Chapter 14: Batch Processing Environment

In this chapter we provide introductory information on **LSF** (Load Sharing Facility), the standard batch processing system at Fermilab, and on **fbatch**, the locally-written interface to **LSF**. We also list the related software components that can be used with **LSF/fbatch**.

You should be able to run and manipulate most batch jobs easily after reading this chapter.

14.1 The Standard Batch System at Fermilab: LSF

LSF, developed by Platform Computing, is a general purpose resource management system that unites a group of UNIX computers into a single system to make better use of the resources on a network. The single system is referred to as a cluster. **LSF** collects resource information from all nodes in the cluster, and uses it to allocate the available host machines for execution of batch jobs.

LSF distinguishes between client machines and server machines. A job can be submitted from either type, but run only on a server (a host). Under **LSF**, jobs that are run remotely behave just like jobs run on the local host. Even jobs with complicated terminal controls behave transparently to the user as if they were being run locally.

LSF is fully documented on-line; see the *LSF User's Guide*. To find it, select *Batch Services* from the Computing Division home, and look under *LSF v3.0 Documentation*. There is also a link to it in the Computing Division document database under **fbatch**.

For the purposes of this chapter, a batch job (also called simply a *job*) is any UNIX executable that is submitted to the **LSF** batch system. Job control information (e.g., name of executable, queue, required resources, and so on) is passed to **LSF** via command line arguments supplied when submitting a job.

14.1.1 Job Queues

Batch jobs are submitted to **LSF** via *job queues*. **LSF** administrators generally configure job queues to control host resource access according to user and application type. A queue can be defined to use a particular subset of the hosts in the **LSF** cluster; the default is to use all hosts.

Each queue represents a different job scheduling and control policy. Users select the job queue that best fits each job. All jobs submitted to the same queue share the same scheduling and control policy. There is a **nice** value associated with each queue (see section 5.5.1), and jobs submitted to a queue are automatically "re**nice**d" accordingly.

14.1.2 Load Monitoring on Hosts

LSF monitors the load of each host in the batch cluster by comparing the values of several built-in load indices against the allowable load thresholds defined by the **LSF** administrator. A load index is simply a measurement of the processing load on a batch host. On an overloaded host, batch jobs can begin interfering with each other or with interactive jobs. Therefore, **LSF** begins suspending jobs on a host when it becomes overloaded (i.e. when one or more load indices exceed the predefined *suspension threshold*). **LSF** resumes any suspended jobs once all the load indices read below the *release threshold*.

If a job queue has been defined with a time window (measured in real time), **LSF** suspends any jobs running on that queue when the current time falls outside of the window. These jobs get released when the time window reopens.

14.1.3 Host Selection

The resources available for processing **LSF** jobs on each host are defined by an **LSF** administrator. Only nodes having resources that match or exceed the resource requirements of a given job are potential hosts for that job. **LSF** compares the resource requirements specified for the job against the load on each of these nodes, and chooses the most favorable host.

If no resource requirements are specified for a job, a host of the same model and type as the machine on which the job was submitted is chosen.

14.1.4 Job Priority

LSF schedules, suspends, and releases submitted jobs by balancing job priority and available resources. Job priority is governed by several factors:

- the options and arguments specified on the command line during batch submission
- the priority of the queue on which the job was submitted; according to **LSF**'s FCFS (first come first serve) protocol
- the number of shares that a job has used, according to the FSS (Fair Share Scheduling) protocol. A *share* is a portion of the resources available on the host or hosts; queues may be defined to limit the number of shares jobs can utilize.

When a host's suspension threshold is reached, **LSF** suspends lower priority jobs first unless the scheduling policy associated with a particular job dictates otherwise. A suspended job can later be resumed by **LSF** if the host's release threshold is again reached (or, if the suspension was due to a time window, as mentioned above, the job resumes when the time window reopens).

LSF does not override the UNIX scheduler.

