

SuperMike-III User Environment

Feng Chen
HPC User Services
LSU HPC & LSU
sys-help@LSU.org

Louisiana State University
Baton Rouge
July 28, 2022

Outline

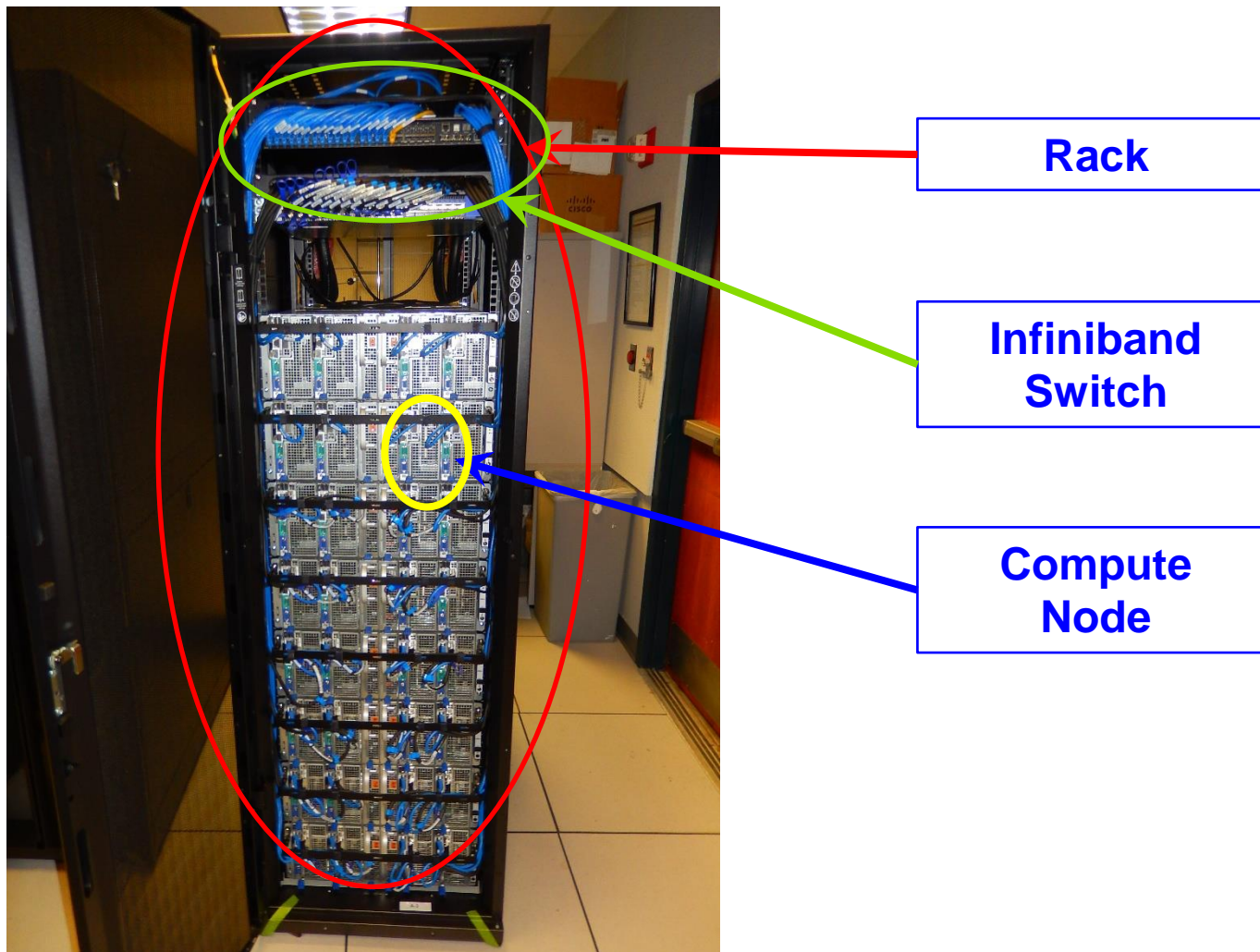
- **Things to be covered in this section**
 - SuperMike-III job queues
 - Job submission on SuperMike-III
 - Interactive and batch jobs using SLURM
 - Serial and parallel jobs
 - Monitor your jobs



