Job Monitoring on SciNet

Ramses van Zon

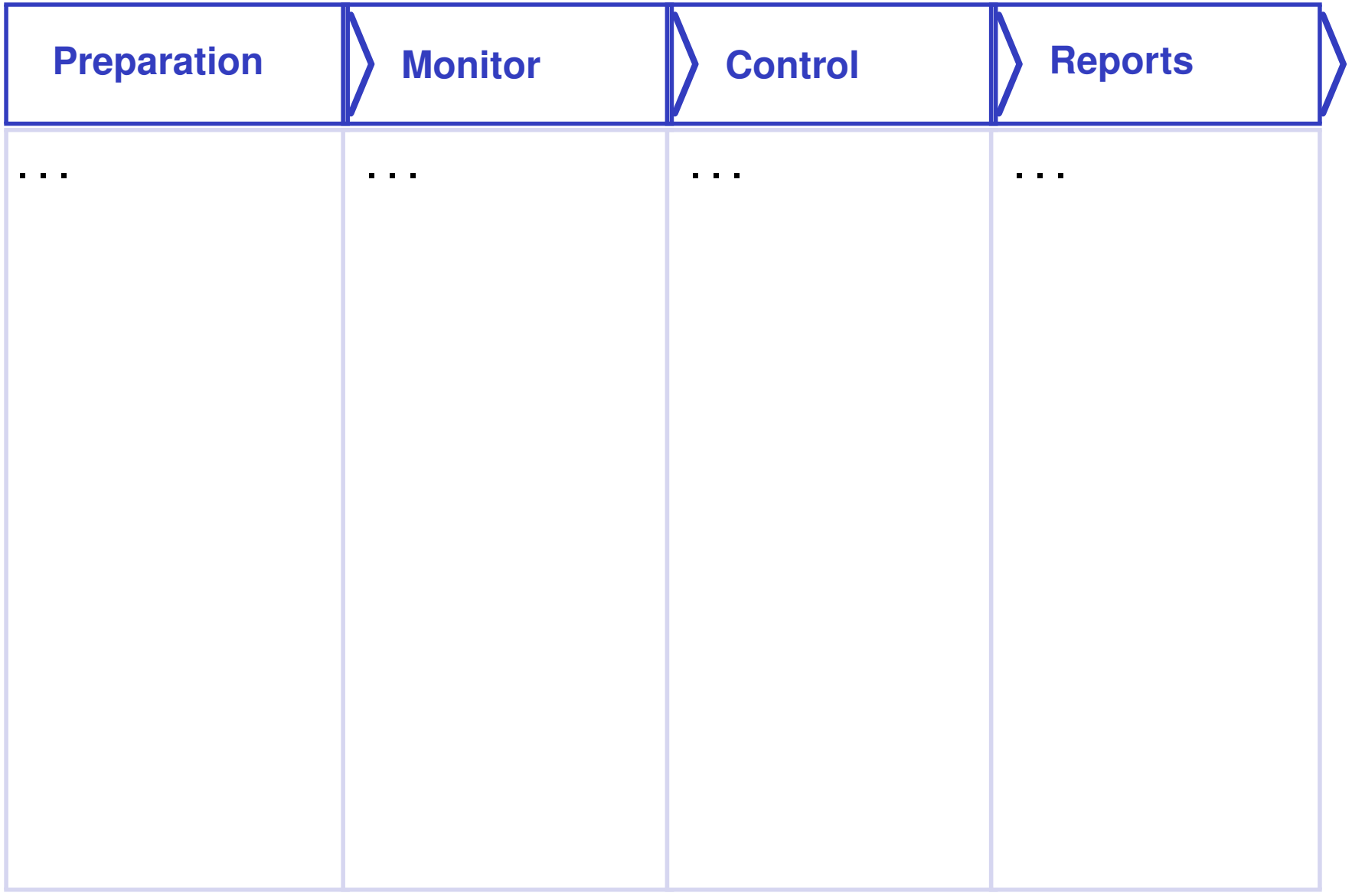
SciNet, Toronto

compute • calcul CANADA

Job Monitoring on SciNet

When you have a job running in the batch queue,
how do you know that it is going well?

