HPC User Environment 2

Jason Li
HPC User Services
LSU HPC / LONI
sys-help@loni.org

Louisiana State University, Baton Rouge
Jul 10, 2024
Outlines

- **HPC User Environment 1**
  1. Intro to HPC
  2. Getting started
  3. Into the cluster
  4. Software environment (modules)

- **HPC User Environment 2**
  1. Basic concepts
  2. Preparing my job
  3. Submitting my job
  4. Managing my jobs
Outlines

- HPC User Environment 2
  1. Basic concepts
  2. Preparing my job
  3. Submitting my job
  4. Managing my jobs
Outlines

HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers

2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues

3. Submitting my job
   1) Interactive job
   2) Batch job

4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
Outlines

- HPC User Environment 2

  1. Basic concepts
     1) Previously on HPC User Environment 1…
     2) Job & Job schedulers
  2. Preparing my job
     1) Basic principles
     2) Job duration (wall time)
     3) Number of nodes & cores
     4) Job queues
  3. Submitting my job
     1) Interactive job
     2) Batch job
  4. Managing my jobs
     1) Useful commands
     2) Monitoring job health
• **HPC User Environment 2**

1. **Basic concepts**
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers
2. **Preparing my job**
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues
3. **Submitting my job**
   1) Interactive job
   2) Batch job
4. **Managing my jobs**
   1) Useful commands
   2) Monitoring job health
Two things needed to run jobs on our clusters:

1) Account
2) Allocation
1) Previously on HPC User Environment 1…
1) Previously on HPC User Environment 1...

Run my code on all the resources you have, however long it takes.

```
sudo!
yum!
apt-get!
...
```

Yes, my master!
1) Previously on HPC User Environment 1…

Run my code on all the resources you have, however long it takes

sudo!
yum!
apt-get!
…

You are not my boss, buddy!

1. Basic concepts
2. Preparing my job
3. Submitting my job
4. Managing my jobs
1) Previously on HPC User Environment 1…

I will ask nicely. Please grant me the use of **24 cores** for **10 hours** to run my code.

Now we are talking. Let me schedule it for you.
1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers
2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues
3. Submitting my job
   1) Interactive job
   2) Batch job
4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
2) Job & Job scheduler

a) What’s a “job”? 

- A user's request to use a number of nodes/cores for a certain amount of time on a cluster.
- Calculation MUST be done via jobs (NO heavy calculation on head nodes!!)
- SUs deducted from allocations based on actual usage of each job.

- Example:
  - My allocation: 50,000 SU
  - Running a job: 24 core * 10 hours = 240 SU
  - Balance: 49,760 SU
2) Job & Job scheduler

b) What’s a “job scheduler”?

I will ask nicely. Please grant me the use of 24 cores for 10 hours to run my code.

Now we are talking. Let me schedule it for you.
b) What’s a “job scheduler”?
b) What’s a “job scheduler”?

i. Decides which job runs when and where

You! Go to Node 002!

I need 1 node w/ 24 cores for 10 hours
2) Job & Job scheduler

b) What’s a “job scheduler”?

i. Decides which job runs when and where

ii. Enforces job policies

I need to reserve 100 nodes exclusively for our group indefinitely.
2) Job & Job scheduler

b) What’s a “job scheduler”?

<table>
<thead>
<tr>
<th>Job scheduler’s responsibilities</th>
<th>Your responsibilities</th>
</tr>
</thead>
</table>
| • Decides which job runs when and where  
  • Enforces job policies | • Decide a job’s size and duration  
  • Understand the job queuing system and policies  
  • Submit/monitor/cancel jobs  
  • Diagnose job health |
2) Job & Job scheduler

b) What’s a “job scheduler”?
2) Job & Job scheduler

b) What’s a “job scheduler”?
2) Job & Job scheduler

b) What’s a “job scheduler”?

- Previously on our clusters…

<table>
<thead>
<tr>
<th></th>
<th>LSU HPC</th>
<th>LONI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job scheduler</td>
<td>Deep Bayou SuperMike III</td>
<td>QB3 QB4</td>
</tr>
<tr>
<td></td>
<td>SMIC</td>
<td>QB2</td>
</tr>
</tbody>
</table>
b) What’s a “job scheduler”?

- Previously on our clusters…

<table>
<thead>
<tr>
<th>LSU HPC</th>
<th>LONI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Bayou, SuperMike III, SMIC</td>
<td>QB3, QB4</td>
</tr>
<tr>
<td>PBS</td>
<td>QB2</td>
</tr>
</tbody>
</table>
b) What’s a “job scheduler”?

- Previously on our clusters…

<table>
<thead>
<tr>
<th></th>
<th>LSU HPC</th>
<th>LONI</th>
</tr>
</thead>
<tbody>
<tr>
<td>slurm</td>
<td>Deep Bayou</td>
<td>QB3</td>
</tr>
<tr>
<td></td>
<td>SuperMike III</td>
<td>QB4</td>
</tr>
<tr>
<td></td>
<td>SMIC</td>
<td></td>
</tr>
<tr>
<td>PBS</td>
<td></td>
<td>QB2</td>
</tr>
</tbody>
</table>
2) Job & Job scheduler

b) What’s a “job scheduler”?
Outlines

HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers

2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues

3. Submitting my job
   1) Interactive job
   2) Batch job

4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
Outlines

• HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers

2. Preparing my job
   1) Basic principles
      2) Job duration (wall time)
      3) Number of nodes & cores
      4) Job queues

3. Submitting my job
   1) Interactive job
      2) Batch job

4. Managing my jobs
   1) Useful commands
      2) Monitoring job health
1) Basic principles

- Two basic principles of requesting resources
  - Amount of resources (node / core number, RAM, duration, ...)

  Large enough …  Small enough …
1) Basic principles

- Two basic principles of requesting resources
  - Amount of resources (node / core number, RAM, duration, …)

<table>
<thead>
<tr>
<th>Large enough …</th>
<th>Small enough …</th>
</tr>
</thead>
</table>
| • To successfully complete your job | • To ensure quick turnaround  
• Not to waste resources for other users |
Outlines

- HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers

2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues

3. Submitting my job
   1) Interactive job
   2) Batch job

4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
2) Job duration (wall time)

- **What is it?**
  - Real-world *(wall)* time, from start to end
  - Required!
  - There is a **maximum** you may request (see later)
2) Job duration (wall time)

- FAQ

<table>
<thead>
<tr>
<th>Q</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>• What if my command is still running when the wall time runs out?</td>
<td>• Job terminated, any running process killed</td>
</tr>
<tr>
<td>• What if all my commands in the job finished before the wall time runs out?</td>
<td>• Job exits successfully when all commands finished</td>
</tr>
<tr>
<td>• If my job exits before requested wall time, how many SUs will I be charged?</td>
<td>• You will be charged based on your actual time used (if less than requested)</td>
</tr>
<tr>
<td>• In that case, why don’t I just request maximum wall time every time?</td>
<td>• Your queuing time may be long…</td>
</tr>
</tbody>
</table>
2) Job duration (wall time)

- Back to basic principles...

