bw | HPC – C5

bwHPC Cluster: Batch System

Funding: Baden-Württemberg
MINISTERIUM FÜR WISSENSCHAFT, FORSCHUNG UND KUNST

Resource management

- **Compute job** – sequence of commands and programs
- **“Resources”:**
  - working time (walltime)
  - nodes, cores (nodes:ppn)
  - memory (pmem, mem)
  - disk space

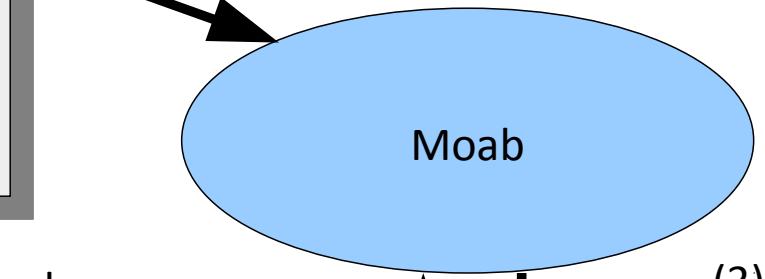
Resource management

- Components of management system (Batch System)
  - **resource manager**
    - control over jobs and distributed compute nodes
    - SLURM (bwUniCluster)
    - TORQUE (bwForCluster JUSTUS)
  - **workload manager (scheduler)**
    - scheduling, managing, monitoring, reporting
    - Moab

Resource and workload manager

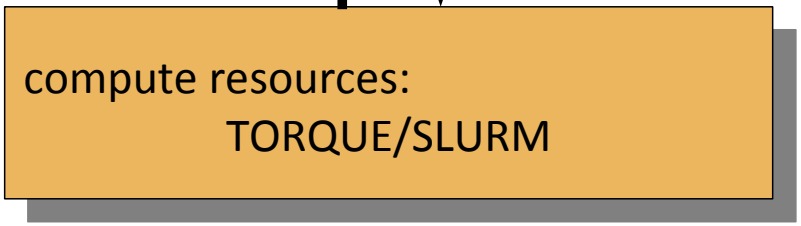
```
#/bin/bash
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=00:01:00
#MSUB -l pmem=50mb

echo “Hello from job”
exit 0
```

(2) Moab parses the job script: --> where & when to run job



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

(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

Job's life circle

- Setup job script:

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

echo “Hello from job”
exit 0
```

- Submit job to workload manager **ONLY with “msub”**

```
$ msub <resource_options> <job_script>
<job_ID>
```

- Job waits for free resources in queue

```
$ showq
<job_ID> state “Idle” → “Running”
```

- Job is finished → check output (default job name)

```
bwUniCluster: job_uc1_<job_ID>.out
```

msub-options

- http://www.bwhpc-c5.de/wiki/index.php/Batch_Jobs#msub_Command

- Msub-options: **command line / job script**

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

→ command line option overwrites script option

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 **nodes** and number of **processes per node**
`-l walltime=600`	Wall-clock time (seconds)
`-l walltime=01:30:00`	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

→ resources can be combined, but must be separated by comma:

```
$ 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

queue	*default resources*	*MIN resources*	*MAX resources*
automatic queue choosing			
develop	*procs*=1, *mem*=4000mb	nodes=1	*walltime*=00:30:00, *nodes*=1:*ppn*=16
singlenode	*procs*=1, mem=4000*mb*	walltime=00:30:01, nodes=1	*walltime*=3:00:00:00, *nodes*=1:*ppn*=16
multinode	***procs*=1, *mem*=4000mb**	***nodes*=2**	***walltime*=2:00:00:00, *nodes*=16:*ppn*=16**
explicit queue definition			
verylong	*procs*=1, *mem*=4000mb	*walltime*=3:00:00:01, *nodes*=1	*walltime*=6:00:00:00, *nodes=1:*ppn=*16*
fat (fat nodes)	*procs*=1, *mem*=32000mb	*nodes*=1	*walltime*=3:00:00:00, *nodes*=1:*ppn*=32

- **Automatic queue choosing** - walltime, nodes, processes

Interactive jobs

- **Common**
  - Access to compute nodes
    → start your application direct there
  - Specify resources what you need
  - Auto logout when job is finished
  - Submit job: "**`msub -I -V`**"

