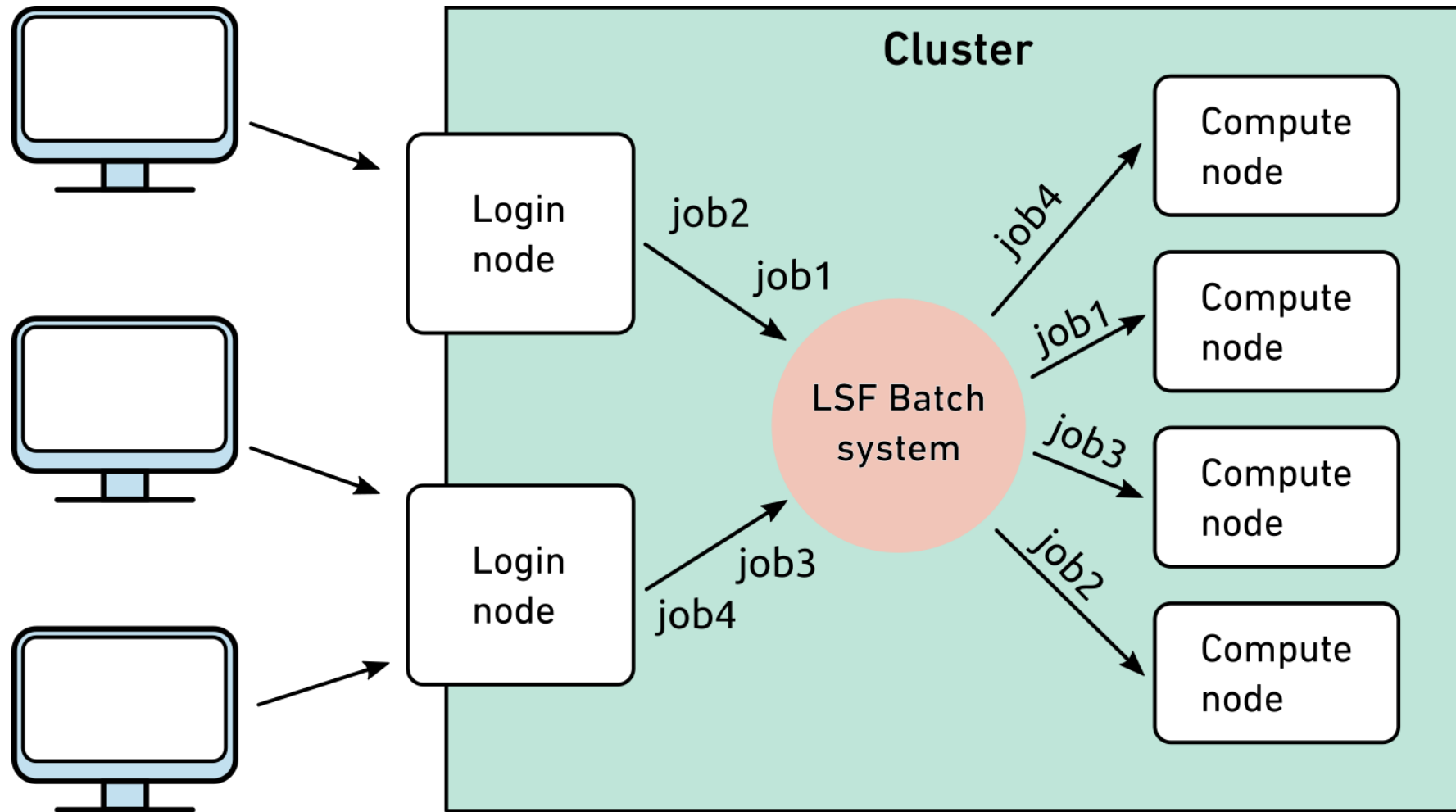


Using the batch system

Samuel Fux, Jarunan Panyasantisuk
High Performance Computing Group
Scientific IT Services, ETH Zurich



Batch > Overview



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
 - a shell script, passed via “<”
 - a [here document](#), passed via “<<”
 - a program, with its path
 - a command or program, with its arguments
 - multiple commands, enclosed in quotes
 - piped commands, enclosed in quotes
 - a command with I/O redirection, quoted
- We'll talk about `bsub`'s options later

```
cmd
< script
<< EOF ... EOF
/path/to/program
cmd arg1 arg2
"cmd1 ; cmd2"
"cmd1 | cmd2"
"cmd <in >out"
```

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 where the job was dispatched – e.g. “normal.4h”

Batch > Basic job submission > Examples

```
[sfux@eu-login-03 ~]$ bsub echo hello
Generic job.
Job <8146539> is submitted to queue <normal.4h>.

[sfux@eu-login-03 ~]$ bsub < hello.sh
Generic job.
Job <8146540> is submitted to queue <normal.4h>.

[sfux@eu-login-03 ~]$ bsub ./bin/hello
Generic job.
Job <8146541> is submitted to queue <normal.4h>.

[sfux@eu-login-03 ~]$ bsub "date; pwd; ls -l"
Generic job.
Job <8146542> is submitted to queue <normal.4h>.

[sfux@eu-login-03 ~]$ 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

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

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

```
[sfux@eu-login-03 ~]$ export OMP_NUM_THREADS=8
[sfux@eu-login-03 ~]$ bsub -n 8 ./hello_omp
Generic job.
Job <8147290> is submitted to queue <normal.4h>.