<table>
<thead>
<tr>
<th>Large enough ...</th>
<th>Small enough ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>• To successfully complete your job</td>
<td>• To ensure quick turnaround</td>
</tr>
<tr>
<td></td>
<td>• Not to waste resources for other users</td>
</tr>
</tbody>
</table>
Outlines

• HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers

2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues

3. Submitting my job
   1) Interactive job
   2) Batch job

4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
### 3) Number of nodes & cores

- Previously in HPC User Environment 1 ...

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Hostname</th>
<th>Peak Performance/TFlops</th>
<th>Compute nodes</th>
<th>Processor/node</th>
<th>Processor Speed</th>
<th>Processor Type</th>
<th>Nodes with Accelerators</th>
<th>Accelerator Type</th>
<th>OS</th>
<th>Vendor</th>
<th>Memory per node</th>
<th>Detailed Cluster Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperMIC</td>
<td>smic.hpc.lsu.edu</td>
<td>925</td>
<td>360</td>
<td>2 10-core</td>
<td>2.8 GHz</td>
<td>Intel Xeon 64bit</td>
<td>360</td>
<td>Xeon Phi 7120P</td>
<td>RHEL v6</td>
<td>Dell</td>
<td>64 GB</td>
<td>User Guide Available Software</td>
</tr>
<tr>
<td>Deep Bayou</td>
<td>db1.lsu.edu</td>
<td>257</td>
<td>13</td>
<td>2 24-core</td>
<td>2.4 GHz</td>
<td>Intel Cascade Lake Xeon 64bit</td>
<td>13</td>
<td>2 x NVIDIA Volta V1005</td>
<td>RHEL v7</td>
<td>Dell</td>
<td>192 GB</td>
<td>User Guide Available Software</td>
</tr>
<tr>
<td>SuperMike III</td>
<td>mike.hpc.lsu.edu</td>
<td>1,285</td>
<td>183</td>
<td>2 32-core</td>
<td>2.6 GHz</td>
<td>Intel Xeon Ice Lake</td>
<td>8</td>
<td>4 NVIDIA A100</td>
<td>RHEL v8</td>
<td>Dell</td>
<td>256/2048 GB</td>
<td>User Guide Available Software</td>
</tr>
</tbody>
</table>
3) Number of nodes & cores

- When submitting you job…
  - Required!
## 3) Number of nodes & cores

### FAQ

<table>
<thead>
<tr>
<th>Q</th>
<th>A</th>
</tr>
</thead>
</table>
| **My code runs slow. Can I request more nodes / cores to make it faster?** | • Not quite! Your code most likely is **NOT** using multiple nodes / cores, if:  
  - You **do not know** if it is using multiple nodes / cores  
  - You **did not tell** it to use multiple nodes / cores  
  - You are **not familiar** with names like “MPI” / “OpenMP”  
  • **Underutilization** is THE most common warning received on our clusters |
| **How many nodes / cores should I request?** | • In short: We can't answer that  
  • Each code / job is different. You must test to determine |
3) Number of nodes & cores

- Back to basic principles…

<table>
<thead>
<tr>
<th>Large enough …</th>
<th>Small enough …</th>
</tr>
</thead>
<tbody>
<tr>
<td>• To successfully complete your job</td>
<td>• To ensure quick turnaround</td>
</tr>
<tr>
<td></td>
<td>• Not to waste resources for other users</td>
</tr>
</tbody>
</table>
Outlines

• HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers

2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues

3. Submitting my job
   1) Interactive job
   2) Batch job

4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
4) Job queues

- Node001
- Node002
- Node003
- Node004

... Job scheduler

- Job1
- Job2
- Job3

...
4) Job queues

1. Basic concepts
2. Preparing my job
3. Submitting my job
4. Managing my jobs
4) Job queues

- Node001
- Node002
- Node003
- Node004...

Job scheduler

- Job1
- Job2
- Job3...

1. Basic concepts
2. Preparing my job
3. Submitting my job
4. Managing my jobs
4. Managing my jobs

2. Preparing my job

3. Submitting my job

4) Job queues

Node001
Node002
Node003
Node004

Job scheduler

single
workq
bigmem
gpu2

Job1
Job2
Job3

Job1
Job2
Job3

Job1
Job2
Job3

Job1
Job2
Job3

...
4) Job queues

a) Definition

– Lines where jobs are waiting to be executed

– Must pick one queue

– Goal: Use the resources efficiently
4) Job queues

a) Definition
### 4) Job queues

#### b) Available queues

**i. workq / checkpt**

<table>
<thead>
<tr>
<th>Purpose</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>General purposes</td>
<td></td>
</tr>
<tr>
<td>Most likely your <strong>default</strong> queue</td>
<td></td>
</tr>
<tr>
<td>Difference: <strong>non-preemptable</strong> (workq) vs. <strong>preemptable</strong> (checkpt)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Names</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>All clusters: <strong>workq / checkpt</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource availability</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td></td>
</tr>
<tr>
<td><strong>Entire</strong> node(s)</td>
<td></td>
</tr>
<tr>
<td>Up to a maximum</td>
<td></td>
</tr>
<tr>
<td>Cores</td>
<td></td>
</tr>
<tr>
<td><strong>All</strong> cores on the node(s)</td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td></td>
</tr>
<tr>
<td><strong>All</strong> memory on the node(s)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Max duration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>72 hours (3 days)</td>
<td></td>
</tr>
</tbody>
</table>
4) Job queues

b) Available queues

ii. single

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Only need a portion of one node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names</td>
<td>All clusters: single</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource availability</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Portion of one node
- 1 ~ all cores
- A portion, proportional to the number of requested cores
- 168 hours (7 days)

[ QB-4 ]

- Total: 64 cores, 256 GB memory
  → 4 GB / core
- Request: 10 cores
  → 40 GB memory
### 4) Job queues

