HPC User Environment 2

Le Yan
LSU & LONI HPC
sys-help@loni.org

Louisiana State University
Baton Rouge
September 21, 2022

Outline

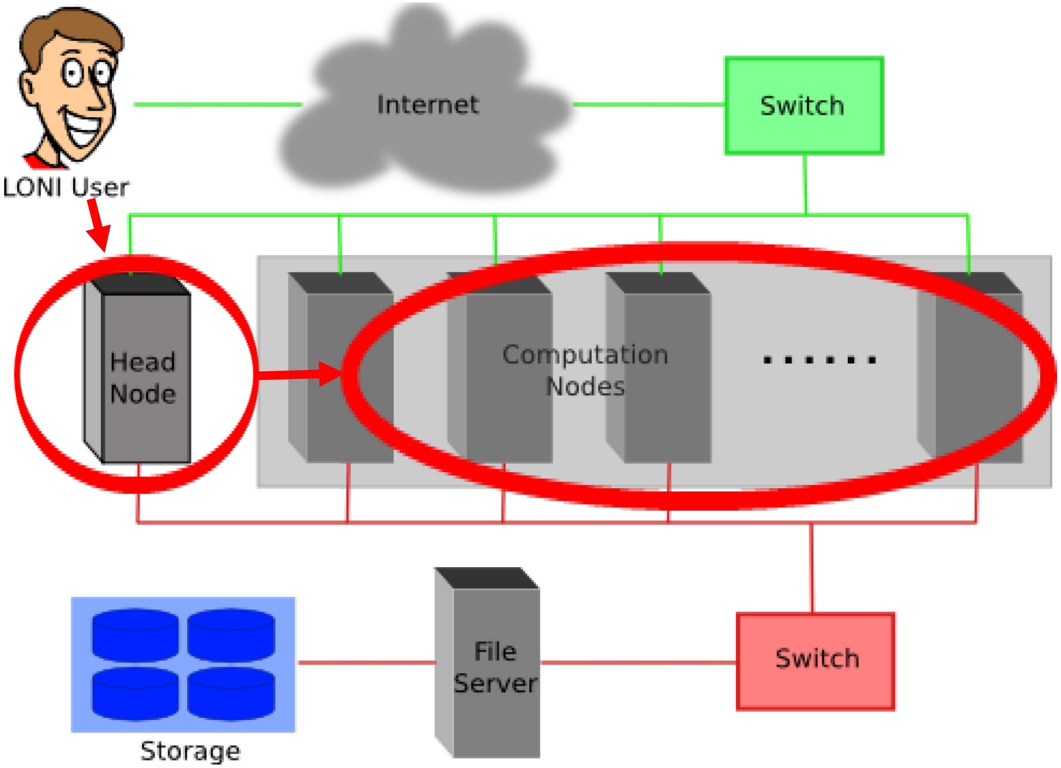
- **Review HPC User Environment 1 topics**
  - Cluster architecture
  - Connect to HPC clusters
  - Software management using module
  - Allocation
- **Things to be covered in this training**
  - Job management
    - Job queue basics
    - Interactive vs Batch jobs
    - Submit and monitor your jobs

HPC User Environment 2

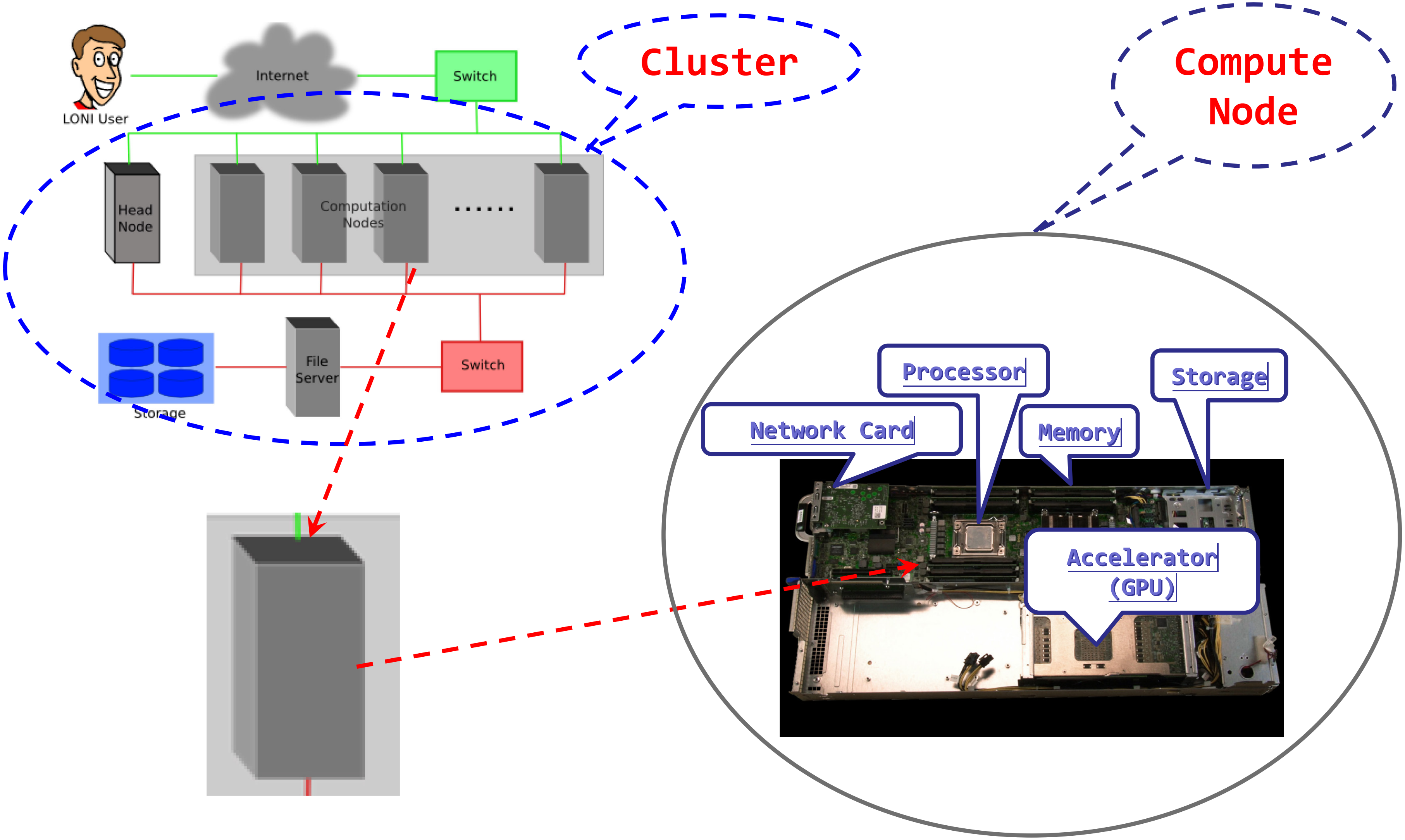
Review of HPC User Environment 1

HPC Cluster Environment

- Multiple compute nodes
- Multiple users
- Each user may have multiple jobs running simultaneously
- Multiple jobs (not necessarily from multiple users) may share the same node

