Selecting and Implementing the PBS Scheduler on an SGI Onyx 2/Orgin 2000

Sandra Bittner
Mathematics and Computer Science Division
Argonne National Laboratory

In the Mathematics and Computer Science Division at Argonne, the demand for resources on the Onyx 2 exceeds the resources available for consumption. To distribute these scarce resources effectively, we need a scheduling and resource management package with multiple capabilities. In particular, it must accept standard interactive user logins, allow batch jobs, backfill the system based on available resources, and permit system activities such as accounting to proceed without interruption. The package must include a mechanism to treat the graphic pipes as a schedulable resource. Also required is the ability to create advance reservations, offer dedicated system modes for large resource runs and benchmarking, and track the resources consumed for each job run. Furthermore, our users want to be able to obtain repeatable timing results on job runs. And, of course, package costs must be carefully considered.

We explored several options, including NQE and various third-party products, before settling on the PBS scheduler.

NQE – Our Initial Choice

We first purchased NQE, a scheduling system found on many CRAY systems and adapted for use on the Onyx 2 systems by SGI. Our experience with the package revealed several major drawbacks. The package had little supporting documentation, and many of the supporting files required to use the package were not included with the distribution. I created the contents of each file by guessing at what information and syntax was needed in configuration files from the errors output by the system. I also attempted to move from NQE 3.2 to NQE 3.3 by downloading the release from the CRAY Web site, only to discover that the binaries available from the site were corrupt. I wrestled with the installation until the product was discontinued.

Third-Party Products

I next considered several third-party products that were similar in function to NQE, including LSF, Miser, Fair Share/Share II, the Maui scheduler, and PBS.

LSF. The product LSF, available from Platform Computing Corporation, provides scheduling capabilities under IRIX and is the product endorsed by SGI to replace NQE on the IRIX platform. While LSF is known in the IBM SP world, it suffers significantly from a lack of OS-level support in IRIX for resource management. The kernel-level hooks needed to obtain critical system information still must be developed. Moreover,

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

LSF is expensive; and the base package does not include the supporting products, such as the analyzer and parallel tools, that would allow the use of message-passing tools such as MPI. In fact, when we considered purchasing the entire product suite plus maintenance, the cost was approximately equal to purchasing an additional system. The lack of kernel support coupled with the high costs reinforced our decision to pursue other options.

Miser. The SGI Miser product was of interest because it was included in our software collection for the system. The tool provides kernel-level structures to access system information. The product was poorly documented, however. Moreover, several features failed to operate reliably, often crashing the system. The latest release does fix many problems, but reliability is still questionable. As the tool matures, we hope to use the additional kernel constructs to extract critical system information for use with our selected scheduling tool. At the moment, Miser is not yet ready for production systems.

Share II. The Share II scheduler (previously called the Fair Share scheduler) from Softway Systems seemed like it would meet our needs because it uses a model that divides resources by allocating each group a percentage of the total resources available. This resource division worked but presented two fundamental problems for our environment: (1) jobs run using the fair share mechanism did not receive repeatable timing results, and (2) the package did not scale to large resource requests. Specifically, it was not possible on a busy system to obtain a significant portion of the resources for a single job without manually interfering and setting the system resource allotment to zero for other individuals. Moreover, since the product was not included in the standard software distribution with our SGI system, purchase of this package would require additional funds. For these reasons we did not pursue Share II for use at our site.

Maui scheduler. The Maui scheduler was a logical system to consider because we use it on our IBM SP system. However, although work is under way to migrate the scheduler to IRIX, the product is not yet in a releasable form.

PBS. The final package examined was PBS version 2.0+. PBS is available free of charge, includes source code, and provides useful documentation (a users guide is not included but is due out in the summer of 1999). The installation directories used spanned many directories on the system, and while the configuration appeared reasonable, it conflicted with our local software installation model. It is a credit therefore to the PBS developers that we were able to modify the installation scripts fairly easily and were able to install the software package as one self-contained unit in a single file system. (I did uncover multiple bugs, but each was promptly fixed.) The remainder of this paper focuses on PBS because this is the system we settled upon and currently are running.