#### b) Available queues

##### iii. bigmem

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Need <strong>large memory</strong> (larger than regular computing nodes have)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names</td>
<td>All clusters: <strong>bigmem</strong></td>
</tr>
<tr>
<td>Resource availability</td>
<td></td>
</tr>
<tr>
<td>Nodes</td>
<td><strong>Entire</strong> node(s)</td>
</tr>
<tr>
<td>Cores</td>
<td><strong>All</strong> cores on the node</td>
</tr>
<tr>
<td>Memory</td>
<td><strong>All</strong> memory on the node</td>
</tr>
<tr>
<td>Max duration</td>
<td><strong>72 hours</strong> (3 days)</td>
</tr>
</tbody>
</table>
### 4) Job queues

#### b) Available queues

#### iv. GPU

<table>
<thead>
<tr>
<th>Purpose</th>
<th>• Need GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Names</strong></td>
<td></td>
</tr>
<tr>
<td>• QB-3:</td>
<td>gpu2</td>
</tr>
<tr>
<td>• SMIC:</td>
<td></td>
</tr>
<tr>
<td>• Deep Bayou:</td>
<td></td>
</tr>
<tr>
<td>• SuperMike 3:</td>
<td></td>
</tr>
<tr>
<td>• QB-4:</td>
<td></td>
</tr>
<tr>
<td><strong>Nodes</strong></td>
<td></td>
</tr>
<tr>
<td>• Entire</td>
<td>node(s)</td>
</tr>
<tr>
<td>• Portion</td>
<td>or <strong>entire</strong> node(s)</td>
</tr>
<tr>
<td><strong>Cores</strong></td>
<td></td>
</tr>
<tr>
<td>• All</td>
<td>cores on the node(s)</td>
</tr>
<tr>
<td>• Portion</td>
<td>or <strong>all</strong> on the node(s)</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td></td>
</tr>
<tr>
<td>• All</td>
<td>memory on the node(s)</td>
</tr>
<tr>
<td>• Portion</td>
<td>or <strong>all</strong> on the node(s)</td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td></td>
</tr>
<tr>
<td>• All</td>
<td>GPUs on the node(s)</td>
</tr>
<tr>
<td>• 1 ~ all</td>
<td>GPU on the node(s)</td>
</tr>
</tbody>
</table>

**Max duration**

• 72 hours (3 days)

gpu\(X\) :
\[ X = \text{[Number of GPUs on one node]} \]

---

[ QB-4 / gpu4 ]
- **Total**: 64 cores, 4 GPUs → 16 cores / GPU
- **Request**: 3 GPUs → 48 cores
4) Job queues

c) Queues by clusters (LSU HPC)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Queue</th>
<th>Cores per node (ppn)</th>
<th>Max running jobs</th>
<th>Max nodes per user</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperMIC</td>
<td>workq</td>
<td>20</td>
<td>45 (global)</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>checkpt</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>single</td>
<td>1 ~ 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>gpu2</td>
<td>18,36</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bigmem</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeepBayou</td>
<td>gpu2</td>
<td>24,48</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>gpu4</td>
<td>12,24,36,48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SuperMike3</td>
<td>workq</td>
<td>64</td>
<td>32 (global)</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>checkpt</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>single</td>
<td>1 ~ 64</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>gpu4</td>
<td>16,32,48,64</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bigmem</td>
<td>64</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 4) Job queues

### c) Queues by clusters (LONI)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Queue</th>
<th>Cores per node (ppn)</th>
<th>Max running jobs</th>
<th>Max nodes per user</th>
</tr>
</thead>
<tbody>
<tr>
<td>QB-3</td>
<td>workq</td>
<td>48</td>
<td></td>
<td>32 (global)</td>
</tr>
<tr>
<td></td>
<td>checkpoint</td>
<td></td>
<td></td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>single</td>
<td>1 ~ 48</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>gpu2</td>
<td>48</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>bigmem</td>
<td>48</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>QB-4</td>
<td>workq</td>
<td>64</td>
<td></td>
<td>32 (global)</td>
</tr>
<tr>
<td></td>
<td>checkpoint</td>
<td></td>
<td></td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>single</td>
<td>1 ~ 64</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>gpu2</td>
<td>32, 64</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>gpu4</td>
<td>16, 32, 48, 64</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>bigmem</td>
<td>64</td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>
d) Choose your queue

<table>
<thead>
<tr>
<th>Large enough …</th>
<th>Small enough …</th>
</tr>
</thead>
</table>
| • To successfully complete your job | • To ensure quick turnaround  
|                               | • Not to waste resources for other users |
4) Job queues

d) Choose your queue

- Do I need a GPU? (Y/N)
  - Y: GPU queues
  - N: Lots of RAM?
- Do I need lots of RAM? (Y/N)
  - Y: bigmem
  - N: Multi-node?
- Do I need a multi-node setup? (Y/N)
  - Y: workq / checkpoint
  - N: single

1. Basic concepts
2. Preparing my job
3. Submitting my job
4. Managing my jobs
4) Job queues

d) Choose your queue

<table>
<thead>
<tr>
<th>My job …</th>
<th>Queue choice? (include number of nodes / cores)</th>
</tr>
</thead>
</table>
| • SMIC  
  • MPI code, needs 100 CPU cores  
    - Hint: SMIC has 20 cores / node | workq / checkpoint  
    (5 nodes, 20 cores per node) |
| • SuperMike 3  
  • Uses 3 GPUs to train a neural network  
    - Hint: SuperMike 3 has 64 cores / node, 4 GPUs / node → 16 cores / GPU | gpu4  
    (1 node, 48 cores per node) |
| • QB-3  
  • Single-core serial code  
  • Needs to store and process 30 GB data in RAM  
    - Hint: QB-3 has 192 GB RAM / node, 4 GB RAM / core | single  
    (1 node, 8 cores per node) |
4) Job queues

e) Useful commands to check queues

i. **sinfo**: Detailed node health information of all queues

```bash
(base) [jasonli3@mike2 ~]$ sinfo
PARTITION AVAIL TIMELIMIT   NODES  STATE NODELIST
single  up 7-00:00:00       2 unval mike[035,138]
single  up 7-00:00:00       1 comp mike144
single  up 7-00:00:00       58 alloc mike[008-026,031-034,036-044,046-050,141-143,148-162,167-169]
single  up 7-00:00:00       108 idle mike[001-007,027-030,045,051-137,139,145-146,163-166,170-171]
single  up 7-00:00:00       2 down mike[140,147]
checkpoint up 3-00:00:00    2 unval mike[035,138]
checkpoint up 3-00:00:00    1 comp mike144
checkpoint up 3-00:00:00    58 alloc mike[008-026,031-034,036-044,046-050,141-143,148-162,167-169]
checkpoint up 3-00:00:00    108 idle mike[001-007,027-030,045,051-137,139,145-146,163-166,170-171]
checkpoint up 3-00:00:00    2 down mike[140,147]
workq   up 3-00:00:00       2 unval mike[035,138]
workq   up 3-00:00:00       1 comp mike144
workq   up 3-00:00:00       58 alloc mike[008-026,031-034,036-044,046-050,141-143,148-162,167-169]
workq   up 3-00:00:00       108 idle mike[001-007,027-030,045,051-137,139,145-146,163-166,170-171]
workq   up 3-00:00:00       2 down mike[140,147]
bigmem  up 3-00:00:00       4 idle mike[172-175]
gpu     up 3-00:00:00       8 idle mike[176-183]
```
4) Job queues

