Using Quality of Service for Scheduling on Cray XT Systems

Troy Baer

HPC System Administrator

National Institute for Computational Sciences, University of Tennessee

Outline

- Introduction
- Scheduling Cray XT systems with TORQUE and Moab
  - Scheduling algorithm
  - Queue structure
  - Job Prioritization
  - Quality of service levels
- Case studies
  - Normal operation on Kraken
  - Nightly weather forecasting on Kraken
  - User-managed scheduling on Athena
- Conclusions

Introduction

- NICS operates two Cray XT systems for the U.S. National Science Foundation's Teragrid system:
  - Kraken, 88 cabinet XT5.
  - Athena, 48 cabinet XT4.
- The two systems are allocated differently:
  - Kraken is allocated through the Teragrid Resource Allocation Committee.
  - Athena is dedicated to one or two projects at a time on a roughly quarterly basis.
- On both systems, there are occasionally needs to give special scheduling consideration to individual projects or users.

System Details

System	Kraken	Athena
Cabinets	88	48
Compute Nodes	8,256	4,512
Processor	AMD Opteron 2.6 GHz hex-core	AMD Opteron 2.3 GHz quad-core
Total Cores	99,072	18,048
Peak Performance (TFLOP/s)	1,030.3	165.9
Memory (TB)	129	17.6
Disk (TB)	2,400	85
Disk Bandwidth (GB/s)	30	12

Scheduling Cray XT Systems with TORQUE and Moab

- Like many large XT systems, Kraken and Athena use TORQUE and Moab for resource management.
  - TORQUE is an open source batch system from Adaptive Computing (nee Cluster Resources).
    - Derived from PBS.
    - Works on everything from a laptop to JaguarPF.
  - Moab is a closed source scheduler from Adaptive Computing that works with many batch systems, including TORQUE.
    - Highly flexible priority system.
    - Advance reservations.
    - Quality of service (QOS) mechanisms.
    - Modular design allows integration with external services (e.g. ALPS on Cray XTs).

Scheduling Algorithm

The basic Moab scheduling algorithm has seven steps:

- **Update status information from resource manager(s). ***
- **Refresh reservations.**
- **Start jobs with reservations (if possible). ***
- **Start jobs with highest priority (if possible). ***
- **Backfill jobs. ***
- **Update statistics.**
- **Handle client requests.**

*** Requires interaction with ALPS on Cray XTs**

Queue Structure

Queue	Max. Walltime	Max. Cores (Athena)	Max. Cores (Kraken)
small	24 hours	512	512
longsmall	60 hours	512	256
medium	24 hours	2,048	8,192
large	24 hours	8,192	49,536
capability	24 hours	18,048	99,072
dmover	24 hours	0	0
hpss	24 hours	0	0

Job Prioritization

In addition to having similar queue structures, Kraken and Athena use the same set of priority weights, which prioritize jobs based on:

- Size.
- Queue time.
- Expansion factor (though this was recently removed).

Quality of Service Levels

- A *quality of service (QOS) level* is an object in Moab enabling a job to request special scheduling considerations.
  - Can be applied automatically to a job by Moab or explicitly requested (e.g. `-l qos=foo` in TORQUE).
  - Access to a QOS can be assigned to any Moab credential (user, group, account, queue/class).
  - Can have a number of effects:
    - Modify priority.
    - Modify throttling policies.
    - Change reservation behavior.
    - Change backfill behavior.
    - Enable preemption.

Quality of Service Levels (con't.)

Kraken and Athena originally had two non-default QOS levels available:

- `sizezero`
  - Applied automatically by Moab to any job requesting `size=0` (typically data transfer jobs).
  - Uses `RUNNOW` flag to cause `size=0` jobs to run in a timely manner despite their low priority.
- `negbal`
  - Applied by the TORQUE submit filter to jobs charging against projects whose balances have gone negative.
  - Large, negative priority modifier.

Case Studies

- Three cases
  - Normal operation on Kraken.
  - Normal operation on Kraken with reservations in support of nightly weather forecasts.
  - Dedicated operation of Athena for a single group.
- In all cases, we'll look at how different QOSes affect the queue times of jobs, as users tend to look at shorter queue times as proof of better service.
- The Moab settings used to accomplish all of the work described here are in the paper's appendix.

Normal Operation on Kraken

- Baseline.
- Time frame: 5 Oct 2009 to 31 Mar 2010.
- System utilization: 65.62% (not compensated for downtime).

Normal Operation on Kraken (con't.)

