Resource Management Analysis and Accounting

Mike Showerman Mark Klein Joshi Fullop Jeremy Enos
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Urbana, IL USA
Email: mshow@ncsa.illinois.edu {mdk,fullop,jenos}@illinois.edu

***Abstract*— Maximizing the return on investment in a large scale computing resource requires policy that best enables the highest value workloads. Measuring the impact of a given policy presents great challenges with a highly variable workload. Defining and measuring the separate components of scheduling and resource management overhead is critical in reaching a valuable conclusion about the effectiveness of the system's availability for your workload. NCSA has developed tools for collecting and analyzing both user workload and system availability to measure the delivered impact of the Blue Waters resource. This publication presents solutions for displaying the scheduler's past and present workloads well as an accounting for the availability and usage at the system and compute node level for application availability.**

NCSA has developed the Nodestats web based utility to evaluate the scheduler's perspective of node availability and job eligibility. This interfaced was motivated by a constant lack of insight into the reason queued workload was not being executed and required a laborious process to understand. This process was additionally impractical due to the lack of state data for investigating behavior for past workloads. Job dependency data and intentional draining of compute resources for prioritized workloads are not readily visible to end users and system managers. In addition, separating nodes draining for future reservations from large fragments that do not have appropriated queued workload is not readily presented. This type of analysis was found crucial in separating the impact of intended policy from workload mismatches for idled compute resources.

I. INTRODUCTION

The Blue Waters supercomputer is a large scale Cray system focused on sustained application performance at scale. Its compute engine incorporates 22640 Cray XE6 nodes and 4224 Cray XK7 nodes. In operating a large capability machine, it is important to allocate the resources in accordance with the goals for the scientific research teams, and to account for the usage accordingly. The Blue Waters must develop policy that provides appropriate turnaround time for jobs while supporting a variety of large differing workloads. Many of the desired criteria conflict with maximal utilization, therefore, it is important to attribute and quantify the impacts of policy decisions and separate them from workload and system characteristics. This has led to a series of tools and methods to work in conjunction with the scheduler and resource manager environments to analyze system utilization and job turnaround. In addition to this activity we have discovered that there are many potential sources of inaccuracy in how we account for the utilization of the system.

II. ANALYZING ACCOUNTING ACCURACY

A. *Moab torqe/alps issues*

The scheduling and resource management environment is comprised of Moab and Torque from Adaptive Computing layered on top of Cray's Application Level Placement Scheduler (ALPS). See figure 1. For real time handling of allocations and accounting, we have utilized Moab's allocation manager interface to communicate directly with our existing allocations database. This allows us to process allocation transactions at job submission, start and end events.

Figure 1. Moab Torque Alps Integration.

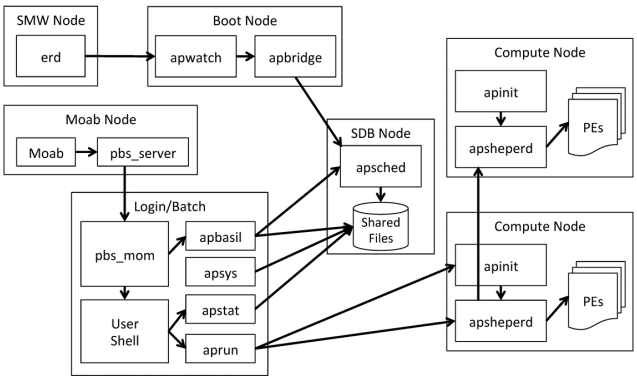


Image reprinted from "Production Experiences with the Cray-Enabled TORQUE Resource Manager" by Matt Ezell and Don Maxwell

It was soon discovered that our charging differed from user experience in many cases. The initial challenge was dueto the interface between the scheduler and allocation manager contains no fault tolerance capabilities. The packets traverse the Cray High Speed Network (HSN) to communicate with the allocation manager, and dropped packets and failed connection attempts appear to be common in a system our size. Therefore, many charge events were simply lost, and the charges would need to be reconciled with the Torque accounting records. In back loading Torque accounting records, it was determined that failure modes in the resource manager can lead to missing accounting records there as well. In addition, in nearly all cases there is variation in the timing reported by Torque compared to the successful Moab records, and that care must be taken in accounting for preempted jobs. Finally,

both sources of timing data potentially differ from the user's timestamps of output files. A fourth source of timing was necessary from ALPS to understand and identify inaccuracies due to system or software failures. The layers of potential timing overheads are detailed in Figure 2.