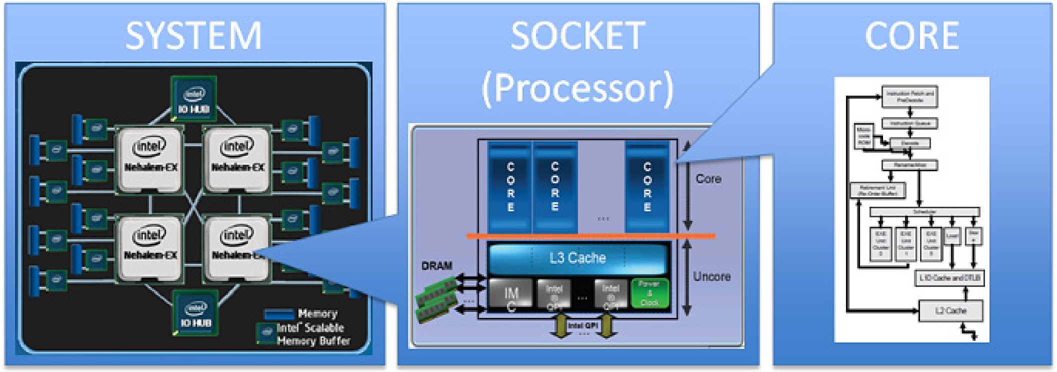


HPC Cluster Environment



Cluster Nomenclature

Term	Definition
Node	A single, named host machine in the cluster.
Core	The basic computing unit of the CPU (processor). For example, a quad-core CPU has 4 cores.
Job	A user's request to use a number of nodes/cores for a certain amount of time on a cluster.



Accessing Cluster Using SSH (Secure Shell)

- **On Unix and Mac**
  - use ssh on a terminal to connect
- **Windows box (ssh client):**
  - MobaXterm (http://mobaxterm.mobatek.net/)
  - Putty, Cygwin (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)
- `ssh username@<cluster host name>`
- **Host name**
  - LONI: <cluster_name>.loni.org
    - <cluster_name> can be:
      - `qb.loni.org (QB-2)`
      - `qbc.loni.org (QB-3)`
  - LSU HPC: <cluster_name>.hpc.lsu.edu
    - <cluster_name> can be:
      - `smic.hpc.lsu.edu (SuperMIC)`
      - `mike.hpc.lsu.edu (SuperMike-3)`
      - `db1.hpc.lsu.edu  (DeepBayou)`

Software Management with Environment Modules

- To list all available or part of packages is: `module av`
  `module av <package name>`
- To see what packages are currently loaded into a user's environment, the command is: `module list`
- The command for loading a package into a user's environment is: `module load <package name>`. If a specific version of a package is desired, the command can be expanded to: `module load <package name>/<package version>`.
- On LSU and LONI clusters, Modules can be loaded automatically upon login by adding the appropriate module load commands to a user's `~/.bashrc` or `~/.modules (recommended)` file

Allocation

- To run jobs, one must have an active allocation.
- Each allocation contains a number of service units (SUs), with one SU equivalent to one core-hour
- All allocations expire after 12 months and are not extensible, even if there are remaining balances.

  - List active allocation balance (and disk usage): showquota

```
[lyan1@db1 ~]$ showquota
User filesystem quotas for lyan1 (uid 24106):
     Filesystem         MB used       quota       files      fquota
     /home                  154       10000        3138           0
     /work /project     4217410           0     3578603     4000000

Storage allocation      MB used       quota       files  expiration
     sa_lyan1                 -           -           -  2000-01-01

CPU Allocation SUs:        remaining   allocated  expiration
    hpc_db_osg01:          818228.33  1100000.00  2023-01-01
    hpc_db_test:            39804.38    50000.00  2022-10-01
    hpc_deepbayou:        2121462.80  2200000.00  2023-01-01
```

Outline

- **Review HPC User Environment 1 topics**
  - Cluster architecture
  - Connect to clusters
  - Software management using and module
  - Allocation
- **Things to be covered in this training**
  - Job management
    - Job queue basics
    - Interactive vs Batch jobs
    - Submit and monitor your jobs

HPC User Environment 2

Job Queue Basics

Basic Concepts

- **Jobs**
  - A user's request to use a number of nodes/cores for a certain amount of time on a cluster.
- **Resource manager/scheduler**
  - A software that
    - Decides which job runs when and where
    - Enforces job policies
  - Two difference resource managers on LONI and LSU clusters
    - PBS: QB-2, SuperMIC
    - Slurm: QB-3, Deep Bayou, and SuperMike-3
- **Job management (as users' responsibilities)**
  - Decide a job's size (in terms of nodes and cores) and duration
  - Understand the job queuing system and policies
  - Submit/monitor/cancel jobs
  - Diagnose job health

Job Queues

- **Nodes on an HPC cluster are organized into queues (PBS) / partitions (Slurm).**
  - They are called "queues", but there might not be a strict "First Come First Serve" policy.
- **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.
- **When submitting a job, a user needs to specify the job parameters such as queue, size (number of nodes/cores), duration, etc.**

Queue Characteristics – LONI Clusters

Machine	Queue	Max Runtime	ppn	Max running jobs	Max nodes per job	Use
QB-2	workq	3 days	20	64	128	Unpreemptable
	checkpt		20		128	Preemptable
	bigmem		48		1	Big memory
	single	7 days	1,2,4,6,8		1	Single node jobs
QB-3	workq	3 days	48	32	96	Unpreemptable
	checkpt		48		96	Preemptable
	gpu		48		8	Preemptable
	bigmem		48		1	Big memory
	single	7 days	1-47		1	Single node jobs

Unpreemptable vs Preemptable

http://www.adaptivecomputing.com/blog-hpc/understanding-moab-scheduling-part-iii/

