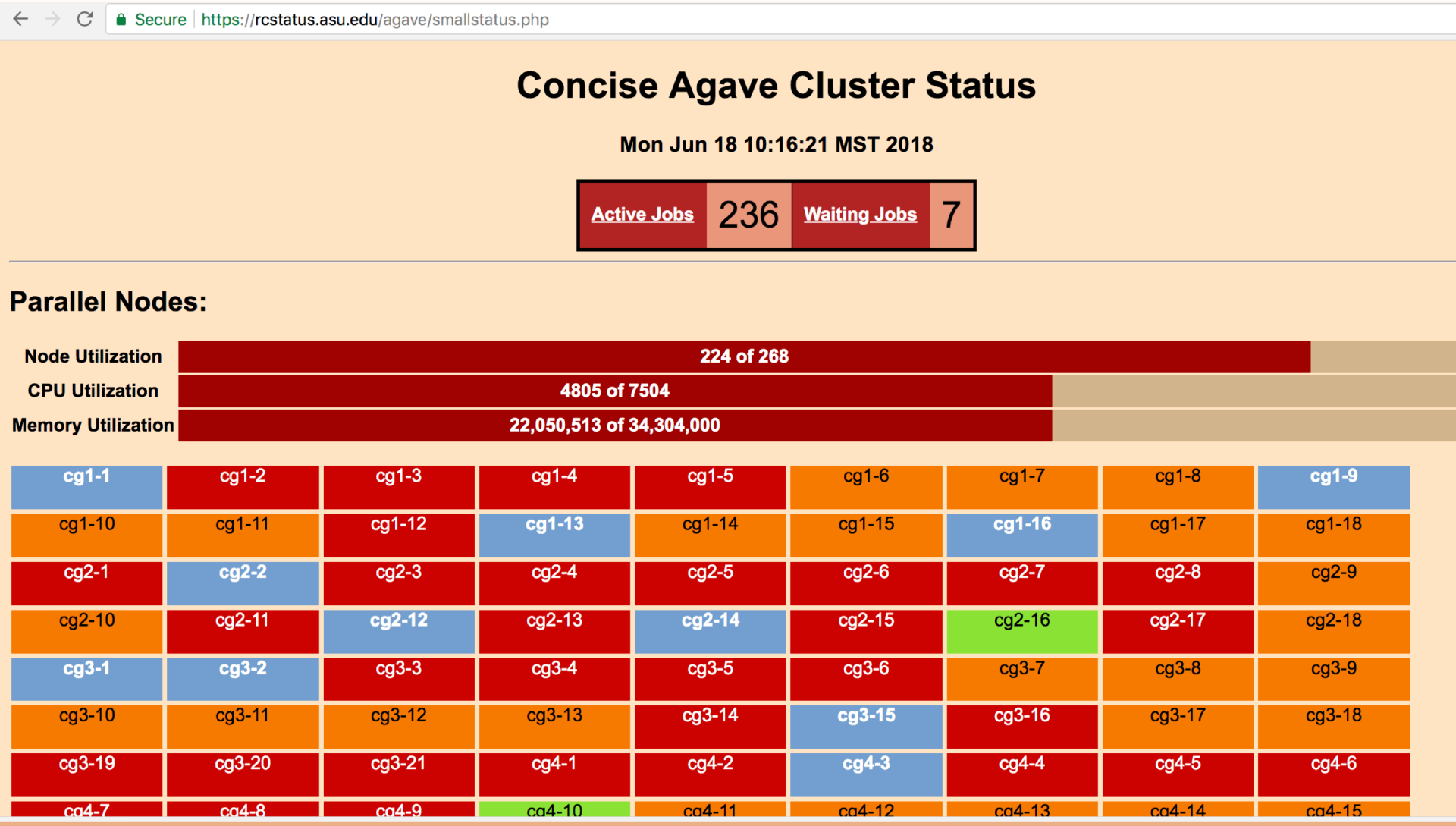
API: OpenMP

Distributed memory (MPP): each Processor has its own local memory. Must do message passing to exchange data between processors. (examples: Clusters)

