bw|HPC – C5

File Systems, Software and Batch System

Simon Raffeiner, Scientific Computing and Simulation Dept., SCC, KIT

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Hochschule für Technik Stuttgart

UNI FREIBURG

UNIVERSITÄT HOHENHEIM 1818

Hochschule Esslingen University of Applied Sciences

Universität Konstanz

UNIVERSITÄT MANNHEIM

Universität Stuttgart

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

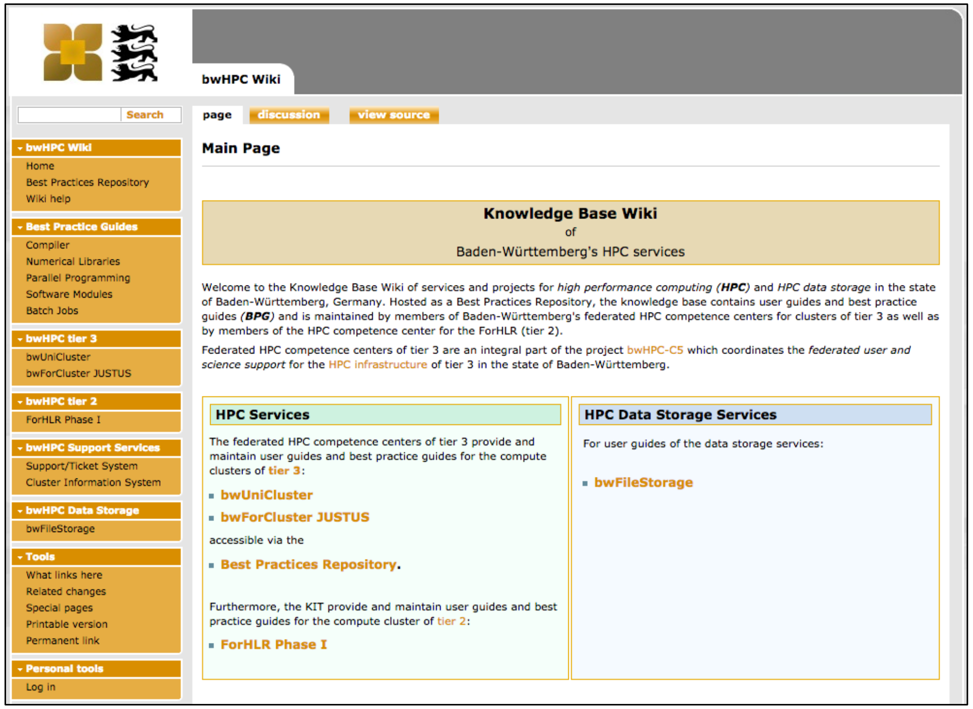
KIT Karlsruher Institut für Technologie

ulm university universität uulm

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

Reference: bwHPC-C5 Best Practices Repository

- Most information given by this talk can be found at http://bwhpc-c5.de/wiki:
  - Category:Hardware_and_Architecture
  - Environment_Modules
  - Batch_Jobs



Main Page

Knowledge Base Wiki
of
Baden-Württemberg's HPC services

Welcome to the Knowledge Base Wiki of services and projects for *high performance computing* (***HPC***) and *HPC data storage* in the state of Baden-Württemberg, Germany. Hosted as a Best Practices Repository, the knowledge base contains user guides and best practice guides (***BPG***) and is maintained by members of Baden-Württemberg's federated HPC competence centers for clusters of tier 3 as well as by members of the HPC competence center for the ForHLR (tier 2).

Federated HPC competence centers of tier 3 are an integral part of the project bwHPC-C5 which coordinates the *federated user and science support* for the HPC infrastructure of tier 3 in the state of Baden-Württemberg.