Queue Characteristics – LSU Clusters

Machine	Queue	Max Runtime	ppn	Max running jobs	Max nodes per job	Use
SuperMIC	workq	3 days	20	34	128	Unpreemptable
	checkpt		20		200	Preemptable
	v100		36		2	Job using GPU
	single	7 days	1,2,4,6,8		1	Single node jobs
DeepBayou	checkpt	3 days	48	4	4	Preemptable
	nvlink		48		1	Job using GPU
	single		1 to 47		1	Single node jobs
SuperMike3	workq	3 days	64	32	84	Unpreemptable
	checkpt		64		84	Preemptable
	gpu		64		8	Job using GPU
	single	7 days	1 to 63		1	Single node jobs

Queue Characteristics

- The “qstat -q” command displays info on the queues

```
  [lyan1@qbc1 ~]$ qstat -q
Queue            Memory CPU Time Walltime Node  Run Que Lm State
---------------- ------ -------- -------- ----  --- --- -- -----
admin              --      --       --      --    0   0 --  E R
single             --      --    168:00:00   1   32   0 --  E R
checkpt            --      --    72:00:00  --   35  33 --  E R
workq              --      --    72:00:00  --   17  17 --  E R
gpu                --      --    72:00:00  --    3   0 --  E R
bigmem             --      --    72:00:00  --    0   0 --  E R
priority           --      --    72:00:00  --    0   0 --  E R
                                               ----- -----
                                                  87    50
```

Queue Querying

- The “showq” command displays information about active, eligible, blocked, and/or recently completed jobs

```
$ showq
active jobs------------------------
JOBID              USERNAME      STATE PROCS   REMAINING            STARTTIME
236875              ebeigi3    Running    16     1:44:29  Mon Sep 15 20:00:22
236934                 mwu3    Running    16    00:03:27  Mon Sep 15 19:04:20
...
eligible jobs----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT            QUEUETIME
236795              dmarce1       Idle  1456    00:15:00  Mon Sep 15 16:38:45
236753               rsmith       Idle  2000     4:00:00  Mon Sep 15 14:44:52
236862              dlamas1       Idle   576     2:00:00  Mon Sep 15 17:28:57
...
121 eligible jobs
blocked jobs-----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT            QUEUETIME
232741              myagho1       Idle  2000  1:00:00:00  Mon Sep  8 07:22:12
235545              tanping       Idle     1  2:21:10:00  Fri Sep 12 16:50:49
235546              tanping       Idle     1  2:21:10:00  Fri Sep 12 16:50:50
...
```

Queue Querying – Free Nodes

- **The “qfree” command queries the free nodes in each queue**

```
[lyan1@qbc1 ~]$ qfree
PBS total nodes: 202,  free: 7,  busy: 188,  down: 4,  use: 93%
PBS single nodes: 192,  free: 3,  busy: 32,  queued: 0
PBS workq nodes: 192,  free: 3,  busy: 51,  queued: 190
PBS checkpt nodes: 192,  free: 3,  busy: 107,  queued: 136
PBS bigmem nodes: 2,  free: 2,  busy: 0,  queued: 0
PBS gpu nodes: 8,  free: 5,  busy: 3,  queued: 0
```

Queue Characteristics (Slurm Only)

- **The “sinfo” command (QB-3 and Deep Bayou) displays more info on the queues**

```
 [fchen14@qbc1 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
single*      up 7-00:00:00      4  drain qbc[114-115,119-120]
single*      up 7-00:00:00    119  alloc qbc[001-002,006-018,021-024,026,031-039,041-057,062-
066,069-076,079-086,088-093,095-113,116-117,121-126,148-151,154-163,166,186-189]
single*      up 7-00:00:00     69   idle qbc[003-005,019-020,025,027-030,040,058-061,067-
068,077-078,087,094,118,127-147,152-153,164-165,167-185,190-192]
checkpt      up 3-00:00:00      4  drain qbc[114-115,119-120]
checkpt      up 3-00:00:00    119  alloc qbc[001-002,006-018,021-024,026,031-039,041-057,062-
066,069-076,079-086,088-093,095-113,116-117,121-126,148-151,154-163,166,186-189]
checkpt      up 3-00:00:00     69   idle qbc[003-005,019-020,025,027-030,040,058-061,067-
068,077-078,087,094,118,127-147,152-153,164-165,167-185,190-192]
workq        up 3-00:00:00      4  drain qbc[114-115,119-120]
workq        up 3-00:00:00    119  alloc qbc[001-002,006-018,021-024,026,031-039,041-057,062-
066,069-076,079-086,088-093,095-113,116-117,121-126,148-151,154-163,166,186-189]
workq        up 3-00:00:00     69   idle qbc[003-005,019-020,025,027-030,040,058-061,067-
068,077-078,087,094,118,127-147,152-153,164-165,167-185,190-192]
gpu          up 3-00:00:00      8   idle qbc[193-200]
bigmem       up 3-00:00:00      2   idle qbc[201-202]
```

Choosing A Queue for Your Jobs

- **Before choosing a queue, understand your needs first**
  - Job size
    - If your code is serial, use the single queue
    - If your code is parallel, then you need to run the same job a few times with incremental core and node counts until the best configuration is found
  - Job duration
    - Should be long enough for your code to finishing running
    - Should be as short as possible to allow quicker turnaround
  - Other considerations
    - Does the code use GPUs? (use the queues where GPU nodes are)
    - Does your job use a lot of memory (use the "bigmem" queue)
- **For most users, the "checkpt" and "single" queues are where their jobs will be submitted**

HPC User Environment 2

Submit and Monitor Jobs

Two Job Types

- **Interactive job**
  - Set up an interactive environment on compute nodes for users
    - Advantage: can run programs interactively
    - Disadvantage: must be present when the job starts
  - Use case: testing and debugging, compiling
    - **NEVER RUN COMPUTATIONALLY INTENSIVE TASKS ON THE HEAD NODE (Login Node)**
    - Try not to run interactive jobs with large core count, which is usually a waste of resources
- **Batch job**
  - Executed without user intervention using a job script
    - Advantage: the system takes care of everything
    - Disadvantage: can only execute one sequence of commands which cannot changed after submission
  - Use case: production run