Figure 2. Timing variations by Layer

B. *Integrating Timing Data*

While user perception of job finish times are not immediately available, the remaining sources are, but in separate logs streams. Blue Waters utilizes a central database to integrate all log and event streams from the system in a facility named The Integrated System Console (ISC). This enables us to create single entries for each job in the database that incorporate the timing from ALPS, Torque and Moab and to generate alerts when timing skews exceed thresholds. This allows us to identify cases where attempted job cancels fail to result in immediate job termination via Alps. Future work includes the plan to take job and accounting actions based on system failure modes. These would include enhancements such as automatic job walltime increases during filesystem recoveries as well as credit to account balances for periods of reduced capability do to system failures.

III. NODE STATE ACCOUNTING

A. *Availability of Nodes*

As part of the overhead described in the previous section, there is potential lost node availability for job execution as part of health checking at the completion of jobs with failure codes. It was our desire to quantify the availability loss due to each system state.

Cray's XTAdmin database on the Cray Service Database (SDB) houses the processor table that stores the current state of the compute and service nodes. When things change in the system, those changes of state are reflected in this table. When various Cray tools need to know or modify the current state, they update the processor table. This table is inherently the de-facto authority of system state. In efforts to account for aggregate node time spent in the various states (up, down, suspect, admindown, etc.), this table is looked to for current status. Periodic sampling of this data can give a simplistic, albeit imperfect, accounting of time spent in these states. However, since the data is housed in MySQL, we are able to construct triggers to log in a separate table when a node changes an attribute in the processor table. This differential data collection method also allows us to store a much smaller amount of data than a periodic snapshot of the entire Blue Waters compute and service node environment. We specifically track two fields: status and mode. We track both the state it changed from and to for better data quality in case of data loss during transmission. We also have to track any change in either, as the two are mutually exclusive of each other.

Generating reports from this data involves querying the changelog for each node individually and sorting their change records by time. Then the list of changes must be traversed and time spent in each state accumulated. This set of times is then aggregated across all nodes and an actual node hours number can be determined for each state. We also provide scripts to manage the potentially unbound growth of the changelog table by migrating it to another home for long term storage and analysis.

B. *Mean Time To Interrupt*

This solution has been implemented on our test system, and upon completion on the Blue Waters system, we intend to use this data determine the operational availability of the compute nodes and to be able to account for down states and time spent in node health checking separately. Finally, we will be able to generate MTTI data for software interrupts and hardware faults on a per node basis.

IV. POLICY ANALYSIS

A. *Nodestats*

An early goal in Blue Waters' operation was to validate that the system was being utilized effectively. Reports were received that the Blue Waters User Portal frequently showed low utilization, while many jobs where waiting to be run. This generated concern that there was either something wrong with the system scheduler, or that our job scheduling policies were excessively contributing to the idle time. With a machine this size, and the mixed architecture of XE/XK nodes, it can be difficult to diagnose and easily explain previous states of the workloads and priorities when investigating past events. The information exists in logs, but this leads to large data challenges due to logs containing hundreds of gigabytes of messages per day. A tool was needed to better visualize the behavior and state of scheduler at a given point in time to give a clear view of the backlog of work compared to the drain of the system.

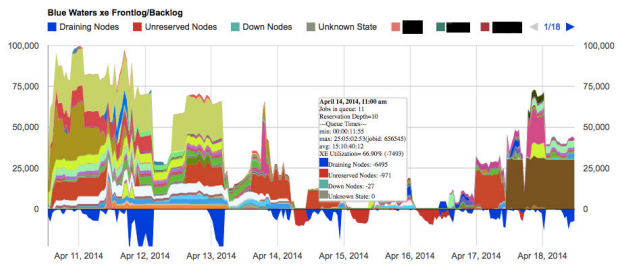
The first step was to gather more data. Various iteration times were tested, but a 5-minute iteration was settled upon. A cron job was created on the internal login node that takes an xml-snapshot of showq every 5 minutes. A list of jobs and the node requirements for each is stored in a database, as well as whether the job is eligible to run or not. We call this the backlog. Also at this time a snapshot of both "xtprocadmin" and "showres –f" is taken. The "showres –f" dumps a list of all free nodes, and when the next reservation

on the node begins. Some light processing is done on these two lists separating them into XE and XK nodes, and also breaking them down into four categories; Busy, Draining, Idle, and Down. This helps show a difference between the purely Idle nodes, with no future reservations; and nodes that are Idle, but are currently being held so a large job can run. We call this portion of data the frontlog.