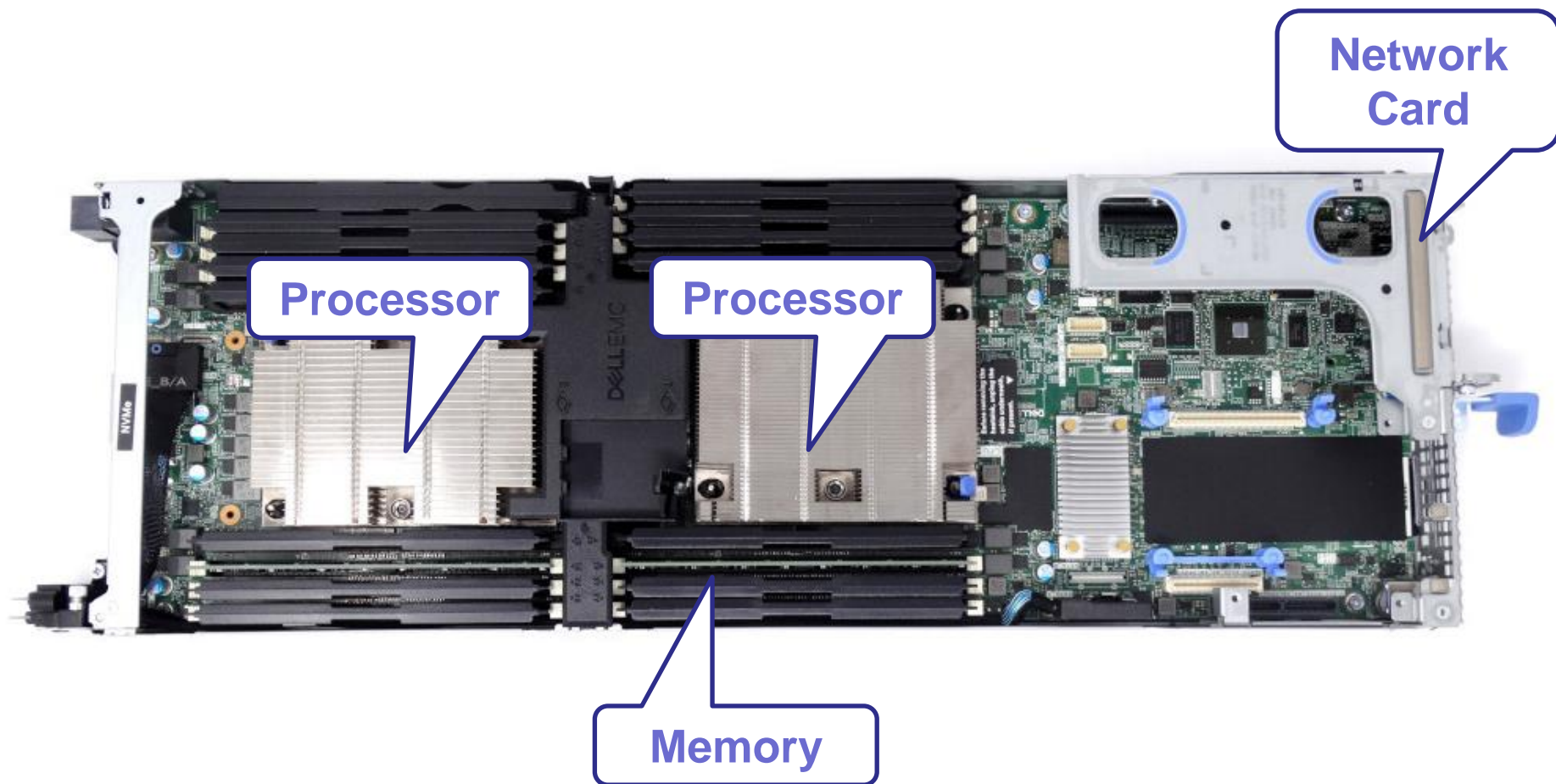
SuperMike-III HPC User Environment

Overview of SuperMike-III

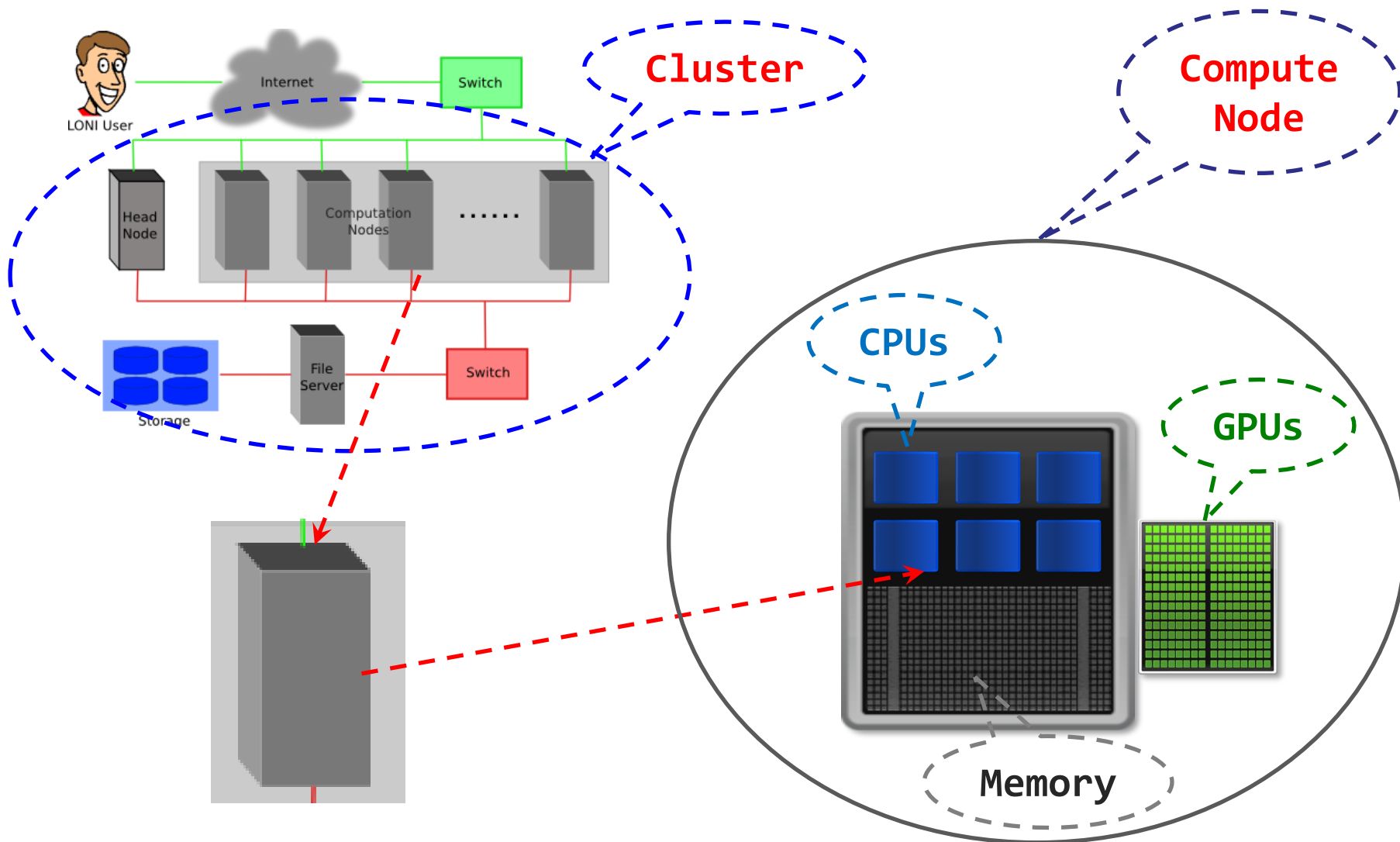
Inside A Cluster Rack



Inside A SuperMike-III Compute Node



Conceptual Relationship



Summary of SuperMike-III

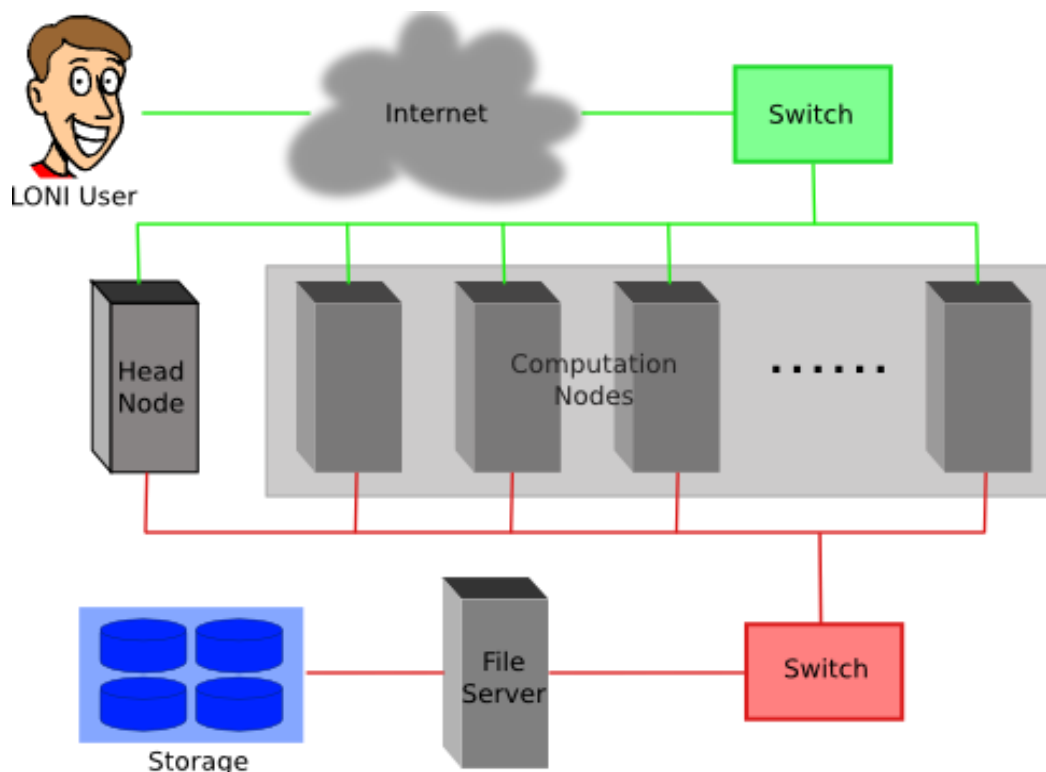
- **SuperMike-III is a 1.3 PetaFlop peak performance cluster with the latest CPUs from Intel and GPUs from NVIDIA, comprised of 183 compute nodes connected by 200 Gbps Infiniband fabric:**
 - 171 regular nodes: two 32-core Intel Ice Lake CPUs, 256 GB RAM
 - 8 GPU nodes: two 32-core Intel Ice Lake CPUs, 256 GB RAM, four NVIDIA Tesla A100 GPUs
 - 4 bigmem nodes: two 32-core Intel Ice Lake CPUs, 2 TB RAM
- **log in SuperMike-III with your current LSU HPC credentials using**
 - `ssh <username>@mike.hpc.lsu.edu`
- **Before you submit jobs on SuperMike-III, please make sure that you review the user guide here:**
 - <http://www.hpc.lsu.edu/docs/guides.php?system=SuperMike3>
- **The biggest difference SuperMIC users would notice on SuperMike-III is that, instead of Torque/Moab (PBS), *SLURM* is employed as the workload and resource manager.**

SuperMike-III HPC User Environment

SuperMike-III Job Queues

Cluster Environment

- Multiple compute nodes
- Multiple users
- Each user may have multiple jobs running simultaneously
- Multiple users may share the same node



Job submission basics

- 1. Find appropriate queue**
- 2. Understand the queuing system and your requirements and proceed to submit jobs**
- 3. Monitor jobs during execution**

Job Queues (Partitions)