Submitting Jobs on Linux Clusters

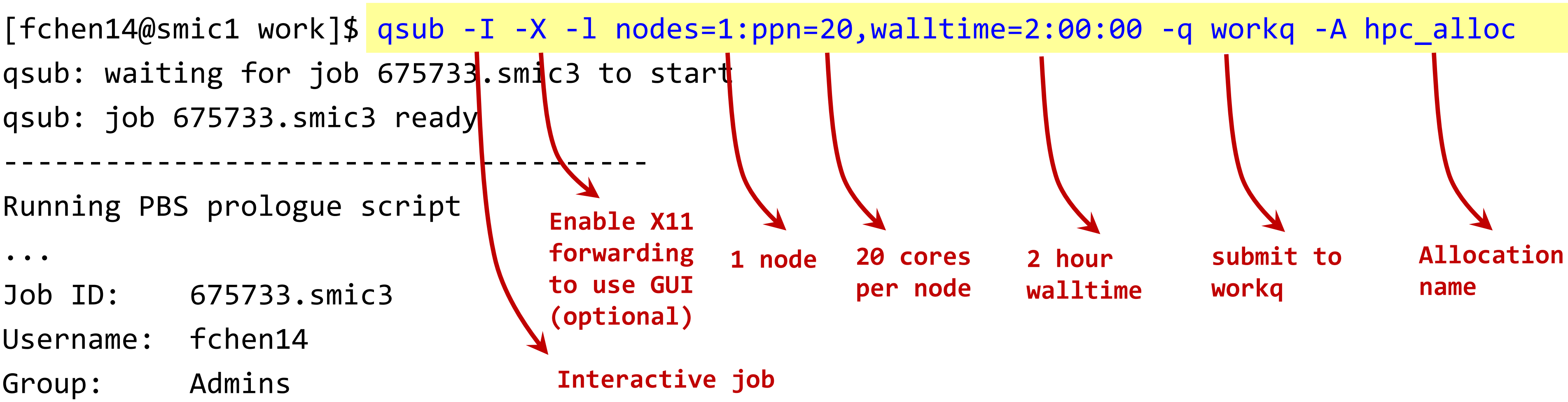
- **Interactive job example:**
  - PBS for SuperMIC and QueenBee2

```
qsub -I \
     -l walltime=<hh:mm:ss>,nodes=<num_nodes>:ppn=<num_cores> \
     -A <Allocation> \
     -q <queue name> \
     -X to enable X11 forwarding (if needed)
```

  - SLURM for DeepBayou and QueenBee3

```
srun -t hh:mm:ss  \
     -N short for --nodes, number of nodes \
     -n short for --ntasks, number of tasks to run job on \
     -c short for --ncpus-per-task, number of threads per process \
     -A <Allocation> \
     -p  <queue name> \
     --x11 enable X11 forwarding (if needed) \
      --pty bash
```

Submit a PBS Interactive Job on SuperMIC



```
[fchen14@smic1 work]$ qsub -I -X -l nodes=1:ppn=20,walltime=2:00:00 -q workq -A hpc_alloc
qsub: waiting for job 675733.smic3 to start
qsub: job 675733.smic3 ready
--------------------------------------
Running PBS prologue script
...
Job ID:     675733.smic3
Username:   fchen14
Group:      Admins
Date:       13-Jun-2017 15:34
Node:       smic044 (62703)
--------------------------------------
PBS has allocated the following nodes:
smic044
A total of 16 processors on 1 nodes allocated
---------------------------------------------
...
Concluding PBS prologue script - 13-Jun-2017 15:34:19
------------------------------------------------------
[fchen14@smic044 ~]$
```

Interactive job

Enable X11 forwarding to use GUI (optional)

1 node

20 cores per node

2 hour walltime

submit to workq

Allocation name

The maximum run time for an interactive job is 12 hours.

Submit a PBS Interactive Job on SuperMIC

```
[fchen14@smic1 work]$ qsub -I -X -l nodes=1:ppn=20,walltime=2:00:00 -q workq -A hpc_alloc
qsub: waiting for job 675733.smic3 to start
qsub: job 675733.smic3 ready
--------------------------------------
Running PBS prologue script
...
Job ID:     675733.smic3
Username:   fchen14
Group:      Admins
Date:       13-Jun-2017 15:34
Node:       smic044 (62703)
--------------------------------------
PBS has allocated the following nodes:
smic044
A total of 16 processors on 1 nodes allocated
---------------------------------------------
...
Concluding PBS prologue script - 13-Jun-2017 15:34:19
------------------------------------------------------
[fchen14@smic044 ~]$
```

- Note the digit change in the host name AND the directory change.

The maximum run time for an interactive job is 12 hours.

Submit a PBS Interactive Job on QB-2

```
[ychen64@qb2 work]$ qsub -I -X -l nodes=1:ppn=20,walltime=02:00:00, -q workq -A loni_alloc
qsub: waiting for job 505851.qb3 to start
qsub: job 505851.qb3 ready
--------------------------------------
Running PBS prologue script
--------------------------------------
User and Job Data:
--------------------------------------
Job ID:     505851.qb3
Username:   ychen64
Group:      loniadmin
Date:       13-Jun-2018 01:27
Node:       qb061 (4497)
--------------------------------------
PBS has allocated the following nodes:
…
-------------------------------------------------------
Concluding PBS prologue script - 13-Jun-2018 01:27:39
-------------------------------------------------------
[ychen64@qb061 ~]$
```

