Batch Scheduling on the Cray XT3

Chad Vizino, Nathan Stone, John Kochmar, J. Ray Scott
{vizino,nstone,kochmar,scott}@psc.edu
Pittsburgh Supercomputing Center

ABSTRACT: The Pittsburgh Supercomputing Center has implemented a custom batch scheduler, Simon, to operate with PBS on one of its massively parallel systems. The design and implementation of an early custom scheduler on the XT3 as well as a functional overview and adaptation of Simon to the XT3 will be discussed.

KEYWORDS: simon, scheduling, pbs, xt3, crms, harness, backfilling, easy.

1.0 Introduction

The Pittsburgh Supercomputing Center has implemented a batch scheduler, Simon, running under OpenPBS[1,3] on one of its massively parallel systems. The scheduler has been in production for the last four years and contains many advanced features. This scheduler is to be ported to the XT3.

We will give an overview of Simon, its design and features, and then discuss current scheduling activities on PSC's XT3, including a specially designed scheduling system, and the progress toward implementing Simon in the XT3 environment.

2.0 Simon

PSC's Terascale Computing System, LeMieux, is an HP AlphaServer SC cluster of 3000 Alpha EV68 processors spanning 750 nodes. Each node is an ES45 quad-processor server with 4 gigabytes of memory. Quadrics Elan comprises the interconnect technology, and the system is capable of achieving 6 teraflops.

The job scheduler, Simon[1], used on this system was custom designed by PSC Systems and Operations staff to work with the AlphaServer SC architecture and to allow us to run large jobs and achieve high overall system utilization. Simon is designed to work under OpenPBS and to function with the AlphaServer SC resource management layer, RMS, and its companion system database. Although a user may request an arbitrary number of processors for his/her job, the job is scheduled to whole nodes so that each node may provide exclusive access to its resources.

Simon has been in use for the last four years that LeMieux has been in production and is written almost entirely in TCL[4], which allows for easy development and adaptation of new functions. As we have gained experience with LeMieux, site policies have been created and extra features have been added to Simon to provide additional capabilities to further enhance overall system responsiveness to user demand while maintaining utilization

2.1 Simon Advanced Features

Simon contains a number of advanced features. Several of these, including backfilling, reservations, co-scheduling, and defensive measures will be discussed.

2.1.1 Favoring Large Jobs

One of the main goals of Simon is to fairly run large jobs. A large job is one that will consume the majority of the machine's processors. Through experience in helping users to scale their codes and so that up to two large jobs may run on the machine at once, we have established a large job to be one requiring at least 1024 processors (see Table 1). We want to run as many large jobs as possible as soon after they are queued as possible, and not allow any one user to dominate the machine, filling in the remainder of the machine with smaller jobs.

The basic strategy of Simon is to completely order all queued jobs and work through this unified queue, top to bottom, starting as many jobs in one scheduling cycle as can fit. Jobs have two required attributes: number of

[1] "Simon" is named for Dr. Herbert Simon (1916-2001), University Professor of Computer Science and Psychology at Carnegie Mellon University, and winner of the 1978 Nobel Prize in Economics. Simon argued that inevitable limits on knowledge and analytical ability force people to choose the first option that "satisfices" or is good enough for them. Scheduling a large computing system often requires making choices with limited knowledge.

processors and duration (wall clock time). The set of queued jobs is ordered into two subsets, or bins, by number of processors requested: An upper bin (>=1024 processors) and a lower bin (< 1024 processors). Upper bin jobs are ordered FIFO while lower bin jobs are ordered in decreasing size by number of processors requested.

Table 1 LeMieux jobs >= 1024 processors.

Year	Percent processor*hours
2002	6.5
2003	28.6
2004	46.7

Using reservations and allowing backfilling, we are able to keep a continuous stream of large (or top priority) jobs running on the system without having to regularly empty the machine to start large jobs. LeMieux can run at most two simultaneous large jobs. The list of large jobs is kept in a FIFO ordered list in such a way as to prevent one user from dominating the machine.