Job cycle



Job cycle

Preparation	Monitor	Control	Reports
• Compile • Test on devel node • Determine resources • Write job script	. . .	. . .	. . .

Job cycle

Preparation	Monitor	Control	Reports
• Compile • Test on devel node • Determine resources • Write job script	• Job queued? • When will it run? • What else is queued? • What has it produced? • Efficiency?	...	...

Job cycle

Preparation	Monitor	Control	Reports
• Compile • Test on devel node • Determine resources • Write job script	• Job queued? • When will it run? • What else is queued? • What has it produced? • Efficiency?	• Cancel job • Ssh to nodes • Interactive jobs • Debug queue	. . .

Job cycle

Preparation	Monitor	Control	Reports
• Compile • Test on devel node • Determine resources • Write job script	• Job queued? • When will it run? • What else is queued? • What has it produced? • Efficiency?	• Cancel job • Ssh to nodes • Interactive jobs • Debug queue	• .o/.e files *jobid*.{o,e} • short-term statistics: showstats -u *user* • year-to-date usage on: https://portal.scinet.utoronto.ca

Job Monitoring on SciNet

When you have a job running in the batch queue,
how do you know that it is going well?

1. Check the queue

1. Check the queue

A bit about that queue

- Computing by submitting batch jobs to the scheduler.
- When you submit a job, it gets placed in a queue.
- Job priority is based on allocation and fairshare.
- When sufficient nodes are free to execute a job, it starts the job on the appropriate compute nodes.
- Jobs remain 'idle' until resources become available.
- Jobs can be temporarily 'blocked' if you submit too much.

Batch computing

Components

Torque: Resource manager providing control over batch jobs and distributed compute nodes.

Moab: A policy-based job scheduler and event engine that enables utility-based computing for clusters.

Fairshare: Mechanism using past utilization for prioritization.

Monitoring not-yet-running jobs

qstat and checkjob

- Show torque status right away on GPC: qstat
- Show moab status (better): checkjob *jobid*
- See more details of the job: checkjob -v *jobid*

Monitoring not-yet-running jobs

qstat and checkjob

- Show torque status right away on GPC: qstat
- Show moab status (better): checkjob *jobid*
- See more details of the job: checkjob -v *jobid*

showq

- See all the jobs in the queue: showq
- See your jobs in the queue: showq -u *user*

Monitoring not-yet-running jobs

qstat and checkjob

- Show torque status right away on GPC: qstat
- Show moab status (better): checkjob *jobid*
- See more details of the job: checkjob -v *jobid*

showq

- See all the jobs in the queue: showq
- See your jobs in the queue: showq -u *user*

showstart and showbf

- Estimate when a job may start: showbf
- Estimate when a queued job may start: showstart *jobid*

Monitoring running jobs

checkjob

- checkjob *jobid*

Monitoring running jobs

checkjob

- checkjob *jobid*

showq

- showq -r -u *user*

  Tells you the nodes it's running on too.

Job Monitoring on SciNet

When you have a job running in the batch queue, how do you know that it is going well?

1. Check the queue
2. Check the processes on the node

2. Check processes

ssh

- On the GPC and P7, all the nodes that your job is running on are yours while it is running.
- So you can log into those nodes from the devel nodes.
- ssh *node* (node name from checkjob or showq -r)
- Not available for TCS.

2. Check processes

top

- ssh *node*
- top: shows process state, memory and cpu usage

2. Check processes

top

- ssh *node*
- top: shows process state, memory and cpu usage

vmstat

- ssh *node*
- vmstat: shows number of running processes, cpu usage, memory usages

3. Check output

output/error files

- ssh *node*
- On the head node, output and error produced so far are in
  - /var/spool/torque/spool/`jobid`.OU
  - /var/spool/torque/spool/`jobid`.ER

own output files

Of course, you can check files that your job creates itself.

Since the file system is shared, you can do this from the devel nodes too.