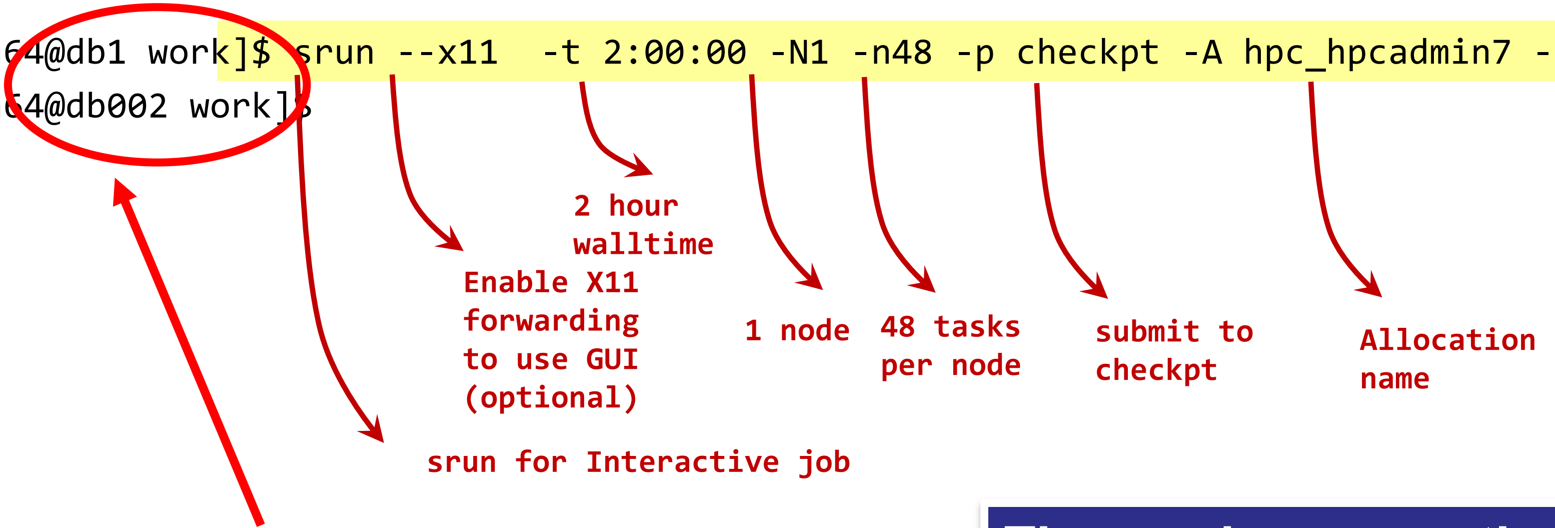
20 cores per node

Allocation name

The maximum run time for an interactive job is 12 hours.

Submit a SLURM Interactive Job on DeepBayou and QB-3

```
[ychen64@db1 work]$ srun --x11  -t 2:00:00 -N1 -n48 -p checkpt -A hpc_hpcadmin7 --pty bash
[ychen64@db002 work]$
```



srun for Interactive job

Enable X11 forwarding to use GUI (optional)

2 hour walltime

1 node

48 tasks per node

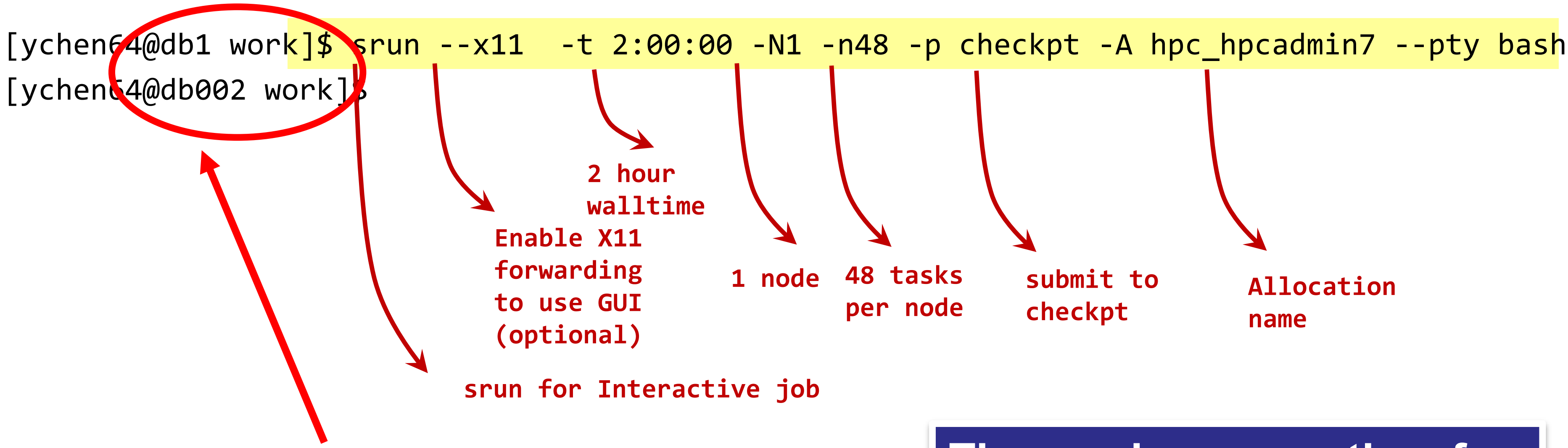
submit to checkpt

Allocation name

- Note the digit change but no change in the directory.

The maximum run time for an interactive job is 12 hours.

Submit a SLURM Interactive Job on DeepBayou and QB-3



```
[ychen64@db1 work]$ srun --x11  -t 2:00:00 -N1 -n48 -p checkpt -A hpc_hpcadmin7 --pty bash
[ychen64@db002 work]$
```

srun for Interactive job

Enable X11 forwarding to use GUI (optional)

2 hour walltime

1 node

48 tasks per node

submit to checkpt

Allocation name

- Note the digit change but no change in the directory.

The maximum run time for an interactive job is 12 hours.

PBS: When a job (interactive and batch) starts, the current directory will be **home**

Slurm: When a job (interactive and batch) starts, the current directory will be the one **where the job is submitted**.

Running MPI Programs in A Slurm Interactive Job

- **The "--overlap" option MUST be used with the srun command.**
  - Otherwise, it will hang

This will hang.

```
[lyan1@qbc016 pi]$ srun -n48 <my_mpi_executable>
```

This will run.

```
[lyan1@qbc016 pi]$ srun --overlap -n48 <my_mpi_executable>
```

PBS Environmental Variables

```
[fchen14@smic315 ~]$ echo $PBS_
$PBS_ENVIRONMENT  $PBS_MOMPORT      $PBS_NUM_PPN      $PBS_O_MAIL
$PBS_QUEUE        $PBS_WALLTIME     $PBS_GPUFILE      $PBS_NODEFILE
$PBS_O_HOME       $PBS_O_PATH       $PBS_SERVER       $PBS_JOBCOOKIE
$PBS_NODENUM      $PBS_O_HOST       $PBS_O_QUEUE      $PBS_TASKNUM
$PBS_JOBID        $PBS_NP           $PBS_O_LANG       $PBS_O_SHELL
$PBS_VERSION      $PBS_JOBNAME      $PBS_NUM_NODES    $PBS_O_LOGNAME
$PBS_O_WORKDIR    $PBS_VNODENUM
```

$PBS_NODEFILE: the list of the nodes allocated to the current job (useful for MPI jobs)