2.1.2 General Reservations

Simon supports chargeable reservations by allowing a queue to be bound to a specific set of processors for a specific duration starting at a specific time. These reservations are generally used to foster parallel development and code scaling and for special, dedicated runs. Unused time within the reservation is billed to the requesting user's account.

2.1.3 Co-Scheduling

Two special purpose clusters, one for visualization services and one for remote networking services, interoperate with LeMieux. Simon has been implemented to co-schedule jobs with these clusters by providing additional internal PBS resources and interaction with the system RMS database.

2.1.4 Defensive Measures

An important feature of Simon is its pre-job scan module (the pre-scanner.) The pre-scanner checks a number of features and services expected to be present on the processors to be assigned to a job. Specifically, it checks the following:

- Are file systems mounted and accessible?
- Are the processors busy with left over processes from a previous run?
- Are the processors busy with system daemon processes that would interfere with computation?
- Are the processors available and performing correctly?
- Is the system interconnect functioning properly?

Failure of any of these conditions causes the pre-scanner to remove these processors from further scheduling and causes Simon to pick a new set of processors for a job.

3.0 XT3 at PSC

In October 2004, PSC received its initial XT3 system. This single cabinet system was demonstrated at PSC's booth at Supercomputing 2004 running a complement of applications. Since then, the XT3 system has been expanded to 10 cabinets in December 2004 with additional cabinets added in February 2005 for a total of 22 cabinets containing over 2000 compute processors.

Our goal is to have an implementation of Simon running on our XT3 system. Progression to this goal is not complete and has happened in a number of smaller steps and these will be discussed next.

3.1 XT3 Resource Manager Challenges

Under the Cray Resource Management System (CRMS), the compute processor allocator (CPA) which functions as the XT3 resource manager layer, has been designed to provide a centralized means to allocate processors to either interactive or batch sessions or both, keeping its state in a system database (the SDB). PBS Pro has been adapted to function with CPA and provides basic scheduling capabilities around CPA.

PSC's initial XT3 ran with an early (pre-CRMS) version of the XT3 software suite, called Dev Harness, which did not contain key CRMS features, namely CPA, the SDB and PBS Pro. These missing features required us to design and implement replacement functions until CRMS was available. Problematic under Dev Harness was the requirement that users needed to interactively address specific nodes using node identifiers, or nids, when running applications. Since no centralized mechanism was present to allocate nids, and manage batch jobs, a user was left to guess which nids were free and then address them using yod (the XT3 application launcher) hoping that he/she did not collide with those nids requested by colleagues. This was unacceptable in our environment.

3.2 XT3 Early Solutions

To work around the limitations of the pre-release XT3 software, transitional replacements were designed and are summarized in Table 2. This table shows the software component comparison between LeMieux and the phases of the XT3 Dev Harness and CRMS Simon integration.

Table 2 – PSC Phases in Simon Implementation on XT3

	LeMieux	**XT3 Initial**	**XT3 Transitional**	**XT3 Final**
Date	Complete	October 2004	April 2005	Planned
Software Suite	Alpha-Server SC	Dev Harness	CRMS	CRMS
Resource Mgr.	RMS	Custom	Custom/CPA	CPA
Database	mSQL	Flat Files	Files/MySQL	MySQL
Batch System	OpenPBS	Torque	Torque	PBS Pro
Scheduler	Simon	Custom	Custom	Simon
App. Launch	prun	pbsyod	pbsyod	yod

3.2.1 Batch system

Since no batch package existed under Dev Harness, we selected Torque[5] for several reasons. Torque is, at its core OpenPBS carried forward with community support and is Open Source. At the PSC, we had developed quite a bit of experience with OpenPBS and were comfortable using it and designing custom schedulers to work with it. Torque also provided a logical path to using PBS Pro under the final machine configuration.

