Using the batch system

Samuel Fux

High Performance Computing Group, Scientific IT Services, ETH Zurich

Batch > Overview

- The batch system of Euler is called *LSF* (Load Sharing Facility)
- LSF manages all resources available on the cluster and allocates them to users' jobs
  - Ensures that resources are used as efficiently as possible
  - Calculates user/job priorities based on a fair share principle
- All computations must be submitted to the batch system
  - There is no other way to access the cluster's compute nodes
- Please do not run computations on the login nodes
  - Login nodes may only be used for file transfer, compilation, code testing and debugging, and quick pre- and post-processing

Batch > Basic job submission

- Use `bsub` to submit a job to the batch system

```
bsub [LSF options] job
```

- A *job* can be either …
  - a single Linux command — `cmd`
  - a shell script, passed via “<” — `< script`
  - a here document, passed via “<<” — `<< EOF ... EOF`
  - a program, with its path — `/path/to/program`
  - a command or program, with its arguments — `cmd arg1 arg2`
  - multiple commands, enclosed in quotes — `"cmd1 ; cmd2"`
  - piped commands, enclosed in quotes — `"cmd1 | cmd2"`
  - a command with I/O redirection, quoted — `"cmd <in >out"`
- We’ll talk about `bsub`’s options later

Batch > Basic job submission

- When you submit a job via `bsub`, the batch system analyzes it and dispatches it to a batch queue
  - LSF always selects the best queue for your job
  - You can not select a queue yourself
- If all goes well, `bsub` tells you
  - The kind of job you have submitted – e.g. “Generic job”
  - The job’s unique identifier (“job ID”) – e.g. “8146539”
  - The queue were the job was dispatched – e.g. “normal.4h”

Batch > Basic job submission > Examples

```
[leonhard@euler03 ~]$ bsub echo hello
Generic job.
Job <8146539> is submitted to queue <normal.4h>.

[leonhard@euler03 ~]$ bsub < hello.sh
Generic job.
Job <8146540> is submitted to queue <normal.4h>.

[leonhard@euler03 ~]$ bsub ./bin/hello
Generic job.
Job <8146541> is submitted to queue <normal.4h>.

[leonhard@euler03 ~]$ bsub "date; pwd; ls -l"
Generic job.
Job <8146542> is submitted to queue <normal.4h>.

[leonhard@euler03 ~]$ bsub "du -sk /scratch > du.out"
Generic job.
Job <8146543> is submitted to queue <normal.4h>.
```

Batch > Resource requirements

- The batch system of Euler works like a black box
  - You do not need to know anything about queues, hosts, user groups, priorities, etc. to use it
  - You only need to specify the resources needed by your job
- The two most important resources are
  - Maximal run-time and the number of processors for parallel jobs
- These resources are passed to `bsub` using options

  `bsub -W` *`HH:MM`* `-n` *`number_of_processors`* `…`
- By default, a job will get 1 processor for 4 hour
  - If you need more time and/or processors, you must request them
  - Standard run-time limits are 4h, 24h, 120h and 30 days

Batch > Advanced resource requirements

- Memory
  - By default LSF gives you 1024 MB of memory per processor (core)
  - If you need more, you must request it
  - For example, to request 2048 MB per processor (core):

```
bsub -R "rusage[mem=2048]" …
```

- Scratch space
  - LSF does not allocate any local scratch space to batch jobs
  - If your job writes temporary files into the local `/scratch` file system, you **must** request it
  - For example, to request 10,000 MB of scratch space:

```
bsub -R "rusage[scratch=10000]" …
```

- Both requirements can be combined

```
bsub -R "rusage[mem=2048,scratch=10000]" …
```

Batch > Other bsub options

`-o` *`outfile`*	append job's standard output to *outfile*
`-e` *`errfile`*	append job's error messages to *errfile*
`-R "rusage[…]"`	advanced resource requirement (memory,…)
`-J` *`jobname`*	assign a *jobname* to the job
`-w "`*`depcond`*`"`	*wait* until dependency condition is satisfied
`-Is`	submit an *interactive* job with pseudo-terminal
`-B / -N`	send an email when the job begins/ends
`-u` *`user@domain`*	use this address instead of *username*@ethz.ch

Batch > Parallel job submission

- Shared memory job (OpenMP)
  - Runs on a single compute node
  - Can use up to 24 processors
  - Number of processors must be defined in `$OMP_NUM_THREADS`

```
export OMP_NUM_THREADS=8
bsub -n 8 ./program
```