- **Nodes are organized into queues. Nodes can be shared.**
- **Each job queue (partition) differs in**
 - Number of available nodes
 - Max run time
 - Max running jobs per user
 - Nodes may have special characteristics: GPU, Large memory, etc.
- **Jobs need to specify resource requirements**
 - Nodes, time, queue
- **It's called a queue for a reason, but jobs don't run on a "First Come First Served" policy,**
 - This will be detailed in later slides

Queue Characteristics – LSU clusters

Machine	Queue	Max Runtime (h)	ppn	Max nodes per job
SuperMIC	workq	72	20	128
	checkpt		20	256
	single	168	1,2,4,6,8	1
	bigmem	72	48	1
Machine	Partition	Max Runtime (h)	cores per node	Max nodes per job
SuperMike-III	workq	72	64	84
	checkpt	72	64	84
	single	168	64	1
	gpu	72	64	4
	bigmem	72	64	1

❖ By default, you job will be submitted to “single” queue (partition).

Queue Characteristics

- “sinfo” will give you more info on the queues

```
[fchen14@mike2 slurmdoc]$ sinfo
```

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	NODELIST
admin	up	infinite	4	mix	mike[001-003,192]
admin	up	infinite	197	idle	mike[004-190,193-202]
admin	up	infinite	1	down	mike191
single*	up	3-00:00:00	4	mix	mike[001-003,192]
single*	up	3-00:00:00	187	idle	mike[004-190]
single*	up	3-00:00:00	1	down	mike191
checkpt	up	3-00:00:00	4	mix	mike[001-003,192]
checkpt	up	3-00:00:00	187	idle	mike[004-190]
checkpt	up	3-00:00:00	1	down	mike191
workq	up	3-00:00:00	4	mix	mike[001-003,192]
workq	up	3-00:00:00	187	idle	mike[004-190]
workq	up	3-00:00:00	1	down	mike191
gpu	up	3-00:00:00	8	idle	mike[193-200]
bigmem	up	3-00:00:00	2	idle	mike[201-202]

