SLAC-PUB-2892
MARCH 1982
(N)

CMS BATCH SYSTEM *

T.Y. Johnston

Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

March 8, 1982

When people talk about batch they have varying images (from a user being able to off-load some work that would tie up his/her terminal for some time to job networking and major production use). The Stanford Linear Accelerator Center (SLAC) is closer to the latter than the former. Three quarters of our VM cpu cycles are delivered to batch, and the other quarter also includes all service machines (including the batch monitor). With this heavy emphasis on batch one might ask why did we opt for VM versus MVS? This gets into another fundamental characteristic of our laboratory.

SLAC is a a research laboratory. This means that "production" programs change frequently. There is a continual interplay between development and production. Consequently, one wants to be able to do very similiar work interactively and in "production". Within the IBM world this argues for a system like VM with strong interactive capability.

In a perfect world computers would be infinitely fast and there would be no need for batch. In our imperfect world there is the need for users to be able to specify units of work that can be run in parallel (with user specified constraints existing within that work). In addition, in our imperfect world 200 users cannot simultaneously ask for 2 hours of computing and get the results back in 2 hours; so it is necessary to schedule and control the computing resource so as to equitably provide service to the user community. These constraints argue for the need for batch. The rest of this paper is devoted to showing how SLAC has approached the solution of these problems. It is broken down into several sections:

* Work supported by the Department of Energy, contract DE-AC03-76F00515

(Presented at the Annual SHARE Conference 1982, Los Angeles, CA., March 15-20, 1982)

As our base for going into VM batch we obtained from INTEL Corp. a batch monitor which had been developed for them. We have done a significant amount of work to that base to create our current system. We chose the INTEL system because it required no CP mods (we have done some); was not a modificaton to a modification to a modification; and had a very reasonable design which allowed significant extensions without major rework.

Scope (From a User Viewpoint)

A user sees batch as a set of services which should interact in a predictable and cohesive style. These include:

1. The ability to give work to the batch system.

2. The ability to find out about the status of any work that has been given to the batch system.

3. The ability to change one's mind after submitting work. This means the ability to modify work after having given it to the batch system.

4. Running the work.

5. Providing results back to the user.

Job submission is handled by a BATCH SUBMIT exec. All batch interfacing is handled by BATCH EXECs. In the following two examples I have used upper case to represent keywords and lower case to represent the value of arguments.

```
BATCH SUBMIT q disk

BATCH SUBMIT (NEWCMS STOR 3m TIME 30 CONTROL test)  myexec parm (option

File TEST BATCH might contain:
      BEFORE startup
      AFTER endit
      NOBEFORE
      CONTROL send1
      RETURN ?

File SEND1 BATCH might contain
      SEND file1
      SEND file2
```

In the first case the command Query DISK would be executed in a batch worker and the results returned to the submittor's reader. This is a not very useful example of submitting a basically trivial command to batch (it would show the user the assignment of disks to the batch worker), but shows the simplicity of submitting a simple function.

The second case illustrates several features of the SUBMIT exec.

1. NEWCMS specifies that the job should run under our new CMS system rather than the standard production one.

2. STOR 3m specifies that the user wants the machine to be 3 meg rather than the default of 2.

3. TIME 30 specifies that the user wants the job to have 30 minutes of cpu time. The batch monitor will put the job into a class based upon this value.

4. CONTROL test specifies that further submittal options will be obtained from a file named TEST BATCH. Up to 10 levels of nesting are allowed.

5. The left and right parentheses bounding NEWCMS and test specify that these are options to the BATCH EXEC and are not part of the command to be executed in batch.

6. myexec parm (option is the command, its parameters, and options that are to be executed in the batch machine.

7. File TEST BATCH (specified as a control file in the submit options) contains 5 more commands:

a. BEFORE - specifies the name of a file which contains the name of execs to be run before the command on the command line.

b. AFTER - the name of a file containing the names of execs to be run after the command terminates.

c. NOBEFORE - a command specifying that any further "before" requests encountered (at this or lower levels of nesting) should be ignored.

d. CONTROL - specifying that another control file (at a lower level of nesting) is requested.

e. RETURN - with the question mark will prompt the user for the name of specific files to be returned when the job terminates rather than running the normal cleanup execs.

8. File SEND1 BATCH (the next level of control file) specifies two separate files which contain the names of files to be sent with the job.

There are other options in addition to those that I have shown in the examples. The design was to allow simple submissions to be easy, but to provide a rich environment for those who have complex requirements.

In addition to the SUBMIT function the user also has QUERY, CHANGE, and CANCEL functions to allow the user to find out about the status of work (his/hers or other people's), the ability to modify some of the characteristics of the work the user has submitted, and the ability to cancel his/her own work.

During the execution phase of a batch job the user has very little control. The user has the need to know if it is progressing satisfactorily, but can do little about a job's progress other than cancelling it. An area that we need to do work for the user is to provide the user better capability of monitoring the progress of a job (particularly a long-running job). Currently, the user can obtain overall I/O counts, CPU utilization, and several other pieces of information, but the user cannot obtain I/O counts by virtual device for instance. We provided similiar capability under OS and think that we should provide capability under VM.