e) Useful commands to check queues

ii. `qfree`: Free nodes in each queue

```
(base) [jasonli3@mike2 ~]$ qfree
PBS total nodes: 183, free: 120, busy: 58, down: 2, use: 31%
PBS workq nodes: 171, free: 108, busy: 54, queued: 0
PBS single nodes: 171, free: 108, busy: 0, queued: 0
PBS checkpt nodes: 171, free: 108, busy: 4, queued: 0
PBS bigmem nodes: 4, free: 4, busy: 0, queued: 0
PBS gpu nodes: 8, free: 8, busy: 0, queued: 0
```
Summary

1. Basic concepts
   a) How job works on clusters
   b) Job scheduler and how it works

2. Preparing my job
   a) Basic principles
      • “large enough” and “small enough”
   b) Information you need to tell job scheduler:
      • Duration
      • Number of nodes & cores
      • Job queue
1) Have your terminal open and ready to connect to HPC
2) Download our testing code (π calculation) to your /home directory
   – http://www.hpc.lsu.edu/training/weekly-materials/Downloads/pi_Jason.tar.gz
   – Hint: use `wget` command
Outlines

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers
2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues
3. Submitting my job
   1) Interactive job
   2) Batch job
4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
3. Submitting a job

- Two types of jobs:

1) **Interactive job**
   - Runs in terminal (just like using a local machine)
   - Can interact with the job while running

2) **Batch job**
   - Submit to server and runs by itself, until finished or error
   - Cannot interact with the job while running
3. Submitting a job

- Two types of jobs:

<table>
<thead>
<tr>
<th></th>
<th>1) Interactive job</th>
<th>2) Batch job</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td>• Can interact and monitor with job in real time</td>
<td>• Submit and leave it&lt;br&gt;• Repeatable for complicated jobs</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td>• Waiting for human intervention is the opposite of “high performance”</td>
<td>• Cannot edit or interact with job while running</td>
</tr>
<tr>
<td><strong>Ideal for</strong></td>
<td>• Debugging and testing&lt;br&gt;• Large compilation</td>
<td>• Production</td>
</tr>
</tbody>
</table>
1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers
2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues
3. Submitting my job
   1) Interactive job
   2) Batch job
4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
1) Interactive job

a) Starting an interactive job (bare minimum)

`salloc [options]`
1) Interactive job

a) Starting an interactive job (bare minimum)

```
salloc
-A <Allocation name> \
-t <HH:MM:SS> \
-p <Queue name> \
-N <# of nodes> \
-n <# of TOTAL cores>
```

4. Managing my jobs

a) Starting an interactive job (bare minimum)

```
salloc \
  -A <Allocation name> \
  -t <HH:MM:SS> \
  -p <Queue name> \
  -N <# of nodes> \
  -n <# of TOTAL cores>
```

Allocation name
1) Interactive job

a) Starting an interactive job (bare minimum)

```
salloc \
-A <Allocation name> \
-t <HH:MM:SS> \
-p <Queue name> \
-N <# of nodes> \
-n <# of TOTAL cores>
```
1) Interactive job

a) Starting an interactive job (bare minimum)

```
salloc \
- A <Allocation name> \ 
- t <HH:MM:SS> \ 
- p <Queue name> \ 
- N <# of nodes> \ 
- n <# of TOTAL cores>
```

Job queue
1) Interactive job

a) Starting an interactive job (bare minimum)

```
salloc \
  -A <Allocation name> \ 
  -t <HH:MM:SS> \ 
  -p <Queue name> \ 
  -N <# of nodes> \ 
  -n <# of TOTAL cores>
```

Number of nodes
1) Interactive job

a) Starting an interactive job (bare minimum)

```
salloc \
-A <Allocation name> \
-t <HH:MM:SS> \
-p <Queue name> \
-N <# of nodes> \
-n <# of TOTAL cores>
```

Number of TOTAL cores
1) Interactive job

a) Starting an interactive job (bare minimum)

```bash
(base) [jasonli3@qbd1 pi]$ salloc -A loni_loniadmin1 -N1 -n64 -p workq -t 1:00:00
salloc: Job estimates 64.00 SUs for -p workq --nodes=1 --ntasks=64 --cpus-per-task=1
salloc: Granted job allocation 23480
salloc: Waiting for resource configuration
salloc: Nodes qbd454 are ready for job
salloc: lua: Submitted job 23480
(base) [jasonli3@qbd454 pi]$  
```
a) Starting an interactive job (bare minimum)

```
(base) [jasonli3@qbd1 pi]$ salloc -A loni_loniadmin1 -N1 -n64 -p workq -t 1:00:00
salloc: Job estimates 64.00 SUS for -p workq --nodes=1 --ntasks=64 --cpus-per-task=1
salloc: Granted job allocation 23480
salloc: Waiting for resource configuration
salloc: Nodes qbd454 are ready for job
salloc: lua: Submitted job 23480
(base) [jasonli3@qbd454 pi]$
```
1) Interactive job

a) Starting an interactive job (bare minimum)

(base) [jasonli3@qbd1 pi]$ salloc -A loni loniadmin1 -N1 -n64 -p workq -t 1:00:00
salloc: Job estimates 64.00 SU's for -p workq --nodes=1 --ntasks=64 --cpus-per-task=1
salloc: Granted job allocation 23480
salloc: Waiting for resource configuration
salloc: Nodes qbd454 are ready for job
salloc: lua: Submitted job 23480
(base) [jasonli3@qbd454 pi]$
1) Interactive job

a) Starting an interactive job (bare minimum)

```
(base) [jasonli3@qbd1 pi]$ salloc -A loni_loniadmin1 -N1 -n64 -p workq -t 1:00:00
salloc: Job estimates 64.00 SUs for -p workq --nodes=1 --ntasks=64 --cpus-per-task=1
salloc: Granted job allocation 23480
salloc: Waiting for resource configuration
salloc: Nodes qbd454 are ready for job
salloc: lua: Submitted job 23480
(base) [jasonli3@qbd454 pi]$ 
```