Queue Querying – Linux Clusters

- Displays information about active, eligible, blocked, and/or recently completed jobs: `showq` command

```
[fchen14@mike1 ~]$ showq
```

ACTIVE JOBS-----

JOBID	JOBNAME	USERNAME	STATE	CORE	REMAINING	STARTTIME
4896	cellranger	lyan1	Running	64	21:42:43	Tue Jul 26 14:55:23
4924	bash	fchen14	Running	64	11:05:50	Tue Jul 26 16:18:30
4946	xhpcg_skx	lyan1	Running	2048	11:33:16	Tue Jul 26 16:45:56
4956	supernova.	lyan1	Running	64	23:42:50	Tue Jul 26 16:55:30
4964	test	yexu	Running	64	3:52:34	Tue Jul 26 17:05:14
4968	gmx_mpi	lyan1	Running	128	11:59:19	Tue Jul 26 17:11:59
4969	bash	lyan1	Running	64	11:59:41	Tue Jul 26 17:12:21

7 active jobs

Total Jobs: 7 Active Jobs: 7 Idle Jobs: 0 Blocked Jobs: 0

SuperMike-III HPC User Environment

Job Submission Through SLURM

PBS to SLURM

➤ **Why SLURM?**

- SLURM has a more open model
- SLURM also feels more modern in its design and implementation
- SLURM scales well, job starts faster, etc.

➤ **What will we cover in this session?**

- Basic SLURM job submission
 - Interactive job
 - Batch job

Two Job Types

➤ Interactive job

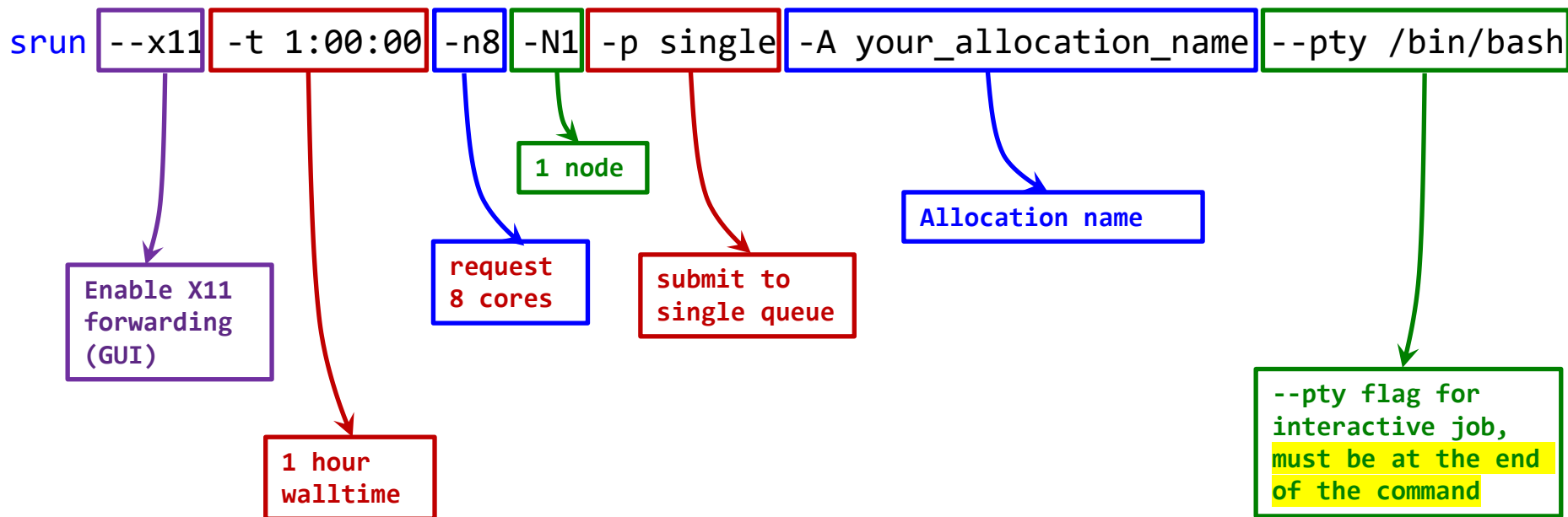
- User can interact with the terminal on the compute node.
- Interactive jobs can be used for **testing and troubleshooting** code.
- Requesting an interactive job will allocate resources and log you into a shell on a compute node.

➤ Batch job

- Executed without user intervention using a job script
- Batch jobs are used for production runs.
- Requesting an interactive job will allocate resources and execute the job commands in a non-interactive shell

SLURM Interactive Job Command

- To start an interactive job, use the **srun** command like the example below:



Check Available Allocations

```
[fchen14.fchen14-t460] ➤ ssh fchen14@mike.hpc.lsu.edu
```

```
#####
```

Send questions and comments to the email ticket system at sys-help@loni.org.

```
#####
```

SuperMike-3 at LSU (Open in friendly user mode)

17-May-2022

SuperMike-3 is a 1.3 PetaFlop peak performance cluster with 11,712 CPU cores,
...Message Of The Day...

```
[fchen14@mike1 ~]$ showquota
```

User filesystem quotas for fchen14 (uid 32584):

Filesystem	MB used	quota	files	fquota
/homem	433	10000	21532	0
/work /project	302299	0	2325239	4000000
Storage allocation	MB used	quota	files	expiration
sa_fchen14	-	-	-	2000-01-01
CPU Allocation SUs:	remaining	allocated	expiration	
hpc_hpcadmin8:	393715.16	2000000.00	2022-07-01	
hpc_train_2022:	49301.54			

Start an SLURM Interactive Job

```
[fchen14@mike1 ~]$ srun --x11 -t 1:00:00 -n8 -N1 -p single -A hpc_train_2022 --pty /bin/bash
srun: Job is in held state, pending scheduler release
srun: job 4799 queued and waiting for resources
Interactive job 4799 running:
srun: job 4799 has been allocated resources
[fchen14@mike156 ~]$ hostname
mike156
[fchen14@mike156 ~]$ some_job_commands # your own job commands
```

