Open Computing ‘‘Hands-On’’ Tutorial: April 1994

Batching in the Unix Environment

Batch processing is easily implemented on Unix with publicly available software that helps you spread the workload among several systems

By Steve Hanson

Batch processing is usually the farthest thing from a Unix user’s mind. People are usually drawn to Unix so they can avoid the old batch environments where users would queue up large processing jobs and come back two days later to see what had happened. Particularly in the workstation world, systems are often single-user machines on which batching may not make a lot of sense at first glance. However, batch processing is becoming more important in the Unix world than it has been in the past.

In the current frenzy to downsize from mainframes to open systems, users often want to regain some of the mainframe features they left behind, including batch processing. Also, the advent of inexpensive high-powered Unix machines makes them good candidates for doing batch processing, particularly if the batching can be distributed across a number of nodes in a network. We’ll look at some examples of batch processing on Unix systems, concentrating on examples for a freely available Unix batch system.

People with a stock Unix system often start up a batch job by using the `at` or `batch` commands. These commands allow a user to run a batch job either immediately or in the future. The job will run, and the results will be mailed back to the user. This method is less than satisfactory for several reasons. For one thing, it is often useful to queue up a whole bunch of jobs and have them run consecutively automatically. This way no individual job will overload the machine. Usually six jobs running sequentially will finish faster than six jobs running at the same time.

In modern environments where there may be many workstations in a workgroup, it would be useful to queue the job up to the workstation that is least busy--after all, if your office mate has gone out for a long lunch, it would be a shame to have that new zippy workstation just sit idle. Modern batching software will automatically dole out jobs to the machines that are least busy.

A true batching system also gives finer control over the batch jobs than do the traditional Unix tools. Using a batch control system, users can assign priorities to jobs, suspend them, kill them, or control them in other ways. In some cases, the software may even make a group of systems appear to be one powerful virtual machine by taking a job and splitting it up among the systems in the cluster.

Thus, the goals of a Unix batching system include some or all of the following:

- Forcing jobs in a queue to run sequentially.
- Distributing jobs between systems and balancing system load, as well as making fuller use of resources that may be idle.
- Giving users finer control over how their jobs run.

- Making jobs parallel across a cluster of machines.

A Sample Batching System

In this tutorial I will use the Distributed Queuing System (DQS) as an example of a batching system. DQS is publicly available and was developed at the Supercomputer Computations Research Institute at Florida State University by Tom Green and Jeff Snyder. It is a fairly simple batching system, but it is good to use as an example. And it compiles easily on many platforms, including IBM's (Armonk, N.Y.) AIX, Silicon Graphics Inc.'s (Mountain View, Calif.) Irix, Sun's (Mountain View, Calif.) SunOS, Convex Computer Corp.'s (Richardson, Texas) ConvexOS, and Cray Research Inc.'s (Eagan, Minn.) Unicos. It also supports batching in AFS, a distributed file system, authenticating the jobs at run time.

DQS runs on one system in the cluster as a ''qmaster'' machine. It keeps track of the different queues on the various systems and starts the jobs on remote machines. Typically each of the worker systems will have one queue (or possibly more) to which jobs are submitted. Within an individual queue, one job will be executed at a time, and the other jobs will wait for resources to become available. Normally each system will have one queue, which will force serialization of the jobs. However, a system that handles multiple jobs well--for example, a multiprocessor system--might be given more than one queue so that multiple jobs will execute simultaneously.

The system may be built so that jobs will be distributed among queues more or less sequentially or so that any incoming jobs will be assigned to the node that has the lowest load currently. This system generally assumes that nodes may be used for something other than batch processing--that is, that you may be dealing with nodes that are sitting on someone's desk, and the batching system is being used to ''soak up'' those excess CPU cycles that may be going to waste.

Normally jobs are not submitted to an individual queue but to a group, which will usually be a number of queues distributed across machines of the same architecture. Jobs are submitted with the `qsub` command, which specifies the name of the command file and the queue the job is being submitted to. The `qsub` command files are fairly normal Bourne shell scripts, with one exception. They have Bourne shell comments--those starting with ''#$''--that control the DQS system. The arguments after the ''#$'' are actually options that are fed to the `qsub` command and could just as well have been typed on the command line.

Listing 1 shows a sample command script for a job to be submitted to a batch group called ''sun'', a cluster of Sun workstations. This batch job recompiles the DQS batching system. It sends electronic mail at the beginning and end of the job and dumps the standard output and standard error output into a file named `batchjob.out`.