Successfully started: on a computing node (3-digit number)
a) Starting an interactive job (bare minimum)

```
(base) [jasonli3@qdell pi]$ salloc -A loni_loniadmin1 -N1 -n64 -p workq -t 1:00:00
salloc: Job estimates 64.00 SUs for -p workq --nodes=1 --ntasks=64 --cpus-per-task=1
salloc: Granted job allocation 23480
salloc: Waiting for resource configuration
salloc: Nodes qbd454 are ready for job
salloc: lua: Submitted job 23480
(base) [jasonli3@qdell454 pi]$  
```

Job starts in where the job was submitted
1) Interactive job

a) Starting an interactive job (bare minimum)

Once a job starts, **type and run commands** as you normally do.
b) Starting an MPI / OpenMP hybrid job (For those who use it)

```bash
salloc \
   -A <Allocation name> \
   -t <HH:MM:SS> \
   -p <Queue name> \
   -N <# of nodes> \
   -n <# of TOTAL cores>
```

Number of TOTAL cores
1) Interactive job

b) Starting an MPI / OpenMP hybrid job (For those who use it)

```
salloc \ 
-A <Allocation name> \ 
-t <HH:MM:SS> \ 
-p <Queue name> \ 
-N # of nodes \ 
-n # of total processes 
-c # of cores per process
```

<# of TOTAL cores>

= <n> * <c>
1) Interactive job

b) Starting an MPI / OpenMP hybrid job (For those who use it)

```
salloc \
 -A <Allocation name> \
 -t <HH:MM:SS> \
 -p workq \
 -N 2 \
 -n 32 \
 -c 4
```

[ QB-4 / workq ]
- 64 / node
- 2 nodes → 128 cores
- 32 processes
- 4 cores / process
b) Starting an MPI / OpenMP hybrid job (For those who use it)

```
salloc
   -A <Allocation name>
   -t <HH:MM:SS>
   -p workq
   -N 2
   -n 32
   -c 4
```

[ QB-4 / workq ]

- 64 / node
- 2 nodes → 128 cores
- 32 processes
- 4 cores / process
b) Starting an MPI / OpenMP hybrid job (For those who use it)

```
salloc \
-A <Allocation name> \
-t <HH:MM:SS> \
-p workq \
-N 2 \n-n 32 \
-c 4
```

[ QB-4 / workq ]

- 64 / node
- 2 nodes → 128 cores
- 32 processes
- 4 cores / process
b) Starting an MPI / OpenMP hybrid job (For those who use it)

```
salloc \
  -A <Allocation name> \
  -t <HH:MM:SS> \
  -p workq \
  -N 2 \
  -n 32 \
  -c 4
```

[ QB-4 / workq ]
- 64 / node
- 2 nodes → 128 cores
- 32 processes
- 4 cores / process
b) Starting an MPI / OpenMP hybrid job (For those who use it)

```
salloc
-A <Allocation name> \
-t <HH:MM:SS> \
-p workq \
-N 2 \ 
-n 32
-c 4
```

[ QB-4 / workq ]
- 64 / node
- 2 nodes → 128 cores
- 32 processes
- 4 cores / process
1) Interactive job

c) Starting a GPU job (For those who use it)

```
salloc \  
  -A <Allocation name> \  
  -t <HH:MM:SS> \  
  -p <Queue name> \  
  -N <# of nodes> \  
  -n <# of TOTAL cores>
```
c) **Starting a GPU job** (For those who use it)

```
salloc \
  -A <Allocation name> \
  -t <HH:MM:SS> \
  -p <Queue name> \
  -N <# of nodes> \
  -n <# of TOTAL cores> \
  --gres=gpu:<# of GPUs per node>
```

Number of GPUs per node
c) Starting a GPU job (For those who use it)

```
salloc \
-A <Allocation name> \
-t <HH:MM:SS> \
-p gpu4 \n-N 1 \n-n 16 \n--gres=gpu:1
```

[ QB-4 / gpu4 ]

- 64 cores, 4 GPUs → 16 cores / GPU
- 1 nodes (only need ¼)
- 16 cores (¼ of total)
- 1 GPU (¼ of total)
1) Interactive job

c) Starting a GPU job (For those who use it)

```
salloc \
   -A <Allocation name> \
   -t <HH:MM:SS> \
   -p gpu4 \
   -N 1 \n   -n 16 \n   --gres=gpu:1
```

- 64 cores, 4 GPUs → 16 cores / GPU
- 1 nodes (only need ¼)
- 16 cores (¼ of total)
- 1 GPU (¼ of total)
1) Interactive job

2. Preparing my job

3. Submitting my job

4. Managing my jobs

1. Basic concepts

2. Preparing my job

3. Submitting my job

4. Managing my jobs

---

c) Starting a GPU job (For those who use it)

```
salloc \
  -A <Allocation name> \
  -t <HH:MM:SS> \
  -p gpu4 \
  -N 1 \
  -n 16 \
  --gres=gpu:1
```

[ QB-4 / gpu4 ]

- 64 cores, 4 GPUs → 16 cores / GPU
- 1 nodes (only need ¼)
- 16 cores (¼ of total)
- 1 GPU (¼ of total)
c) **Starting a GPU job** (For those who use it)

```bash
salloc
  -A <Allocation name> \
  -t <HH:MM:SS> \
  -p gpu4 \
  -N 1 \
  -n 16
  --gres=gpu:1
```

[ QB-4 / gpu4 ]
- 64 cores, 4 GPUs → 16 cores / GPU
- 1 nodes (only need ¼)
- 16 cores (¼ of total)
- 1 GPU (¼ of total)
1) Interactive job

c) Starting a GPU job (For those who use it)

```
salloc \
   -A <Allocation name> \
   -t <HH:MM:SS> \
   -p gpu4 \
   -N 1 \
   -n 16 \
   --gres=gpu:1
```

[ QB-4 / gpu4 ]

- 64 cores, 4 GPUs → 16 cores / GPU
- 1 nodes (only need ¼)
- 16 cores (¼ of total)
- 1 GPU (¼ of total)
1) Interactive job