Several modifications were made to the Torque source code to accommodate it under Dev Harness. A “size” resource, similar to the one to be provided under CRMS and PBS Pro was added to allow a user to specify the number of compute nodes and a “nid_list” resource was added to record which nodes the job was using. The qstat command was altered to properly show the “size” resource for each job. Finally, a “nidmask” resource was added that allowed users provide a mask to guide the scheduler in picking (or not picking) certain node ids.

3.2.2 SDB and CPA Replacements

No system database (SDB) and no compute processor allocator (CPA) existed under Dev Harness, and so replacements were designed for them. For the SDB replacement, flat files were used to hold processor state and were managed by the batch scheduler also functioning as the CPA replacement. The application launcher, yod, was wrapped with a script called pbsyod which translated pbsyod to “yod –list <nidlist>” with <nidlist> set to a specific processor list passed in through an environment variable, YOD_NIDLIST, set by the batch scheduler. This variable provided an interface between the system and user layers (see Figure 1). This setup also provided users with a way to avoid using messy processor id lists. For example, “yod –list 10..20,30..60 myapp” became “pbsyod myapp”.

To query the state of the system, a new command, “shownids”, was designed to query the current available node set, report availability and show batch job assignment to nids.

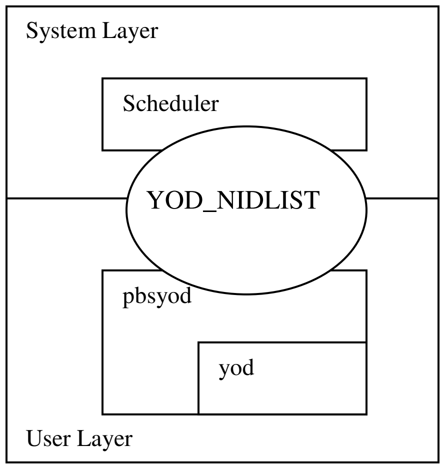

Figure 1 – System—User Interface under Dev Harness

3.2.3 Early XT3 Scheduler

The last missing piece under the Dev Harness system was a scheduler to tie the others together. Under Dev Harness the scheduler functioned not only as a decision maker for selecting compute nodes under Torque but also as a resource manager managing the allocation process of recording allocation state in flat files.

Several interesting features were added to this initial scheduler to adapt it to the early environment. The scheduler was coded in TCL for easy development of new modules. TCL is one of three supported languages under Torque and is the only script-based one. Simon’s scheduler was written almost entirely in TCL and so this had a proven record of working well for us.

Important to PSC was the ability to drain the machine for maintenance along with fairly scheduling jobs of all sizes in a FIFO ordering. Target drains were implemented so that we could drain the machine for a specific time. We also adopted the EASY[2] scheduling algorithm, which set a reservation for the top job in queue, permitting other jobs to jump ahead (backfill) in the queue as long as they did not delay the start of the top job in queue.

As the machine configuration evolved, we ran the machine as several Dev Harness systems and set up Torque and the scheduler to dispatch jobs to these smaller systems.

Booting was a fairly frequent activity, so the scheduler was designed to recognize boot states and not assign jobs to a booting Dev Harness system. The scheduler was even allowed to control the booting of these Dev Harness systems so that each could be rebooted after a job ran on them.

As previously mentioned, a "nidmask" resource was added. Users could specify a nidmask to guide the scheduler to select or avoid certain ranges of compute nodes. This helped users navigate around certain parts of the machine as problems were encountered between boots.

One of the more important features added to the early, custom scheduler was the ability to check node responsiveness in advance of a job running on a set of nodes. As discussed previously, on PSC's LeMieux system, Simon provided an extensive pre-job scan or pre-scan before each job that was very successful in avoiding nodes with problems. Using the ping_node utility, Torque performed a ping_node check to each node in the job using a job prologue script. Any failures caused the job to fail to start (put back in queue) with the failing nodes put into a "bad" list recognized by the scheduler.

