Batch System Introduction

Thorsten Zirwes, SCC, KIT
Most information given by this talk can be found at [http://bwhpc-c5.de/wiki: Batch_Jobs](http://bwhpc-c5.de/wiki: Batch_Jobs).
## How to read the following slides

<table>
<thead>
<tr>
<th>Abbreviation/Colour code</th>
<th>Full meaning</th>
</tr>
</thead>
</table>
| $ command -option value  | $ = prompt of the interactive shell  
The full prompt may look like:  
  user@machine:path$  
The command has been entered in the interactive shell session |
| <integer>                | <> = Placeholder for integer, string etc |
| <string>                 |              |
Resource management

- Jobs are **not** executed by the user

- Instead, there is a management system (Batch System)
  - **workload manager (scheduler)**
    - scheduling, managing, monitoring, reporting
    - MOAB
  - **resource manager**
    - control over jobs and distributed compute nodes
    - SLURM (bwUniCluster, ForHLR I)
    - TORQUE (ForHLR II, all bwForClusters)
(1) User creates a job script and submits it to MOAB via the “msub” command

```
#!/bin/bash
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=00:01:00
#MSUB -l pmem=50mb
./your_simulation
```

(2) MOAB parses the job script: → where & when to run job

(3) Job execution: delegated to resource manager on the node

(4) The resource manager (TORQUE/SLURM) executes the job and communicates status information to MOAB
Resource and workload manager (2)

- All clusters:
  - compute job will **only** be processed by the batch system
  - Running jobs on login nodes not allowed

- Waiting time:
  - fairshare based queue

- depends on:
  - your job demands
  - your demand history
  - your university's share (bwUniCluster only)
Job's life circle

1. Setup `job.sh`:

```bash
#!/bin/bash
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=00:01:00
#MSUB -l pmem=50mb
./your_simulation
```

Options for the job

Actual work to be executed on the cluster

2. Submit job **ONLY with “msub”**

```bash
$ msub job.sh <job_ID>
```

3. Job pending/running

```bash
$ showq <job_ID> state "Idle" → "Running"
```

4. Job is finished → check output (default job name)

- `bwUniCluster/ForHLR1/2`: `job_{uc1,fh1,fh2}_{<job_ID>}.out`
- `bwForCl. JUSTUS/NEMO`: `<jobscriptname>.o<job_ID>`
- `bwForCl. BinAC`: `<jobscriptname>.o<job_ID>`
1. Job Submit: msub options


- msub options: command line or in your job script

<table>
<thead>
<tr>
<th>Command line</th>
<th>Script</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-l resources</td>
<td>#MSUB -l resources</td>
<td>Defines the resources that are required by the job. See the description below for this important flag.</td>
</tr>
<tr>
<td>-N name</td>
<td>#MSUB -N name</td>
<td>Gives a user specified name to the job.</td>
</tr>
<tr>
<td>-q queue</td>
<td>#MSUB -q queue</td>
<td>Defines the queue class</td>
</tr>
<tr>
<td>-m bea</td>
<td>#MSUB -m bea</td>
<td>Send email when job begins (b), ends (e) or aborts (a).</td>
</tr>
</tbody>
</table>
1. Job Submit: \textit{resource\_list}

- http://www.bwhpc-c5.de/wiki/index.php/Batch_Jobs#msub\_l_resource\_list

<table>
<thead>
<tr>
<th>Resource</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>-l nodes=2:ppn=16</td>
<td>Number of \textit{nodes} and number of \textit{processes per node}</td>
</tr>
<tr>
<td>-l walltime=600</td>
<td>Wall-clock time (seconds)</td>
</tr>
<tr>
<td>-l walltime=00:01:30:00</td>
<td>DD:HH:MM:SS format</td>
</tr>
<tr>
<td>-l pmem=1000mb</td>
<td>Max. amount of physical memory used by one process of the job (kb,mb,gb)</td>
</tr>
<tr>
<td>-l mem=1000mb</td>
<td>Max. total physical memory used by the job</td>
</tr>
</tbody>
</table>