Methods of memory access :
- various topological interconnects

API: Message Passing Interface (MPI)

Graphical View of Agave



Agave system Information

- **Agave ~7.5K Broadwell cores**
  - **>300 trillion floating point operations per second (TFLOPs)**
  - **>34TB aggregate RAM**
  - **>6PB aggregate disk**
  - **Omnipath Interconnect – 100Gb/s**
  - **Located in ISTB1**
- **~5K Xeon Phi cores**
- **GPU partition(s)**
- **Agave is a true *cluster* architecture**
  - **Each Broadwell node has 28 processors and 128GB of RAM (~4.5GB/CPU).**
  - **Programs needing more resources *must* use parallel programming**
  - **Normal, single processor applications *do not go faster* on Agave**

Initial Login

- Login with SSH
  `ssh agave.asu.edu`
- Connects you to a login node
- Don't overwrite ~/.ssh/authorized_keys
  - Feel free to add to it if you know how to use it:
    - `ssh-keygen`
    - `ssh-copy-id -i ~/.ssh/id_rsa.pub userid@agave.asu.edu`
  - SSH is used for job start up on the compute nodes. Mistakes can prevent your jobs from running
- For X forwarding, use `ssh -X`
- Nomachine: rcstatus.asu.edu/agave/howto
  - Click on "Logging in to Agave cluster with NoMachine Remote Desktop"
  - Get nomachine client and nomachine profile
- Xming
- **From outside ASU campus, use VPN**: download from sslvpn.asu.edu

Transferring files

- Secure copy

```
scp projectfile
 user1@agave.asu.edu:/home/user1/projectdir
scp –r projectdir
 user1@agave.asu.edu:/home/user1/projectdir
```

- **`rsync -e ssh`** for large file transfers
- **`rsync –avtr bigfiledir user1@agave.asu.edu:/home/user1/projectdir`**
- Winscp
- ftp: Filezilla

Packages

- Modules are used to setup your PATH and other environment variables
- They are used to setup environments for packages & compilers

```
[user1@agave1 ~]$ module          {lists options}
[user1@agave1 ~]$ module avail    {lists available packages}
[user1@agave1 ~]$ module load <package> <…> {add one or more packages}
[user1@agave1 ~]$ module unload <package> {unload a package}

[user1@agave1 ~]$ module list     {lists loaded packages}

[user1@agave1 ~]$ module purge    {unloads all packages}
```

- Multiple compiler families available, so make sure you are consistent between libraries, source and run script!