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

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

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

Check status of your jobs

- Job → submission → <job_ID>

```
$ msub job.sh

12345
```

- **Check commands:**
  - `showq` — all your active, eligible, blocked, and/or recently completed jobs
  - `showq -r` — active (running) jobs
  - `showq -i` — eligible (idle) jobs
  - `showq -b` — blocked jobs
  - `showq -c` — recently completed jobs

  - `showstart <job_ID>` — Get information about start time of job with <job_ID>
  - `showstart 16@12:00:00` — Info about start time of 16 procs with run time of 12 hours

Check status of your jobs

- **Command “showq”:**

```
$ showq

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

eligible jobs----------------------
JOBID          USERNAME     STATE   PROCS   REMAINING            STARTTIME
12346          ///            Idle     1        00:04:58   Thu Jan 22 19:21:56
1 eligible job

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

Check status of your jobs

- **Check, why job can not start:**

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

Check status of your jobs (example, MAXNODE limit)

- Job with ID 12345

checkjob 12345:

```
State: Idle
class:quick
...
NodeCount:  5
...
```

```
BLOCK MSG: job 12345 violates active
           HARD MAXNODE limit of 2 for class quick user partition ALL
           (Req: 5  InUse: 0) (recorded at last scheduling iteration)
```

Change status of your jobs

- **Change commands**

`canceljob <job_ID>`	cancel the job with <job_ID>
`canceljob ALL`	cancel ALL your jobs!!!
`mjobctl -c <job_ID>`	cancel the job (new command)
`mjobctl -c -w state=Idle`	cancel ALL idle jobs
`mjobctl -c -w state=Running`	cancel ALL running jobs
`mjobctl -c -w state=BatchHold`	cancel ALL hold jobs

Simple Example Job

```
#!/bin/bash
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=5:00
#MSUB -l pmem=2gb
#MSUB -N serial-test

~/hello
```

- lines starting with **#** are ignored by the shell
- lines starting with **#MSUB** are interpreted by Moab
- lines not starting with **#MSUB** are ignored by Moab
  - → file commands are equal to:

```
$ msub -l nodes=1:ppn=1,walltime=00:05:00,pmem=2gb \
-N serial-test  jobscript.moab
```

Common Problems

Common problems

- Wrong „ppn“ setting:

```
$ msub -l nodes=3:ppn=38,walltime=00:01:00,pmem=1gb <job_script>
```

- „mem“ instead of „pmem“:

```
$ msub -l nodes=4:ppn=16,walltime=00:01:00,mem=1gb <job_script>
```

- Wrong queue

- Data in $HOME instead of $WORK

- `# MSUB` instead of `#MSUB` he space...)

Thank you!

- **Support (bwUniCluster):** bwunicluster-hotline@lists.kit.edu
- **Support (JUSTUS) :** CompChem@bwhpc-c5.de

- **Ticket system:** http://www.support.bwhpc-c5.de

Addition:
chain jobs, job array

Chain jobs

- Submit a chain of jobs
  → each job runs after the previous job has completed

- **bwUniCluster**
  - www.bwhpc-c5.de/wiki/index.php/Batch_Jobs_-_bwUniCluster_Features#Chain_Jobs
  - example script for Moab + SLURM

- **bwForCluster JUSTUS**
  - www.bwhpc-c5.de/wiki/index.php/Batch_Jobs_-_bwForCluster_Chemistry_Features#Chain_jobs
  - example script for Moab + TORQUE
  - change dependencies
    - after
    - afterany
    - afterok
    - afternotok

Job arrays

- **Only on bwForCluster JUSTUS**
  - www.bwhpc-c5.de/wiki/index.php/Batch_Jobs_-_bwForCluster_Chemistry_Features#Job_arrays
  - to run same job script with different parameters
- **Submit a job array**

```
$ msub -t [<jobname>]<indexlist>[%<limit>] jobarray.sh
```

- **Count and order of submitted sub-jobs (`<indexlist>`)**

```
$ msub -t job1.[1-10:2] jobarray.sh
$ msub -t job1.[2-10:2] jobarray.sh
```

- **Special environment variables**
  - MOAB_JOBARRAYINDEX and MOAB_JOBARRAYRANGE
- **Check job array** → only one job_ID
  - `checkjob <job_ID>` or `checkjob <job_ID>[<index>]`
  - **output files:** <jobname>.o(e)<job_ID>-<index>