

LSF Job Manage System HOWTO

The High Performance Computing Center(HPCC) at SJTU

<http://hpt.sjtu.edu.cn>

Updated at 2013-10-08

目录

1	Check the Cluster's Running Status via LSF	2
1.1	Check the LSF Computing Nodes List <code>bhosts</code>	2
1.2	Check the LSF Queues <code>bqueues</code>	2
1.3	Check the load of computing nodes <code>lsload</code>	3
2	Submit the Jobs via LSF <code>bsub</code>	3
2.1	Submit Jobs Manually	3
2.2	Interactive Batch Submit	4
2.3	Write a LSF scripts to submit jobs	4
2.4	A more complicated LSF script	5
3	Other Job Manage Operations	6
3.1	Check the jobs' status <code>bjobs</code>	6
3.2	Kill the Jobs <code>bkill</code>	6
3.3	Monitor the Output of Jobs <code>bpeek</code>	7
3.4	Jobs' History Information <code>bhist</code>	7

This document will instruct you to submit and manage jobs via LSF, it contains the information about using LSF to submit, check and delete jobs.

Following the document's operation and feedback's methods will help you complete the job successfully. It will be very kind to provide advices, thank you!

1 Check the Cluster's Running Status via LSF

1.1 Check the LSF Computing Nodes List `bhosts`

```
# bhosts
```

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
fat01	ok	-	16	0	0	0	0	0
fat02	ok	-	16	0	0	0	0	0
fat03	ok	-	16	0	0	0	0	0
fat04	ok	-	16	0	0	0	0	0
fat05	ok	-	16	0	0	0	0	0
fat06	ok	-	16	0	0	0	0	0
fat07	ok	-	16	0	0	0	0	0
fat08	ok	-	16	0	0	0	0	0
fat09	ok	-	16	0	0	0	0	0
fat10	ok	-	16	0	0	0	0	0
.....								

1.2 Check the LSF Queues `bqueues`

Check the whole queues' overall information:

```
# bqueues
```

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
cpu	40	Open:Active	-	-	-	-	2072	0	2072	0
fat	40	Open:Active	-	-	-	-	0	0	0	0
gpu	40	Open:Active	-	-	-	-	288	0	288	0
mic	40	Open:Active	-	-	-	-	0	0	0	0
cpu-fat	40	Open:Active	-	-	-	-	16	0	16	0

Check for some queue's information:

```
# bqueues fat
QUEUE_NAME      PRIO STATUS      MAX JL/U JL/P JL/H NJOBS  PEND  RUN  SUSP
fat              40  Open:Active    -   -   -   -      0     0    0    0
```

1.3 Check the load of computing nodes lsload

Check the overall load:

```
# lsload
HOST_NAME      status  r15s  r1m  r15m  ut    pg  ls    it    tmp  swp  mem
node011        ok     0.0  0.3  0.4  0%   0.0  0 49024 193G  62G  61G
node039        ok     0.0  0.6  0.5  0%   0.0  0 49024 194G  62G  61G
node041        ok     0.0  0.0  0.0  0%   0.0  0 49024 194G  62G  61G
node050        ok     0.0  0.3  0.4  0%   0.0  0 49024 194G  62G  60G
node064        ok     0.0  0.7  0.6  0%   0.0  0 49024 194G  62G  61G
node077        ok     0.0  0.7  0.5  0%   0.0  0 49024 193G  62G  61G
.....
```

Check for some node's load:

```
# lsload node001
HOST_NAME      status  r15s  r1m  r15m  ut    pg  ls    it    tmp  swp  mem
node01         ok     0.3  0.1  0.1  1%   0.0  0  332 152G  62G  61G
```

2 Submit the Jobs via LSF bsub

2.1 Submit Jobs Manually

LSF uses `bsub` to submit jobs. The format of `bsub` is:

```
bsub -n Z -q QUEUE_NAME -i INPUTFILE -o OUTPUTFILE COMMAND
```

`z` is the number of threads needed, `-q` assign the queue. If there is no option `-q`, the system will submit the jobs to the default queue. `INPUTFILE` is name of the file read by the program, `OUTPUTFILE` is the output file's name.