$PBS_O_WORKDIR: the directory where the job is submitted

SLURM Environmental Variables

```
[ychen64@qbc025 ~]$ echo $SLURM_
$SLURM_CLUSTER_NAME           $SLURM_JOB_UID                $SLURM_STEPID
$SLURM_CPU_BIND               $SLURM_JOB_USER               $SLURM_STEP_ID
$SLURM_CPU_BIND_LIST          $SLURM_LAUNCH_NODE_IPADDR     $SLURM_STEP_LAUNCHER_PORT
$SLURM_CPU_BIND_TYPE          $SLURM_LOCALID                $SLURM_STEP_NODELIST
$SLURM_CPU_BIND_VERBOSE       $SLURM_MPI_TYPE               $SLURM_STEP_NUM_NODES
$SLURM_CPUS_ON_NODE           $SLURM_NNODES                 $SLURM_STEP_NUM_TASKS
$SLURM_GTIDS                  $SLURM_NODEID
$SLURM_STEP_TASKS_PER_NODE
$SLURM_JOB_ACCOUNT            $SLURM_NODELIST               $SLURM_SUBMIT_DIR
$SLURM_JOB_CPUS_PER_NODE      $SLURM_NPROCS                 $SLURM_SUBMIT_HOST
$SLURM_JOB_GID                $SLURM_NTASKS                 $SLURM_TASK_PID
$SLURM_JOBID                  $SLURM_PRIO_PROCESS           $SLURM_TASKS_PER_NODE
$SLURM_JOB_ID                 $SLURM_PROCID                 $SLURM_TOPOLOGY_ADDR
$SLURM_JOB_NAME               $SLURM_PTY_PORT
$SLURM_TOPOLOGY_ADDR_PATTERN
$SLURM_JOB_NODELIST           $SLURM_PTY_WIN_COL            $SLURM_UMASK
$SLURM_JOB_NUM_NODES          $SLURM_PTY_WIN_ROW            $SLURM_WORKING_CLUSTER
$SLURM_JOB_PARTITION          $SLURM_SRUN_COMM_HOST
$SLURM_JOB_QOS                $SLURM_SRUN_COMM_PORT
```

Demo/Exercise

- **Start an interactive job session with 1 node for 1 hour**
  - Find out your allocation name if you don't remember
  - Decide which queue to use
  - Use "`qsub -I`" or "`srun`" , including all necessary options
  - Once the job starts, verify that you are NOT on the head node

Demo/Exercise (Continued)

- **Computing an approximate value for Pi**
  - cd to your work directory

```
$ cd /work/$USER
```

  - Download the tarball from HPC website to the home directory

```
$ wget http://www.hpc.lsu.edu/training/weekly-materials/Downloads/pi.tar.gz
```

  - Untar it

```
$  tar -xvzf pi.tar.gz
```

  - cd to the directory “pi”

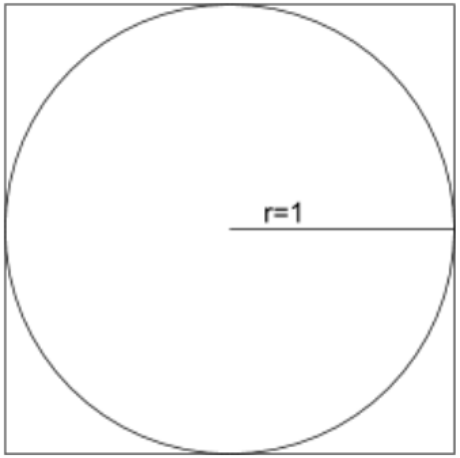
```
$ cd pi
```

  - Use “module list” to make sure the mvapich2 is loaded.
  - Execute serial or mpi version

```
$ ./serialpi.out #serial version, if no argument given, default value 1000000000
# MPI version:
# QueenBee2 or SuperMIC:
$ mpirun -np 20 ./mpi_pi.out 100000000000  # default 1000000000000
# DeepBayou or QueenBee3
$ srun --overlap -n48./ mpi_pi.out 100000000000  # default 1000000000000
```

Computing an approximate value for Pi

- The executables in this training calculate the value for PI based on the math which is actually quite simple: Imagine a square dartboard with circle inscribed within it such that the diameter of the circle is the length of a side of the square.



- We can observe that the ratio of the area of the circle to the area of the square is equal to some constant, π/4 (since the square's area is 2*2 = 4 and area_circle = π*r^2 = π). If we randomly place many points (darts) inside the square, we can count how many are also inside the circle (satisfy x^2+y^2 <= 1) vs the total number of points and compute an estimate for the value of π. (Problem description is from Jared Baker, UW; Ben Matthews, NCAR)

During the break…

- **Finish the exercise run.**
- **If you are not familiar with the Linux commands used in the exercise, review the Linux commands cheat sheet in the next slide.**

Linux Commands Cheat Sheet

- **History** # Show the history of commands
- **mkdir <name of directory>** # creates a directory
- **ls** # list files/directories
  - **-a** list all files including hidden ones
  - **-l** shows files with a long listing format
- **cd** # change directory
- **pwd** # shows the current working directory
- **cp** # copy
- **rm** # Remove files (careful)
- **Up arrow (↑)** # moves back in command history
- **Tab** -> fills in unique file name
- **Tab Tab** -> press tab twice, shows all available file names

Two Job Types

- **Interactive job**
  - Set up an interactive environment on compute nodes for users
    - Advantage: can run programs interactively
    - Disadvantage: must be present when the job starts
  - Use case: testing and debugging, compiling
    - **NEVER RUN COMPUTATIONALLY INTENSIVE TASKS ON THE HEAD NODE (Login Node)**
    - Try not to run interactive jobs with large core count, which is usually a waste of resources
- **Batch job**
  - Executed without user intervention using a job script
    - Advantage: the system takes care of everything
    - Disadvantage: can only execute one sequence of commands which cannot changed after submission
  - Use case: production run

Submit a Batch Job

- **PBS batch Job example:**

```
[ychen64@qb2 pi]$ qsub qsub.submit
```

Job submission command

Job script

- **SLURM batch Job example:**

```
[ychen64@qbc1 pi]$ sbatch sbatch.submit
```

Job submission command

Job script

- **Batch job cannot be submitted when you are on the compute node**

```
[ychen64@qb023 pi]$ qsub qsub.submit
qsub: Bad UID for job execution MSG=ruserok failed validating
ychen64/ychen64 from qb023
```

PBS Job Script – Parallel Job

- Note: don’t let your <path_to_executable> <options> be the EndOfFile
  - EOF can be <shell commands>, comments or a blank line.

SLURM Job Script – Parallel Job

- Note: don’t let your <path_to_executable> <options> be the EndOfFile
  - EOF can be <shell commands>, comments or a blank line.