By splitting the system into XK and XE, we have a backlog of jobs that require XK nodes separate which does not present itself in the XE graph this removes any confusion on jobs being incorrectly held. The portal feed was rewritten to use the data from this tool to create a more accurate state of jobs on the system. Prior to this, all jobs were represented in the portal as a single pool for the system. Also, by grabbing the snapshot every 5 minutes, jobs in a held state are no longer included in the list of jobs waiting. Prior to these changes, all jobs, even held jobs, were considered idle. This increased accuracy of the user portal greatly.

With the data now gathered in an easy to parse and query format, a web frontend was developed to better present the state of the system for analysis. Taking these datapoints, we use the Google Chart API to generate graph of the system over time separated between the xk and xe node types. Backlog is plotted in the +y axis, and frontlog is plotted around the -y axis. The different states of the nodes are color-coded for easy visual processing: Down nodes show up as light blue, draining nodes are dark blue, idle nodes are red. Information at a given point is provided by mouse-over hover. Figure 3

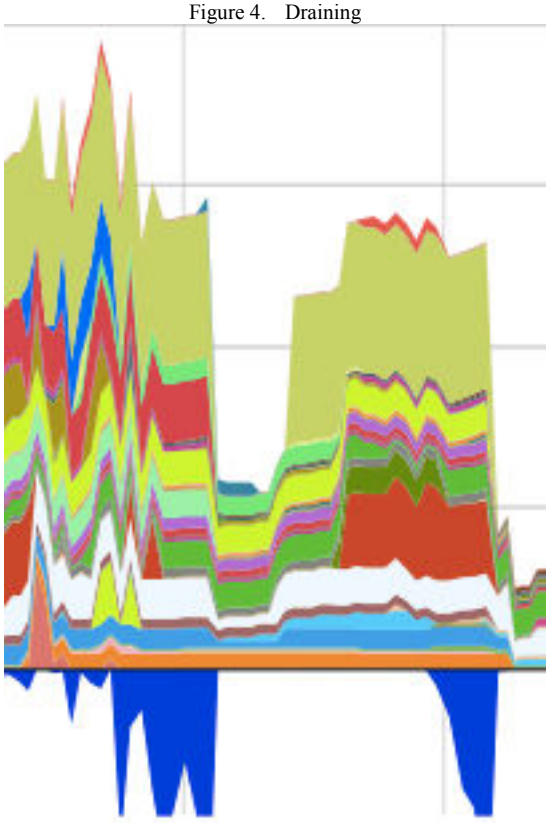
Figure 3. NodeStats display



In the ideal case, there will be nothing on the -y axis, as all nodes will be busy. Light blue should not be heavily represented except during maintenance. Dark blue means there's workload scheduled on the nodes within our max-walltime hours into the future. Anything scheduled past our max walltime into the future, and the node is classified as red for idle. With these charts in hand, it became trivial for the admin team to determine when there's a problem with the system or a problem with the workload.

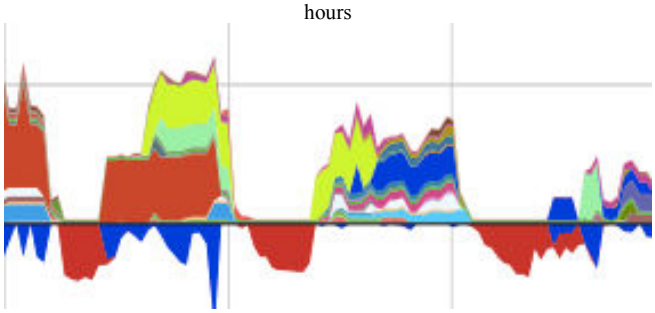
The scheduling policy for Blue Waters is to prefer large jobs, which will causes more draining than similar systems that prefer to maximize system utilization by maximizing job placement. This frequently causes a large dark blue frontlog as shown in the chart, but it also shows that small jobs are backfilling, as the backlog is shrinking even with the large blue drain. For these small jobs to be scheduled, a proper walltime for placement within these drain periods must be specified. “showbf” is helpful for figuring out the job requirements that are available. Large areas of drains denoted by the blue regions is not ideal for system utilization, but is not cause for alarm as long as we maintain this policy. See Figure 4