HPC Services

The federated HPC competence centers of tier 3 provide and maintain user guides and best practice guides for the compute clusters of tier 3:

- **bwUniCluster**
- **bwForCluster JUSTUS**

accessible via the

- **Best Practices Repository.**

Furthermore, the KIT provide and maintain user guides and best practice guides for the compute cluster of tier 2:

- **ForHLR Phase I**

HPC Data Storage Services

For user guides of the data storage services:

- **bwFileStorage**

Material: Slides & Scripts

- https://indico.scc.kit.edu/indico/event/263/
- @bwUniCluster/ForHLR I/ForHLR II:
  /pfs/data1/software_uc1/bwhpc/kit/workshop/2016-12-06

How to read the following slides

Abbreviation/Colour code	Full meaning
`$ 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>`
`<string>`
	<> = Placeholder for integer, string etc

`foo, bar`	Metasyntactic variables

File Systems

File Systems

- bwUniCluster, ForHLR I / ForHLR II, bwForCluster, …
  - Too many file systems to list them all, please see documentation!

- Common „rules of thumb“
  - There is a small, global, permanent $HOME directory

    → Usually backed up

  - There are one or more large, global temporary $WORK directories

    → Old files might be deleted automatically

    → Usually NOT backed up

  - There might be local, non-global, temporary storage on the computing nodes

    → Usually named $TMP, $SCRATCH or something similar

    → Will be wiped when the job ends

$HOME = Home directory

- $HOME:

  @ bwUniCluster/ForHLR:

  - Current quota:
    ```
    $ lfs quota -u $(whoami) $HOME
    ```
  - Diskusage:
    ```
    $ grep -E "$(whoami)|Account" ~/../../diskusage
    ```

  @ KIT: $HOME directories of bwUniCluster, ForHLR I / II are the same

  - But: different hardware, libraries, queueing etc.

    → generalise your scripts to work on all systems using $CLUSTER

```
if [ ${CLUSTER} == “uc1“ ]; then
  <operations>
fi
```

  @bwForCluster:

  - Please check the documentation!

$PROJECT = Project directory of ForHLR I/II

- ONLY ForHLR I/II:
  - All features of $HOME
  - Access granted based on approved projects
    → assigned „name/acronym“
    → $PROJECT_GROUP
  - Access project home directory:
    ```
    $ cd $PROJECT
    ```
  - Do not use: $HOME → it has very low quota for the project group!
  - Quota of Project:
    ```
    $ lfs quota -g ${PROJECT_GROUP} ${PROJECT}
    ```

Workspaces = Working directory

- **Workspaces**: lifetime on allocated folder
  - HowTo:

    → http://www.bwhpc-c5.de/wiki/index.php/BwUniCluster_File_System#Workspaces

`$ ws_allocate foo 10`	Allocate a workspace named *foo* for 10 days
`$ ws_list -a`	List all your workspaces
`$ ws_find foo`	Get absolute path of workspace *foo*
`$ ws_extend foo 5`	Extend lifetime of your workspace *foo* by 5 days from now. You can extend 3 times → max. lifetime of *foo* = 240 days (U+F) 90 days (J)
`$ ws_release foo`	Manually erase your workspace foo

Example:

```
$ ws_allocate scratch
$ SDIR=$(ws_find scratch)
$ echo $SDIR
/work/workspace/scratch/ab1234-scratch-0
```

Software System

Environment modules

- Default → manual setup of
  - compilers, libraries and software packages etc.
    → complicated if multiple versions of same software installed
- Solution:
  - dynamic modification of the session environment by
    → instruction sets stored in *modulefiles*
- HowTo?
  - *load* and *unload* instruction sets (= modulefiles)
- How to use modulefiles in general?