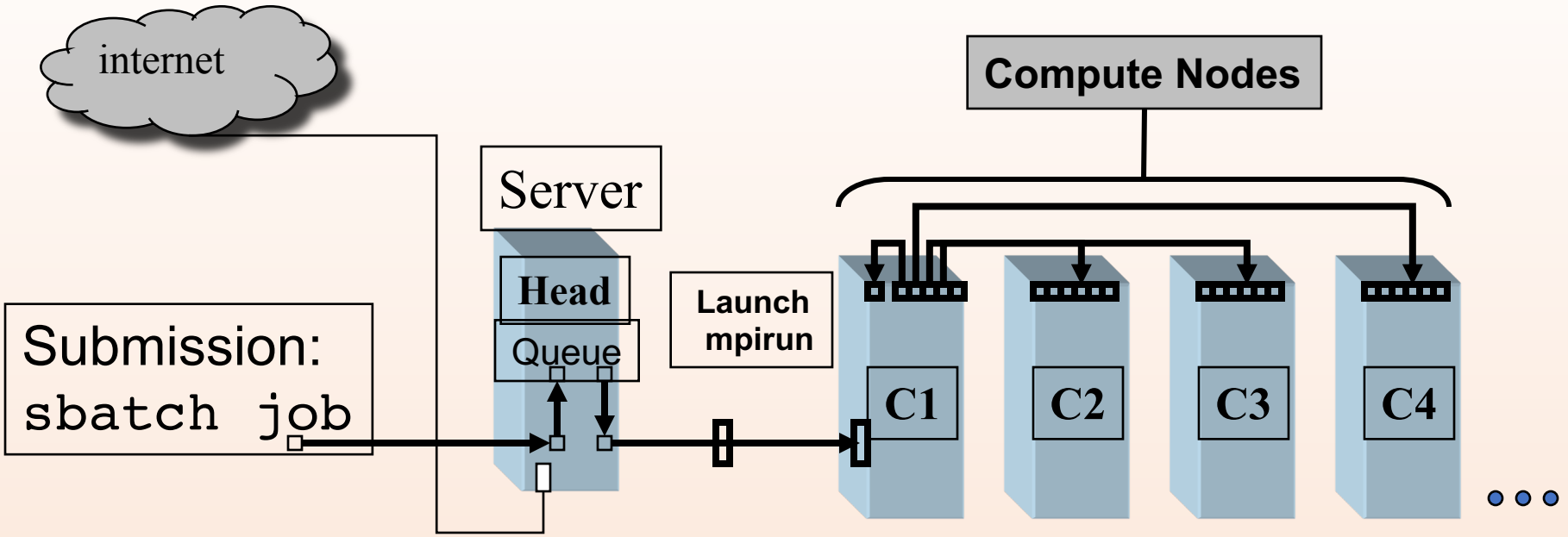
Compile

- After linking (via `module` command)
- Different compilers unfortunately means different flags

```
[user1@agave1 ~]$ module load intel/2018x
[user1@agave1 ~]$ icc -qopenmp -o program program.c

[user1@agave1 ~]$ module purge
[user1@agave1 ~]$ module load gcc/8x
[user1@agave1 ~]$ gcc -fopenmp -o program program.c
[user1@agave1 ~]$ gfortran -fopenmp -o program
program.f
[user1@agave1 ~]$ export OMP_NUM_THREADS=28
```

- Multiple compiler families available, so make sure you are consistent between libraries, source and run script!

Batch Submission Process



Queue: Job script waits for resources on Server
Compute nodes execute the job script, launching MPI processes

Launch: contact each compute node to start executable (e.g. a.out)

Batch Systems

- *Agave* systems use slurm for batch queuing and scheduling
- Batch jobs are submitted on the front end and are subsequently executed on compute nodes as resources become available
- Order of job execution depends on a variety of parameters:
  - Submission Time
  - Resources requested
  - Backfill Opportunities: small jobs may be back-filled while waiting for bigger jobs to complete
  - Advanced Reservations: jobs my be blocked in order to accommodate advanced reservations (for example, during maintenance windows)
  - Number of Actively Scheduled Jobs: there are limits on the maximum number of concurrent processors used by each user

SLURM Commands

sbatch, srun, salloc	Submit a job
squeue	Check on the status of jobs
sinfo	Get info on nodes/partitions
scancel	Delete running and queued jobs
scontrol	Alter/hold/release jobs

