Optimizing Batch Submission and Job Performance

Matt Gitzendanner: magitz@ufl.edu

4/10/14
HiPerGator

The University of Florida Supercomputer for Research

- 16,384 cores (total of ~20,000 today)
- Infiniband interconnect
- >3PB fast, high-availability, storage
Cluster basics

User interaction
Login server (Head node)

Scheduler
Tell the scheduler what you want to do

Compute resources
Your job runs on the cluster
Scheduling a job

- Need to tell scheduler what you want to do
  - **How many CPUs** you want and how you want them grouped
  - **How much RAM** your job will use
  - **How long** your job will run
  - The commands that will be run
Ordinary Shell Script

```bash
#!/bin/bash
date
module load test_app
test_app -i file.txt
```

Read the manual for your application

Commands typed on the command line can be put in a script.
UF Research Computing

- Submission Script

```bash
#!/bin/bash
#PBS -N My_Job_Name
#PBS -M Joe_Shmoe@ufl.edu
#PBS -m abe
#PBS -o My_Job.log
#PBS -e My_Job.err
#PBS -l nodes=1:ppn=1
#PBS -l pmem=900mb
#PBS -l walltime=00:05:00

cd $PBS_O_WORKDIR
date
module load test_app
test_app -i file.txt
```
How do you know???

- Experience
- Trial and Error
- Run on development node:
  
  ```bash
  $ ssh dev1
  $ module load my_app
  $ my_app -i file1.txt &
  $ top
  ```

---

**top** - 15:30:21 up 196 days, 21:26, 44 users, load average: 1.09, 4.71, 8.34
Tasks: 1874 total, 2 running, 1864 sleeping, 8 stopped, 0 zombie
Cpu(s): 1.6%us, 0.2%sy, 0.0%ni, 98.2%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 264523424k total, 180940792k used, 83582632k free, 38652k buffers
Swap: 8388600k total, 2031204k used, 6357396k free, 72171180k cached

<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>45338</td>
<td>jmeyer</td>
<td>20</td>
<td>0</td>
<td>92.6g</td>
<td>85g</td>
<td>4540</td>
<td>R</td>
<td>99.2</td>
<td>33.9</td>
<td>2454:48</td>
<td>mira</td>
</tr>
<tr>
<td>24915</td>
<td>malex</td>
<td>20</td>
<td>0</td>
<td>138m</td>
<td>9824</td>
<td>2192</td>
<td>S</td>
<td>6.9</td>
<td>0.0</td>
<td>8:03:62</td>
<td>perl</td>
</tr>
<tr>
<td>41</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>2.6</td>
<td>0.0</td>
<td>1121:42</td>
<td>ksoftirqd/9</td>
</tr>
<tr>
<td>40467</td>
<td>magitz</td>
<td>20</td>
<td>0</td>
<td>18476</td>
<td>2672</td>
<td>976</td>
<td>R</td>
<td>2.0</td>
<td>0.0</td>
<td>0:00:61</td>
<td>top</td>
</tr>
<tr>
<td>37</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>1.6</td>
<td>0.0</td>
<td>1157:49</td>
<td>ksoftirqd/8</td>
</tr>
<tr>
<td>26491</td>
<td>greulich</td>
<td>20</td>
<td>0</td>
<td>18480</td>
<td>2716</td>
<td>988</td>
<td>S</td>
<td>1.6</td>
<td>0.0</td>
<td>1:33:53</td>
<td>top</td>
</tr>
<tr>
<td>40493</td>
<td>arnh</td>
<td>20</td>
<td>0</td>
<td>144m</td>
<td>5900</td>
<td>2180</td>
<td>S</td>
<td>1.9</td>
<td>0.0</td>
<td>0:00:44</td>
<td>vim</td>
</tr>
</tbody>
</table>
How do you know???

- Start general and refine
  - `#PBS -M magitz@ufl.edu`
  - `#PBS -m abe`
  - `#PBS -l nodes=1:ppn=1`
  - `#PBS -l pmem=4gb`
  - `#PBS -l walltime=24:00:00`
- Look at the ending or abort email for time and ram usage and adjust

Job deleted at request of root@moab.ufhpc job 4189661 exceeded MEM usage hard limit (2325 > 2252) (MB)
How do you know???