14.2 Local Interface to LSF: fbatch

The **UPS** product **fbatch** supplies the commands that you enter to run and manipulate batch jobs. It is a set of locally-written shell scripts and C programs that provides a wrapper around **LSF**, and thus characterizes the batch processing environment on a cluster. **fbatch** is installed on many Fermilab systems, including FNALU. **fbatch** supports all the **LSF** batch functionality, and provides in addition:

- a consistent interface to **LSF** commands, regardless of UNIX flavor
- supplementary commands not directly supported by **LSF**
- an interface to the Fermilab products **spacall** and **needfile** (see section 14.3)
- AFS token renewal at job execution time
- a remote shell facility allowing commands to be executed from any host; if the host is neither a client nor a server, **fbatch** issues the **LSF** command via the **rsh** facility (see section 13.3)
- case-insensitive resource names

You need to run the command **`setup fbatch`** before accessing **fbatch** commands.

When you setup **fbatch**, you will also be able to access the man pages for these commands. Running **`man fbatch`** returns a list of all the commands supported under **fbatch**. For a complete description of the **fbatch** product, refer to the *fbatch User's Guide* (PU0152).

Several of the **fbatch** commands are illustrated below, organized by function. For complete information on each of the commands, see the man pages.

14.2.1 View Host Information

To see which hosts and resources are defined in your cluster, you can issue the command:

`% fbatch_hostinfo`

The configuration information returned includes: host name, host type, host model, CPU factor, number of CPUs, total memory, total swap space, whether the host runs **LSF** servers or not, available resources denoted by resource names. The host name, host type, and host model fields are truncated if too long. The CPU factor is used to scale the CPU load value so that differences in CPU speeds are considered by LIM[1]. The faster the CPU, the larger the CPU factor.

The output is returned in this format:

```
HOST_NAME      type    model  cpuf ncpus maxmem maxswp server RESOURCES
fsgi02          SGI R4400Ch2  84.0    16   511M  2755M    Yes (irix any fsgi02)
fsui02       SUNSOL ULTRA167  93.0     4   320M   889M    Yes (sparc any sun fsui02)
fibb01          AIX     I560  39.0     1   192M   400M    Yes (aix any fibb01)
fncl10          AIX     I370  49.0     1   128M  1136M    Yes (aix any clubs fncl10)
fibi01          AIX     I590  62.0     -      -      -     No ()
fsgi01          SGI   I4D420  30.0     -      -      -     No ()
```

14.2.2 View Queue Information

The **`fbatch_queues`** command lists the available **LSF** batch queues:

`% fbatch_queues`

The output returned is in the following format. A dash (`-`) in any entry means that the column does not apply to the row. In this example some queues have no per-queue, per-user or per-processor job limits configured, so the `MAX`, `JL/U` and `JL/P` entries are dashes. The man page describes each of the fields.

1. Load Information Manager (LIM) is a daemon process that keeps track of the load indices.

```
QUEUE_NAME     PRIO      STATUS      MAX  JL/U JL/P JL/H NJOBS  PEND  RUN  SUSP
test_queue      99    Open:Active      -    -    -    -     0     0    0     0
e831_long       16    Open:Active      1    1    -    -     0     0    0     0
e831_short      14    Open:Active      -   10    -    -     0     0    0     0
30min           10    Open:Active      -    5    -    -     1     1    0     0
30min_disk      10    Open:Active      -    5    -    -     3     3    0     0
4hr              8    Open:Active      -    5    -    -     2     0    1     1
4hr_disk         8    Open:Active      -    5    -    -     5     2    3     0
12hr             6    Open:Active      -    5    3    -     3     0    3     0
12hr_disk        6    Open:Active      -    5    2    -     7     4    3     0
1day             4    Open:Active      -    5    1    -     0     0    0     0
1day_disk        4    Open:Active      -    5    1    -     7     0    7     0
4day             2    Open:Active      -    5    0    -    33    17   12     4
```

You can submit jobs to a queue as long as its `STATUS` is `Open`. However, jobs are not dispatched unless the queue is `Active`.