man pages for all of these commands

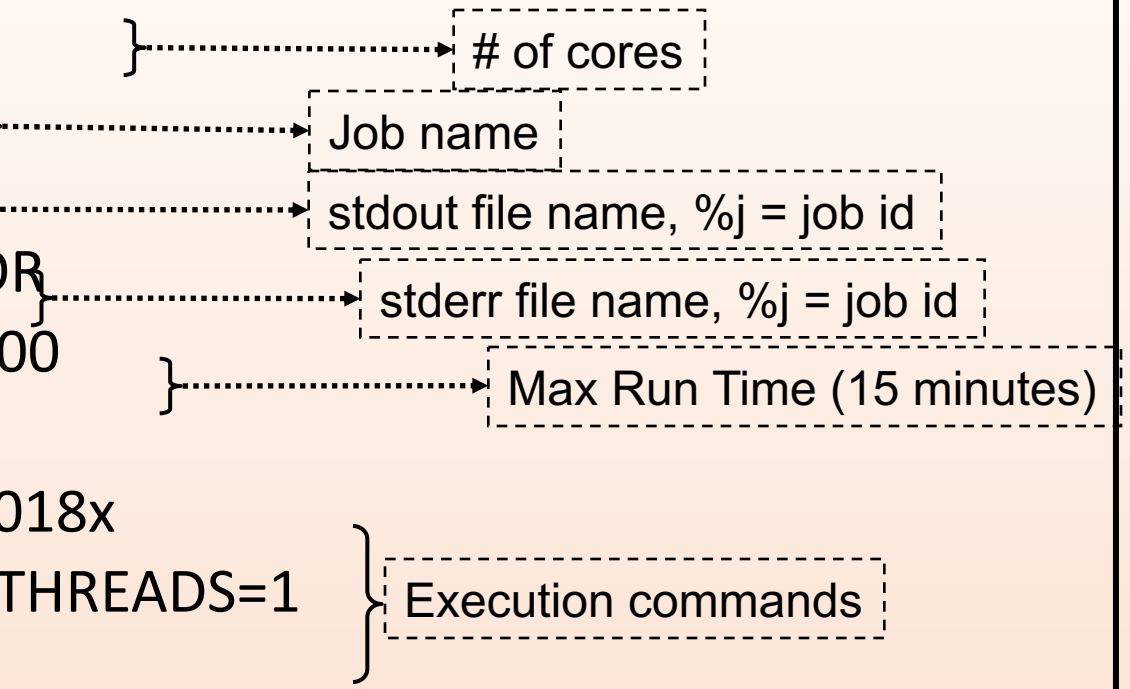
showjobs	Running jobs with queue info
longjob <jobid>	Details on job

Batch System Concerns

- Submission (* need to know)
  - Required Resources*
  - Estimated runtime*
  - Run-time Environment
  - Directory of Submission/Execution
  - Files for stdout/stderr Return
  - Email Notification
- Job Monitoring
- Job Deletion
  - Queued Jobs
  - Running Jobs

SLURM: OpenMP “job” Script

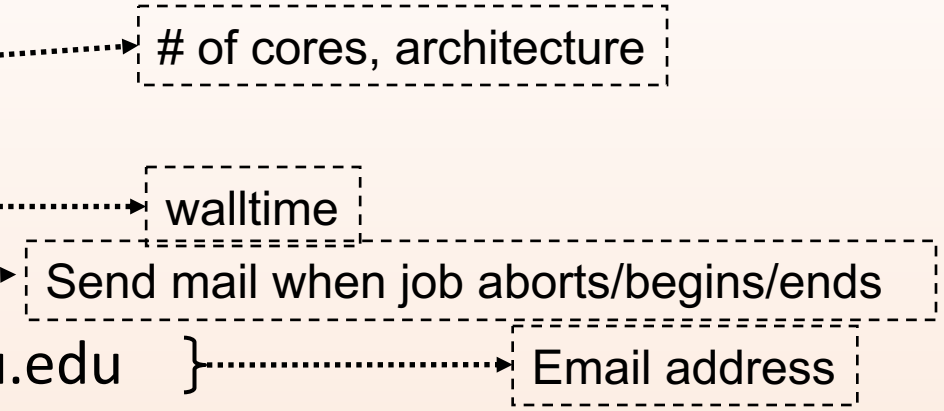
```
#!/bin/bash
#SBATCH -n 1
#SBATCH -J hello
#SBATCH -o %j.OUT
#SBATCH -e %j.ERROR
#SBATCH -t 0-00:15:00

module load intel/2018x
export OMP_NUM_THREADS=1
./hello
```




```
[user1@agave1 ~]$ sbatch job
```