For the serial job, `COMMAND` can directly be your program's name. Example: submit the serial program `mytest` via LSF:

```
bsub -n 1 -q q_default -o mytest.out ./mytest
```

For the MPI parallel program, the format of `COMMAND` is `-a mpich_gm mpirun.lsf PROG_NAME`. Example: submit the parallel program `mytest` via LSF which uses 16 threads:

```
bsub -n 16 -q q_default -o mytest.out -a mpich_gm mpirun.lsf ./mytest
```

2.2 Interactive Batch Submit

You can start up an interactive shell environment by using `bsub` to submit multiple parallel jobs whose running arguments are the same:

```
# bsub
bsub> -n 16
bsub> -q q_default
bsub> -o output.txt
bsub> COMMAND1
bsub> COMMAND2
bsub> COMMAND3
```

It is equal to:

```
bsub -n 16 -q q_default -o output.txt COMMAND1
bsub -n 16 -q q_default -o output.txt COMMAND2
bsub -n 16 -q q_default -o output.txt COMMAND3
```

2.3 Write a LSF scripts to submit jobs

```
#BSUB -n 16
#BSUB -q q_default
#BSUB -o output.txt
-a mpich_gm mpirun.lsf ./mytest
```

`bsub` also accepts the state of jobs from stdin, that means we can write the LSF script to submit jobs. `bsub`'s script is easy to write, the code above is an example named `bsub.script`, submit `bsub.script` to LSF via input redirection:

```
bsub < bsub.script
```

It is equal to:

```
bsub -n 16 -q q_default -o output.txt -a mpich_gm mpirun.lsf ./mytest
```

2.4 A more complicated LSF script

```
#BSUB -J HELLO_MPI
#BSUB -o job.out
#BSUB -e job.err
#BSUB -n 256

source /lustre/utility/intel/composer_xe_2014.3.163/bin/compilervars.sh intel64
source /lustre/utility/intel/mkl/bin/intel64/mklvars_intel64.sh
source /lustre/utility/intel/impi/4.1.1.036/bin64/mpivars.sh

MPIRUN=`which mpirun`
EXE="./mpihello"

CURDIR=$PWD
cd $CURDIR
rm -f nodelist nodes >& /dev/null
touch nodelist
touch nodes
NP=0

for host in `echo $LSB_MCPU_HOSTS | sed -e 's/ /:/g' | sed 's/:n/\nn/g'`
do
echo $host >> nodelist
echo $host | cut -d ":" -f1 >> nodes
nn=`echo $host | cut -d ":" -f2`
NP=`echo $NP+$nn | bc`
done
```

3 Other Job Manage Operations

3.1 Check the jobs' status `bjobs`

Check the submitted jobs' running status:

```
bjobs
```

Display the jobs' running status as wide format:

```
bjobs -w
```

Display all the jobs:

```
bjobs -a
```

Display the running jobs:

```
bjobs -r
```

Display the pending jobs and reasons:

```
bjobs -p
```

Display the suspending jobs and reasons:

```
bjobs -s
```

Display detailed information of job `JOBID`:

```
bjobs -l JOBID
```

3.2 Kill the Jobs `bkill`

Kill the jobs unwanted:

```
bkill
```

Kill the job `JOBID`:

```
bkill JOBID
```

Remove the job `JOBID` from LSF instead of waiting its progresses killed by the operating system:

```
bikill JOBID
```

3.3 Monitor the Output of Jobs `bpeek`

Display the stdout and stderr output of a unfinished batch job

```
bpeek
```

Display the output of the job with the specified ID `JOBID`

```
bpeek JOBID
```

3.4 Jobs' History Information `bhist`

display the history of batch jobs

```
bhist
```

Display the specified job(s) `JOBID` only

```
bhist JOBID
```