SLURM Environmental Variables

```
[fchen14@mike2 slurmdoc]$ srun --x11 -t 1:00:00 -n8 -N1 -p single -A
LSU_LSUadmin1 --pty /bin/bash
```

```
[fchen14@mike156 slurmdoc]$ echo $SLURM_
```

\$SLURM_CLUSTER_NAME	\$SLURM_JOB_NAME
\$SLURM_NPROCS	\$SLURM_STEP_NODELIST
\$SLURM_CPU_BIND	\$SLURM_JOB_NODELIST
\$SLURM_NTASKS	\$SLURM_STEP_NUM_NODES
\$SLURM_CPU_BIND_LIST	\$SLURM_JOB_NUM_NODES
\$SLURM_PRIO_PROCESS	\$SLURM_STEP_NUM_TASKS
\$SLURM_CPU_BIND_TYPE	\$SLURM_JOB_PARTITION
\$SLURM_PROCID	\$SLURM_STEP_TASKS_PER_NODE
...	
\$SLURM_JOB_GID	\$SLURM_NNODES
\$SLURM_STEPID	\$SLURM_TOPOLOGY_ADDR_PATTERN
\$SLURM_JOBID	\$SLURM_NODEID
\$SLURM_STEP_ID	\$SLURM_UMASK
\$SLURM_JOB_ID	\$SLURM_NODELIST
\$SLURM_STEP_LAUNCHER_PORT	\$SLURM_WORKING_CLUSTER

Notes about srun inside interactive job

- If you need to start another MPI job using **srun** inside an interactive job, you need to add "**--overlap**" after the srun command to avoid command hang:

```
[fchen14@mike1 ~]$ srun -N1 -n8 -p single --pty bash # interactive job
```

```
srun: Job is in held state, pending scheduler release
```

```
srun: job 5224 queued and waiting for resources
```

```
Interactive job 5224 waiting:
```

```
srun: job 5224 has been allocated resources
```

```
[fchen14@mike001 ~]$ srun -n 4 hostname # this command will hang
```

```
[fchen14@mike001 ~]$ srun --overlap -n 4 hostname # this command will work
```

```
mike001
```

```
mike001
```

```
mike001
```

```
mike001
```

- For batch job, you don't need to add --overlap

SLURM Batch Job Script

- To create a batch SLURM script, use your favorite editor (e.g. vi or emacs, nano) to create a text file with both SLURM instructions and commands how to run your job.
- All SLURM directives (special instructions) are prefaced by the **#SBATCH**.

```
#!/bin/bash
#SBATCH -N 1          # request one node
#SBATCH -t 2:00:00    # request two hours
#SBATCH -p single     # in single partition (queue)
#SBATCH -A your_allocation_name
#SBATCH -o %x-%j.out-%N # optional, name of the stdout, using the job number (%j) and the
                        # hostname of the node (%N)
#SBATCH -e %x-%j.err-%N # optional, name of the stderr, using job and hostname values
# below are job commands
```

Tells the job scheduler how much resource you need.

```
date
cd /work/$USER/myjob
./mydemo
# Mark the time it finishes.
date
# exit the job
exit 0
```

How will you use the resources?

Common SLURM Switches

- **#SBATCH -A allocation_name:**
 - short for --account, charge jobs to your allocation named allocation_name.
- **#SBATCH -N <number_of_nodes>:**
 - short for --nodes, number of nodes on which to run.
- **#SBATCH -n <number_of_cores/processes>:**
 - short for --ntasks, number of tasks (CPU cores) to run job on. The memory limit for jobs is 4 GB of MEM per CPU core requested.
- **#SBATCH -c <cores_per_process>:**
 - short for --ncpus-per-task, number of threads per process.
- **#SBATCH -p partition:**
 - short for --partition, submit job to the partition queue. Allowed values for partition: single, checkpt, workq, gpu, bigmem. Depending on cluster (use [sinfo](#) command)
- **#SBATCH -t hh:mm:ss:**
 - short for --time, request walltime.
- **#SBATCH -o filename.out:**
 - short for --output, write standard output.
- **#SBATCH -e filename.err:**
 - short for --error, write standard error.
 - Note that by default, SLURM will merge standard error and standard output.

Submit SLURM Batch Job

- To submit the above job to the scheduler, save the above script as a text file, e.g., `singlenode.sh`, then use the `sbatch` command to submit, the output will be something like the below:

```
[fchen14@mike2 slurmdoc]$ sbatch singlenode.slm
```

```
Submitted batch job 37355 estimates 9 SUs from allocation LSU_train_2020.
```

```
Estimated remaining SUs: 37352
```

```
See running job information with: scontrol show job 37355
```

- To check the status of your job, use the `squeue` command:

```
[fchen14@mike2 slurmdoc]$ squeue -u $USER
```

JOBID	PARTITION	QOS	NAME	USER	ACCOUNT	STATE	PRIORITY	TIME
SUBMIT_TIM TIME_LIMI NODES CPUS MIN_MEMORY NODELIST(REASON)								
1	0:06	2020-09-18	1:00:00	2	96 3958M	LSU_tra	RUNNING	

Common SLURM Commands (1)

- **queue** is used to show the partition (queue) status. Useful options:
 - **-u <username>**: limit output to jobs by username **--state=pending**: limit output to pending (i.e. queued) jobs **--state=running**: limit output to running jobs
 - Below is an example to query all jobs submitted by current user (fchen14)

```
[fchen14@mike2 slurmdoc]$ squeue -u fchen14
```

JOBID	PARTITION	NAME	USER	ST	TIME_LIMIT	TIME	CPUS	NODES	NODELIST(REASON)
37876	workq	hybrid_job	fchen14	CF	5:00	0:04	96	2	mike[005-006]

Common SLURM Commands (2)

➤ **sinfo** is used to view information about SLURM nodes and partitions.

```
[fchen14@mike156 ~]$ sinfo
```