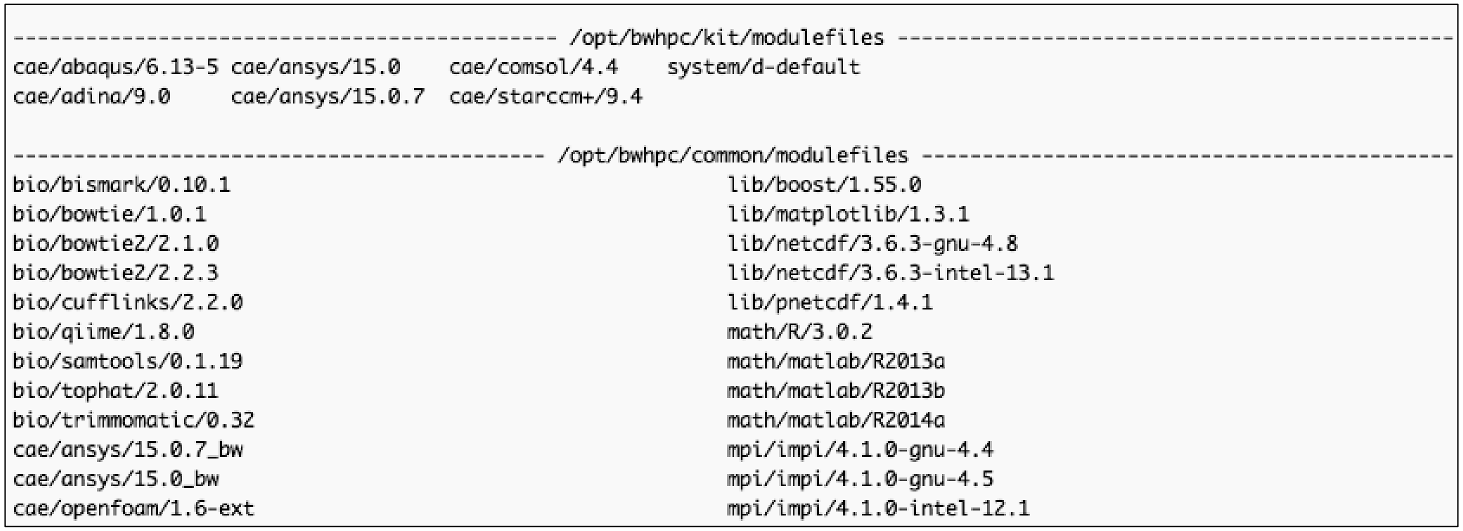
```
$ module help
```

- More information:
  - http://www.bwhpc-c5.de/wiki/index.php/Environment_Modules

modulefiles: available / search

- Display all modulefiles

```
$ module avail
```

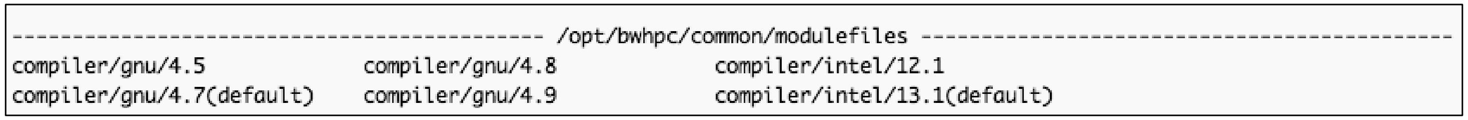


```
--------------------------------------------- /opt/bwhpc/kit/modulefiles ---------------------------------------------
cae/abaqus/6.13-5 cae/ansys/15.0    cae/comsol/4.4    system/d-default
cae/adina/9.0     cae/ansys/15.0.7  cae/starccm+/9.4

-------------------------------------------- /opt/bwhpc/common/modulefiles --------------------------------------------
bio/bismark/0.10.1                                          lib/boost/1.55.0
bio/bowtie/1.0.1                                            lib/matplotlib/1.3.1
bio/bowtie2/2.1.0                                           lib/netcdf/3.6.3-gnu-4.8
bio/bowtie2/2.2.3                                           lib/netcdf/3.6.3-intel-13.1
bio/cufflinks/2.2.0                                         lib/pnetcdf/1.4.1
bio/qiime/1.8.0                                             math/R/3.0.2
bio/samtools/0.1.19                                         math/matlab/R2013a
bio/tophat/2.0.11                                           math/matlab/R2013b
bio/trimmomatic/0.32                                        math/matlab/R2014a
cae/ansys/15.0.7_bw                                         mpi/impi/4.1.0-gnu-4.4
cae/ansys/15.0_bw                                           mpi/impi/4.1.0-gnu-4.5
cae/openfoam/1.6-ext                                        mpi/impi/4.1.0-intel-12.1
```