SLURM: OpenMP Job Script II

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=28
#SBATCH --time=96:00:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=user1@asu.edu

module load openmpi/3.0.0-gnu-7x
export OMP_NUM_THREADS=28
export OMP_DYNAMIC=TRUE
time srun –mpi=pmi2 lmp_g++_openmpi –in in.eam –v x 2
```



Batch Script Debugging Suggestions

- Echo issuing commands
  - (“set -x” or “set echo” for bash or csh).
- Abort job when a critical command fails.
- Print environment
  - Include the "env" command if your batch job doesn't execute the same as in an interactive execution.
- Use “./” prefix for executing commands in the current directory
  - The dot means to look for commands in the present working directory. Not all systems include "." in your $PATH variable. (usage: ./a.out).
- Track your CPU time

Normal and wildfire queues

- The mybalance command:

```
[user1@agave1 ~]$ mybalance

User                    : user1
CPU Hours Allocated     : 25000
CPU Hours Used          : 526.36
CPU Hours Available     : 24473.63
```

- If the resource request for your submitted job (#CPUs X walltime requested) fits within your available CPU hours, the job is **non-preemptable**, i.e. will run uninterrupted in the **normal** queue.
- If your resource request exceeds the available CPU hours, the job is **preemptable**, and will run in the **wildfire** queue.
- Limit of 50 running jobs per user

SLURM partitions and environment

- Serial and parallel partitions
  - No need to specify – automatically determined based on requested resources
- `#SBATCH -p phi`
- `#SBATCH -p physicsgpu1 # Use physicsgpu1 partition`
- `#SBATCH -p cidsegpu1 # Use cidsegpu1 partition`

```
#SBATCH -q wildfire # Run job in wildfire QOS queue
#SBATCH --gres=gpu:2
```

Variable	Purpose
SLURM_JOB_ID	Batch job id
SLURM_SUBMIT_DIR	Directory where job was submitted
SLURM_JOB_NODELIST	Filename containing list of nodes
SLURM_ARRAY_TASK_ID	Slurm array index (next slide)

SLURM arrays and small jobs

- Arrays can loop to submit many jobs: `--array`
- `sbatch --array=0-20 job`

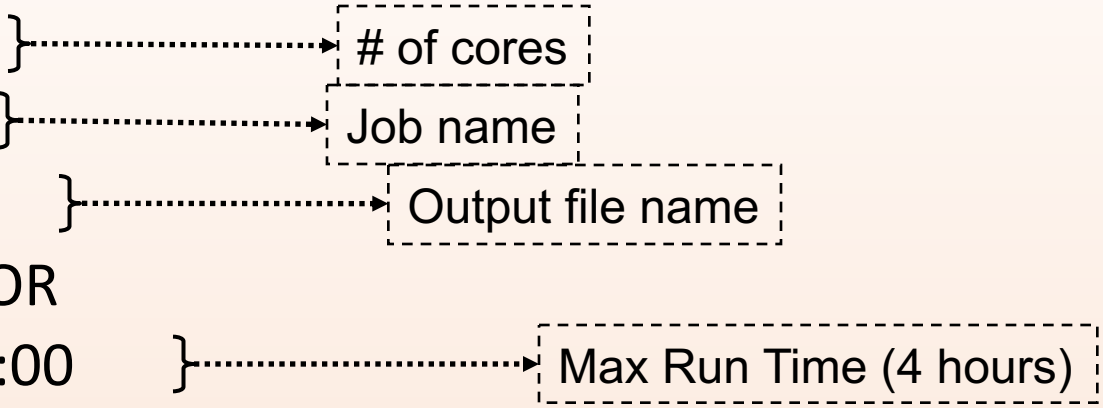
```
(in job) ./executable.x < $SLURM_ARRAY_TASK_ID.inp
 or ./executable2.x $SLURM_ARRAY_TASK_ID
```

- For single line short jobs: `--wrap`