Listing 2A shows how to submit the batch script depicted in Listing 1. DQS also includes powerful commands for examining and manipulating queues. Once a job has been submitted with `qsub`, it may be examined with the `qstat` command, which recognizes several options. Listing 2B shows how the `-l` option is used with `qstat` to give more detailed information about the queue status. Note that two jobs were submitted to the ''sun'' group, and they were distributed automatically across both of the machines in the group.

The `qmod` command may be used by the owner of a queue--for example, the workstation owner--or a system administrator to suspend and restart a queue. Listing 2C demonstrates these functions.

Additionally, there is a `qdel` command, which may be used to delete a DQS job from a queue. It can only be used by the owner of the job, a system operator, or a DQS manager. So, to delete a job, a user would usually first run `qstat` to determine the request ID of the job he or she wants to delete, then use `qdel` to terminate the running job (Listing 2D).

The last command demonstrated in the listing is `qacct`. There is an accounting system built into DQS that tracks the system usage by job and by user. This facility makes it easy to track what your systems are being used for. Listing 2E shows `qacct -u` displaying the accounting history by user. The `qacct -j` *`jobname`* option provides a more detailed accounting record for a particular job.

The `qmon` program allows the user to do most of the features of all of the commands above while working in an X Window System graphical-user-interface environment. This tool uses Motif widgets and is nicely put together though the current version of `qmon` doesn't support all of the DQS features and is therefore currently lagging a little behind the rest of the DQS system.

There is also a `qidle` program, which can be run on a system that has a graphics display running X. This program monitors the X display and will automatically suspend the the queues on the machine when the X display is being used interactively. This method will keep the batch jobs from slowing down the responsiveness of X on the workstation but will allow the excess CPU cycles to be used when the workstation is idle.

Interactive Access to Queues

DQS includes a program called `qsh`, which allows users to submit commands to a queue interactively. It is somewhat analogous to the difference between using the shell interactively and executing a shell script.

A queue has several options that may be configured in DQS. These options will modify the behavior of the queue and limit the resources that a job may use. The queues are configured with the `qconf` command, which is also used to add and delete hosts and users from the system configuration. Some of the options to `qconf` are shown in Table 1. In general, the DQS stategy is to define access to the cluster at two levels: access to the cluster as a whole and access to individual queues in the cluster. Each cluster or queue may have access set to one of the following levels:

Free
: No restrictions on the users.

Open
: Access is permitted to the queue unless the access list for the queue contains the user's account name.

Restricted
: Access is denied unless the user's name is on the access list.

This access permission system is fairly flexible and allows policy to be set regarding which computing resources are available to which users.

In addition to configuring queues, hosts, and users, the editing features of `qconf` allow the administrator to change the characteristics of the system. Several features of the system may be configured either when the system is built or changed dynamically by using `qconf`. These features include the default

shell, minimum user ID and group ID values that are allowed to use the batching system, the number of jobs that an individual user can have running at once, whether jobs are re-runnable, and so forth. Several features of queues may also be configured at run time or by `qconf`. Queues may be defined as batch, interactive, or either. Only batches that are defined as interactive may be used with the `qsh` command. Overload levels may be defined for a queue: If the load on the queue node exceeds the overload level, the icon for the queue will turn red in the `qmon` program display.

As was mentioned previously, systems may be configured to assign jobs according to the system's load average. This feature may also be tinkered with somewhat by assigning load multipliers for each queue, so that some queues are more ''expensive'' than others. Therefore, these queues will be less likely to run jobs. This capability is a way of changing the priorities of different queues.

Queues may also be assigned ''nice'' values. Adding niceness to a process reduces its priority, so other processes will be more likely to be scheduled first. Therefore, a system may be given two different queues, one of which nices jobs down quite a lot (useful for long-running jobs) and another that doesn't nice down jobs (intended for quick, low-resource jobs).

So how do you keep your users from cheating? We all know computer users--if there's a high-speed queue and a low-speed queue, nobody will use the low-speed queue. After all, their thinking goes, my job is more important than all those other ones. Each queue may also be assigned limits on the amount of CPU time, core file size, total memory usage, and disk output each job may use. This way, you can set the limits low on queues that are intended to be express queues and set the limits higher on queues that are niced.

