Increasing HPC Resiliency Leads to Greater Productivity

Roger Moye
University of Texas – MD Anderson Cancer Center
rvmoye@mdanderson.org
November 14, 2016
## Our HPC Environment

<table>
<thead>
<tr>
<th></th>
<th>Shark HPC</th>
<th>Nautilus HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute nodes</td>
<td>80</td>
<td>336</td>
</tr>
<tr>
<td>CPU cores per node</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Total cores</td>
<td>1920</td>
<td>8064</td>
</tr>
<tr>
<td>Memory per node</td>
<td>384GB</td>
<td>64GB, 128GB, 192GB</td>
</tr>
<tr>
<td>Filesystem</td>
<td>1.6 PB GPFS</td>
<td>800TB GPFS</td>
</tr>
<tr>
<td>Primary workload</td>
<td>Next generation sequencing.</td>
<td>Basic science (biostatistics, radiation physics, etc).</td>
</tr>
</tbody>
</table>

- ~200 applications installed.
- ~125 active users between the two clusters.
- ~3 FTE.
- Utilization typically around 70% or higher.
Node Availability

- Memory over-subscription of compute nodes a major problem on Nautilus.
  - Jobs using (*a lot*) more memory than expected.
  - Users unable to provide reasonable estimates for memory utilization.
  - Users instructed to request a number of CPUs commiserate with the amount of memory they needed:
    - A job that needed 40% of the memory should request 40% of the CPUs on a node.
    - In theory this would prevent over-subscription.
    - In practice this was marginally successful and a difficult concept to convey to the user community.
Node Availability

• Memory over-subscription resulted in node crashes.
  • *Node crash event* – one or more nodes crashing.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node crash events</td>
<td>1.4 per day</td>
</tr>
<tr>
<td>Scope of event</td>
<td>3 nodes (almost 1% of the cluster)</td>
</tr>
<tr>
<td>Largest event</td>
<td>55 nodes (16% of the cluster)</td>
</tr>
</tbody>
</table>

* Metrics are for Nautilus.

• In extreme cases a large number of node failures would have an adverse impact on the shared filesystem which would in turn impact the entire cluster.
Lost Time and Productivity

• Each event resulted in:
  • Lost compute jobs
    • For the offending user.
    • For other users sharing the node.
  • Lost CPU capacity while the nodes were offline.
  • Lost time for the systems administrators
    • Identify the offending jobs.
    • Notify the affected users.
    • Reboot the nodes.
  • These issues often resulted in Help Desk tickets from the user community.
    • Typically ~2 hours per day was devoted to this problem, including weekends.

• Not sustainable in many respects!
Single Threaded versus Multithreaded

- Over-subscription of CPUs on Nautilus was a secondary problem.
  - Users unaware of their software being multithreaded.
  - Software using more cores than reserved by the scheduler.
  - Too many multithreaded jobs being assigned to the same node(s).
    - Caused jobs to run much slower than expected.
    - Jobs exceeded their run time and were terminated.
    - Users submitted tickets asking for help.
    - Users required to resubmit jobs.
  - *More lost time and lost productivity.*
Solution: Job Scheduler to the Rescue!

- Shark HPC was designed to avoid memory over-subscription via LSF configuration:

<table>
<thead>
<tr>
<th>LSF Config File</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsb.conf</td>
<td>LSB_MEMLIMIT_ENFORCE=y</td>
</tr>
<tr>
<td>lsb.queues</td>
<td>MEMLIMIT = 1 377856, RES_REQ = &quot;rusage[mem=8192] span[hosts=1]&quot;</td>
</tr>
</tbody>
</table>

- Job scheduler will terminate a job that exceeds the amount of memory it has requested.
- Jobs that do not specify a memory value will be assigned the soft limit.
- Soft and hard limits are set with the MEMLIMIT option above.
- Hard limit chosen to be 369GB (369 * 1024MB = 377856MB).
  - Allows operating system and GPFS pagepool to occupy 15GB.
- All jobs will reserve at least 8GB ([rusage[mem=8192]]).
Solution: Job Scheduler to the Rescue!