QOS	Jobs	CPU Hours	Min Queue Time	Max Queue Time	Mean Queue Time
default	220,197	241.7M	00:00:03	858:59:59	03:59:26
negbal	13,137	39.5M	00:00:04	398:04:00	08:51:31
sizezero	2,231	0.0M	00:00:05	262:51:39	04:11:42

Normal Operation on Kraken (con't.)

- `negbal` jobs generally wait longer than jobs with other QOSes.
- `sizezero` jobs wait slightly longer than jobs in the default QOS.
  - This is because the `RUNNOW` flag was added to the `sizezero` QOS after some time after the start of the period in question in response to long queue times for those jobs.

Nightly Weather Forecasting on Kraken

- The Center for the Analysis and Prediction of Storms (CAPS) at Oklahoma University received a Teragrid allocation on Kraken for their 2009 Spring Experiment:
  - WRF-based weather forecasts and associated post-processing five nights a week, 10:30pm to 6:30am.
  - Standing reservations held resources (~10k cores) available.
  - Access to reservations controlled by `capsforecast` and `capspostproc` QOSes, which also gave priority bumps.
- Time frame: 16 Apr 2009 to 12 June 2009.
- System utilization: 66.02% (not compensated for downtime).

Nightly Weather Forecasting on Kraken (con't.)

QOS	Jobs	CPU Hours	Min Queue Time	Max Queue Time	Mean Queue Time
default	35,257	55.6M	00:00:02	466:24:39	03:51:50
negbal	2,692	3.0M	00:00:04	146:53:14	02:15:37
sizezero	611	0.0M	00:00:03	132:26:36	01:04:35
caps forecast	68	3.0M	00:00:05	42:01:04	01:48:31
caps postproc	2,155	0.1M	00:00:03	26:42:12	00:02:49

Nightly Weather Forecasting on Kraken (con't.)

- CAPS jobs generally received excellent service.
  - The largest queue time component for forecast jobs was the fact that they were generally submitted around 9PM with a flag that prevented them from being eligible to run before 10:30PM.
- However, this caused significant impact on other users.
  - `longsmall` jobs' mean queue time went from ~52 hours to ~96.5 hours.
  - `capability` jobs could not run longer than 16 hours during the week.
    - The `caps*` QOSes also caused `capability` jobs to lose reservations, resulting in "near-miss" behavior and several user complaints.
    - The `capability` queue was subsequently assigned a `bigjob` QOS with its own reservation pool.

User-Managed Scheduling on Athena

- The first group given dedicated access to Athena was a climate modeling project from the Center for Ocean-Land-Atmosphere Studies (COLA) at the Institute of Global Environment and Society (IGES).
  - Two codes:
    - IFS from EMCWF.
    - NICAM from University of Tokyo (previously run only on the Earth Simulator).
  - Two QOSes added to allow COLA users to manage their own scheduling:
    - `bypass` (has NTR "next-to-run" flag).
    - `bottomfeeder` (no backfill or reservations).
- Time frame: 1 Oct 2009 to 31 Mar 2010.
- System utilization: 90.51% (not compensated for downtime).

User-Managed Scheduling on Athena (con't.)

QOS	Jobs	CPU Hours	Min Queue Time	Max Queue Time	Mean Queue Time
default	11,851	37.4M	00:00:02	138:33:50	01:12:50
negbal	57	0.5M	00:00:02	06:19:08	00:37:19
sizezero	4,822	0.0M	00:00:01	148:22:35	01:31:55
bypass	1,540	32.2M	00:00:02	43:32:33	01:01:22
bottom feeder	85	1.3M	00:00:03	137:09:04	22:27:44

User-Managed Scheduling on Athena (con't.)

- Jobs with `bypass` QOS waited slightly less than default and `sizezero` jobs.
- Jobs with `bottomfeeder` QOS waited much, much longer than others.
  - Users initially complained that Moab scheduled `bottomfeeder` jobs too aggressively.
  - It turned out that the complaining users had a simulation job that submitted a `size=0` data transfer job that in turn submitted another simulation job, resulting in many situations where only `size=0` and `bottomfeeder` jobs were eligible to run.
  - After the users restructured their jobs to have their simulation jobs submit both data transfer and subsequent simulation jobs, things behaved as expected.

Conclusions

- The use of QOS levels has become integral to scheduling on Kraken and Athena:
  - Pushing through data transfer jobs.
  - Deprioritizing jobs from projects with negative balances.
  - Scheduling policy modifications and exceptions for individual users, groups, and projects.
  - In the extreme, allowing users of a dedicated system to manage their own scheduling.