WEXAC

(Weizmann EXAscale Cluster)

SGE-to-LSF Migration

Users Training

Job Submission & Control

- `bsub` - submits a batch job to LSF
- `bjobs` - displays information about LSF jobs
- `bhist` - displays historical information about jobs
- `bkill` - sends signals to kill, suspend, or resume unfinished jobs
- `bmod` - modifies job submission options of a job
- `bpeek` - displays the stdout and stderr output of an unfinished job
- `bstop` - suspends unfinished jobs
- `bresume` - resumes unfinished jobs
- `bswitch` - switches jobs from one queue to another

bsub - Commonly Used Options

`-q` *qname*	submits the job to the specified queue
`-o` *file* information	redirect stdout, stderr and resource usage of the job to the specified *output file*
`-e` *file*	redirect stderr to the specified error file
`-oo/-eo` *file*	same as -o/-e, but overwrite file if it exists
`-i` *filename*	use the specified file as standard input for the job
`-n` number	specify number of job slots
`-g` *jobgroup*	submit job to specified group
`-J` *jobname*	assigns the specified name to the job
`-R` res_req	runs job on a host that meets the specified resource requirements
`-L` *shell*	Initializes the execution environment using the specified login shell

bsub - Methods for Submitting Jobs

- **By script or command**

```
$ bsub -q all.q -J example -o example-%J.o -e example-%J.e date
```

- **By job spooling**

```
$ bsub < job_file
```

job_file example:

```
#BSUB -q all.q
#BSUB -J example
#BSUB -o example-%J.o
#BSUB -e example-%J.e
date
```

bsub - Methods for Submitting Jobs

- **Interactively**

```
$ bsub
bsub> #BSUB -q all.q
bsub> #BSUB -J example
bsub> #BSUB -o example-%J.o
bsub> #BSUB -e example-%J.e
bsub> date
bsub> ^D
Job <2387> is submitted to queue <all.q>.
```

- **Array Jobs:**

```
    bsub –J “ArrN[1-300]%10" –i "inp.%I“ ascript
where:
    –J "ArrN[…]"   names and creates the job array
    1-300 can be start[-end[:step]]
    –i inp.%I is the input file to be used and %I is the array index
      reference value
    ascript is the script to be executed
```

Resource Requirements (-R)

- **Resource requirement string is divided into following sections:**

Selection	-	`select[`*`selection_string`*`]`
Usage	-	`rusage[`*`rusage_string`*`]`
Ordering	-	`order[`*`order_string`*`]`
Locality	-	`span[`*`span_string`*`]`
Same	-	`same[`*`same_string`*`]`
CU	-	`cu[`*`cu_string`*`]`

- **Span and same sections are used for parallel jobs**

Resource Requirements Examples

Example 1. Select execution host candidates that have at least 2GB free RAM

```
$ bsub -R "select[mem>2GB]" myJob
```

Example 2. Select execution host candidates with Infiniband interconnect

```
$ bsub -R "defined(ib)" myJob
```

Example 3. Select candidate hosts that have 1000MB free swap and order by amount of available memory

```
$ bsub -R "select[swp>1000] order[mem]" myJob
```

Resource Requirements Examples

8

Example 4. Candidate hosts should have min 500MB free RAM, job will reserve 400MB RAM.

```
$ bsub -n 4 -R "select[mem>500] rusage[mem=400]" myJob
```

Example 5. All slots required for a parallel job should reside on the same host

```
$ bsub -n 4 -R "span[hosts=1]" parallelJob
```

Example 6. 12-CPU parallel job should run on up to 4 CPUs per host , all candidate hosts must have the same CPU model

```
$ bsub -n 12 -R "span[ptile=4] same[model]" parallelJob
```

bjobs - View Job Information

`-a`	Display information about jobs in all states, including recently finished jobs
`-A`	Displays summarized information about job arrays
`-d`	Display information about jobs that finished recently
`-l|-w`	Display information in long or wide format
`-p`	Display information about pending jobs
`-r`	Display information about running jobs
`-g job_group`	Display information about jobs in specified group
`-J job_name`	Display information about specified job or array
`-q queue`	Display information about jobs in specified queue
`-u user`	Display information about jobs for specified users/groups