1) Basic concepts

2) Preparing my job

3) Submitting my job

4) Managing my jobs

d) Other useful flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--x11</td>
<td>Enable x11 forwarding for GUI (exclusive to interactive job)</td>
</tr>
<tr>
<td>-J</td>
<td>Job name</td>
</tr>
<tr>
<td>--dependency=afterok:[jobid]</td>
<td>Dependent job (starts after another job finishes)</td>
</tr>
<tr>
<td>--mail-type:FAIL</td>
<td>Send email when …</td>
</tr>
<tr>
<td></td>
<td>Job aborts / fails</td>
</tr>
<tr>
<td>--mail-type:BEGIN</td>
<td>Job begins</td>
</tr>
<tr>
<td></td>
<td>Job ends</td>
</tr>
<tr>
<td>--mail-type:END</td>
<td>Email address (will check against registered institutional email)</td>
</tr>
</tbody>
</table>

1) Interactive job

- Running an interactive job:

<table>
<thead>
<tr>
<th>Serial (Single-thread)</th>
<th>Parallel (MPI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Run commands as you normally do</td>
<td>• Method 1 (Recommended)</td>
</tr>
<tr>
<td>$ &lt;Executable&gt; [options]</td>
<td>$ srun -N[...] -n[...] -c[...] &lt;mpi_executable&gt; [options]</td>
</tr>
<tr>
<td></td>
<td>• Method 2</td>
</tr>
<tr>
<td></td>
<td>$ module load &lt;desired MPI&gt;</td>
</tr>
<tr>
<td></td>
<td>$ export OMP_NUM_THREADS=[..]</td>
</tr>
<tr>
<td></td>
<td>$ mpirun -np [..] &lt;mpi_executable&gt; [options]</td>
</tr>
</tbody>
</table>
### f) Useful environmental variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SLURM_JOBID</td>
<td>Job ID</td>
</tr>
<tr>
<td>$SLURM_SUBMIT_DIR</td>
<td>Job submit directory</td>
</tr>
<tr>
<td>$SLURM_JOB_NODELIST</td>
<td>A temp file, contains a list of allocated nodes’ names (useful for MPI)</td>
</tr>
<tr>
<td>$SLURM_NNODES</td>
<td>Number of allocated nodes</td>
</tr>
<tr>
<td>$SLURM_NTASKS</td>
<td>Number of processes (tasks)</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

[1] [http://www.hpc.lsu.edu/docs/slurm.php](http://www.hpc.lsu.edu/docs/slurm.php)
Outlines

• HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers

2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues

3. Submitting my job
   1) Interactive job
   2) Batch job

4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
2) Batch job

• What do you need?

i. A batch file (containing job parameters and bash scripts)

ii. Submit this batch file with submission command
2) Batch job

a) Batch file

```bash
#!/bin/bash
#SBATCH -A <Allocation name>
#SBATCH -p workq
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 1000000000
```

[Header]
Job parameters

[Body]
Commands to run after job starts
2) Batch job

a) Batch file

```bash
#!/bin/bash
#SBATCH -A <Allocation name>
#SBATCH -p workq
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 100000000
```
2) Batch job

a) Batch file

```bash
#!/bin/bash
#SBATCH -A <Allocation name>
#SBATCH -p workq
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 100000000
```

Shell type ("shebang")
2) Batch job

a) Batch file

```bash
#!/bin/bash
#SBATCH -A <Allocation name>
#SBATCH -p workq
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 100000000
```
2) Batch job

a) Batch file

```bash
#!/bin/bash

#SBATCH A <Allocation name>
#SBATCH -p workq
#SBATCH -l 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 100000000
```

Queue name
2) Batch job

a) Batch file

```bash
#!/bin/bash
#SBATCH -A <Allocation name>
#SBATCH -p workq
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 100000000
```

Wall time
2) Batch job

a) Batch file

```
#!/bin/bash
#SBATCH -A <Allocation name>
#SBATCH -p workq
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 100000000
```
### 2) Batch job

#### a) Batch file

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-o</td>
<td>Standard output file (exclusive to batch job)</td>
</tr>
<tr>
<td>-e</td>
<td>Standard error file (exclusive to batch job)</td>
</tr>
<tr>
<td>-J</td>
<td>Job name</td>
</tr>
<tr>
<td>--dependency=afterok:[jobid]</td>
<td>Dependent job (starts after another job finishes)</td>
</tr>
<tr>
<td>--mail-type</td>
<td>Job aborts / fails</td>
</tr>
<tr>
<td>BEGIN</td>
<td>Job begins</td>
</tr>
<tr>
<td>END</td>
<td>Job ends</td>
</tr>
<tr>
<td>--mail-user</td>
<td>Email address (will check against registered institutional email)</td>
</tr>
</tbody>
</table>

[1] [http://www.hpc.lsu.edu/docs/slurm.php](http://www.hpc.lsu.edu/docs/slurm.php)
2) Batch job

a) Batch file

```
#!/bin/bash
#SBATCH -A <Allocation name>
#SBATCH -p workq
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 100000000
```

[Body]
Commands to run after job starts
2) Batch job

a) Batch file

```bash
#!/bin/bash
#SBATCH -A <Allocation name>
#SBATCH -p workq
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 100000000
```

[Recommended]
Explicitly load modules here if needed!
2) Batch job

a) Batch file

Whatever commands you need to run your job...

```bash
#!/bin/bash
#SBATCH -A <Allocation name>
#SBATCH -p workq
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 1000000000
```
2) Batch job

a) Batch file

```bash
#!/bin/bash
#SBATCH -A <Allocation name>
#SBATCH -p workq
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 64

module load python

cd $SLURM_SUBMIT_DIR
./pi_serial.out 100000000
```

Empty line to avoid error
2) Batch job

b) Submit

```
sbatch <batch file name>
```
Outlines

- HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers

2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues

3. Submitting my job
   1) Interactive job
   2) Batch job

4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
• Running jobs on HPC ≠ “Submit and done”
  – Monitoring and managing jobs are part of the work
HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers
2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues
3. Submitting my job
   1) Interactive job
   2) Batch job
4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
### 1) Useful commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>squeue</code></td>
<td>List all jobs</td>
</tr>
<tr>
<td><code>-j &lt;Job ID&gt;</code></td>
<td>List the job of specific ID</td>
</tr>
<tr>
<td><code>-u &lt;Username&gt;</code></td>
<td>List all jobs belong to a specific user</td>
</tr>
<tr>
<td><code>-p &lt;Queue name&gt;</code></td>
<td>List all jobs in a particular queue</td>
</tr>
<tr>
<td><code>--start</code></td>
<td>Estimated start time of queuing jobs</td>
</tr>
<tr>
<td><code>scontrol show job &lt;Job ID&gt;</code></td>
<td>Show job details</td>
</tr>
<tr>
<td><code>scancel &lt;Job ID&gt;</code></td>
<td>Cancel &lt;Job ID</td>
</tr>
</tbody>
</table>