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	NODELIST
admin	up	infinite	2	inval	mike[147,175]
admin	up	infinite	1	comp	mike183
admin	up	infinite	1	mix	mike156
admin	up	infinite	178	idle	mike[001-146,148-155,157-174,176-181]
admin	up	infinite	1	down	mike182
single*	up	7-00:00:00	1	inval	mike147
single*	up	7-00:00:00	1	mix	mike156
single*	up	7-00:00:00	169	idle	mike[001-146,148-155,157-171]
checkpt	up	3-00:00:00	1	inval	mike147
checkpt	up	3-00:00:00	1	mix	mike156
checkpt	up	3-00:00:00	169	idle	mike[001-146,148-155,157-171]
workq	up	3-00:00:00	1	inval	mike147
workq	up	3-00:00:00	1	mix	mike156
workq	up	3-00:00:00	169	idle	mike[001-146,148-155,157-171]
bigmem	up	3-00:00:00	1	inval	mike175
bigmem	up	3-00:00:00	3	idle	mike[172-174]
gpu	up	3-00:00:00	1	comp	mike183
gpu	up	3-00:00:00	6	idle	mike[176-181]
gpu	up	3-00:00:00	1	down	mike182

Common SLURM Commands (3)

➤ **scancel is used to signal or cancel jobs. Typical usage with squeue.**

```
[fchen14@mike1 ~]$ squeue -u fchen14
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
341	checkpt	bash	fchen14	R	0:13	1	mike001
340	checkpt	bash	fchen14	R	1:50:57	1	mike002

```
# cancel (delete) job with JOBID 340
```

```
[fchen14@mike1 ~]$ scancel 340
```

```
# job status might display a temporary "CG" ("Completing") status immediately after scancel
```

```
[fchen14@mike1 ~]$ squeue -u fchen14
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
340	checkpt	bash	fchen14	CG	1:51:08	1	mike002
341	checkpt	bash	fchen14	R	0:41	1	mike001

```
[fchen14@mike1 ~]$ squeue -u fchen14
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
341	checkpt	bash	fchen14	R	1:08	1	mike001

Common SLURM Commands (4)

- **scontrol** is used to view or modify SLURM configuration and state.

Typical usage for the user is to check job status:

```
[fchen14@mike1 ~]$ squeue -u fchen14 # show all jobs
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(Reason)
341	checkpt	bash	fchen14	R	1:29:20	1	mike001

```
[fchen14@mike1 ~]$ scontrol show job 341
```

```
JobId=341 JobName=bash
```

```
UserId=fchen14(32584) GroupId=Admins(10000) MCS_label=N/A
```

```
Priority=1 Nice=0 Account=hpc_hpcadmin6 QOS=normal
```

```
JobState=RUNNING Reason=None Dependency=(null)
```

```
... some details omitted...
```

```
MinCPUsNode=1 MinMemoryNode=22332M MinTmpDiskNode=0
```

```
Features=(null) DelayBoot=00:00:00
```

```
OverSubscribe=NO Contiguous=0 Licenses=(null) Network=(null)
```

```
Command=/bin/bash
```

```
WorkDir=/home/fchen14/test
```

```
Power=
```


Serial and Parallel (Multi-Threaded and MPI) Job Templates

- **Serial Jobs**

- **Parallel Jobs**
 - SMP (Shared Memory Parallelism)
 - OpenMP
 - Python's Multiprocessing
 - Pthread
 - R's mapply
 - MPI
 - Hybrid

Serial Job Script Template

```
#!/bin/bash
#SBATCH --job-name=serial_job_test    # Job name
#SBATCH --ntasks=1                    # Using a single core
#SBATCH --time=00:10:00               # Time limit hh:mm:ss
#SBATCH --output=%x_%j.log            # Standard output and error log,
#                                     # %x: job name
#                                     # %j: job-id
```

```
module load python
```

```
echo "Running job on a single CPU core"
```

```
date
```

```
/home/user/single_core_job.py
```

```
date
```

Shared Memory Parallelism (SMP) Jobs

- **Shared-Memory Parallelism (SMP) is when workload is shared among different CPU cores using multiple threads or processes running within a single compute node and these cores have access to common (shared) memory.**
 - SMP jobs cannot make use of multiple nodes and all the cores must be physically located the same node.
 - When running SMP jobs, you must make the SMP application aware of how many cores to use.
 - How that is done depends on the specific type of application. Typical examples:
 - OpenMP (Open Multi-Processing)
 - First set `--ntasks=1`, and then set `OMP_NUM_THREADS` to a value less than or equal to the number of cpus-per-task
 - Pthreads
 - Python's multiprocessing module
 - R's mapply

SMP Job Script (OpenMP Based)

```
#!/bin/bash
#SBATCH --job-name=smp_job           # Job name
#SBATCH --nodes=1                     # Run all processes on a single node
#SBATCH --ntasks=1                   # Run a single task
#SBATCH --cpus-per-task=8             # Number of CPU cores per task
#SBATCH --time=00:10:00               # Time limit hh:mm:ss
#SBATCH --output=%x_%j.log           # Standard output and error log
```

date

use this line if your job uses OpenMP

export OMP_NUM_THREADS=\$SLURM_CPUS_PER_TASK

/home/user/smp_job.out

date

MPI (Message Passing Interface) Job

- According to SLURM documentation (https://slurm.schedmd.com/mpi_guide.html), "there are three fundamentally different modes of operation used by various MPI implementation with SLURM:
 - *"SLURM directly launches the tasks and performs initialization of communications through the PMI2 or PMIx APIs. (Supported by most modern MPI implementations.)"*
 - *Use mpirun launches tasks using SLURM's infrastructure (not using PMIx).*
 - *SLURM creates a resource allocation for the job and then mpirun launches tasks using some mechanism other than SLURM." (We do not recommend HPC/LSU users use this method to launch their MPI jobs.)"*

MPI Job - (PMIx Versions)

