Using a cluster effectively
Scheduling and Job Management

• Log into Jasper.westgrid.ca:
  – ssh -X yourusername@jasper.westgrid.ca
  – use putty if you are working in windows

• Copy the working directory to your own and go into it.
  – cd scheduling-wg-2014

• You can find a copy of the slides and materials for this workshop in the following link
  https://www.westgrid.ca/events/scheduling_and_job_management_how_get_most_cluster
Using a cluster effectively

Scheduling and Job Management
Presentation contents

Job submission
Scheduling Theory
Troubleshooting
Batch Scheduling

• Is not used when you need a service for example a webserver that runs all the time.
• Is preferred when you have one or more jobs (simulations) that need to be run and you wish to get the results back sometime in the future.
• Your job automatically started by the scheduler when enough resources are available, and you get results back, you may be notified when your job starts and finishes.
Typical HPC Cluster
Goals of scheduling

• Fairness and policy
• Efficiency / Utilization / Throughput
• Minimize turnaround
Fairness and policy

- Does not necessarily mean everyone or every group gets the same usage.
- An important science project may get a larger allocation.
- Scheduler fairly allocates according to usage policy.
Efficiency, Utilization and Throughput

• We want all resources cpus, gpus, memory, disk, software licenses, bandwidth, and more to be all used as much as possible.

• How many gaps are there in scheduling between jobs.
Minimize turnaround

• Goal here is return an answer or result to a user as fast as possible
• Important to users which use iterative process to their goal.
• Minimize time to scientific discovery
Some insights

• The shorter the walltime which is the maximum time a job will run before being killed, the better we can meet the 3 goals of scheduling.

• Jobs using large amount of resources per job result in a reduction of fairness, efficiency, responsiveness of the scheduling system.

• The more nodes we have the better we can meet these goals.
Advantages of Large Clusters

• Larger clusters are more fair, efficient, responsive just by being larger.
• Larger clusters are capable of running larger jobs expanding capability, but if larger jobs are run exclusively we loose the advantage of a large cluster.
• Shared resources such as WestGrid are better and are used more efficiently than multiple small clusters. The larger the scope of shared resources the better.
Visualizing single node cluster

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>Day 4</th>
<th>Day 5</th>
<th>Day 6</th>
<th>Day 7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time

Now
Running jobs
Scheduling jobs in order of priority
Scheduling jobs in order of priority
Scheduling jobs in order of priority
Scheduling jobs in order of priority
Scheduling jobs in order of priority
Scheduling jobs in order of priority
Scheduling jobs in order of priority
A Job finishes early
Jobs are rescheduled
Jobs are rescheduled
Jobs are rescheduled
Jobs are rescheduled

Day 1: Core 1, Core 2, Core 3
Day 2: Core 1, Core 2
Day 3: Core 1, Core 3
Day 4: Core 1
Day 5: Core 1
Day 6: Core 1
Day 7: Core 1
Jobs are rescheduled
Jobs are rescheduled
Jobs are rescheduled
Single node cluster
Backfill

• Showbf command tells you how many processors are available for immediate use by anyone no matter the priority and for how long.

• This allows you to tailor your jobs to fit into perfectly into the gaps left by the scheduler
Short serial jobs
Myths

If there is a large number of jobs in the queue my job will not run quickly.

– Most of the time these jobs belong to users with very low priority, because they are running a large number of jobs.
– Most of these jobs may not be capable of running as number of running jobs per user may be limited.
– The cluster may have empty processors available for immediate use.
– Deciding if a cluster is busy by number of queued jobs does not work.

It is better not to submit to many jobs at a time so that other users can run.

– The scheduling system is more efficient if you submit your jobs earlier, as long as you don’t go over the usage limits.
– Fairness is insured by the scheduling system.
Tips

• Make sure your job can run on the resources available on the cluster.
• Look at the state of cluster/account/Jobs and how to get the information.
• If the cluster is empty and you are able to run shorter jobs to evade the limits.
Basic Job submission
Submitting a Job

• If you have a program that you wish to run you need to figure out the resource requirements of your Job. These requirements include:
  – walltime: maximum length of time your will take to run
  – number of cpus, memory, nodes, gpus
  – The queue you are submitting to.