- **Common misconceptions**
  - More cores (processors) will make my application run faster
  - More RAM will make my application run faster
  - *The University of Florida Supercomputer for Research* will run my application faster than on my laptop
Scheduling a job

- Need to tell scheduler what you want to do
  - How many CPUs you want and how you want them grouped
  - How much RAM your job will use
  - How long your job will run
  - The commands that will be run

Scheduler

Tell the scheduler what you want to do
Walltime

#PBS -l walltime=00:50:00

• Fairly straight forward
• As with all resource requests, accuracy helps ensure your jobs and all other jobs will run sooner

<table>
<thead>
<tr>
<th></th>
<th>Maximum</th>
<th>Short</th>
<th>Long</th>
</tr>
</thead>
<tbody>
<tr>
<td>Investor</td>
<td>31 days</td>
<td>&lt;12 hrs</td>
<td>7 days</td>
</tr>
<tr>
<td>Other</td>
<td>7 days</td>
<td>&lt;12 hrs</td>
<td>3 days</td>
</tr>
</tbody>
</table>
RAM

`#PBS -l pmem=900mb`

- Lots to consider, but do your best at estimating RAM needed for job
- Over about 4GB of RAM, “costs” toward CPU allocation

Wasted RAM leads to idle CPUs and low job throughput
RAM– bigmem queue

- For jobs asking for over 16GB per core (pmem)
- `#PBS -q bigmem`
- 1TB, 750GB and 512GB nodes
End-of-job emails: #PBS -M Joe_Shmoe@ufl.edu
#PBS -m abe

PBS Job Id: 358634.moab.ufhpc
Job Name: NR.25.nex
Exec host: c7a-s1/60
Execution terminated
Exit_status=0
resources_used.cput=07:16:09
resources_used.mem=251348kb
resources_used.vmem=318916kb
resources_used.walltime=07:16:52
Nodes and processors

Single processor apps:
#PBS -l nodes=1:ppn=1

Threaded (& MPI) apps:
#PBS -l nodes=1:ppn=4

MPI apps:
#PBS -l nodes=2:ppn=32
Nodes—Processors—“Cores”

A compute node or server

Each node has 4 processors

Most HiPerGator nodes have 64 cores or ppn and 256GB of RAM

The scheduler uses “processors” where most think of “cores”.

The processor request (ppn) is what most people think of as cores.

Each processor has 16 “cores”
Processor Requests

- Is your application parallel?
- Can it use CPUs on multiple nodes?
- How well does it scale?
HiPerGator
The University of Florida Supercomputer for Research

- **64 cores per node**
  - If RAM allows, MPI jobs under ~32 cores, should use `nodes=1:ppn=##`

- Some older nodes have 4-16 cores
**MPI**

- Up to 32 cores: `nodes=1:ppn=#`
- Use the `--bind-to-core` flag
  - `mpiexec --bind-to-core my_app ...`

- If you use MPI a lot, talk to us
  - Several optimization flags
Task Arrays

- `#PBS -t 1-1000` Runs 1000 tasks all submitted at once
- `#PBS -t 1-1000%20` Will throttle to run 20 tasks at a time
- `$PBS_ARRAYID`

```bash
#PBS -t 1-100%10
cd $PBS_O_WORKDIR
module load my_app
file=`ls *.txt | head -n $PBS_ARRAYID | tail -n 1`
my_app -i $file
```
**Task Arrays**

- The "[]" is part of the job number
  - `qdel 1234[]`
  - `qdel -t 45 1234[]`
  - `qdel -t 1,10,50-100 1234[]`
  - `qstat -t 1234[]`

- Resource requests are per task
  - `#PBS -l nodes=1:ppn=4`
  - Each task gets 4 processors

- Can’t assume tasks are executed in order
Checking resources

- How many jobs are running in my group?
  - `showq -w group=<group_name>`

- How many resources will this job take?
  - `pbs_info -f script_file.pbs`

- Why isn’t this job running?
  - `checkjob -v <job_id>`

- **NOTE:** Job violates constraints for partition base
  (job 46226XX violates active HARD MAXPE limit of 310
  for group XXXX partition ALL (Req: 1 InUse: 310))
Some helpful environment variables

- $PBS_O_WORKDIR: the directory where you typed qsub
- $PBS_JOBID: the unique job id: e.g. 24461774.moab.ufhpc
- $TMPDIR: temporary directory for each job on compute node’s local disk, good for jobs with lots of small I/O
- $PBS_NUM_PPN: Number of processors for single node job, use this when starting a threaded application to tell it how many processors to use. Prevents needing to change in multiple places. E.g. nodes=1:ppn=4, blastn –num_threads $PBS_NUM_PPN
- $PBS_JOBNAME: Name your gave your job with #PBS –N
UF Research Computing

- Help and Support
  - Help Request Tickets
    - [https://support.hpc.ufl.edu](https://support.hpc.ufl.edu)
    - For any kind of question or help requests
    - Searchable database of solutions
  - We are here to help!
    - [support@hpc.ufl.edu](mailto:support@hpc.ufl.edu)
Help and Support (Continued)

- [http://wiki.hpc.ufl.edu](http://wiki.hpc.ufl.edu)
  - Documents on hardware and software resources
  - Various user guides
  - Many sample submission scripts
- [http://hpc.ufl.edu/support](http://hpc.ufl.edu/support)
  - Frequently Asked Questions
  - Account set up and maintenance