**Alter jobs after submission? → NOT allowed!**

[1] [http://www.hpc.lsu.edu/docs/slurm.php](http://www.hpc.lsu.edu/docs/slurm.php)
Outlines

• HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers

2. Preparing my job
   1) Basic principles
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues

3. Submitting my job
   1) Interactive job
   2) Batch job

4. Managing my jobs
   1) Useful commands
   2) Monitoring job health
2) Monitoring job health

A job requesting \( n \) cores \( \neq \) A job utilizing \( n \) cores

• Goal
  – Use the allocated resources (CPU cores, RAM, time, ...) \textit{as fully and efficiently as possible}
  – No serious underutilizing
  – No serious overutilizing

• Things to check
  – CPU / GPU load
  – Memory usage
a) Method 1: `qshow <Job ID>`

- Displays diagnostic information of a *running job*
- Can be run on *head node*
### 2) Monitoring job health

#### a) Method 1: `qshow <Job ID>`

<table>
<thead>
<tr>
<th>What to look at</th>
<th>Normal behavior</th>
<th>You should be concerned if</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBS job, nodes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hostname, Days, Load, CPU, U#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User:Process, Virtual Memory, Memory, Hours</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- `<Job ID>`
- PBS job: 38581, nodes: 1
- Hostname: Days: Load: CPU: U#
- User:Process:VirtualMemory:Memory:Hours
- Job ID: [Output of `qshow` command]
2) Monitoring job health

a) **Method 1: qshow <Job ID>**

<table>
<thead>
<tr>
<th>What to look at ...</th>
<th>Normal behavior ...</th>
<th>You should be concerned if ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg_load</td>
<td>Close to allocated number of cores on the node</td>
<td>Consistently too low or too high</td>
</tr>
</tbody>
</table>
2) Monitoring job health

a) **Method 1: qshow <Job ID>**

<table>
<thead>
<tr>
<th>What to look at ...</th>
<th>Normal behavior ...</th>
<th>You should be concerned if ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg_load</td>
<td>Close to allocated number of cores on the node</td>
<td>Consistently too low or too high</td>
</tr>
<tr>
<td>ave_mem</td>
<td>Does not exceed total allocated memory</td>
<td>Exceeds total allocated memory</td>
</tr>
</tbody>
</table>
2) Monitoring job health

b) Method 2: top

• Displays dynamic real-time view of a computing node

• Must run on computing nodes!

* ssh to computing nodes while job running (cannot ssh if you do not have jobs on it)
2) Monitoring job health

b) Method 2: `top`

```
top - 02:23:58 up 278 days, 19:17, 2 users, load average: 63.63, 39.81, 17.49
Tasks: 981 total, 65 running, 916 sleeping, 0 stopped, 0 zombie
%CPU(s): 90.2 us, 9.2 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.5 hi, 0.0 si, 0.0 st
MiB Mem: 257004.8 total, 211261.0 free, 41926.9 used, 3816.9 buff/cache
MiB Swap: 16641.0 total, 16580.7 free, 60.2 used. 212737.8 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2701318 jasonli3 20 0 595668 582356 2568 R 100.0 0.2 4:08.94 TDSE_np3_e0
2701342 jasonli3 20 0 595668 581944 2616 R 100.0 0.2 4:08.99 TDSE_np3_e0
2701249 jasonli3 20 0 595668 581792 2464 R 99.7 0.2 4:08.97 TDSE_np3_e0
2701252 jasonli3 20 0 595668 514684 2520 R 99.7 0.2 4:09.00 TDSE_np3_e0
2701261 jasonli3 20 0 595668 393828 2616 R 99.7 0.1 4:08.97 TDSE_np3_e0
2701264 jasonli3 20 0 595668 581856 2532 R 99.7 0.2 4:08.92 TDSE_np3_e0
2701267 jasonli3 20 0 595668 582480 2432 R 99.7 0.2 4:08.95 TDSE_np3_e0
2701270 jasonli3 20 0 595668 581776 2448 R 99.7 0.2 4:08.81 TDSE_np3_e0
2701273 jasonli3 20 0 595668 582160 2568 R 99.7 0.2 4:08.98 TDSE_np3_e0
2701276 jasonli3 20 0 595668 582964 2544 R 99.7 0.1 4:08.98 TDSE_np3_e0
```

What to look at … | Normal behavior … | You should be concerned if …
2) Monitoring job health

b) Method 2: top

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>2701318</td>
<td>jasonli3</td>
<td>20</td>
<td>0</td>
<td>595668</td>
<td>582356</td>
<td>2568</td>
<td>R</td>
<td>100.0</td>
<td>0.2</td>
<td>4:08.94</td>
<td>TDSE_np3_e0</td>
</tr>
<tr>
<td>2701342</td>
<td>jasonli3</td>
<td>20</td>
<td>0</td>
<td>595668</td>
<td>581944</td>
<td>2616</td>
<td>R</td>
<td>100.0</td>
<td>0.2</td>
<td>4:08.90</td>
<td>TDSE_np3_e0</td>
</tr>
<tr>
<td>2701249</td>
<td>jasonli3</td>
<td>20</td>
<td>0</td>
<td>595668</td>
<td>581792</td>
<td>2464</td>
<td>R</td>
<td>99.7</td>
<td>0.2</td>
<td>4:08.97</td>
<td>TDSE_np3_e0</td>
</tr>
<tr>
<td>2701252</td>
<td>jasonli3</td>
<td>20</td>
<td>0</td>
<td>595668</td>
<td>514684</td>
<td>2520</td>
<td>R</td>
<td>99.7</td>
<td>0.2</td>
<td>4:09.00</td>
<td>TDSE_np3_e0</td>
</tr>
<tr>
<td>2701261</td>
<td>jasonli3</td>
<td>20</td>
<td>0</td>
<td>595668</td>
<td>393828</td>
<td>2616</td>
<td>R</td>
<td>99.7</td>
<td>0.1</td>
<td>4:08.97</td>
<td>TDSE_np3_e0</td>
</tr>
<tr>
<td>2701264</td>
<td>jasonli3</td>
<td>20</td>
<td>0</td>
<td>595668</td>
<td>581856</td>
<td>2532</td>
<td>R</td>
<td>99.7</td>
<td>0.2</td>
<td>4:08.92</td>
<td>TDSE_np3_e0</td>
</tr>
<tr>
<td>2701270</td>
<td>jasonli3</td>
<td>20</td>
<td>0</td>
<td>595668</td>
<td>582480</td>
<td>2432</td>
<td>R</td>
<td>99.7</td>
<td>0.2</td>
<td>4:08.95</td>
<td>TDSE_np3_e0</td>
</tr>
<tr>
<td>2701273</td>
<td>jasonli3</td>
<td>20</td>
<td>0</td>
<td>595668</td>
<td>581776</td>
<td>2448</td>
<td>R</td>
<td>99.7</td>
<td>0.2</td>
<td>4:08.81</td>
<td>TDSE_np3_e0</td>
</tr>
<tr>
<td>2701276</td>
<td>jasonli3</td>
<td>20</td>
<td>0</td>
<td>595668</td>
<td>582160</td>
<td>2568</td>
<td>R</td>
<td>99.7</td>
<td>0.2</td>
<td>4:08.98</td>
<td>TDSE_np3_e0</td>
</tr>
</tbody>
</table>