- Display all modulefiles with prefix „compiler“

```
$ module avail compiler
```



```
-------------------------------------------- /opt/bwhpc/common/modulefiles --------------------------------------------
compiler/gnu/4.5             compiler/gnu/4.8             compiler/intel/12.1
compiler/gnu/4.7(default)    compiler/gnu/4.9             compiler/intel/13.1(default)
```

modulefiles: help / whatis

- Show help of modulefiles, e.g. `$ module help compiler/intel`

```
----------- Module Specific Help for 'compiler/intel/13.1' --------


This module provides the Intel(R) compiler suite version 13.1.3 via
commands 'icc', 'icpc' and 'ifort', the debugger 'idb' as well as the Intel(R)
Threading Building Blocks TBB and the Integrated Performance Primitives IPP
libraries (for details see also 'http://software.intel.com/en-us/intel-compilers/').

The related Math Kernel Library MKL module is 'numlib/mkl/11.0.5'.
The related Intel MPI module is 'mpi/impi/4.1.1-intel-13.1'.
The Intel icpc should work well with GNU compiler 4.7.

Commands:
  icc           # Intel(R) C compiler
  icpc          # Intel(R) C++ compiler
  ifort         # Intel(R) Fortran compiler
  idb           # Intel(R) debugger in GUI mode
  idbc          # Intel(R) debugger in console mode

Local documentation:
  Man pages: man icc; man icpc; man ifort
  firefox $INTEL_DOC_DIR/documentation_c.htm
  firefox $INTEL_DOC_DIR/documentation_f.htm
```