Experiences with PBS

My work with PBS included evaluation of different schedulers and modifications for improving interactive login capabilities.

Schedulers

PBS offers several scheduling systems:

- *TCL.* I experienced some initial difficulties in installing TCL. For example, I had to upgrade our versions of TCL, TCLX, and TK to version 8.0.4. In addition, it proved necessary to install each package in the same location, a fact that the guide failed to indicate. The TCL scheduler provided two sample scheduling methods:
  1. System load average.
  2. Cluster scheduling of jobs using fair share (parallel job support is not provided).

  Neither method specifically considered processor or memory availability. The routines necessary to modify the sample to meet our needs exceeded the modifications necessary to modify the C scheduler.

- *C.* The default scheduler, called C (or "FIFO"), was quite powerful. It was sufficient for evaluating the PBS package and, when modified to address resource balancing, worked for production testing. The C scheduler as provided supports seven scheduling methods that may be specified for standalone, prime time, or nonprime time hours:
  1. Decaying fair share algorithm.
  2. Round robin for jobs and queues.
  3. Queue priority.
  4. Job type or job ordering.
  5. Strict FIFO (First In First Out).
  6. Load balancing jobs between hosts.
  7. Dedicated times/modes.

- *BASL.* To avoid having to learn another scripting language, I did not use the BASL scheduler. The BASL scheduler provided five sample scheduling methods:
  1. System load average check and release of one job per cycle.
  2. System load average check and release of two jobs per cycle.
  3. System load average check and release of three jobs per cycle.
  4. System load average check and time processor (cput).
  5. Best-fit algorithm similar to early schedulers such as NQS.

Interactive Login

Following the administration guide, I next established the necessary queues and began experimenting with the operation of PBS. My attempts to limit the resources in the queues by setting minimums, maximums, and default values proved insufficient, but through e-mail to the PBS staff I was guided to an item called *resources_available* that must be set to control resources. This worked. I was successful in performing batch processing and interactive use through batch queues. PBS worked reliably and as advertised.

PBS works quite well at handling batch processing. Our users, however, require interactive logins to the system. PBS does offer a partial solution to this: One can issue a "qsub –I" and obtain an interactive batch job that is extremely close to a true interactive session. I considered rewriting the standard login sequence to feed interactive logins to an interactive queue, but such an approach has the disadvantage of having to rewrite the login sequence every time an operating system upgrade occurs. Since upgrades are a quarterly occurrence, I decided to find an alternative.

Collaborating with the author of the scheduler and our knowledgeable onsite SGI representative, I made minor modifications so that PBS could use our supplied programs to obtain resource information about the usage level and therefore the availability of items such as memory and the number of processors on the system. We subtracted for known problematic resources and system overhead, to prevent the system from becoming overloaded and thus crash, and fed this information into PBS. Each queue then looked at its own limits and adjusted what it could allocate for the next job based on the numbers it had received. We had to use gross approximations because the kernel structures to obtain this information are significantly limited at this time. This has been noted by SGI, and development is under way to produce a robust resource ABI for the operating system/kernel to obtain such information more easily.

Admittedly, our current method is not without problems. First, using a single moment in time to influence the next moment in time is not strictly accurate. Second, resource collision, while less likely, can still occur. Third, some parties may still choose to take advantage of the mechanism, to the detriment of other users; therefore it is important to retain the authority to rein in abusive users by first requesting compliance and then forcing compliance through suspension of privileges. Fourth, it is still necessary to use outside measures and utilities to fill in for offerings not available within PBS; one example is scripts to force the draining of resources prior to an advance reservation.

The advantage of this method, nevertheless, outweighs the disadvantages. Retaining interactive logins allows development to proceed with standard tools in their current form without restricting base code editing. This frees our scientists to focus on the tools at hand and allows users to concentrate on the documentation without expending effort to force their problem to fit the batch system. Once they have determined what is needed, they can formulate the problem as a bounded problem and submit it to the batch system. PBS supports execution queues, advance scheduling, dedicated system modes, and guarantee of resource availability.