• The command to submit your job is qsub, although qsub allows you to specify your requirements on the command line, however you should put your requirements in a job script.
  • qsub jobscript.pbs
Simple PBS job script

#PBS -S /bin/bash
#PBS -l nodes=1:ppn=1
#PBS -l walltime=5:00:00
#PBS -m abe
#PBS -M yourEmail@ualberta.ca
#PBS –N <my_job_name>

cd $PBS_O_WORKDIR; # Strongly recommended
sleep 1000; # Replace with a line running code
# Basic PBS script commands

<table>
<thead>
<tr>
<th>PBS script command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -S /bin/bash</td>
<td>Sets the shell that the job will be executed on the compute node.</td>
</tr>
<tr>
<td>#PBS -l nodes=1:ppn=1</td>
<td>Requests for 1 processors on 1 node.</td>
</tr>
<tr>
<td>#PBS -l procs=1</td>
<td></td>
</tr>
<tr>
<td>#PBS -l walltime=5:00:00</td>
<td>Sets the maximum runtime of 5 hours for your job.</td>
</tr>
<tr>
<td>#PBS –M &lt;email&gt;</td>
<td>Sets the email address for sending notifications about your job state.</td>
</tr>
<tr>
<td>#PBS -m abe</td>
<td>Sets the scheduling system to send you email when a mail is sent when the job is aborted by the batch system. b mail is sent when the job begins execution. e mail is sent when the job terminates.</td>
</tr>
<tr>
<td>#PBS –q &lt;quename&gt;</td>
<td>Puts your job into &lt;quename&gt; queue</td>
</tr>
<tr>
<td>#PBS –N myjobName</td>
<td>Gives your job a name</td>
</tr>
</tbody>
</table>
Running basic Jobs

BREAK FOR PRACTICE

compute • calcul
C A N A D A

UNIVERSITY OF ALBERTA
Jobs Types: Parallelism

- Many Serial Jobs
- Message Passing (MPI)
- Single node muti-core (OpenMP, Guassian)
- Hybrid/ Advanced

<table>
<thead>
<tr>
<th></th>
<th>1 Nodes</th>
<th>N Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 cpu</td>
<td>Serial</td>
<td>MPI</td>
</tr>
<tr>
<td>X cpus</td>
<td>OpenMP</td>
<td>Hybrid</td>
</tr>
</tbody>
</table>
Visualizing Multinode cluster

<table>
<thead>
<tr>
<th>Node</th>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>Day 4</th>
<th>Day 5</th>
<th>Day 6</th>
<th>Day 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1</td>
<td>Core 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Core 2</td>
<td>Core 3</td>
<td>Core 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 2</td>
<td>Core 1</td>
<td>Core 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Core 3</td>
<td>Core 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node 3</td>
<td>Core 1</td>
<td></td>
<td>Core 2</td>
<td>Core 3</td>
<td>Core 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Day 1: Core 1 and Core 2 active.
Day 2: Core 1 and Core 2 active.
Day 3: Core 1 and Core 2 active.
Day 4: Core 3 and Core 4 active.
Day 5: No active cores.
Day 6: No active cores.
Day 7: Core 1 and Core 2 active.
Many Serial Jobs
Many Serial Jobs

- Use 1 cpu per job
- Easiest and most efficient to schedule
- Excellent scaling linear speedup
- Example job would be a parameter searches
- In your pbs file one asks for a serial job in one of the following ways
  - #PBS -l nodes=1:ppn=1
  - #PBS -l procs=1
Serial job PBS example

#PBS -S /bin/bash
#PBS -l nodes=1:ppn=1
#PBS -l walltime=5:00:00
#PBS -m abe
#PBS –M <your_email@ualberta.ca>
#PBS –N <my_job_name>
cd $PBS_O_WORKDIR
sleep 1000; # Replace with a line running code
Tips for running more Serial Jobs

• Submit shorter serial jobs
• Many short serial jobs will run before larger job
• Checkpoint longer jobs and submit them as short jobs, this will also save you when the cluster suffers hardware or power failure.
Job array

- Job arrays are used when you have need to submit a large number of Jobs using the same job script.
- There is a naming convention for jobs in array, which is useful as you don’t need to remember a large number of unique job ids or job names: jobname[0]
- Job arrays are preferred as they don’t require as much computation by the scheduling system to schedule, as they are evaluated as a group instead of individually. Ask for a job array in one of the following ways:
  - #PBS -t 0-99  job array 100 jobs numbered 0 -99
  - #PBS -t 1,2,3,5,7  Job array with 5 jobs with indexes [1,2,3,5,7]
  - #PBS -t 0-99%5  job array 100 jobs numbered 0 -99 with a maximum of 5 running at any time
Job array

• To display information on a Job array job in csh/tcsh shell one has to use double quotes
  – qstat -f "4485414[920]"

• To get qstat to display individual jobs in an array, do:
  – qstat -t
Job array sample script

#PBS -S /bin/bash
#PBS –l procs=1
#PBS –t 1-10
#PBS -l walltime=10:00
#PBS -m abe
#PBS –M email@ualberta.ca
#PBS –N myjob

cd $PBS_O_WORKDIR
export $MYOUTPUT= $PBS_JOBNAME-$PBS_ARRAYID

echo “This jobs name is:  $PBS_JOBNAME”  > $MYOUTPUT

echo “This jobs jobid is:  $PBS_JOBID”  >> $MYOUTPUT

echo “This jobs arrayid is:  $PBS_ARRAYID”  >> $MYOUTPUT

echo “This jobs is running on host:  $PBS_O_HOST”  >> $MYOUTPUT

sleep 100;  # Replace with a line running code
### MPI Job

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td></td>
<td></td>
<td></td>
<td>MPI Job part 1 of 4</td>
</tr>
<tr>
<td>Day 2</td>
<td></td>
<td></td>
<td></td>
<td>MPI Job part 2 of 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node 2</th>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td></td>
<td></td>
<td></td>
<td>MPI Job part 3 of 4</td>
</tr>
<tr>
<td>Day 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node 3</th>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td></td>
<td></td>
<td></td>
<td>MPI Job part 4 of 4</td>
</tr>
<tr>
<td>Day 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MPI Jobs