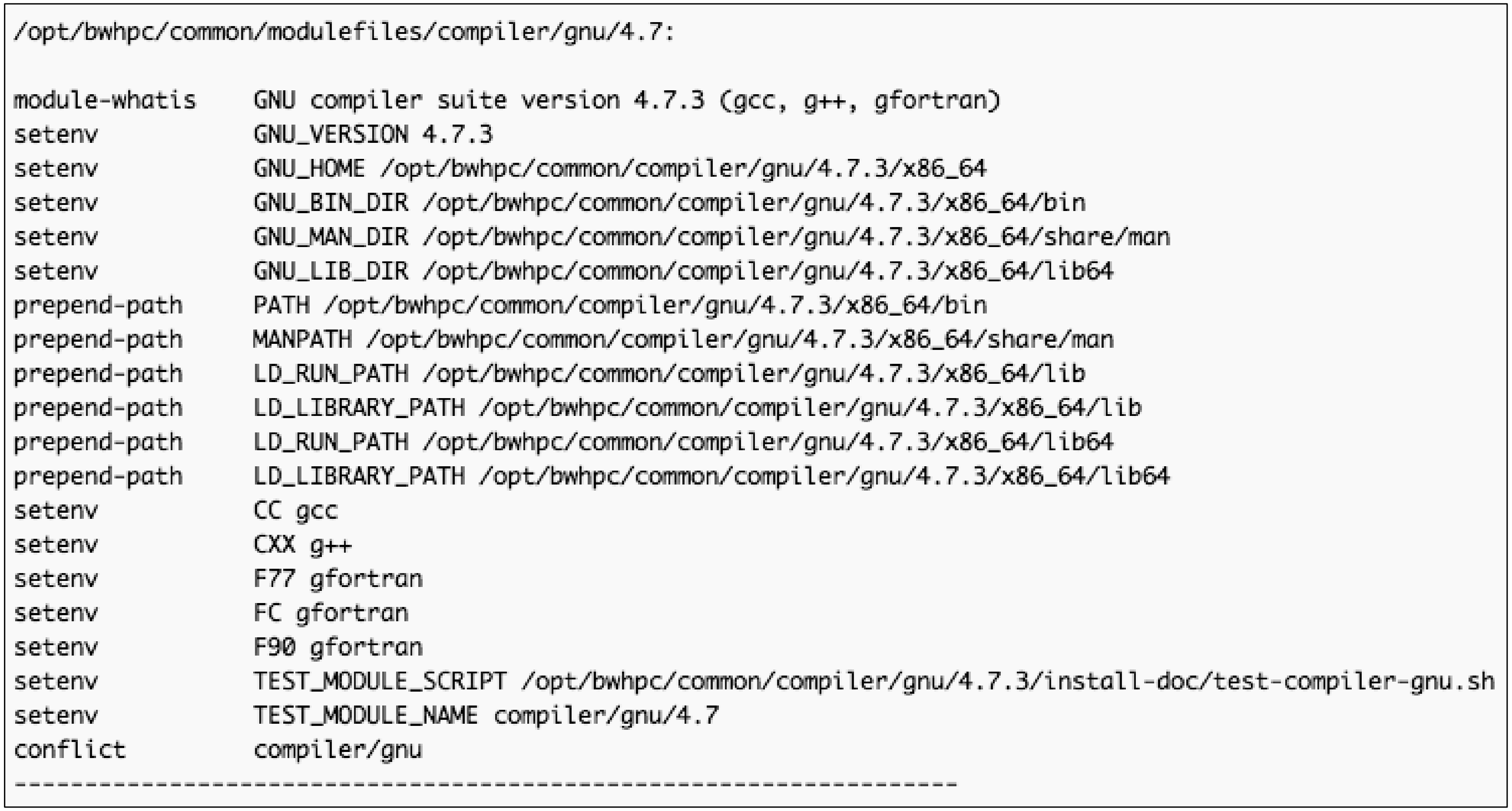
- Show short info modulefile

```
$ module whatis compiler/intel
```

```
compiler/intel    : Intel(R) compiler suite (icc, icpc, ifort), debugger (gdb-ia), IPP and TBB ver 16.0.4
```

modulefiles: show

- Show all instructions of modulefile `$ module show compiler/gnu/4.7`



```
/opt/bwhpc/common/modulefiles/compiler/gnu/4.7:

module-whatis    GNU compiler suite version 4.7.3 (gcc, g++, gfortran)
setenv           GNU_VERSION 4.7.3
setenv           GNU_HOME /opt/bwhpc/common/compiler/gnu/4.7.3/x86_64
setenv           GNU_BIN_DIR /opt/bwhpc/common/compiler/gnu/4.7.3/x86_64/bin
setenv           GNU_MAN_DIR /opt/bwhpc/common/compiler/gnu/4.7.3/x86_64/share/man
setenv           GNU_LIB_DIR /opt/bwhpc/common/compiler/gnu/4.7.3/x86_64/lib64
prepend-path     PATH /opt/bwhpc/common/compiler/gnu/4.7.3/x86_64/bin
prepend-path     MANPATH /opt/bwhpc/common/compiler/gnu/4.7.3/x86_64/share/man
prepend-path     LD_RUN_PATH /opt/bwhpc/common/compiler/gnu/4.7.3/x86_64/lib
prepend-path     LD_LIBRARY_PATH /opt/bwhpc/common/compiler/gnu/4.7.3/x86_64/lib
prepend-path     LD_RUN_PATH /opt/bwhpc/common/compiler/gnu/4.7.3/x86_64/lib64
prepend-path     LD_LIBRARY_PATH /opt/bwhpc/common/compiler/gnu/4.7.3/x86_64/lib64
setenv           CC gcc
setenv           CXX g++
setenv           F77 gfortran
setenv           FC gfortran
setenv           F90 gfortran
setenv           TEST_MODULE_SCRIPT /opt/bwhpc/common/compiler/gnu/4.7.3/install-doc/test-compiler-gnu.sh
setenv           TEST_MODULE_NAME compiler/gnu/4.7
conflict         compiler/gnu
-------------------------------------------------------------------
```