What to look at … Normal behavior … You should be concerned if …

| Load average | Close to allocated number of cores on the node | Consistently too low or too high |

2) Monitoring job health

b) Method 2: `top`

```
%top - 02:23:58 up 278 days, 19:17, 2 users, load average: 63.63, 39.81, 17.49
Tasks: 981 total, 65 running, 916 sleeping, 0 stopped, 0 zombie
%Cpu(s): 90.2 us, 9.2 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.5 hi, 0.0 si, 0.0 st
MiB Mem : 257004.8 total, 211261.0 free, 41926.9 used, 3816.9 buff/cache
MiB Swap: 16641.0 total, 16500.7 free, 140.3 used, 212737.8 avail Mem

   PID USER      PR  NI  VIRT  RES  SHR  %CPU %MEM    TIME+  COMMAND
      2701318 jasonli3  20  0  595668  502356  2568  100.0   0.2   4:08.94  TDSE_np3_e0
      2701342 jasonli3  20  0  595668  581944  2616  100.0   0.2   4:08.90  TDSE_np3_e0
      2701249 jasonli3  20  0  595668  581792  2464  99.7   0.2   4:08.97  TDSE_np3_e0
      2701252 jasonli3  20  0  595668  514684  2520  99.7   0.2   4:09.00  TDSE_np3_e0
      2701261 jasonli3  20  0  595668  393828  2616  99.7   0.1   4:08.97  TDSE_np3_e0
      2701264 jasonli3  20  0  595668  518586  2532  99.7   0.2   4:08.92  TDSE_np3_e0
      2701270 jasonli3  20  0  595668  582480  2432  99.7   0.2   4:08.95  TDSE_np3_e0
      2701273 jasonli3  20  0  595668  581776  2448  99.7   0.2   4:08.81  TDSE_np3_e0
      2701276 jasonli3  20  0  595668  582160  2568  99.7   0.2   4:08.98  TDSE_np3_e0
      2701279 jasonli3  20  0  595668  527056  2544  99.7   0.1   4:08.98  TDSE_np3_e0
```

<table>
<thead>
<tr>
<th>What to look at</th>
<th>Normal behavior</th>
<th>You should be concerned if</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load average</td>
<td>Close to allocated number of cores on the node</td>
<td>Consistently too low or too high</td>
</tr>
<tr>
<td>Memory usage (not virtual memory)</td>
<td>Does not exceed total allocated memory</td>
<td>Exceeds total allocated memory</td>
</tr>
</tbody>
</table>
2) Monitoring job health

c) Method 3: nvidia-smi (for GPU only)

What to look at ... | Normal behavior ... | You should be concerned if ...

![nvidia-smi output image]
2) Monitoring job health

- Method 3: **nvidia-smi** (for GPU only)

<table>
<thead>
<tr>
<th>What to look at ...</th>
<th>Normal behavior ...</th>
<th>You should be concerned if ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU usage</td>
<td>Close to 100%</td>
<td>Consistently too low</td>
</tr>
</tbody>
</table>
c) Method 3: `nvidia-smi` (for GPU only)

![nvidia-smi output]

<table>
<thead>
<tr>
<th>What to look at</th>
<th>Normal behavior ...</th>
<th>You should be concerned if ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU usage</td>
<td>Close to 100%</td>
<td>Consistently too low</td>
</tr>
<tr>
<td>Memory usage (not virtual memory)</td>
<td>Not used up</td>
<td>Used up</td>
</tr>
</tbody>
</table>
## 2) Monitoring job health

### d) Common issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>What would happen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exceeded memory allocation (e.g., using more memory than allocated w/ single queue)</td>
<td>Terminated. Receive email notice.</td>
</tr>
<tr>
<td>Exceeded ppn/core allocation (e.g., using more cores than allocated w/ single queue)</td>
<td>Terminated. Receive email notice.</td>
</tr>
<tr>
<td>Seriously underutilize node CPU cores / unused nodes (e.g., Requested multiple nodes but only runs on one node)</td>
<td>Receive email warning. (* Killed if <strong>completely idle</strong> for a long time)</td>
</tr>
<tr>
<td>Submitting to bigmem but only using little memory</td>
<td>Receive email warning.</td>
</tr>
<tr>
<td>Running intensive calculation on head nodes</td>
<td>Terminated. Receive email notice.</td>
</tr>
<tr>
<td>Submitting too many (i.e., hundreds of) single-thread jobs</td>
<td>Poor parallelization and bad for server. We may reach out to you to help. (Better yet, reach out to us first)</td>
</tr>
</tbody>
</table>
Summary

- A typical workflow --

1. Basic concepts
2. Preparing my job
3. Submitting my job
4. Managing my jobs

- Test your code using *interactive jobs*
- Monitor your jobs to determine resource usage
- Submit production jobs using *batch jobs*
- Keep monitoring your jobs and edit your production jobs as needed
Outlines

HPC User Environment 2

1. Basic concepts
   1) Previously on HPC User Environment 1…
   2) Job & Job schedulers → All calculation must be submitted as jobs

2. Preparing my job
   1) Basic principles → Large enough & small enough
   2) Job duration (wall time)
   3) Number of nodes & cores
   4) Job queues

3. Submitting my job
   1) Interactive job → Good for testing and debugging
   2) Batch job → Good for production

4. Managing my jobs
   1) Useful commands
   2) Monitoring job health → How to monitor jobs health, and how to create health jobs
Next week

- Basic Shell Scripting
Contact us

- Contact user services
  - Email Help Ticket: sys-help@loni.org
  - Telephone Help Desk: +1 (225) 578-0900