Top example

```
gpc-f103n084-$ ssh gpc-f109n001
gpc-f109n001-$ top
```

```
top - 21:56:45 up  5:56,  1 user,  load average: 5.55, 1.73, 0.88
Tasks: 234 total,   1 running, 233 sleeping,   0 stopped,   0 zombie
Cpu(s): 11.4%us, 36.2%sy,  0.0%ni, 52.2%id,  0.0%wa,  0.0%hi,  0.2%si,  0.0%st
Mem:  16410900k total,  1542768k used, 14868132k free,        0k buffers
Swap:        0k total,        0k used,        0k free,   294628k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  P COMMAND
22479 ljdursi   18   0  108m 4816 3212 S 98.5  0.0   1:04.81  6 gameoflife
22480 ljdursi   18   0  108m 4856 3260 S 98.5  0.0   1:04.85 13 gameoflife
22482 ljdursi   18   0  108m 4868 3276 S 98.5  0.0   1:04.83  2 gameoflife
22483 ljdursi   18   0  108m 4868 3276 S 98.5  0.0   1:04.82  8 gameoflife
22484 ljdursi   18   0  108m 4832 3232 S 98.5  0.0   1:04.80  9 gameoflife
22481 ljdursi   18   0  108m 4856 3256 S 98.2  0.0   1:04.81  3 gameoflife
22485 ljdursi   18   0  108m 4808 3208 S 98.2  0.0   1:04.80  4 gameoflife
22478 ljdursi   18   0  117m 5724 3268 D 69.6  0.0   0:46.07 15 gameoflife
 8042 root       0 -20 2235m 1.1g  16m S  2.3  6.8   0:30.59  8 mmfsd
10735 root      15   0  3792  452  372 S  1.3  0.0   0:16.80  0 cat
```

Top example

```
gpc-f103n084-$ ssh gpc-f109n001
gpc-f109n001-$ top
```

```
top - 21:56:45 up  5:56,  1 user,  load average: 5.55, 1.73, 0.88
Tasks: 234 total,   1 running, 233 sleeping,   0 stopped,   0 zombie
Cpu(s): 11.4%us, 36.2%sy,  0.0%ni, 52.2%id,  0.0%wa,  0.0%hi,  0.2%si,  0.0%st
Mem:  16410900k total,  1542768k used, 14868132k free,        0k buffers
Swap:        0k total,        0k used,        0k free,   294628k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+   P COMMAND
22479 ljdursi   18   0  108m 4816 3212 S 98.5  0.0   1:04.81  6 gameoflife
22480 ljdursi   18   0  108m 4856 3260 S 98.5  0.0   1:04.85 13 gameoflife
22482 ljdursi   18   0  108m 4868 3276 S 98.5  0.0   1:04.83  2 gameoflife
22483 ljdursi   18   0  108m 4868 3276 S 98.5  0.0   1:04.82  8 gameoflife
22484 ljdursi   18   0  108m 4832 3232 S 98.5  0.0   1:04.80  9 gameoflife
22481 ljdursi   18   0  108m 4856 3256 S 98.2  0.0   1:04.81  3 gameoflife
22485 ljdursi   18   0  108m 4808 3208 S 98.2  0.0   1:04.80  4 gameoflife
22478 ljdursi   18   0  117m 5724 3268 D 69.6  0.0   0:46.07 15 gameoflife
 8042 root       0 -20 2235m 1.1g  16m S  2.3  6.8   0:30.59  8 mmfsd
10735 root      15   0  3792  452  372 S  1.3  0.0   0:16.80  0 cat
```

Job Monitoring on SciNet

When you have a job running in the batch queue, how do you know that it is going well?

1. Check the queue
2. Check the processes on the node
3. Check output files

Control

canceljob

- If you spot a mistake: canceljob *jobid*
- Other than that, there is unfortunately little control.

Job Monitoring on SciNet

When you have a job running in the batch queue, how do you know that it is going well?