- If you compiled your MPI application using our default mvapich2 libraries (which is compiled with PMIx enabled), you should start the application directly using the **srun** command.

```
#!/bin/bash
#SBATCH --job-name=mpi_job_test      # Job name
#SBATCH --partition=workq            # For jobs using more than 1 node, submit to workq
#SBATCH --nodes=2                    # Number of nodes to be allocated
#SBATCH --ntasks=96                  # Number of MPI tasks (i.e. processes/cores)
#SBATCH --time=00:05:00              # Wall time limit (hh:mm:ss)
#SBATCH --output=%x_%j.log           # Standard output and error

echo ""
echo "SLURM Nodes Allocated           = $SLURM_JOB_NODELIST"
echo "Number of Nodes Allocated       = $SLURM_JOB_NUM_NODES"
echo "Number of Tasks Allocated        = $SLURM_NTASKS"

module load mvapich2/2.3.3/intel-19.0.5
srun -n $SLURM_NTASKS ./a.out
```

MPI Job - (Non-PMIx Versions)

- If your MPI application did not use our default module key `mvapich2/2.3.3/intel-19.0.5`, you should start the application using the `mpirun` command.

```
#!/bin/bash
#SBATCH --job-name=mpi_job_test      # Job name
#SBATCH --partition=workq            # For jobs using more than 1 node, submit to workq
#SBATCH --nodes=2                    # Number of nodes to be allocated
#SBATCH --ntasks=96                  # Number of MPI tasks (i.e. processes/cores)
#SBATCH --time=00:05:00              # Wall time limit (hh:mm:ss)
#SBATCH --output=mpi_test_%j.log     # Standard output and error

echo ""
echo "SLURM Nodes Allocated"          = $SLURM_JOB_NODELIST"
echo "Number of Nodes Allocated"     = $SLURM_JOB_NUM_NODES"
echo "Number of Tasks Allocated"     = $SLURM_NTASKS"

module load mvapich2/2.3.3/intel-19.0.5-hydra
mpirun -n $SLURM_NTASKS ./a.out
```


Hybrid (MPI + SMP) Job

- Hybrid jobs are MPI applications where each MPI process is multi-threaded and can use multiple cores across multiple nodes. If the MPI implementation is compiled with PMIx enabled, use the `srun` command to start the hybrid job, otherwise, use the `mpirun` command to start it.

```
#!/bin/bash
#SBATCH --partition=workq      # Need to submit workq for multiple node jobs
#SBATCH --nodes=2             # Maximum number of nodes to be allocated
#SBATCH --ntasks=4            # Number of MPI tasks (i.e. processes)
#SBATCH --cpus-per-task=24     # Number of cores per MPI task
#SBATCH --time=00:05:00       # Wall time limit (hh:mm:ss)
#SBATCH --output=hybrid_test_%j.log # Standard output and error file
```

```
echo "Number of Nodes Allocated"      = $SLURM_JOB_NUM_NODES"
echo "Number of Tasks Allocated"      = $SLURM_NTASKS"
echo "Number of Cores/Task Allocated" = $SLURM_CPUS_PER_TASK"
```

```
module load mvapich2/2.3.3/intel-19.0.5
```

```
# PMIx version
```

```
srun -n $SLURM_NTASKS -c $SLURM_CPUS_PER_TASK ./a_pmix.out
```

```
# Non-PMIx version
```

```
module swap mvapich2/2.3.3/intel-19.0.5 mvapich2/2.3.3/intel-19.0.5-hydra
```

```
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
```

```
mpirun -n $SLURM_NTASKS ./a_hydra.out
```

Inspect Job Results

- **Check job standard output/standard error by inspecting the redirected files:**

```
[fchen14@mike2 slurmdoc]$ cat hybrid_test_37723.log
```

```
Date                = Tue Sep 22 15:37:44 CDT 2020
```

```
Hostname            = mike001
```

```
Working Directory = /home/fchen14/slurmdoc
```

```
Number of Nodes Allocated      = 2
```

```
SLURM Nodes Allocated         = mike[001-002]
```

```
Number of Tasks Allocated     = 4
```

```
Number of Cores/Task Allocated = 24
```

```
Loading mvapich2/2.3.3/intel-19.0.5
```

```
  Loading requirement: intel/19.0.5
```

```
Autoloading hdf5/1.10.6/intel-19.0.5-mvapich-2.3.3
```

```
Autoloading fftw/3.3.8/intel-19.0.5-mvapich-2.3.3
```

```
Autoloading netcdf-c/4.7.3/intel-19.0.5-mvapich-2.3.3
```

SuperMike-III HPC User Environment

Job Monitoring

Job Monitoring on SuperMike-III

➤ **Check details on your job using `qstat`**

`$ queue -u $USER` : For quick look at nodes assigned to you

`$ scontrol show job <jobid>` : For details on your job

`$ scancel jobid` : To delete job

➤ **Check memory usage of your job using `qshow`**

`$ qshow jobid`

❖ **Please pay close attention to the load and the memory consumed by your job!**

Using the “top” command

- The top program provides a dynamic real-time view of a running system.

```
top - 23:30:16 up 51 days, 16:18, 4 users, load average: 0.16, 0.05, 0.06
Tasks: 692 total, 2 running, 690 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.1 us, 1.0 sy, 0.0 ni, 97.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 19647060+total, 18699553+free, 8677504 used, 797560 buff/cache
KiB Swap: 13421772+total, 13405440+free, 163328 used. 18702988+avail Mem
```

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
208754	fchen14	20	0	7731040	5.5g	20108	R	100.0	2.9	0:16.50	lmp
208999	fchen14	20	0	172868	2948	1624	R	0.7	0.0	0:00.07	top
1	root	20	0	191624	2832	1544	S	0.0	0.0	21:18.21	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:04.81	kthreadd
4	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0.0	0.0	1:06.85	ksoftirqd/0

Check memory Usage for Multi-Node Job

- Check health of your job using `qshow`
\$ `qshow <jobid>`

```
[fchen14@mike2 slurmdoc]$ sbatch ex_lmp_hybrid.sh
```

```
Submitted batch job 37888 estimates 8 SUs from allocation LSU_LSUadmin1.
```

```
Estimated remaining SUs: 521696
```