```
sbatch -n 2 --wrap="module load gcc/7x;gcc -fopenmp
hello_world.c;export OMP_NUM_THREADS=2;./a.out"
```

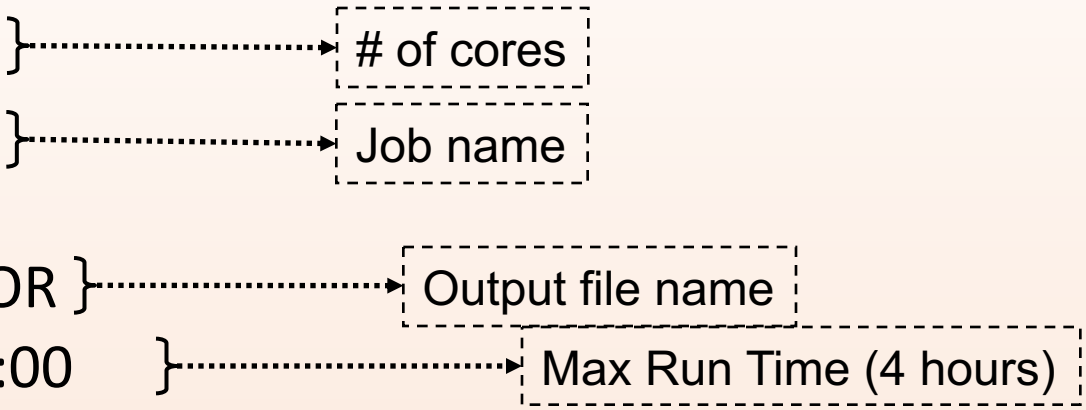
Matlab Job Script

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -J hello
#SBATCH -o %j.OUT
#SBATCH -e %j.ERROR
#SBATCH -t 0-04:00:00

module load matlab/R2018a
matlab –nodisplay –nodesktop –nosplash < hello.m
matlab –nodisplay –nodesktop –nosplash –r “hello, quit”
```



R Job Script

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -J hello
#SBATCH -o %j.OUT
#SBATCH -e %j.ERROR
#SBATCH -t 0-04:00:00

module load r/3.5.1
R --no-save --quiet --slave < regression.r
Rscript regression.r 20000
```



Job Monitoring (*squeue* utility)

```
[user1@agave1 ~]$ squeue
      JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
   208952_[0-199]    serial   COLD_3   mrline PD       0:00      1 (BeginTime)
          207709    serial R-px-OSa epopplet PD       0:00      1 (AssocMaxJobsLimit)
          207710    serial R-px-OSa epopplet PD       0:00      1 (AssocMaxJobsLimit)
          207711    serial R-px-OSa epopplet PD       0:00      1 (AssocMaxJobsLimit)
          207712    serial R-px-OSa epopplet PD       0:00      1 (AssocMaxJobsLimit)
          207713    serial R-px-OSa epopplet PD       0:00      1 (AssocMaxJobsLimit)
        .
        .
        .
```

Basic *squeue* options:

-u username Display jobs belonging to specified user

-l, --long Display extended job information

SLURM Job Manipulation/Monitoring

- To kill a running or queued job (takes ~30 seconds to complete):

```
scancel <jobID>
```

- To suspend a queued job:

```
scontrol hold job <jobId>
```

- To resume a suspended job:

```
scontrol release job <jobID>
```

- To alter job information in queue:

```
scontrol update job <jobID> NumNodes=6-6
NumCPUs=6 NumTasks=6
```

- To see more information on the partitions and their status:

```
sinfo
```

- To see a detailed info on job:

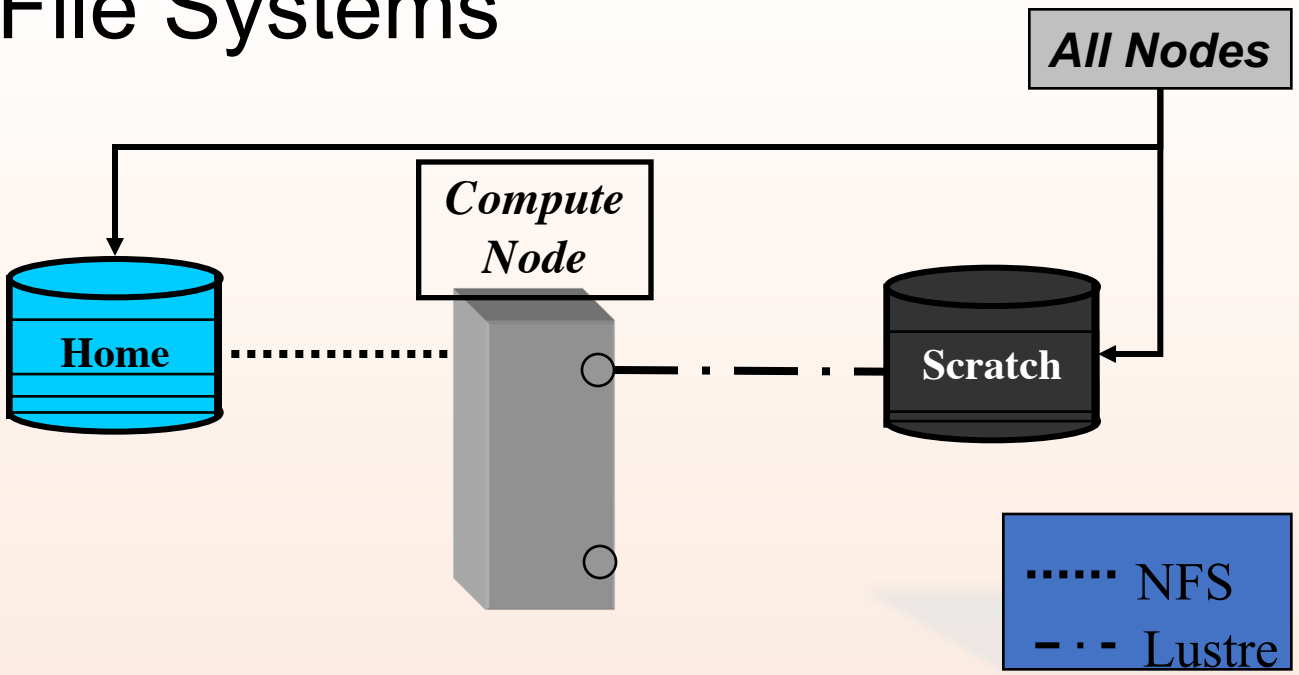
```
scontrol show job <jobID>
Scontrol show partition
```

Interactive Mode

- **`interactive`** Interactive mode
- **`interactive -p phi`** Interactive on Xeon phi
- **`screen`** Detach "`^ad`" interactive jobs.
  Reattach with **`screen -r`**. (Also **`tmux`**)
  1. **`screen`** (on login node)
  2. **`interactive`**

- **`interactive -N 1 -n 28`** Entire Broadwell node

Available File Systems



Mount point	User Access Limit	Lifetime
/home	500GB quota	Project
/scratch	no quota	30 days

Good citizenship

- Shared login node: Do not compute on the login nodes
- Shared filesystem: Run I/O intensive jobs on scratch
- Shared network: Do not start 20 scps
- Shared compute resource: Give good estimate of runtime. Test submission scripts before submitting them at large scale.
- Shared help desk: Do some homework before submitting ticket. Do not submit multiple tickets on same topic. Describe issue in detail (e.g. job ID, full path to failing sbatch script, etc.). Be patient.

Conclusion

For any assistance please contact us:

support@hpchelp.asu.edu

Office hours: GWC546 1-4pm Tues (and 1-4pm Wed during academic year)

Info at: researchcomputing.asu.edu and rcstatus.asu.edu/agave