When a job is running it may direct its output to a user or directly to "unit record" devices. At SLAC most users direct most output back to the submittor's reader. The user must then have reasonable tools for looking at this output. All batch jobs have their output tagged with the jobname. We have an EXEC which allows a user to first get an overview by job of the output in the user's reader, select a job, and obtain files from the job. One is put into an editor with the file to peruse it. One is also given a series of other functions such as printing a file, putting it on disk, purging it, joining multiple spool files into 1 file, etc.

Constraints

One of the major concepts of a batch system is that of constraints. The system is finite. Instantaneously, requests can be made that are significantly more than the system can handle. It is thus necessary for a batch system to include scheduling capability. There are a great variety of constraints and requirements that end up being melded into a scheduling algorithm. The major ones that we have chosen to implement are:

1. A discrete number of worker machines to service batch requests.

2. A limited number of batch classes to be serviced by each worker.

3. A set of batch classes which partition the work by their consumption of resources (primarily CPU time).

4. The ability to dynamically change the number of batch workers and the classes of work that they service.

5. A monitoring capability which reduces the number of batch workers as the interactive workload picks up.

6. The ability to prevent a user from "flooding" the queue of work. This is done by establishing priorities within job classes.

7. The ability to expedite work.

8. The ability to recognize multiple batch jobs requesting the same tape volume at the same time and to single thread such requests.

9. Intercommunication with our setup system. When the setup processor gets a request from a batch worker to setup a tape it signals to the batch monitor, so that the batch monitor knows that the job is not stalled. The setup processor also signals when the setup is complete.

10. The setup processor also signals the batch monitor when it determines that a batch worker is requesting a tape that either is in use or in operator hold. In this case the job is restarted and placed in volume unavailable hold. When the volume comes out of hold status, the setup processor signals the batch monitor and the job(s) are released from volume unavailable hold.

Metrics

Whenever one has a system which does queueing it is important (if one wants to understand the system) to have a set of metrics which measure its performance. We have done some work in this area and have much more to do.

To date all of our batch performance reporting has been written in SAS, EXEC2, or XEDIT Macros.

The most important daily and monthly report that we currently have is the turnaround report. This report shows a series of measures of turnaround and utilization for each job class, plus overall statistics. Using SAS we obtain for several variables the mean, 50%, 70%, 90%ile, minimum, maximum, and sum of the variable. The variables are :

1. Queue time - how long did the jobs in this class wait before they went into execution.

2. The execution elapsed time for jobs in this class.

3. Total turnaround - queue time plus execution.

4. CPU used by this class

5. Lines produced by this class.

6. Cards punched by this class.

Similiar data are available for all classes. With this information (online) one is able to make certain that we are making our performance goals. A simple EXEC and XEDIT Macro make the viewing of this data reasonable at either a 3270 or ASCII terminal. Under OS all of these reports were on paper, now only the monthly reports are printed.

Scope (From an Operations Viewpoint)

An operator sees batch as a major service provided by the computer center. As such, it is necessary that the operator monitor and have some control over the providing of the service. Some of the capabilities are:

1. The ability to monitor the batch system and ascertain that it is providing the expected service.

2. The ability to change the normal work so as to expedite particular jobs.

3. The ability to modify the mix of work being provided.

4. The ability to affect particular jobs such as to cancel or hold them.

In addition to providing various execs to support these functions we have also created a service machine which monitors the state of many functions including batch. Its function is to alert the operator to problems.

SLAC Environment

Currently the SLAC VM system is a 16 Meg 16 channel 3081 model D which supports:

1. More than 1000 userids

2. Around 30 hours per weekday of batch work

3. 40+ hours per weekend day of batch work.

4. More than 600 jobs on a busy day.

5. More than 300 VM tape mounts per day.

6. Six job classes:

    a. X - 1 minute of CPU.

    b. S - 2 minutes of CPU.

    c. B - 4 minutes of CPU.

    d. M - 8 minutes of CPU.

    e. L - 1/2 hour of CPU.

    f. J - 2 hours of CPU.

7.

We run a maximum of 14 batch workers. The mix of work varies by shift. Where I mention that a worker services class XS it means that it will run either class but with X preferred over S.

Our weekday daytime mix is:

X	-	2	XS	-	2	S	-	1
SB	-	2	B	-	1	BM	-	2
M	-	1						

Our overnight and weekend mix is:

X	-	2	XS	-	2	S	-	1
SB	-	2	BM	-	2	ML	-	2
LJ	-	1	JL	-	2			

Other Thoughts

The batch system is merely a CMS virtual machine running a special program. Because of this, development and the running of test systems has required no stand alone time. We have never had a failure of the batch system which required a system outage. We have had very few failures of the batch system. This is probably in large part because it has been easy to develop and thoroughly test new versions of the monitor interactively.

To be Done

Major areas that we have not completed are:

1. Job Networks. The Ability to set up a complex network of jobs that will run dependent on the completion status of various jobs within the network. We will do some work in this area, but not a great deal. We currently have the capability of running simple networks and for one job to release another.

2. "Delayed" scheduling. We plan to implement in the near future the ability to submit jobs to be started at some specified time.

3. Generalized queueing. This would allow the submission of multiple jobs by independent users to use a single resource and the use of that resource would be serialized.

4. As mentioned previously we also need more status information available to both the user and the operator.

5. Job flow accounting records. From these records it would be possible to better understand our users' submission patterns and our ability to service their requests.

Summary

Although the providing of VM batch service has been a significant undertaking, we feel that we can now provide our users both a friendlier and richer batch environment than we ever provided in the OS world.