

# Batch System Introduction

Thorsten Zirwes, SCC, KIT



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

Hochschule  
für Technik  
Stuttgart



**Hochschule Esslingen**  
University of Applied Sciences

Universität  
Konstanz



UNIVERSITÄT  
MANNHEIM



Universität Stuttgart

EBERHARD KARLS  
UNIVERSITÄT  
TÜBINGEN



**KIT**  
Karlsruher Institut für Technologie



ulm university universität  
**uulm**



# Reference: bwHPC-C5 Best Practices Repository

- Most information given by this talk can be found at <http://bwhpc-c5.de/wiki>:

- Batch\_Jobs



The screenshot displays the bwHPC Wiki interface. On the left is a navigation menu with categories such as 'bwHPC Wikid', 'Best Practice Guides', 'bwHPC tier 3', 'bwHPC tier 2', 'bwHPC Support Services', 'bwHPC Data Storage', 'Tools', and 'Personal tools'. The main content area is titled 'Main Page' and features a 'Knowledge Base Wiki of Baden-Württemberg's HPC services' header. Below this, there is a welcome message and a list of HPC services and data storage services, including 'bwUniCluster', 'bwForCluster JUSTUS', and 'Best Practices Repository'.

# Material: Slides & Scripts

- <https://indico.scc.kit.edu/indico/event/363/>
- @bwUniCluster:  
/pfs/data1/software\_uc1/bwhpc/kit/workshop/2018-04-10

## How to read the following slides

Abbreviation/Colour code	Full meaning
<code>\$ command -option value</code>	<code>\$</code> = <b>prompt</b> of the interactive shell The full prompt may look like: <code>user@machine:path\$</code> The command has been entered in the interactive shell session
<code>&lt;integer&gt;</code> <code>&lt;string&gt;</code>	<code>&lt;&gt;</code> = Placeholder for integer, string etc

# Batch System

# 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)

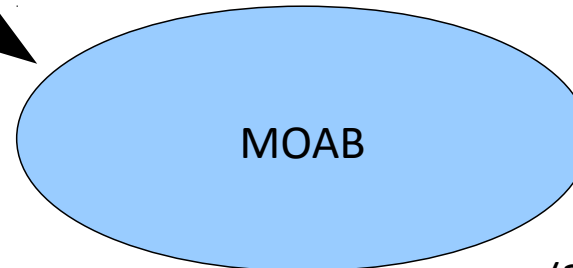
# Resource and workload manager (1)

(1) User creates a **job script** and submits it to MOAB via the “**msub**” command

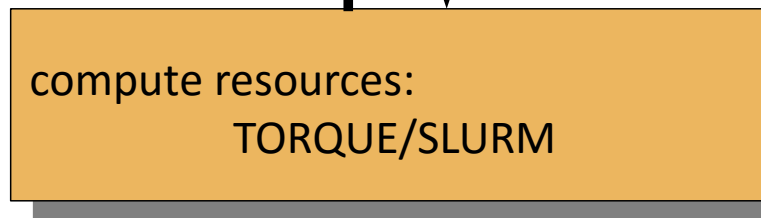
```
#!/bin/bash
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=00:01:00
#MSUB -l pmem=50mb

./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_script.sh`:

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

./your_simulation
```

1) Options for the job

2) Actual work to be executed on the cluster

## 2. Submit job **ONLY** with “msub”

```
$ msub job_script.sh
<job_ID>
```

## 3. Job pending/running

```
$ 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   : <jobscripname>.o<job_ID>
bwForCl. BinAC         : <jobscripname>.o<job_ID>
```



# 1. Job Submit: msub options

- [http://www.bwhpc-c5.de/wiki/index.php/Batch\\_Jobs#msub\\_Command](http://www.bwhpc-c5.de/wiki/index.php/Batch_Jobs#msub_Command)
- msub options: command line or in your job script

Command line	Script	Purpose
<code>-l</code> <i>resources</i>	<code>#MSUB -l</code> <i>resources</i>	Defines the resources that are required by the job. See the description below for this important flag.
<code>-N name</code>	<code>#MSUB -N name</code>	Gives a user specified name to the job.
<code>-q queue</code>	<code>#MSUB -q queue</code>	Defines the queue class
<code>-m bea</code>	<code>#MSUB -m bea</code>	Send email when job begins (b), ends (e) or aborts (a).

