

LSF usage on Coffee

The SGI Altix 4700 is the production computer for ECU researchers. It has 128 processors, 128 GB of shared memory and over 10 terabytes of disk space available for research and educational programs at ECU. The SGI Altix 4700 is named "jasta". Since "jasta" is a production computer, users do not log onto it directly to run their jobs, but use the batch scheduler LSF to run jobs on it. To facilitate this process, an 8 processors (16 GB of shared memory) SGI altix serves as a front-end to "jasta". This front-end computer is named "coffee" (coffee.ecu.edu). The home directories on "coffee" and "jasta" are the same. Thus user files are shared across the two computers. Users log onto "coffee" to submit jobs to LSF which will be run on "jasta".

Since "coffee" is a front-end computer to "jasta", ECU restricts its use for production runs. Interactive use is limited to a maximum of 10 minutes of CPU time and 4 processors. That is, its main purpose is for testing and setting up jobs to run on "jasta". Users who violate these restrictions can lose their user privileges.

To run a job on "jasta" from coffee, we use the LSF batch scheduler. Since "coffee" and "jasta" share the same home directory for each user, files that are located on "coffee" can be manipulated and run on "jasta" using LSF. For instance, suppose you want to know the version of gcc on "jasta", one could create a file on coffee, call it "gcc_version.job" and put the lines

```
#!/bin/csh
gcc --version
exit
```

To run this on jasta, use the command line:

```
bsub -q 1c_30d -o output -e error < gcc_version.job
```

Near the end of the file "output", you will see that the executed command on "jasta" shows that the gcc version is 4.1.0. The file "output" contains standard output from running LINUX commands and the file "error" will contain any errors (if one or more errors occur).

Using the above procedure, users can run any LINUX command(s) on "jasta".

To monitor jobs, use the command:

```
bjobs
```

This will tell you if your job is running or pending. If nothing is returned it means you have no jobs in the LSF batch que. To see all users and number of processors being used by "jasta", execute the command:

```
/usr/local/apps/bin/queues.csh
```

As another example, suppose you wanted to untar a file using the tar command on "jasta". You would first go to the directory containing the tar file (also the directory where you want the file untared) and create a file (call it untar.job) that contains:

```
#!/bin/csh
tar -xvf file.tar
exit
```

To run this tar command on jasta, you would use the following command to submit the file to LSF:

```
bsub -q 1c_30d -o output -e error < untar.job
```

Again, any errors should appear in the file "error" (or whatever you wish to call it).

Now suppose you want to run your mpi executable (called cruz.exe) using 8 processors, You would create a file (say called "fly.job"):

```
#!/bin/csh
#BSUB -o fly.logfile -e fly.error
#BSUB -W 4320
#BSUB -n 8
#BSUB -J fly
#BSUB -m Jasta

mpirun -np 8 cruz.exe < input_file > output_file

exit
```

Submit this job by typing

```
bsub < fly.job
```

Note: all the BSUB commands are in the "fly.job" file, thus you do not need to put them on the command line. Also, LSF will read the # processors (8) and send it to the correct "jasta" que.

Suppose you need to run a script to create the file to be run. In this example, the script "prp_ms_44" creates a file called "h2o.job" which is then executed. The "prp_ms_44" script needs 4 questions answered [name of job, amount of computer time in minutes, number of processors and basis set (a 2 in this case)]. Your job file ("run.job") would look like:

```
#!/bin/csh
#BSUB -o job.logfile -e job.error
#BSUB -W 4320
#BSUB -n 8
#BSUB -J job
#BSUB -m Jasta
```

```
prp_ms_44 << EOF
h2o
4320
8
2
EOF
```

```
h2o.job >&! h2o.log
```

```
exit
```

To submit to LSF, type:

```
bsub < run.job
```

The ques on "jasta" are (1, 8, 16, 32 and 64 processor; 30, 10, 10, 7, 7 days respectively):

1c_30d
8c_10d
16c_10d
32c_7d
64c_7d