- Distributed memory job (MPI)
  - Runs on multiple compute nodes
  - Can use tens or even hundreds of processors
  - Program must be launched using `mpirun`

```
module load compiler
module load mpi_library
bsub -n 240 mpirun ./program
```

Batch > Parallel job submission > Examples

```
[leonhard@euler03 ~]$ export OMP_NUM_THREADS=8
[leonhard@euler03 ~]$ bsub -n 8 ./hello_omp
Generic job.
Job <8147290> is submitted to queue <normal.4h>.

[leonhard@euler03 ~]$ unset OMP_NUM_THREADS

[leonhard@euler03 ~]$ bsub -n 240 mpirun ./hello_mpi
MPI job.
Your environment is not configured for MPI.
Please load the module(s) needed by your job before executing 'bsub'.
Request aborted by esub. Job not submitted.

[leonhard@euler03 ~]$ module load intel open_mpi
[leonhard@euler03 ~]$ bsub -n 240 mpirun ./hello_mpi
MPI job.
Job <8147303> is submitted to queue <normal.4h>.
```

Batch > Job array

- Multiple similar jobs can be submitted at once using a so-called “job array”
  - All jobs in an array share the same JobID
  - Use job index between brackets to distinguish between individual jobs in an array
  - LSF stores job index and array size in environment variables
  - Each job can have its own standard output
- Examples:

```
bsub -J "array_name[1-N]" ./program     # submit N jobs at once
bjobs -J array_name                     # all jobs in an array
bjobs -J jobID                          # all jobs in an array
bjobs -J array_name[index]              # specific job in an array
bjobs -J jobID[index]                   # specific job in an array
```

Batch > Job array > Example

```
[leonhard@euler03 ~] bsub -J "hello[1-8]"
bsub> echo "Hello, I am job $LSB_JOBINDEX of $LSB_JOBINDEX_END"
bsub> ctrl-D
Job array.
Job <29976045> is submitted to queue <normal.4h>.
[leonhard@euler03 ~]$ bjobs
JOBID      USER      STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
29976045   leonhard  PEND  normal.4h  euler03                 hello[1]   Oct 10 11:03
29976045   leonhard  PEND  normal.4h  euler03                 hello[2]   Oct 10 11:03
29976045   leonhard  PEND  normal.4h  euler03                 hello[3]   Oct 10 11:03
29976045   leonhard  PEND  normal.4h  euler03                 hello[4]   Oct 10 11:03
29976045   leonhard  PEND  normal.4h  euler03                 hello[5]   Oct 10 11:03
29976045   leonhard  PEND  normal.4h  euler03                 hello[6]   Oct 10 11:03
29976045   leonhard  PEND  normal.4h  euler03                 hello[7]   Oct 10 11:03
29976045   leonhard  PEND  normal.4h  euler03                 hello[8]   Oct 10 11:03
[leonhard@euler03 ~]$ bjobs -J hello[6]
JOBID      USER      STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
29976045   leonhard  PEND  normal.4h  euler03                 hello[6]   Oct 10 11:03
```

Batch > #BSUB pragmas

- `bsub` options can be specified either on the command line or inside a job script using the #BSUB pragma, for example

```
#!/bin/bash
#BSUB -n 24                          # 24 cores
#BSUB -W 8:00                        # 8-hour run-time
#BSUB -R "rusage[mem=4000]"          # 4000 MB per core
cd /path/to/execution/folder
command arg1 arg2
```

- In this case, the script **must** be submitted using the “<” operator

```
bsub < script
```

- `bsub` options specified on the command line override those inside the script

```
bsub -n 48 < script
```

Batch > Light-weight job

- Light-weight jobs are jobs that do not consume a lot of CPU time, for example
  - Master process in some type of parallel jobs
  - File transfer program
  - Interactive shell
- Some compute nodes are specially configured for light-weight jobs
  - They allow multiple light-weight jobs to run on the same core at the same time
  - This is more efficient than allocating 100% of a core to a job that would use only 10%
- Use the option “-R light” to submit a light-weight job
  - Example: submit a 15-minute interactive bash shell

```
bsub -W 15 -Is -R light /bin/bash
```