• Use the network for message passing
• Each job uses multiple cpus each of which can be on a different node.
• Each process uses a different memory address space
• More difficult to write parallel code than OpenMP as deadlocks are more common.
• Can scale higher than OpenMP as clusters are typically larger than even large SMP machines
MPI Job Submission

- This type of job can have its processes running on any node, multiple processes can run on a single node.
- #PBS –l procs=64
Single node multi-core job (OpenMP, Gaussian, Threads)

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>🟠</td>
<td>🟠</td>
<td></td>
<td>🟠</td>
</tr>
<tr>
<td>Node 2</td>
<td>Core 1</td>
<td>Core 2</td>
<td>Core 3</td>
<td>Core 4</td>
</tr>
<tr>
<td></td>
<td>🟠</td>
<td>🟠</td>
<td>🟠</td>
<td>🟠</td>
</tr>
<tr>
<td></td>
<td>🟠</td>
<td>🟠</td>
<td>🟠</td>
<td>🟠</td>
</tr>
<tr>
<td>Node 3</td>
<td>Core 1</td>
<td>Core 2</td>
<td>Core 3</td>
<td>Core 4</td>
</tr>
<tr>
<td></td>
<td>🟠</td>
<td>🟠</td>
<td></td>
<td>🟠</td>
</tr>
<tr>
<td></td>
<td>🟠</td>
<td>🟠</td>
<td></td>
<td>🟠</td>
</tr>
</tbody>
</table>

Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 |
|-------|-------|-------|-------|-------|-------|-------|

OpenMP job 4 nodes=1:ppn=4
Single node multi-core job

• All the threads must run on a single node.
• The threads share a single memory address space
• Can compile serial and parallel executables from the same source code
• OpenMP is one of the easiest methods of parallel programming, can be done incrementally.
OpenMP job submission

• This type of job must have its thread running on one node, sharing the same memory.
• Communication between parts of the job is done via memory.
• #PBS –l nodes=1:ppn=8
• One can ask the program to run a number of threads via an environment variable:
  – export OMP_NUM_THREADS=8
• Usually set it to the requested cores:
  – export OMP_NUM_THREADS=$PBS_NUM_PPN
  or
  – export OMP_NUM_THREADS=$PBS_NP
Tips for running OpenMP Jobs

• Check the state of the cluster to see if your job will run quickly.

• If you need to run a larger job use Breezy or Hungabee.

• If you have a number of OpenMP style jobs you should consider running longer jobs using less cpus per job instead.
  – It is faster and more efficient to schedule single/smaller processor jobs.
  – This advice may not apply when you need other resources like large amount of RAM per job.
Hybrid Job

Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7

Node 1
Core 1
Core 2
Core 3
Core 4

Node 2
Core 1
Core 2
Core 3
Core 4

Node 3
Core 1
Core 2
Core 3
Core 4

Hybrid Job nodes=2:ppn=2 part 1 of 2

Hybrid Job nodes=2:ppn=2 part 2 of 2
Why use a hybrid job

• It’s possible to combine OpenMP and MPI for running on clusters of SMP machines
• Need more memory or other resource than is available per core.
• Advanced systems of running parallel jobs can utilize resources more efficiently. Communication between cores is faster than between distant nodes. These systems include Chapel language as well as Partitioned global address space languages (PGAS) such as Unified Parallel C, Co-array Fortran.
#PBS script commands

<table>
<thead>
<tr>
<th>PBS script command</th>
<th>Description</th>
</tr>
</thead>
</table>
| #PBS -l nodes=1:ppn=1
PB#S -l procs=1 | Requests for 1 processors on 1 node. (Serial) |
| #PBS -l nodes=1:ppn=X | Requests for X processors on the same node (OpenMP) |
| #PBS -l nodes=X:ppn=1
PB#S -l procs=X | Requests for X processors which may be running on any nodes (MPI) |
| #PBS -l nodes=X:ppn=Y | Requests X nodes with Y processes (Hybrid) |
| #PBS –t 0-4
PB#S –t 0,1-4
PB#S –t 0,1,2,3,4 | Requests Job array of 5 jobs |
| #PBS –t 1-X%Y
ex: #PBS –t 1-100%10 | Requests requests Job array of X jobs with only a maximum of Y jobs running at the same time |
Serial, mpi, openmp, hybrid, jobarrays

BREAK FOR PRACTICE
QUESTIONS?
The End