1. Check the queue
2. Check the processes on the node
3. Check output files as they are produced
4. Check reports afterwards

output/error files

- *.e / *.o
  In submission directory by default, unless set in *script*.

```
---------------------------------------
Begin PBS Prologue Tue Sep 14 17:14:48 EDT 2010 1284498888
Job ID:     3053514.gpc-sched
Username:   ljdursi
Group:      scinet
Nodes:      gpc-f134n009 gpc-f134n010 gpc-f134n011 gpc-f134n012
gpc-f134n043 gpc-f134n044 gpc-f134n045 gpc-f134n046 gpc-f134n047 gpc-f134n048
[...]
End PBS Prologue Tue Sep 14 17:14:50 EDT 2010 1284498890
---------------------------------------
[ Your job's output here... ]
---------------------------------------
Begin PBS Epilogue Tue Sep 14 17:36:07 EDT 2010 1284500167
Job ID:     3053514.gpc-sched
Username:   ljdursi
Group:      scinet
Job Name:   fft_8192_procs_2048
Session:    18758
Limits:     neednodes=256:ib:ppn=8,nodes=256:ib:ppn=8,walltime=01:00:00
Resources:  cput=713:42:30,mem=3463854672kb,vmem=3759656372kb,walltime=00:21:07
Queue:      batch_ib
Account:
Nodes:  gpc-f134n009 gpc-f134n010 gpc-f134n011 gpc-f134n012 gpc-f134n043
[...]
Killing leftovers...
gpc-f141n054:   killing gpc-f141n054 12412

End PBS Epilogue Tue Sep 14 17:36:09 EDT 2010 1284500169
---------------------------------------
```

4. Reports

Statistics

- Short term: showstats -u *USER*
- Year-to-date: SciNet Portal

  Usage stats for past year, showing a breakdown of TCS, GPC, ... Should get updated every 24 hours.

Job Monitoring on SciNet

When you have a job running in the batch queue, how do you know that it is going well?

1. Check the queue
2. Check the processes on the node
3. Check output files
4. Check reports afterwards
5. Preemptive checking

5. Preemptive checking

qsub for interactive and debug jobs (GPC)n

- `-I`:
  - Interactive
  - After qsub, waits for jobs to start.
  - Usually combined with:
- -q debug:
  - Debug queue has 10 nodes reserved for short jobs.
  - You can get 1 node for 2 hours, but also
  - 8 nodes, for half an hour.

5. Preemptive checking

Modify the job script to track resources

- You could add a vmstat command to your job script:

```
vmstat -a 5 > vmstat.out &
```

- You could use Allinea MAP to track resources/mpi:

```
module load ddt
map -profile -n N APP ARGS
```