# 1. Job Submit: *resource\_list*

■ [http://www.bwhpc-c5.de/wiki/index.php/Batch\\_Jobs#msub\\_-l\\_resource\\_list](http://www.bwhpc-c5.de/wiki/index.php/Batch_Jobs#msub_-l_resource_list)

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



Example for bwUni cluster:

The cluster consists of 512 compute nodes

Each compute node has two CPUs with 8 cores each

# 1. Job Submit: *resource\_list*

■ [http://www.bwhpc-c5.de/wiki/index.php/Batch\\_Jobs#msub\\_-l\\_resource\\_list](http://www.bwhpc-c5.de/wiki/index.php/Batch_Jobs#msub_-l_resource_list)

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

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](http://www.bwhpc-c5.de/wiki/index.php/Batch_Jobs_-_bwUniCluster_Features#msub_-q_queues)

<i>queue</i>	<i>default resources</i>	<i>MIN resources</i>	<i>MAX resources</i>
<b>automatic queue routing</b>			
develop	<i>procs=1, pmem=4000mb</i>	<i>nodes=1</i>	<i>walltime=00:30:00, nodes=1:ppn=16</i>
singlenode	<i>procs=1, pmem=4000mb</i>	<i>walltime=00:30:01, nodes=1</i>	<i>walltime=3:00:00:00, nodes=1:ppn=16</i>
multinode	<i>procs=1, pmem=4000mb</i>	<b><i>nodes=2</i></b>	<b><i>walltime=2:00:00:00, nodes=128:ppn=28</i></b>
<b>Manual queue selection</b>			
verylong	<i>procs=1, pmem=4000mb</i>	<i>walltime=3:00:00:01, nodes=1</i>	<i>walltime=6:00:00:00, nodes=1:ppn=16</i>
fat	<i>procs=1, pmem=32000mb</i>	<i>nodes=1</i>	<i>walltime=3:00:00:00, nodes=1:ppn=32</i>

■ **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

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

[?????]
```

- **2)** After defining these options, insert the command to be executed at the end of the jobscript (“**printenv**”)
- **3)** Save the jobscript and submit it to the Batch System with  

```
$ msub 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

# Tutorial 1a - Solution

- Create a file named „**submit\_script.sh**“ with the following content:

```
#!/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 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 “printenv” the value of „**MOAB\_PROCCOUNT**“ is printed (Hint: Use **echo**)
- Submit your job again, but this time use **msub** to specify the number of processes:

```
$ msub -l nodes=1:ppn=2 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 directly with **msub**

# Tutorial 1b - Solution

- Modify your submit script to print the variable **MOAB\_PROCCOUNT**

```
#!/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 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](http://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>

```
$ msub job.sh
```

```
659562
```

- **commands:**

\$ showq	Show status of all your jobs
\$ showq -n	Show status of all your jobs, showing job names
\$ showq -r \$ showq -i \$ showq -b \$ showq -c	All your active (running) jobs eligible(idle) jobs blocked jobs completed jobs
\$ checkjob <job-ID>	Get detailed information of your job → explains why your job is pending
\$ canceljob <job-ID>	Cancel the job with <job-ID>

## Check status of your jobs (2)

### ■ Command “showq”:

```
$ showq

active jobs-----
JOBID          USERNAME      STATE      PROCS    REMAINING      STARTTIME
12345          xy_ab1234    Running    1        00:04:58      Thu Jan 22 19:21:56
1 active job

eligible jobs-----
JOBID          USERNAME      STATE      PROCS    WCLIMIT        QUEUETIME
12346          xy_ab1234    Idle       1        00:05:00      Thu Jan 22 19:21:47
1 eligible job

blocked jobs-----
JOBID          USERNAME      STATE      PROCS    WCLIMIT        QUEUETIME
12347          xy_ab1234    Idle       1        00:05:00      Thu Jan 22 19:21:47
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

# Check status of your jobs (3)

## ■ 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:

#### **blocked jobs-----**

JOBID	USERNAME	STATE	PROCS	WCLIMIT	QUEUETIME
12345	xy_ab1234	Idle	32	00:05:00	Fri Jan 23 15:31:05

**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 <jobscrip>  
55555
```

showq:

```
blocked jobs-----  
JOBID          USERNAME      STATE  PROCS    WCLIMIT      QUEUETIME  
55555          xy_ab1234    Idle   1        00:10:00    Fri Jan 21 15:31:05
```

checkjob -v -v -v 55555:

```
State: Idle  
class:develop  
...
```

limits for university\_X!

```
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 with **msub**
- Use **showq** to check the status of your job
- Use **showq -n** to see the name of your job
- Use **canceljob <job-ID>** or **mjobctl -c <job-ID>** to cancel your job

# Tutorial 2 - Solution

- Modify your submit script

```
#!/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 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



# Moab: Interactive jobs

- **Jobs on login nodes are not permitted**

- **Solution: interactive moab jobs**

- Access compute nodes and work on them interactively

- HowTo:

```
$ 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](http://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
```

Job running. You are now on a compute node

```
mb1337@uc1n257$ {Now you can work on the 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
```

Requested time for the interactive job ran out

```
mb1337@uc1n996:
```

Back on the login node