Comparison of the Schedulers

Each package I considered had a high learning curve. In each, dynamic resource allocation was never considered. In addition, none of the packages examined could schedule the graphics pipes on our system.

Many of the schedulers expected to be the only resource manager and the only controlling process on the system. This is because the model they supported is leftover

from the mainframe days, when large systems did batch processing and supported limited interactive work. Batch processing forces the user to guess at what resources a job may require through time. Interactive use of a system does not require the user to predict the resources required. Batch processing is an easier problem to solve because it is constrained, limited, or bounded and therefore predictable. Historically, interactive use was performed on smaller front-end systems and jobs were submitted to the larger back end system for processing. Our enhanced version of PBS supports the new model that allows interactive and batch processing to coexist on systems. This is particularly important because individual systems can now contain up to 256 processors.

Future Work

We have adopted the PBS system and have, at the same time, formulated a plan to address our additional needs for a fully effective scheduler and resource management system. During our development efforts, system usage will continue, with our users running jobs in weightless mode (using the command "npri –w"). Weightless mode in IRIX is defined in the manual page as follows: "A weightless process executes at a priority strictly less than any other process in the system." This is not the most effective or most efficient way to perform calculations, but we believe it is a reasonable approach to be taken while development efforts proceed.

Two hurdles remain:

- *Integration of graphics pipes.* Graphics pipes are critical to successful scientific visualizations and are key components in immersive environments such as CAVEs and ImmersaDesks. It is therefore essential to develop the mechanisms necessary to integrate graphic pipes as schedulable resources. An important part of our future development effort will require resolving the existing hardware/software problems. Specifically, graphics pipes have a tendency to hang; and clearing a hung graphics pipe often requires the system to be rebooted or power cycled – a method that is disruptive to all other activities happening on the system, particularly computation and development work. Currently, we have isolated the IR pipes from the remainder of the system, thereby allowing important graphics development to proceed while the remainder of the system is devoted to computation and code development without interruption. In the long term, however, we will need to address this issue with SGI.

- *ABI/hooks and resource management tools.* We await the development of the ABI/hooks needed to effectively exploit the resources of our Onyx 2 from an operating system and kernel level.

Summary and Concluding Remarks

I examined multiple packages for scheduling and resource management for our Onyx 2; these included NQE, LSF, Miser, Maui, Fair Share/Share II, and PBS. The most versatile and adaptable product available was PBS version 2.0+. The documentation and availability of source code in combination with knowledgeable developers made it

possible to modify PBS version 2.0+ to balance interactive jobs, batch, and system activities all within one system.

Initial testing of the modified PBS scheduling system has been successful, but only releasing the system to our user community will truly show its effectiveness. I hope to deliver a positive report by the next conference.

We will continue to partner with SGI and PBS to deliver more effective resource mechanisms. For example, to move further along the resource management curve will require SGI to finish development of an open ABI for resource management. The modifications we made to PBS should show up in future versions of PBS. Unfortunately, no package addressed treating the graphics pipes as a schedulable resource, and thus that remains an open issue. We will work directly with SGI to address this in the future. Finally, I note that PBS is already positioned to address scheduling on LINUX clusters. Therefore, it will be possible to leverage our PBS investment and transfer the knowledge gained to meet the scheduling needs on our LINUX cluster now and in the future.

Acknowledgments

I thank the following individuals and organizations: Robert L. Henderson, MRJ Technology Solutions; Bhroam A. Mann, NASA Ames Research Center; Jens Petersohn, NASA Ames Research Center; Joe Boyd, Daryl Coulthart, and Dan Higgins from SGI; Rohan Story, Aurema previously Softway Systems; Dinesh Kumar Kaushik, MCS staff, and the MCS Systems Group from Argonne National Laboratory Mathematics and Computer Science Division; Cray User's Group, CUG; Society of Women Engineers.

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.