  instead of

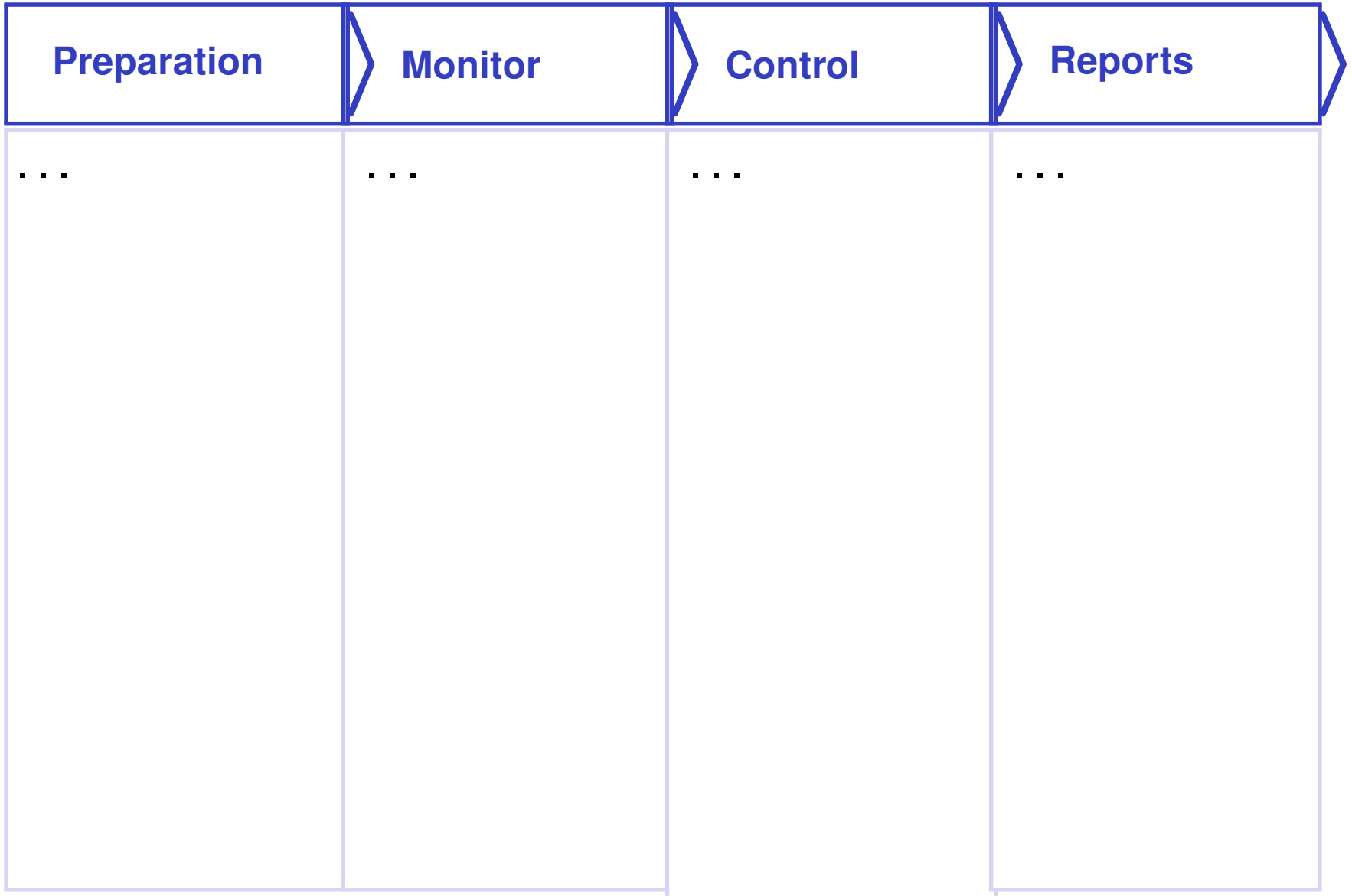
```
mpirun -np N APP ARGS
```

  Will collect info into a .map file, to view later with

```
map MAPFILENAME
```

- wiki:Performance_and_Profiling_Course,_April_2013

Job cycle



Job cycle

Preparation	Monitor	Control	Reports
• Compile • Test on devel node • Determine resources • Write job script llsubmit *script* qsub *script* returns *'jobid'*	...	...	...

Job cycle

Preparation	Monitor	Control	Reports
• Compile • Test on devel node • Determine resources • Write job script llsubmit *script* qsub *script* returns *'jobid'*	• Job queued? • When will it run? • What else is queued? • Efficiency? qstat -f *jobid* checkjob *jobid* showstart *jobid* showbf showq showq -r -u *user*	...	...

Job cycle

Preparation	Monitor	Control	Reports
• Compile • Test on devel node • Determine resources • Write job script llsubmit *script* qsub *script* returns *'jobid'*	• Job queued? • When will it run? • What else is queued? • Efficiency? qstat -f *jobid* checkjob *jobid* showstart *jobid* showbf showq showq -r -u *user*	• Cancel job • Ssh to nodes • Interactive jobs • Debug queue canceljob *jobid* ssh *node* top vmstat qsub -I qsub -q debug	. . .

Job cycle

Preparation	Monitor	Control	Reports
• Compile • Test on devel node • Determine resources • Write job script llsubmit *script* qsub *script* returns *'jobid'*	• Job queued? • When will it run? • What else is queued? • Efficiency? qstat -f *jobid* checkjob *jobid* showstart *jobid* showbf showq showq -r -u *user*	• Cancel job • Ssh to nodes • Interactive jobs • Debug queue canceljob *jobid* ssh *node* top vmstat qsub -I qsub -q debug	• Check .o/.e *jobid*.{o,e} • short-term statistics: showstats -u *user* • year-to-date usage on: https://portal.scinet.utoronto.ca