bjobs - Example 1


```
$ bjobs -a
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
2388    demo    RUN   all.q      access1.wex cn001       test       Apr 11 10:06
2389    demo    RUN   all.q      access1.wex cn002       test       Apr 11 10:06
2390    demo    PEND  all.q      access1.wex             test       Apr 11 10:06
2391    demo    PEND  all.q      access1.wex             test       Apr 11 10:06
2392    demo    PEND  all.q      access1.wex             test       Apr 11 10:06
2393    demo    PEND  all.q      access1.wex             test       Apr 11 10:06
2394    demo    PEND  all.q      access1.wex             test       Apr 11 10:06
2395    demo    PEND  all.q      access1.wex             test       Apr 11 10:06
2396    demo    PEND  all.q      access1.wex             test       Apr 11 10:06
2397    demo    PEND  all.q      access1.wex             test       Apr 11 10:06
```

bjobs - Example 2


```
$ bjobs -l 2395

Job <2395>, Job Name <test>, User <demo>, Project <default>, Status <DONE>, Que
                     ue <all.q>, Command <sleep 200>, Share group charged </dem
                     o>
Tue Apr 11 10:06:16: Submitted from host <access1.wexac.weizmann.ac.il>, CWD <$
                     HOME>;
Tue Apr 11 10:07:20: Started on <cn002>, Execution Home </home/demo>, Execution
                      CWD </home/demo>;
Tue Apr 11 10:10:40: Done successfully. The CPU time used is 0.0 seconds.

 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -
```

bhist - Jobs History

-a Display information about all jobs

-b|-l|-w Display information in brief, long, or wide format

-d Display information about finished jobs

-p Display information about pending jobs

-s Display information about suspended jobs

-t Display job events chronologically

-C|-D|-S|-T *start_time,end_time*

Display information about completed, dispatched, submitted, or all jobs in specified time window

-q *queue* Display information about jobs submitted to specified queue

-u *username|all* Display information about jobs submitted by user or all users

bhist - Example 1


```
$ bhist
Summary of time in seconds spent in various states:
JOBID   USER    JOB_NAME   PEND    PSUSP   RUN     USUSP   SSUSP   UNKWN   TOTAL
2402    demo    test       265     0       196     0       0       0       461
2403    demo    test       265     0       196     0       0       0       461
2404    demo    test       406     0       55      0       0       0       461
2405    demo    test       406     0       55      0       0       0       461
2406    demo    test       426     0       35      0       0       0       461
2407    demo    test       426     0       35      0       0       0       461
```

bhist - Example 2

14

```
$ bhist -l 2403

Job <2403>, Job Name <test>, User <demo>, Project <default>, Command <sleep 200
                     >
Tue Apr 11 10:06:16: Submitted from host <access1.wexac.weizmann.ac.il>, to Que
                     ue <all.q>, CWD <$HOME>;
Tue Apr 11 10:10:41: Dispatched to <cn002>;
Tue Apr 11 10:10:41: Starting (Pid 26555);
Tue Apr 11 10:10:41: Running with execution home </home/demo>, Execution CWD </
                     home/demo>, Execution Pid <26555>;
Tue Apr 11 10:14:01: Done successfully. The CPU time used is 0.0 seconds;
Tue Apr 11 10:14:01: Post job process done successfully;

Summary of time in seconds spent in various states by  Tue Apr 11 10:14:01
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  265      0        200      0        0        0        465
```

Manipulating Jobs

- `bkill` - send signals to kill, suspend, or resume unfinished jobs
  - Tip: use JobID 0 to kill all your jobs
- `bmod` - modify job submission options of a job
- `bpeek` - display the stdout and stderr output of an unfinished job
- `bstop` - suspend unfinished jobs
- `bresume` - resume unfinished jobs
- `bswitch` - switch jobs from one queue to another

Job Manipulation Examples

Example 1:

```
$ bsub -q all.q sleep 500
Job <2411> is submitted to queue <all.q>.
$ bjobs 2411
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
2411    demo    RUN   all.q      access1     cn501       sleep 500  Apr 11 10:19
$ bstop 2411
Job <2411> is being stopped
$ bjobs 2411
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
2411    demo    USUSP all.q      access1     cn501       sleep 500  Apr 11 10:19
$ bresume 2411
Job <2411> is being resumed
$ bjobs 2411
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
2411    demo    RUN   all.q      access1     cn501       sleep 500  Apr 11 10:19
```