Other Batching Systems

There are several other systems for batching Unix jobs, some commercially available, others free. Note that some of these are simple batching systems or load-leveling systems, while others are more ambitious and aim to divide an individual job in parallel across a number of different systems, which may be the same architecture or different architectures. NQS is somewhat similar in flavor to DQS; it also supports the notion of using the batch queues for other functions, such as printing queues. Condor is a somewhat newer package that was written at the University of Wisconsin and is freely available in source form.

IBM Corporation has taken the Condor package and added features to it, including usability in an AFS environment and some more powerful batch job and queuing control mechanisms, and called the package Load Leveler. It is available commercially from IBM for RS/6000, Sun, and Silicon Graphics platforms. Load Leveler is pretty nicely implemented and is one of the packages we've been using at Fermilab for production batching. It has somewhat more sophisticated queue controls than DQS, although DQS has more features in the works.

Finally, Parallel Virtual Machine (PVM) is a more extended package that allows an application developer to treat an extended group of systems as if they were one very large virtual machine. Unlike the other packages in this article, PVM requires source modification of the C or Fortran program to call the PVM libraries. It is particularly useful for classes of programs that are easily split up into many parallel jobs, such as certain kinds of data analysis. It may also be used as a back end for the DQS package so that users may submit jobs through the normal DQS mechanism but have them run on many machines in parallel.

Which package should you use? A lot will depend on a site's feelings about publicly available software. The publicly available software is somewhat more of an adventure but brings with it the advantage of having source code available, either to fix bugs or add features. The commercial packages have the advantage of providing vendor support, although it's often the case that the absence of support for publicly available packages is better than the formal support of some commercial packages. Take a look at some of the public packages, gather up some of the information on the commercial packages, and decide what suits your needs best. And happy batching!

Some Sources for Batching and Load-Leveling Systems

DQS
: Available by anonymous FTP from ftp.scri.fsu.edu in directory /pub/DQS.

NQS
: The public domain version is available via anonymous FTP from ftp.cosmic.uga.edu in /pub/software/ directory. This version is described in the ''read me'' file as:

> NQS - NETWORK QUEUING SYSTEM, VERSION 2.0
>
> The Network Queuing System, NQS, is a versatile batch and device queuing facility for a single Unix computer or a group of networked computers. With the Unix operating system as a common interface, the user can invoke the NQS collection of user-space programs to move batch and device jobs freely around the different computer hardware tied into the network. NQS provides facilities for remote queuing, request routing, remote status, queue status controls, batch request resource quota limits, and remote output return.
>
> NQS is written in C language and has been successfully implemented on a variety of UNIX platforms, including Sun3 and Sun4 series computers, SGI IRIS computers running IRIX 3.3, DEC computers running ULTRIX 4.1, AMDAHL computers running UTS 1.3

: It requires quite a lot of work to get going. More recent releases are sold by Sterling Software (Palo Alto, Calif.) and several different computer manufacturers.

Condor
: Source available by anonymous FTP from ftp.cs.wisc.edu in the /condor directory. This package is difficult to build from source so binary versions of the latest release (5.5.4b) are available for DEC Alpha (OSF/1 v1.0), Sun sun4m (SunOS 4.1.3), and DEC MIPS (Ultrix 4.3b). Slightly older binary versions are available for IBM R6000 (AIX3.2) and Sun sun4c (SunOS 4.1.3).

Load Leveler
: Available from IBM Corporation (Armonk, N.Y.) for several different platforms, including RS/6000, Silicon Graphics Inc. (Mountain View, Calif.), and Sun Microsystems Inc. (Mountain View, Calif.).

Load Balancer
: This program is more a load-balancing system than batching system. It will let you balance log-in load across a number of systems and do some other nice tricks. Freedman Sharp and Associates Inc., Ste. 508, 1011 1st St. SW Calgary, Alberta Canada T2R 1J2; 403-264-4822 or via e-mail at

lb@fsa.ca.

Qmaster

GD Associates Ltd., 160 Bayview Dr., SW Calgary, Alberta Canada T2V 3N8; 403-281-6923.

Parallel Virtual Machine (PVM)

This program is a somewhat more complex virtual machine paralleling program. It can also be used in conjunction with DQS as a front end. This software is available from the ‘‘netlib’’ mail server at Oak Ridge National Laboratory. Send mail to netlib@ornl.gov containing the message body:

```
send index from pvm3??
```

to receive a listing of the files on PVM version 3 that may be obtained from the server.

Copyright © 1995 The McGraw-Hill Companies, Inc. All Rights Reserved.
Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com

Go To Content

Search

Last Modified: Thursday, 30-Nov-95 16:53:06 PST