• Required lines in the LSF Job submission:

<table>
<thead>
<tr>
<th>LSF parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#BSUB –M 8192</td>
<td>Memory limit</td>
</tr>
<tr>
<td>#BSUB –R rusage[mem=8192]</td>
<td>Memory reservation</td>
</tr>
</tbody>
</table>

• Memory limit parameter is the memory hard limit.
• Memory reservation is the amount that has been reserved and can not be scheduled/reserved by other jobs.

• This prevented the memory over-subscription problem from ever occurring on Shark!
Solution: Job Scheduler to the Rescue!

- Porting the changes to Nautilus and expanding the scope:

<table>
<thead>
<tr>
<th>Moab parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVERSUBMITFILTER /opt/moab/etc/jobFilter.pl</td>
<td>Used to confirm that required submission parameters are present.</td>
</tr>
<tr>
<td>RESOURCELIMITPOLICY JOBMEM: ALWAYS, ALWAYS: NOTIFY, CANCEL</td>
<td>Notify the user when job memory exceeds soft limit. Cancel job when it exceeds hard limit.</td>
</tr>
<tr>
<td>RESOURCELIMITPOLICY PROC: ALWAYS, ALWAYS: NOTIFY, CANCEL</td>
<td>Notify the user when job CPU core utilization exceeds soft limit. Cancel job when it exceeds hard limit.</td>
</tr>
<tr>
<td>RESOURCELIMITPOLICY WALLTIME: ALWAYS, ALWAYS: NOTIFY, CANCEL</td>
<td>Notify the user when job walltime exceeds soft limit. Cancel job when it exceeds hard limit.</td>
</tr>
</tbody>
</table>

- If soft and hard limits are not specified in the scheduler configuration, then those limits become the hard limit as specified by the job submission.
Solution: Job Scheduler to the Rescue!

• Required lines in the Moab job submission:

<table>
<thead>
<tr>
<th>LSF parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -l nodes=1:ppn=1</td>
<td>Node and CPU hard limit</td>
</tr>
<tr>
<td>#PBS -l walltime=1:00:00</td>
<td>Run time hard limit</td>
</tr>
<tr>
<td>#PBS -l mem=1gb</td>
<td>Memory hard limit</td>
</tr>
</tbody>
</table>

• Memory limit parameter is the memory hard limit.

• This prevented the memory over-subscription problem from ever occurring on Nautilus!
Solution: Job Scheduler to the Rescue!

- The **JOBMEM** policy will only work with the Moab **MAXMEM** job submission parameter.
- **SERVERSUBMITFILTER** script rewrites the user’s submit script:

  ```bash
  #PBS -l mem=1gb
  becomes
  #PBS -W x=MAXMEM:1gb
  Or adds this if **mem** is absent
  #PBS -W x=MAXMEM:1mb
  **NOTE:** Must submit jobs with **msub** (not **qsub**) to use **SERVERSUBMITFILTER**.
Results!

- 525 days of uptime on Shark.
- 90% of Shark compute nodes NEVER rebooted.
- 360 days of uptime on Nautilus.
- 70% of Nautilus compute nodes NEVER rebooted.

- In the first year of operation for Nautilus:
  - 850,000 jobs submitted.
  - 12,157 jobs exceeded their memory hard limit and were terminated.
  - 2,187 jobs exceeded their CPU cores hard limit and were terminated.
  - 0 compute nodes rebooted due to over-subscription.
Self Diagnosis

- Examples of error messages sent to the user:

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moab</td>
<td>job 3546 exceeded MEM usage hard limit (6135 &gt; 5120).</td>
</tr>
<tr>
<td>LSF</td>
<td>TERM_MEMLIMIT: job killed after reaching LSF memory usage limit.</td>
</tr>
</tbody>
</table>
Error Messages

Job has exceeded its walltime (in seconds):
Job 3558 exceeded WALLTIME usage hard limit (68 > 66).

Job has exceeded its maximum memory allowed (in MB):
job 3546 exceeded MEM usage hard limit (6135 > 5120).

Job has exceeded its maximum number of CPU cores allowed:
job 3526 exceeded PROC usage hard limit (278 > 110).

In order to receive these error messages you must enable email notification in your job submission script with the following line:

#PBS –M MyEmailAddress

Optionally, if you want to receive email when the job aborts, begins, and ends, add this line:

#PBS –m abe
Unintended Consequences

- Users tend to deliberately over-estimate their memory and CPU core requirements, resulting in under-utilization of the cluster.

70% CPU util.  
100% Node util.
Hardening the Resources

• Other items that can be precursors to node crashes:

<table>
<thead>
<tr>
<th>Item</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage block and inode usage</td>
<td>&gt; 90%</td>
</tr>
<tr>
<td>Remote storage unmounted</td>
<td>Y/N</td>
</tr>
<tr>
<td>Memory available</td>
<td>&lt; 2GB</td>
</tr>
<tr>
<td>Swap available</td>
<td>&lt; 15GB</td>
</tr>
<tr>
<td>sshd is running</td>
<td>Y/N</td>
</tr>
</tbody>
</table>

• Monitored with LSF External Load Information Manager (ELIM) script on Shark, and with LBNL Node Health Check (NHC) from Lawrence Berkeley National Laboratory [3] on Nautilus.

• Nodes are automatically marked as “offline” if any of the above conditions are met.
Hardening the Resources

• Deploy a RAMDisk for applications with poor I/O patterns:

Execute from /etc/rc.local:

```
mount -t tmpfs -o size=15g tmpfs /mnt/tmpfs
```

• Clean up RAMDisk with tmpwatch:

```
/etc/cron.daily/tmpwatch
flags=-umc
/usr/sbin/tmpwatch "flags" 1ld /mnt/tmpfs
```

• The size of the RAMDisk and the interval of the tmpwatch clean up is dependent on the requirements and duration of the jobs using the RAMDisk.
Hardening the Resources

• Use `tmpwatch` to clean up `/var`.

<table>
<thead>
<tr>
<th>etc/cron.daily/tmpwatch</th>
</tr>
</thead>
<tbody>
<tr>
<td>flags=-umc</td>
</tr>
<tr>
<td>/usr/sbin/tmpwatch &quot;flags&quot; 10d /var/spool/torque/undelivered</td>
</tr>
<tr>
<td>/usr/sbin/tmpwatch &quot;flags&quot; 60d /var/spool/torque/mom_priv/jobs</td>
</tr>
<tr>
<td>/usr/sbin/tmpwatch &quot;flags&quot; 60d /var/spool/torque/spool</td>
</tr>
</tbody>
</table>

• The interval for **undelivered** was chosen somewhat arbitrarily.
• The interval for **jobs** and **spool** was chosen to exceed the longest possible run time of a job on the cluster.
Accomplishments!

• Nearly 100% uptime on compute nodes is possible.
• Users able to self-diagnose their job failures because the cluster tells them why jobs were terminated.
• Training of users regarding memory reservation via the job scheduler is much easier.
• Nodes are marked offline when a node error condition occurs, and automatically recover when the error condition is resolved.
• Compute job efficiency is more often discussed now than compute node availability.
• Compute nodes are cleaned up (tmpwatch) automatically.
• Recovered 10 hours per week for FTE to work on other things.
• *Increased productivity for HPC staff and customers.*
Accomplishments!

• Since June 1, 2016 (176 days as of 11/07/16):
  • Nautilus 100% cluster uptime
    • 89% of nodes never rebooted.
    • 96% of nodes up over 145 days.

• Shark 100% cluster uptime
  • 95% of nodes never rebooted.
Acknowledgements

- Nathan Baca – IBM Platform Professional Services
- Larry Adams – IBM Platform Computing
- Deepak Khosla – X-ISS
- Bryan Bales – X-ISS
- Waco Millican – X-ISS
- Many at Adaptive Computing
- Dr. Bradley Broom – Professor, Bioinformatics and Computational Biology, UT-MD Anderson Cancer Center
- Dan Jackson, Sally Boyd, Jenny Chen, Eric Sisson, Rong Yao, and Andrew Adams – Research Information Systems, UT-MD Anderson Cancer Center
- Jonathan Fosburgh, Larry Henson, Dan Metts, and Reginald Maxwell – IT Operations Storage Team, UT-MD Anderson Cancer Center
- William Joe Allen – Texas Advanced Computing Center
References