14.2.3 Submit a Batch Job

The **`fbatch_sub`** command is used to submit a job to the batch system. The most common arguments used are **`-q`** (queue name), **`-R`** (resource requirements), **`-o`** (stdout redirection), **`-e`** (stderr redirection), and **`-N`** (notify via email).

As an example, here we submit a script called `myjob` to the `4hr` queue, specify an IRIX host, and request notification. The stdout is redirected to `myjob.out`, and the stderr to `myjob.err`:

`% fbatch_sub -N -q 4hr -o myjob.out -e myjob.err -R "irix" myjob`

AFS

On systems running AFS, **fbatch** will prompt you for your AFS password[1]:

```
Enter AFS Password...
Reenter AFS Password...
fbatch_sub executing LSF command locally on fsui02....
```

When your job begins, you will automatically receive a renewed AFS token on the execution host.

14.2.4 Monitor Submitted Batch Jobs

fbatch provides several commands that allow you to monitor your job. The usage examples below use a sample job number 1022:

Display a listing of running jobs

`% fbatch_jobs`

If no options are supplied, the list will contain only your running jobs. To see *all* running jobs, use the **`-u all`** option. Output is returned in this format:

```
JOBID USER     STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
 1022 aheavey  PEND  30min      fsui02                  sleep1     Sep 10 09:56
```

Display the stdout and stderr of a job

`% fbatch_peek 1022`

The format of the output varies according to the files.

1. Your password gets encrypted by **fbatch** using PGP.

Display history information about a job

% fbatch_hist 1022

Output is returned in the format:

```
Summary of time in seconds spent in various states:
JOBID USER    JOB_NAME   PEND    PSUSP   RUN     USUSP   SSUSP   UNKWN   TOTAL
 1022 aheavey sleep1     7       0       35      0       0       0       42
```

14.2.5 Control Submitted Batch Jobs

For jobs that are in a queue awaiting execution, **fbatch** provides commands to move jobs within the queue, and to modify the resource requirements of the job. The usage examples below use a sample job number 1022:

Move Job within Queue

Move job to the bottom of the queue:

% fbatch_bot 1022

Move job to the 2nd position from the top of the queue:

% fbatch_top 1022 2

The **fbatch_bot** and **fbatch_top** commands, above, move jobs within queues *relative to the user's own jobs*. You cannot move your job ahead of another user's job with these commands.

Change Job Parameters

Change the resource requirements of job:

% fbatch_modify -R aix 1022

Migrate a batch job to another host:

% fbatch_mig -m newhost 1022

Suspend, Resume, or Kill a Job

Suspend (stop, but do not cancel) job:

% fbatch_stop 1022

Resume job:

% fbatch_resume 1022

Cancel job:

% fbatch_kill 1022

14.3 Related Software Components

This section describes other software components that can be used with the **LSF/fbatch** batch system.

needfile

needfile is an interface to the **HPSS/Unitree** central mass storage system. The **needfile** system provides two functions. If the user’s data does not yet exist in **HPSS**, **needfile** tells the system to copy the data there from its original medium. **needfile** then provides access to this data for the user. This data remains in the HSM based on a least-recently-used policy. There is no guarantee how long data will remain in the HSM.

To access **needfile**, first run the command `setup nt` (setup not required if accessed via **fbatch**).

A **needfile** reference manual exists on the Web. To find it, start on the Computing Division home page, select *Mass Storage* under **Systems and Networking** and then *fmss* to find **needfile tool**. There is also documentation there on HPSS.

spacall

The **spacall** utility (space allocator) provides scratch disk storage for a job. **spacall** is invoked under **fbatch** by submitting a job to a specially defined queue; for example, on FNALU the `*_disk` queues have been configured for it. The path to the scratch space is stored in the environment variable *$CLUBS_WORKDIR*.