Job Manipulation Examples

Example 2:

```
$ bmod -q test.q 2411
Parameters of job <2411> are being changed
$ bjobs 2411
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
2411    demo    PEND  test.q     access1                 test       Apr 11
    10:19
```

Cluster Query Commands

`lsinfo`	- displays load sharing configuration information
`lshosts`	- displays hosts and their static resource information
`lsload`	- displays load information for hosts
`bhosts`	- displays hosts and their static and dynamic resources
`bqueues`	- displays information about queues
`bmgroup`	- displays information about host groups and compute units
`busers`	- displays information about users and user groups
`bugroup`	- displays information about user groups

Query Commands Examples

Example 1. Show resource information for compute nodes cn001 and cn002

```
$ lshosts cn001 cn002
HOST_NAME  type    model     cpuf ncpus maxmem maxswp server RESOURCES
cn001      X86_64 Intel_EM  60.0     4   750M  1503M  Yes    ()
cn002      X86_64 Intel_EM  60.0     4   750M  1503M  Yes    ()
```

Example 2. Show load information for compute nodes cn001 and cn002

```
$ lsload cn001 cn002
HOST_NAME  status  r15s  r1m  r15m  ut  pg  ls  it    tmp   swp    mem
cn001          ok   0.0  0.0   0.0  0%  0.0 0   28784 1786M 1503M  654M
cn002          ok   0.0  0.0   0.0  0%  0.0 0   28784 1934M 1503M  662M
```

Example 3. Show statistics for hosts cn001, cn002

```
$ bhosts cn001 cn002
HOST_NAME     STATUS   JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
cn001         ok          -      4      0      0      0      0      0
cn002         ok          -      4      0      0      0      0      0
```

Query Commands Examples

Example 4. Show queues

```
$ bqueues
QUEUE_NAME PRIO STATUS      MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
test.q     20   Open:Active   -   50    -    -     0     0     0     0
all.q      10   Open:Active   -   30    -    -     0     0     0     0
```

Example 4. Display information about all users

```
$ busers all
USER/GROUP      JL/P    MAX  NJOBS   PEND    RUN  SSUSP  USUSP    RSV
adhoc              -      -      0      0      0      0      0      0
default            -      -      -      -      -      -      -      -
demo               -    100      0     10      2      0      0      0
test1              -      -      0      0      0      0      0      0
test2              -      -      0      0      0      0      0      0
```

SGE to LSF Cheat Sheet

SGE	Platform LSF	Description
Queue Commands		
qstat	bjobs -u all	Shows list of all jobs running on the cluster
qstat -g t	bjobs -u all -X	Shows list of all jobs running on the cluster expanded to show nodes
qstatme	bjobs	Shows list of your running jobs (also accepts -X flag to expand to show nodes)
qstat -j JOBID	bjobs -l JOBID	Shows detailed information about a job (where JOBID is the job ID number)
qstat -g c	bqueues	Shows list of available queues inc number of jobs and slots
Job Commands		
qsub jobname.job	bsub < jobname.bsub	Submit job called jobname
qdel JOBID	bkill JOBID	Delete running job (where JOBID is the job ID number)

SGE to LSF Cheat Sheet

SGE	Platform LSF	Description
Job Flags		
#$ -q queuename.q	#BSUB -q queuename	Submit job to queue queuename
#$ -N JobName	#BSUB -J JobName	Job Name
#$ -t 1-10	#BSUB -J JobName[1-10]	Array job with 10 elements
#$ -pe openmpi 16	#BSUB -n 16 -a openmpi	Request number of slots (16 in this case)
#$ -m b	#BSUB -B	Email when job starts
#$ -m e	#BSUB -N	Email when job finishes
#$ -o output.log	#BSUB -o output.log	Write output to output.log (LSF - I for array JOBINDEX)
#$ -e error.log	#BSUB -e error.log	Write error to error.log (LSF - I for array JOBINDEX)
Job Variables		
$JOB_ID	$LSB_JOBID	Job ID
$SGE_TASK_ID	$LSB_JOBINDEX	Array job index (when using -t in SGE and -J Name[1-X] in LSF)
$NSLOTS	$LSB_DJOB_NUMPROC	Number of parallel slots requested

Resources

23

http://www.weizmann.ac.il/grid/

THANK YOU