JOBID	NAME	PARTITION	TIME_LIMIT	ST	CPUS	NODES	REASON
37888	hybrid_job_test	workq	5:00	PD	96	2	None

```
[fchen14@mike2 slurmdoc]$ qshow 37888
```

```
PBS job: 37888, nodes: 2
```

```
Hostname Days Load CPU U# (User:Process:VirtualMemory:Memory:Hours)
```

```
mike005      0 Autoloading 211  6 fchen14:lmf:5847M:3.2G fchen14:lmf:5846M:3.3G
fchen14:slurm_scr+:113M:2M fchen14:srun:388M:5M fchen14:srun:50M:1M
```

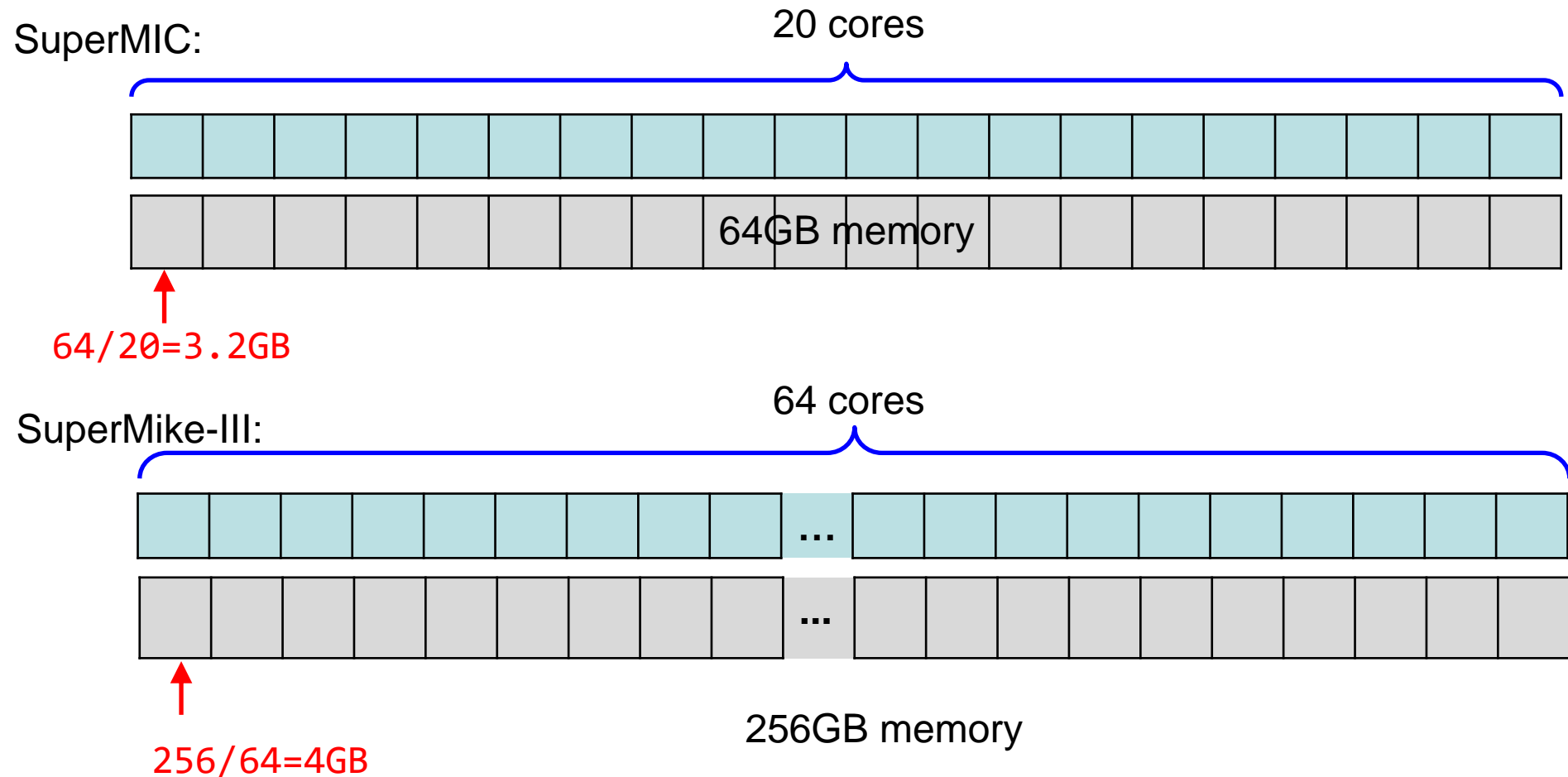
```
mike006      0 Autoloading 216  3 fchen14:lmf:5870M:5.1G fchen14:lmf:4447M:3.3G
```

```
PBS_job=37888 user=fchen14 allocation=LSU_LSUadmin1 queue=workq total_load=0.00
cpu_hours=0.00 wall_hours=0.00 unused_nodes=0 total_nodes=2 ppn=48 avg_load=0.00
avg_cpu=213% avg_mem=7640mb avg_vmem=11746mb
top_proc=fchen14:lmf:mike006:5870M:5.1G:0.0hr:105% node_processes=3
```

Pay attention to single queue usage

- **Single queue** - Used for jobs that will only execute on a single node, i.e. **-N1 -n1-64**.
- **Jobs in the single queue should not use:**
 - More than 3GB memory per core for QB2 (64G/20).
 - More than 4GB memory per core for SuperMike-III (256G/64).
- **If applications require more memory, scale the number of cores (**--ntasks/-n**) to the amount of memory required: i.e. max memory available for jobs in single queue is 16GB for **--ntasks 4** on SuperMike-III.**

Core and Memory in Single queue



Question:

On SuperMIC, if my job needs 17GB memory, what `-ppn` value should I use?

On SuperMike-III, if my job needs 17GB memory, what `--ntasks (-n)` value should I use?

Short Summary

- **How to check queue (partition) information in SLURM?**
- **How to submit SLURM interactive job?**
 - Note the usage of `--overlap`
- **How to submit SLURM batch job?**
 - SMP (OpenMP)
 - MPI
 - Hybrid
- **Monitor your job?**

❖ *Questions?*

Next Sessions

- **11:15 - 12:00**
 - Performance benchmarks and tuning
- **12:00 - 01:30**
 - Lunch break
- **01:30 - 03:30**
 - Q&A + On-ramp sessions (breakout sessions)