Use these options in the job script:

```
#!/bin/bash
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=00:01:00
#MSUB -l pmem=50mb
...
```

Or use them with msub:

```
$ msub -l nodes=1:ppn=1,walltime=00:01:00,pmem=1gb <job_script>
```
msub -q queues (bwUniCluster)

www.bwhpc-c5.de/wiki/index.php/Batch_Jobs_-_bwUniCluster_Features#msub_-q_queues

<table>
<thead>
<tr>
<th>queue</th>
<th>default resources</th>
<th>MIN resources</th>
<th>MAX resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>develop</td>
<td>procs=1, pmem=4000mb</td>
<td>nodes=1</td>
<td>walltime=00:30:00, nodes=1:ppn=16</td>
</tr>
<tr>
<td>singlenode</td>
<td>procs=1, pmem=4000mb</td>
<td>walltime=00:30:01, nodes=1</td>
<td>walltime=3:00:00:00, nodes=1:ppn=16</td>
</tr>
<tr>
<td>multinode</td>
<td>procs=1, pmem=4000mb</td>
<td>nodes=2</td>
<td>walltime=2:00:00:00, nodes=128:ppn=28</td>
</tr>
</tbody>
</table>

**Manual queue selection**

<table>
<thead>
<tr>
<th>queue</th>
<th>default resources</th>
<th>MIN resources</th>
<th>MAX resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>verylong</td>
<td>procs=1, pmem=4000mb</td>
<td>walltime=3:00:00:01, nodes=1</td>
<td>walltime=6:00:00:00, nodes=1:ppn=16</td>
</tr>
<tr>
<td>fat</td>
<td>procs=1, pmem=32000mb</td>
<td>nodes=1</td>
<td>walltime=3:00:00:00, nodes=1:ppn=32</td>
</tr>
</tbody>
</table>

**Automatic queue choosing** - walltime, nodes, processes
Tutorial 1a

■ **Goal:** Use the Batch System to execute "`printenv`" on the cluster

■ **1)** Create a file „submit_script.sh“ and set the following options in the submit script:
  - nodes=1:ppn=1
  - pmem=50mb
  - walltime=00:01:00

■ **2)** After defining these options, write the command to be executed in the jobscript ("`printenv`")

■ **3)** Save the jobscript and submit it to the Batch System with

  ```bash
  $ msub -l advres=workshop_single.50 submit_script.sh
  ```

  - You can use "`showq`" to see the status of your job

■ **4)** Look in the output file of your job (`job_uc1_<jobID>.out`) for variables starting with „`MOAB_`“. These can be used to get information on how the job was started
Create a file named „submit_script.sh“ with the following content:

```bash
#!/bin/bash
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=00:01:00
#MSUB -l pmem=50mb

printenv
```

Save the file and submit it with

```
$ msub -l advres=workshop_single.50 submit_script.sh
```

In the output file, you can find the MOAB variables:

- For example: „MOAB_CLASS=develop“ means:

  In the job script, we have not defined a queue class but the job was automatically submitted to the „develop“ queue
Tutorial 1b

1) Modify your submit script so that instead of \texttt{\texttt{\texttt{printenv}}} the value of \texttt{MOAB\_PROCCOUNT} is printed (Hint: Use \texttt{echo})

Submit your job again, but this time use msub to specify the number of processes:

$ msub -l nodes=1:ppn=2 -l advres=workshop_single.50 submit_script.sh

2) Check in your output file if the number of processes is 1 as specified in the submit script or 2 as specified with \texttt{msub}
Modify your submit script to print the variable **MOAB_PROCCOUNT**

```bash
#!/bin/bash
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=00:01:00
#MSUB -l pmem=50mb

echo $MOAB_PROCCOUNT
```

Save the file and submit it with

```
$ msub -l nodes=1:ppn=2 -l advres=workshop_single.50 submit_script.sh
```

In the output file the number of processes is printed:

```
job_uc1_<job-ID>.out
```

```
2
```

The options given directly to msub take precedence over the options in the submit script.