Single Queue Jobs (1)

- **The "single" queue is for jobs that do not need all on a compute node**
  - Example: your job may only need 1 core + 2 GB memory, or 4 cores + 12 GB memory
- **Jobs in the "single" queue share nodes, i.e. there could multiple single queue jobs running on the same node**
- **The maximum amount of CPU cores and memory allowed for a "single" queue job is determined by the value of "ppn" (PBS) or "-n" (Slurm) flag**

Cluster	Job manager	Memory per core (GB)	Max cores for job	Max memory (GB) for job
QB-3	Slurm	192/48 = 4	-n value	(-n value) * 4
Deep Bayou	Slurm	192/48 = 4		
SuperMike-3	Slurm	256/64 = 4		
QB-2	PBS	64/20 = 3.2	ppn= value	(-n value) * 3.2
SuperMIC	PBS	64/20 = 3.2		

Single Queue Jobs (2)

- **When the maximum allowed cores or memory is exceeded, the owner of the job may receive warning messages:**
  - E124 - Exceeded memory allocation. This Job XXXX appears to be using more memory (GB) than allocated (9 > 3).
  - E123 - Exceeded ppn/core allocation. This Job XXXX appears to be using more cores than allocated (6 > 1). Please allocate the number of cores that the job will use, (ppn=6). This Job has 1 core(s) allocated (ppn=1).

Single Queue Jobs (3)

- **On PBS clusters, only a handful of "ppn" values are allowed: 1/2/4/6/8**
- **On Slurm clusters, "-n" can be any value between 1 and N-1, where N is the number of cores on a node**
- **How to calculate the value of "ppn" (PBS) or "-n" (Slurm) when submitting a job to the "single" queue**
  - Step 1: calculate the amount of available memory per core
  - Step 2: get a "ppn" or "-n" number by dividing the total memory usage by the amount of available memory per core
  - Step 3: compare the "ppn" or "-n" number obtained in Step 2 to the number of cores needed by the job, and select the greater one as the "ppn" or "-n" for the job

Cluster: QB-3 (Slurm)
Job needs: 4 cores and 27 GB memory

Step 1: memory per core for QB-3 is 4 GB
Step 2: "-n" = 27/4 ≈ 7
Step 3: Since 7 > 4, so "-n" should be 7

Cluster: QB-2 (PBS)
Job needs: 6 cores and 14 GB memory

Step 1: memory per core for QB-2 is 3.2 GB
Step 2: "ppn" = 14/3.2 ≈ 5
Step 3: Since 6 > 5, so "ppn" should be 6

PBS Job Script – Single Queue

```
#!/bin/bash
#PBS -l nodes=1:ppn=4        # Number of nodes and processor
#PBS -l walltime=24:00:00    # Maximum wall time
#PBS -N myjob                # Job name
#PBS -o <file name>          # File name for standard output
#PBS -e <file name>          # File name for standard error
#PBS -q single               # The queue for serial jobs
#PBS -A <allocation>         # Allocation name
#PBS -m e                    # Send mail when job ends
#PBS -M <email address>      # Send mail to this address

<shell commands>
<path_to_executable> <options>
<shell commands>
```

Job parameters for PBS

Commands to execute when the jobs starts

- Note: don’t let your <path_to_executable> <options> be the EOF
  - EOF can be <shell commands>, comments or a blank line.

SLURM Job Script – Single Queue

```
#!/bin/bash
#SBATCH -N 1                       #number of nodes
#SBATCH -n 2                       #total number of MPI processes
#SBATCH -t hh:mm:ss                #short for --time
#SBATCH -o <file name>             #File name for standard output
#SBATCH -e <file name>             #File name for standard error
#SBATCH -p single                  #Queue name
#SBATCH -A <allocation>            #Allocation name
#SBATCH --mail-type END            #Send mail when job ends
#SBATCH --mail-user <email>        #Send mail to this address
```

Job parameters for Slurm

```
<shell commands>
<path_to_executable> <options>
<shell commands>
```

Commands to execute when the jobs starts

- Note: don’t let your <path_to_executable> <options> be the EndOfFile
  - EOF can be <shell commands>, comments or a blank line.

Job Deleting/Monitoring - PBS

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

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

- **Delete job using `qdel`**

```
$ qdel <jobid>
```

- **Check details of your job using `checkjob`**

```
$ checkjob <jobid>
```

- **More information on PBS can be found at http://hpc.loni.org/docs/pbs.php**

Job Deleting/Monitoring - SLURM

- Check details on your job using `squeue`

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

- Delete job using `scancel`

```
$ scancel -c <job-id>
```

- Check details of your job using `scontrol`

```
$ scontrol show job <job-id>
```

- More information on Slurm can be found at http://hpc.loni.org/docs/slurm.php

Job Health Diagnosis

- **A healthy job**
  - Uses the allocated resources fully and efficiently
  - Does not underutilize allocated nodes/cores/memory
  - Does not overutilize allocated nodes/cores/memory

A job requesting N nodes ≠ A job utilizing N nodes

- **User responsibilities**
  - Check the number of processes on each node
  - Check CPU load
  - Check memory usage

~~We reserve the rights to refuse services to any customer.~~

You will receive bunch of warning emails from us.

Using the “qshow” command

- **The `qshow <job id>` command collects and displays information about a running job.**
  - How busy the CPU cores are
  - What are the running user processes and their memory consumption
- **It should be run on the head node.**

```
$ squeue -u lyan1
JOBID      PARTITION  NAME                   USER      ST  TIME_LIMIT   TIME
CPUS  NODES  NODELIST(REASON)
263160     checkpt    bash                   lyan1     R   12:00:00     3:30
48    1       qbc186

$ qshow 263160
```

Using the “qshow” command

```
$ qshow 263160

PBS job: 263160, nodes: 1
Hostname  Days Load CPU U# (User:Process:VirtualMemory:Memory:Hours)
qbc186      57 46.11 4637 52 lyan1:mpi_pi:864M:35M lyan1:mpi_pi:863M:35M
lyan1:mpi_pi:863M:35M lyan1:mpi_pi:863M:35M lyan1:mpi_pi:863M:35M
lyan1:mpi_pi:863M:35M lyan1:mpi_pi:863M:35M lyan1:mpi_pi:863M:35M
lyan1:mpi_pi:863M:35M lyan1:mpi_pi:863M:35M lyan1:mpi_pi:863M:35M
lyan1:mpi_pi:863M:35M lyan1:mpi_pi:863M:35M lyan1:mpi_pi:863M:35M
lyan1:mpi_pi:863M:35M lyan1:srun:325M:5M lyan1:srun:43M:1M
PBS_job=263160 user=lyan1 allocation=loni_loniadmin1 queue=checkpt
total_load=46.11 cpu_hours=1.60 wall_hours=0.00 unused_nodes=0 total_nodes=1
ppn=48 avg_load=46.11 avg_cpu=4637% avg_mem=1700mb avg_vmem=42390mb
top_proc=lyan1:mpi_pi:qbc186:864M:35M:0.0hr:100%  node_processes=52
```