[sfux@eu-login-03 ~]$ unset OMP_NUM_THREADS

[sfux@eu-login-03 ~]$ 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.

[sfux@eu-login-03 ~]$ module load intel open_mpi
[sfux@eu-login-03 ~]$ 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

```
[sfux@eu-login-03 ~] bsub -J "hello[1-8]"
bsub> echo "Hello, I am job $LSB_JOBINDEX of $LSB_JOBINDEX_END"
bsub> ctrl-D
Job array.
Job <29976045> is submitted to queue <normal.4h>.
[sfux@eu-login-03 ~]$ bjobs
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
29976045	sfux	PEND	normal.4h	euler03		hello[1]	Oct 10 11:03
29976045	sfux	PEND	normal.4h	euler03		hello[2]	Oct 10 11:03
29976045	sfux	PEND	normal.4h	euler03		hello[3]	Oct 10 11:03
29976045	sfux	PEND	normal.4h	euler03		hello[4]	Oct 10 11:03
29976045	sfux	PEND	normal.4h	euler03		hello[5]	Oct 10 11:03
29976045	sfux	PEND	normal.4h	euler03		hello[6]	Oct 10 11:03
29976045	sfux	PEND	normal.4h	euler03		hello[7]	Oct 10 11:03
29976045	sfux	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	sfux	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

```
[sfux@eu-login-03 ~]$ bsub -W 15 -Is -R light /bin/bash
Generic job.
Job <27877012> is submitted to queue <light.5d>.
<<Waiting for dispatch ...>>
<<Starting on eu-c7-133-05>>
[sfux@eu-c7-133-05 ~]$ pwd
/cluster/home/sfux
[sfux@eu-c7-133-05 ~]$ hostname
eu-c7-133-05
[sfux@eu-c7-133-05 ~]$ exit
exit
[sfux@eu-login-03 ~]$
```


Batch > Job control commands

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

Commands shown in **red** are not standard LSF command but specific to the HPC clusters at ETH

Batch > Job control > Main bjobs options

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

```
[sfux@eu-login-05 ~]$ bbjobs 146773638
Job ID                               : 146773638
Status                               : RUNNING
Running on node                       : 24*eu-c7-113-09
User                                  : sfux
Queue                                 : normal.120h
Command                              : python
launch_solver.py -p /cluster/home/sfux[...]
Working directory                     :
$HOME/python/solver/test1
Requested resources
  Requested cores                     : 24
  Requested runtime                   : 120 h 0 min
  Requested memory                    : 2500 MB per core,
60000 MB total
  Requested scratch                   : not specified
  Dependency                         : -
Job history
  Submitted at                       : 10:03 2020-10-13
  Started at                         : 10:03 2020-10-13
  Queue wait time                    : 18 sec
Resource usage
  Updated at                         : 11:44 2020-10-13
  Wall-clock                         : 1 h 41 min
  Tasks                             : 52
  Total CPU time                     : 39 h 47 min
  CPU utilization                     : 98.4 %
  Sys/Kernel time                    : 0.0 %
  Total resident Memory              : 40788 MB
  Resident memory utilization        : 68.0 %
```

Batch > Job control > Main bkill options

<i>job-ID</i>	kill <i>job-ID</i>
0	kill <u>all</u> jobs (yours only)
-J <i>jobname</i>	kill most recent job called <i>jobname</i>
-J <i>jobname</i> 0	kill all jobs called <i>jobname</i>
-q <i>queue</i>	kill most recent job in <i>queue</i>
-q <i>queue</i> 0	kill all jobs in <i>queue</i>

Batch > Job output

- By default a job's output is stored in a file named "lsf.ojob-*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

Sender: LSF System <lsfadmin@eu-ms-024-36>
Subject: Job 146799736: <./test.py> in cluster <euler> Done

Job <./test.py> was submitted from host <eu-login-30> by user <sfux> in cluster <euler> at Tue Oct 13 11:28:50 2020
Job was executed on host(s) <eu-ms-024-36>, in queue <normal.4h>, as user <sfux> in cluster <euler> at Tue Oct 13 11:29:00 2020
</cluster/home/sfux> was used as the home directory.
</cluster/home/sfux/test/python/matplotlib> was used as the working directory.
Started at Tue Oct 13 11:29:00 2020
Terminated at Tue Oct 13 11:29:02 2020
Results reported at Tue Oct 13 11:29:02 2020

Your job looked like:

```
-----  
# LSBATCH: User input  
./test.py  
-----
```

Successfully completed.

Resource usage summary:

CPU time :	16182 sec.
Max Memory :	879.00 MB
Average Memory :	602.00 MB
Total Requested Memory :	1000.00 MB
Delta Memory :	121.00 MB
Max Swap :	-
Max Processes :	4
Max Threads :	5
Run time :	18423 sec.
Turnaround time :	19004 sec.

The output (if any) follows:

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?