To facilitate much faster checking of nodes, we developed a new Portals-based utility called "ping_list" that overcame some of the limitations of ping_node. With ping_node it was possible to check only one node at a time with a failed check costing about 5 seconds. This was unacceptable as it often took an hour or more to check all nids in the machine when many were failing. Ping_list, however, proved to be much faster. It could easily scan the entire system in about 1 second, providing lists of nids that passed and ones that failed. It also contained a feature to show when the check hung, providing an indicator utilized by the scheduler to initiate automatic reboots of some or all Dev Harness systems.

The "bad" list was viewable using "shownids" and the scheduler avoided assigning nodes from this list. The list was cleared at boot time since most problems were transient and cleared with a boot.

3.2.4 Integration with CRMS

In April 2005, the machine was transitioned to CRMS and was combined into one system of 22 cabinets. With CRMS CPA, the SDB and PBS Pro were available and so we transitioned our custom components to use part of this environment while we investigated complete integration.

As a transitional step and since the SDB was present under CRMS, the scheduler running under Torque was changed to synchronize its flat files with the SDB. Even though CPA was present, it was bypassed by using the YOD_STANDALONE environment variable, allowing the previous pbsyod mechanism to continue to work.

Pre-job scanning was maintained and continued to help provide success in starting applications on sets of operational compute nodes.

3.3 Future

While the transitional environment has functioned well under CRMS, we will be evaluating changes to PBS Pro to accommodate a TCL scheduler and enhance PBS Pro to function with CPA in a way that will allow the batch system to query available nids and pass selected nids on to the CPA through the batch system. We plan to sustain the location of the node allocation decision making process within the scheduler which will provide useful features to us in the future.. When experimentation is complete, we will transition to using PBS Pro with a custom scheduler and will plan to integrate Simon more completely.

Having the allocation decision making process reside in the scheduler will allow us to investigate node allocation algorithms by easily replacing allocation modules within the scheduler, possibly even providing users the ability to provide hints to the scheduler in helping it to select certain nids optimal to the application.

As the system matures, we will begin to expand the number of mom (execution agent) daemons which will help to provide load balancing among the login nodes.

We also plan to investigate alternative scheduling languages, possibly Python, to help us better manage and develop new scheduler code.

Future directions also include, co-scheduling service nodes such as visualization and data handling ones.

4.0 Conclusion

Having a custom batch scheduler has allowed us to implement our site scheduling policies within the scheduler and provide flexibility in policy change. Using a custom scheduler, Simon, on our AlphaServer SC system has been successful in accomplishing our scheduling objectives and we will begin porting features from it to our Cray XT3 machine environment.

We have met initial challenges with early XT3 software limitations by implementing our own customized batch solutions which have helped us to move forward with application development and benchmarking.

Ongoing investigation and testing will help us to fully implement our successful Simon scheduler in the XT3 environment.

5.0 References

[1] B. Henderson, "Job Scheduling Under the Portable Batch System." Proceedings of the Job Scheduling Strategies for Parallel Processing, Santa Barbara, CA April 1995.

[2] D. Lifka, The ANL/IBM SP Scheduling System. Proceedings of the Job Scheduling Strategies for Parallel Processing, Santa Barbara, CA April 1995.

[3] Portable Batch System: http://www.openpbs.org/.

[4] Tool Control Language: http://www.tcl.tk/.

[5] Torque Resource Manager: http://www.clusterresources.com/products/torque/.

About the Authors

Chad Vizino is a member of the Systems and Operations group currently working on scheduling systems, resource management and accounting. Nathan Stone is Senior Research Analyst and is responsible for software infrastructures for data management and administrative scalability issues. John Kochmar is High Performance Systems and Operations Manager. Ray Scott is Director, Systems and Operations, in charge of all computing platforms at the PSC. They can be reached at PSC, 4400 5th Ave., Pittsburgh, PA 15213. Phone: 412-268-4960. E-mail: {vizino,nstone,kochmar,scott}@psc.edu.