Figure 4. Draining



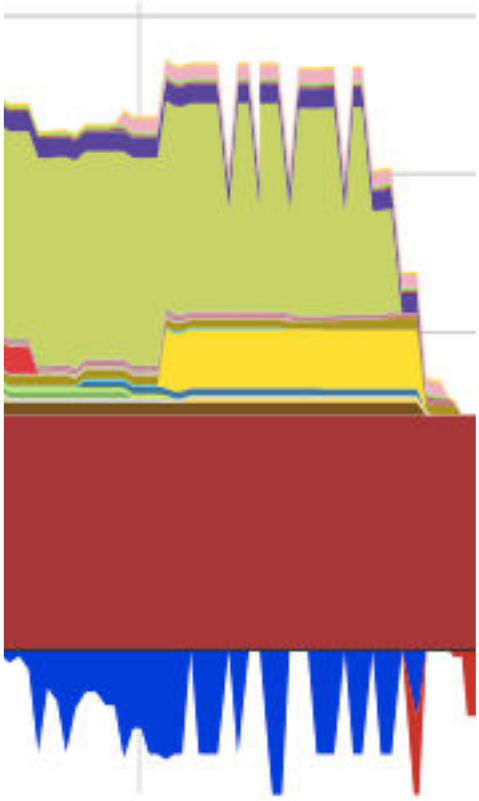
When we see a red frontlog; however, it does represent cause for concern. There are two cases where this will show up. An excessive red frontlog with no backlog means that there is not enough workload being submit to the system. This is a utilization problem outside our control. A red frontlog that shows up with sizeable backlog means there is something not allowing Moab to schedule a job at this time. This usually means a requested feature conflicts with the available nodes, but is worth checking the logs around that time period to see what was blocking the job. Transient small amounts of red are occasionally seen when the snapshot is taken when a node is free, but a Moab iteration hasn’t been completed yet.

Figure 5. Daily Cycle of undersubmitted workload during overnight hours



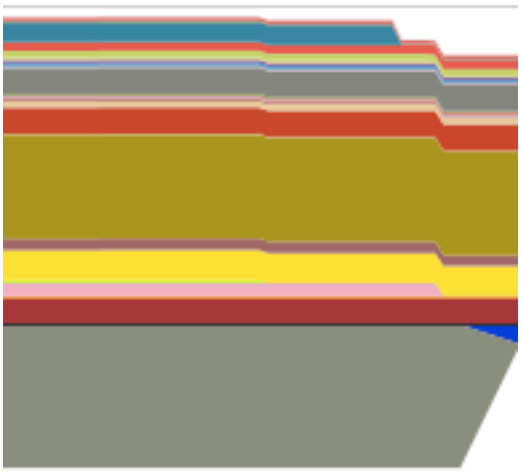
Other unplanned benefits have shown up on these graphs. Occasionally a sawtooth (see figure 6) pattern will show up. This generally means that there was a sliding job reservation, which can be cause for concern. Moab is trying to start a job, but is failing, and the reservation gets pushed back. This can block other jobs and needs to be remedied. Another problem that can be seen is when Moab is unresponsive. If there hasn't been any new data in two iterations, the graph is marked unknown. Investigation is needed to see why Moab is taking longer than 5 minutes to schedule a job.

Figure 6. Sawtooth due to sliding job



By giving us an overall visual view of the system's workload, this web tool allows us to easily answer concerns about the system utilization. It has given us an easy way to show exactly what was going on at various span time. An overall trend is needed to diagnose many of the issues with a system of this size. It can either show that the system is performing correctly, even if at a given point it looks like it might not be; or it can show that there is something wrong when it might look correct. This goes well beyond what is shown by the normal daily utilization numbers, or a single snapshot in time on a user portal. Seeing these trends allows for insight on whether a configuration change was valuable or not, by easily showing over time the impact is has made on the system. This can be either shown in the utilization, or a change in the number of scheduling issues seen in the graph.

Figure 7. Unavailable Node Representation



V. CONCLUSIONS

1) Regularly occuring failures result in accounting inaccuracies at ever level of resource management.

2) While no direct utility is available for accouning for the possible states of the compute nodes, the existing SDB database can be modified to make the node based stistics readily available

3) Seperating policy impacts on utilization from workload characteristics requires new tools and visualation methods to clarify effective system use.

ACKNOWLEDGMENT

I would like to acknowledge Michael Pitcher and Mark Dalton from Cray for their efforts in the accounting investigations. Also David and Scott Jackson for the expertise in allocation management interfaces.

REFERENCES

[1] Matt Ezell and Don Maxwell David Beer “Production Experiences with the Cray-Enabled TORQUE Resource Manager” Cray User’s Group, 2013

[2] Scott Jackson “Unifying Heterogeneous Cray Resources and Systems into an Intelligent Single-Scheduled Environment” cray Users Group 2009