Read also: www.bwhpc-c5.de/wiki/index.php/Batch_Jobs#Environment_Variables_for_Batch_Jobs
Check status of your jobs (1)

- after submission → msub returns <job-ID>

```bash
$ msub job.sh
```

659562

- commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$ showq</code></td>
<td>Show status of all your jobs</td>
</tr>
<tr>
<td><code>$ showq -n</code></td>
<td>Show status of all your jobs, showing job names</td>
</tr>
<tr>
<td><code>$ showq -r</code></td>
<td>All your active (running) jobs</td>
</tr>
<tr>
<td><code>$ showq -i</code></td>
<td>eligible(idle) jobs</td>
</tr>
<tr>
<td><code>$ showq -b</code></td>
<td>blocked jobs</td>
</tr>
<tr>
<td><code>$ showq -c</code></td>
<td>completed jobs</td>
</tr>
<tr>
<td><code>$ checkjob &lt;job-ID&gt;</code></td>
<td>Get detailed information of your job</td>
</tr>
<tr>
<td></td>
<td>→ explains why your job is pending</td>
</tr>
<tr>
<td><code>$ canceljob &lt;job-ID&gt;</code></td>
<td>Cancel the job with &lt;job-ID&gt;</td>
</tr>
</tbody>
</table>
Check status of your jobs (2)

Command “showq”:

```bash
$ showq
```

**active jobs**

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USERNAME</th>
<th>STATE</th>
<th>PROCS</th>
<th>REMAINING</th>
<th>STARTTIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>xy_ab1234</td>
<td>Running</td>
<td>1</td>
<td>00:04:58</td>
<td>Thu Jan 22 19:21:56</td>
</tr>
</tbody>
</table>

1 active job

**eligible jobs**

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USERNAME</th>
<th>STATE</th>
<th>PROCS</th>
<th>WCLIMIT</th>
<th>QUEUETIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>12346</td>
<td>xy_ab1234</td>
<td>Idle</td>
<td>1</td>
<td>00:05:00</td>
<td>Thu Jan 22 19:21:47</td>
</tr>
</tbody>
</table>

1 eligible job

**blocked jobs**

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USERNAME</th>
<th>STATE</th>
<th>PROCS</th>
<th>WCLIMIT</th>
<th>QUEUETIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>12347</td>
<td>xy_ab1234</td>
<td>Idle</td>
<td>1</td>
<td>00:05:00</td>
<td>Thu Jan 22 19:21:47</td>
</tr>
</tbody>
</table>

1 blocked job

Check why job can not start:

- `checkjob <job_ID>` get information of your job
- `checkjob -v -v -v <job_ID>` All detailed information
States:

- **Pre-execution states:**
  - **Idle**: Job is waiting for free resources
  - **Deferred**: Job cannot be scheduled right now
  - **BatchHold**: Job is blocked by scheduler..
    - Reasons: no resources, limits, failure

  Idle → Running → Canceling == OK

  Idle → Deferred → Idle → Deferred → ... → BatchHold → Canceling

- **Execution states**
  - **Starting**: Job is starting
  - **Running**: Job is running
  - **Suspended**: Job has exceeded specified walltime
Check status of your jobs (4)

example: MAXPROC limit

Submitted job (bwUniCluster)

$ msub -l nodes=1:ppn=32 -q fat <jobscript>
12345

showq:

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USERNAME</th>
<th>STATE</th>
<th>PROCS</th>
<th>WCLIMIT</th>
<th>QUEUETIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>xy_ab1234</td>
<td>Idle</td>
<td>32</td>
<td>00:05:00</td>
<td>Fri Jan 23 15:31:05</td>
</tr>
</tbody>
</table>

checkjob -v -v -v 12345:

State: Idle
Creds: user:xy_ab1234 group:xyz account:kit class:fat
...
NOTE: job violates constraints for partition uc1 (job 12345 violates active HARD MAXPROC limit of 64 for class fat user partition ALL (Req: 32  InUse: 64))

BLOCK MSG: job 12345 violates active HARD MAXPROC limit of 64 for class fat user partition ALL (Req: 32  InUse: 64) (recorded at last scheduling iteration)
Check status of your jobs (6)

example: organisation limits

Submitted job (bwUniCluster)

```
$ msub -l nodes=1:ppn=1 <jobscript>
55555
```

showq:

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USERNAME</th>
<th>STATE</th>
<th>PROCS</th>
<th>WCLIMIT</th>
<th>QUEUETIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>55555</td>
<td>xy_ab1234</td>
<td>Idle</td>
<td>1</td>
<td>00:10:00</td>
<td>Fri Jan 21 15:31:05</td>
</tr>
</tbody>
</table>

checkjob -v -v -v 55555:

State: Idle

class: develop

**BLOCK MSG:** job 55555 violates active **SOFT MAXPROC limit of 1000** for acct `university_X` partition ALL (Req: 1 InUse: 1010)
Change status of your jobs

- Change command: **mjobctl**

  - `mjobctl -c <job_ID>`
  - `mjobctl -c -w state=Idle`
  - `mjobctl -c -w state=Running`
  - `mjobctl -c -w state=BatchHold`
  - `mjobctl -c -w user=$USER`

- cancel the job (new command)
- cancel ALL idle jobs
- cancel ALL running jobs
- cancel ALL hold jobs
- cancel ALL your jobs!
Tutorial 2

1) Modify your submit script so that it executes a command to wait for 600 seconds (sleep 600)

Increase the walltime to 10 minutes and give your job a name

Submit your job script

Use showq to check where your job is

Use showq -n to see the name of your job

Use canceljob <job-ID> or mjobctl -c <job-ID> to cancel your job
Modify your submit script

```bash
#!/bin/bash
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=00:10:00
#MSUB -l pmem=50mb
#MSUB -N myJobName

sleep 600
```

Save the file and submit it with

```
$ msub -l advres=workshop_single.50 submit_script.sh
```

Show the status of your jobs with `showq`

```
active jobs------------------------
JOBID   USERNAME  STATE  PROCS   REMAINING          STARTTIME
11705861  mb1337  Running  1    00:08:12  Tue Apr  5 14:30:01
```

Use `checkjob -v -v -v <job-ID>` to see further information

Use `canceljob <job-ID>` or `mjobctl -c <job-ID>` to cancel your job
Interactive jobs on login nodes are not permitted

**Solution:** interactive moab jobs

- Access to compute nodes and work on them interactively

**HowTo:**

```bash
$ msub -I -V -l nodes=1:ppn=1,walltime=02:00:00
```

Attention: Restrictions may apply (shared nodes, single node etc.)

- `-I` = interactive
- `-V` = all environment variables are exported to the compute node

**Details @ bwUniCluster**

- www.bwhpc-c5.de/wiki/index.php/Batch_Jobs_-_bwUniCluster_Features#Interactive_Jobs

**Details @ bwForClusters**

- see bwHPC Wiki for bwForCluster of interest
Moab: Interactive jobs (Example)

mb1337@uc1n996: msub -I -V -l nodes=1:ppn=1,walltime=00:02:00

Job is waiting to start

salloc: Job is in held state, pending scheduler release
salloc: Pending job allocation 11631324
salloc: job 11631324 queued and waiting for resources

salloc: job 11631324 has been allocated resources
salloc: Granted job allocation 11631324

mb1337@uc1n257$ {Now you can work on the compute node}

Job running. You are now on a compute node

allocation has been revoked.
srun: Job step aborted: Waiting up to 32 seconds for job step to finish.
srun: error: uc1n257: task 0: Killed
mb1337@uc1n996:

Requested time for the interactive job ran out

Back on the login node