The normal behavior is	You should be suspicious when
If using whole node, the load should be close to the total number of cores on the node	The load is consistently low
The number of processes should match the value of “ppn” or “-n”	The values do not match (either too high or too low)
The memory (not virtual memory) should not exceed the per core value	The memory exceeds the per core value

Using the “top” command

- The Linux `top` command provides a dynamic real-time view of a running system.
- Should be used on the compute node assigned to you (ssh to it first)

```
$ squeue -u lyan1
JOBID      PARTITION  NAME                 USER      ST   TIME_LIMIT   TIME
CPUS  NODES  NODELIST(REASON)
263160     checkpt    bash                 lyan1     R    12:00:00     3:30
48    1       qbc186

$ ssh qbc186

$ top –u lyan1
```

Using the “top” command

```
$ top -u lyan1
top - 19:46:09 up 57 days,  8:49,  1 user,  load average: 48.05, 44.54, 27.42
Tasks: 707 total,  49 running, 658 sleeping,   0 stopped,   0 zombie
%Cpu(s):100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 19663684+total, 18857572+free,  4611844 used,  3449280 buff/cache
KiB Swap: 13421772+total, 13387878+free,   338944 used. 19058388+avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
238750 lyan1     20   0  864708  34720  20776 R 100.0  0.0  12:22.75 mpi_pi.out
238761 lyan1     20   0  863312  35052  20432 R 100.0  0.0  12:22.93 mpi_pi.out
238763 lyan1     20   0  863312  35108  20428 R 100.0  0.0  12:23.19 mpi_pi.out
238765 lyan1     20   0  863312  35060  20436 R 100.0  0.0  12:23.13 mpi_pi.out
238769 lyan1     20   0  863312  35056  20432 R 100.0  0.0  12:23.24 mpi_pi.out
238779 lyan1     20   0  863312  35120  20436 R 100.0  0.0  12:23.23 mpi_pi.out
238780 lyan1     20   0  863312  35020  20400 R 100.0  0.0  12:23.23 mpi_pi.out
238781 lyan1     20   0  863312  35060  20440 R 100.0  0.0  12:23.20 mpi_pi.out
238784 lyan1     20   0  863312  35128  20444 R 100.0  0.0  12:23.13 mpi_pi.out
238797 lyan1     20   0  863312  35056  20436 R 100.0  0.0  12:23.28 mpi_pi.out
```

The normal behavior is	You should be suspicious when
All processes should be close to 100% busy	The %CPU value is consistently low

Using the “free” command

- The Linux `free` command displays the total amount of free and used physical and swap memory in the system
- Should be used on the compute node assigned to you (ssh to it first)

```
$ qstat -n -u lyan1
…
   smic032/19+smic032/18+smic032/17+smic032/16+smic032/15+smic032/14+smic032/13
   +smic032/12+smic032/11+smic032/10+smic032/9+smic032/8+smic032/7+smic032/6
   +smic032/5+smic032/4+smic032/3+smic032/2+smic032/1+smic032/0
$ ssh smic032
$ free -h
                   total        used         free       shared     buffers      cached
Mem:                 62G        3.1G          59G         177M         31M        1.3G
-/+ buffers/cache:              1.7G          61G
Swap:               127G          0B         127G
```

The normal behavior is	You should be suspicious when
The amount “used” should be significantly lower than the “total”	The “used” is very close to the “total” and the “free” is very low

A Few More Things

- If you run jobs in the “bigmem” queue, make sure that they do need more memory than available on the regular nodes
  - 64 GB for QB-2 and SuperMIC, 192 GB for QB-3
- Before you submit an excessive number (e.g. thousands) of single queue jobs, please consult us
- If you run jobs in the “gpu” queue, make sure that the GPUs are used
  - How to check: ssh to the node where the job is running and run the “nvidia-smi” command – if there are processes in the output, your job is fine.
- Again, the goal is to estimate job needs as accurately as possible and avoid under- and over-utilizing allocated resources

Most Common User Mistakes

- Use more memory than allowed. (e.g. use 5GB memory on SuperMIC with ppn=1)
- Seriously underutilize node resources (e.g. allocate 32 nodes but just use 1 core)
- Submit a job to the big memory queue but use only few MB of memory
- Repeatedly running intensive jobs on the headnode (login node)

Demo/Exercise

- **Submit a batch job**
  - cd to the directory “pi”

```
$ cd pi
```

  - edit qsub.submit (change allocation name, email, ppn=, mpirun etc.)

```
$ vi qsub.submit
```

  - submit job

```
$ qsub qsub.submit
```

- **Check details on your job using `qstat` or `squeue`**

```
$ qstat -n -u $USER   #PBS
$ squeue -l -u $USER  #SLURM
```

- **Monitor the job**
  - `qshow` **or** `scontrol`
  - `top` (must ssh to the compute node assigned to your job)
  - `free` (must ssh to the compute node assigned to your job)

Summary

- **A job is a user's request to use a number of nodes/cores for a certain amount of time on a cluster.**
- **Resource manager/scheduler decides which job runs when and where and enforces job policies**
  - PBS for QB-2 and SuperMIC, and Slurm for QB-3, Deep Bayou, and SuperMike-3
- **Job management as users’ responsibilities**
  - Decides a job’s size (in terms of nodes and cores) and duration
  - Understand the job queuing system and policies
  - Submit/monitor/cancel jobs
  - Diagnose job health (CPU core and memory usage compared to the requested amounts)

Future Training

- **1. September 21, 2022: HPC User Environment 2**
- ***2. September 28, 2022: Basic Shell Scripting***

- **Keep an eye on:**
  - http://www.hpc.lsu.edu/training/tutorials.php#upcoming

HPC User Services

- **Services provided**
  - Access to HPC clusters (2 for LONI, 3 for LSU)
  - Access to the most commonly used software packages
    - Compilers, libraries, applications
  - User support and consultation
  - HPC training
    - Linux, bash, Python, container etc.
- **Contact HPC user services**
  - Email Help Ticket: ***sys-help@loni.org***