  - Do not forget to logout (type “logout” or “exit”) when you’re done

Batch > Light-weight job > Example

```
[leonhard@euler03 ~]$ bsub -W 15 -Is -R light /bin/bash
Generic job.
Job <27877012> is submitted to queue <light.5d>.
<<Waiting for dispatch ...>>
<<Starting on e2002>>
[leonhard@e2002 ~]$ pwd
/cluster/home/leonhard
[leonhard@e2002 ~]$ hostname
e2002
[leonhard@e2002 ~]$ exit
exit
[leonhard@euler03 ~]$
```

Batch > Job control commands

`busers`	user limits, number of pending and running jobs
`bqueues`	queues status (open/closed; active/inactive)
`bjobs`	more or less detailed information about pending and running jobs, and recently finished jobs
`bbjobs`	better bjobs ☺
`bhist`	info about jobs finished in the last hours/days
`bpeek`	display the standard output of a given job
`lsf_load`	show the CPU load of all nodes used by a job
`bjob_connect`	login to a node where your job is running
`bkill`	kill a job

Commands shown in blue are not standard LSF command but specific to Euler

Batch > Job control > Main bjobs options

(no option)	list all your jobs in all queues
`-p`	list only *pending* (waiting) jobs and indicate why they are pending
`-r`	list only *running* jobs
`-d`	list only *done* job (finished within the last hour)
`-l`	display status in *long* format
`-w`	display status in *wide* format
`-o "`*`format`*`"`	use custom *output* format (see LSF documentation for details)
`-J` *`jobname`*	show only job(s) called *jobname*
`-q` *`queue`*	show only jobs in a specific *queue*
`job-ID(s)`	list of job-IDs (this **must** be the last option)

Batch > Job control > bbjobs

- Displays more human-friendly information than `bjobs`
  - Requested number of cores, memory and scratch
  - Queue wait time
  - Wall-clock time
  - Number of tasks
- Shows the efficiency of a job
  - CPU utilization
  - Memory utilization

```
[leonhard@euler05 ~]$ bbjobs 8619658
Job information
  Job ID               : 8619658
  Status               : RUNNING
  Running on node      : 24*e2218 24*e2212 …
  User                 : leonhard
  Queue                : normal.24h
  Command              : #!/bin/csh -f; #BS…
  Working directory    : $HOME/cesm122-trun…
Requested resources
  Requested cores      : 144
  Requested runtime    : 23 h 59 min
  Requested memory     : 1024 MB per core, …
  Requested scratch    : not specified
Job history
  Submitted at         : 15:05 2015-07-07
  Started at           : 15:11 2015-07-07
  Queue wait time      : 0 h 6 min
Resource usage
  Updated at           : 16:10 2015-07-07
  Wall-clock           : 18 min
  Tasks                : 442
  Total CPU time       : 42 h 2 min
  CPU utilization      : 93.2 %
  Sys/Kernel time      : 0.0 %
  Total Memory         : 51235 MB
  Memory utilization   : 34.7 %
```

Batch > Job control > Main bkill options

`job-ID`	kill *job-ID*
`0`	kill all jobs (yours only)
`-J jobname`	kill most recent job called *jobname*
`-J jobname 0`	kill all jobs called *jobname*
`-q queue`	kill most recent job in *queue*
`-q queue 0`	kill all jobs in *queue*

Batch > Job output

- By default a job's output is stored in a file named "lsf.o*job-ID*" located in the submission directory
- In addition to your program's standard output, this file shows
  - The command that you submitted to the batch system
  - The queue where the job was dispatched
  - The date and time when the job started/ended
  - The name(s) of the compute node(s) that executed the job
  - The directory where your program ran
  - The CPU time and memory used by the job
  - The number of processes and threads executed by the job
- This can be used to fine-tune the resources requirements of your next jobs

Batch > Troubleshooting

- `bsub` rejects my job
  - If the error message is not self-explanatory, please report it to cluster-support@id.ethz.ch
- My job is stuck in the queue since XXX hours/days
  - Use `bjobs -p` to find out why your job is pending
  - "Individual host-based reasons" means that the resources requested by your jobs are not available at this time
  - Some resources may *never* become available (e.g. mem=10000000)
  - Some resource requirements may be *mutually exclusive*
- My job was sent to the "purgatory" queue
  - This queue is designed to catch jobs that were not submitted properly, either due to a user error or a bug in the batch system
  - **Always** report this type of problem to cluster-support@id.ethz.ch

Questions?