Load modulefiles (3)

- Modulefiles are sorted in categories, software name and versions:

```
$ module load <category>/<software_name>/<version>
```

-

- Load a default software:

```
$ module load <category>/<software_name>
```

  - e.g. Intel compiler

```
$ module load compiler/intel mpi/impi
```

→ loads currently Intel compiler suite 16

→ loads currently Intel-MPI 5.1.3 for Intel compiler 16.0

```
$ module list
```

- Display all loaded modules

```
Currently Loaded Modulefiles:
  1) compiler/intel/16.0(default)    2) mpi/impi/5.1.3-intel-16.0(default)
```

modulefiles: categories & dependencies

- Module names already implicate dependencies:

  → **Category/softwarename/version_attributes-dependencies**

  e.g **numlib/fftw/3.3.5-impi-5.1.3-intel-16.0**

  → fftw package version 3.3.5, compiled with Intel 16.0 and Intel-MPI 5.1.3

- Categories:

`compiler/`	for compiler, e.g. intel, gnu, pgi, open64
`devel/`	for debugger, e.g. ddt, and development tools, e.g. cmake, itrac
`mpi/`	for MPI libraries, e.g. impi, openmpi, mvapich(2)
`numlib/`	for numerical libraries, e.g. Intel MKL, ACML, nag, gsl, fftw
`lib/`	for other libraries, e.g. netcdf, global array
`bio/`	for biology software, e.g. bowtie, abyss, mrbayes
`cae/`	for CAE software, e.g. ansys, abaqus, fluent
`chem/`	for chemistry software, e.g. gromacs, dacapo, turbomole
`math/`	for mathematics software, e.g. matlab, R
`phys/`	for physics software, e.g. geant4
`vis/`	for visualisation software, e.g. vmd, tigervnc

modulefiles: conflicts

- Conflicts:
  - a) load different software version in the same session, e.g. Intel:

```
$ module load compiler/intel/14.0
$ module load compiler/intel/15.0
```

```
compiler/intel/13.1(394):ERROR:150: Module 'compiler/intel/15.0' conflicts
with the currently loaded module(s) 'compiler/intel/14.0'
```

  - b) load module with dependencies on other modules

```
$ module load mpi/openmpi/1.10-intel-16
```

```
Loading module dependency 'compiler/intel/16.0'.
compiler/intel/16.0(394):ERROR:150: Module 'compiler/intel/16.0' conflicts
with the currently loaded module(s) 'compiler/intel/15.0'
```

→ **NOT** an issue if the cluster uses **Lmod** (ForHLR I/II)

modulefiles: unload/swap

- To remove module *foo*:

```
$ module unload foo
```

```
$ module remove foo
```

  be aware that you might create inconsistencies,

  e.g. you can remove

  compiler/intel/16.0 while *mpi/openmpi/1.10-intel-16.0* is still loaded

- Swap = remove + load

  e.g.:

```
$ module swap compiler/intel/15.0 compiler/intel/16.0
```

Private modulefiles

- Each user can create own modulefiles:

  e.g. modulefiles that adds path of own programs, $HOME/special, to $PATH

  → content of this modulefile „*mybin*“

```
#%Module1.0

Append-path   PATH  “$env(HOME)/special”
```

  → place „*mybin*“ under $HOME/privatemodules

  → to make all own modules visible to “module avail” command, enter:

```
$ module load use.own
```

or

```
$ module use $HOME/privatemodules
```

  → former: own modules have lower priority than system ones if equally named

  → latter: own module have higher priority

- Remove own modules:

```
$ module unload use.own
```

or

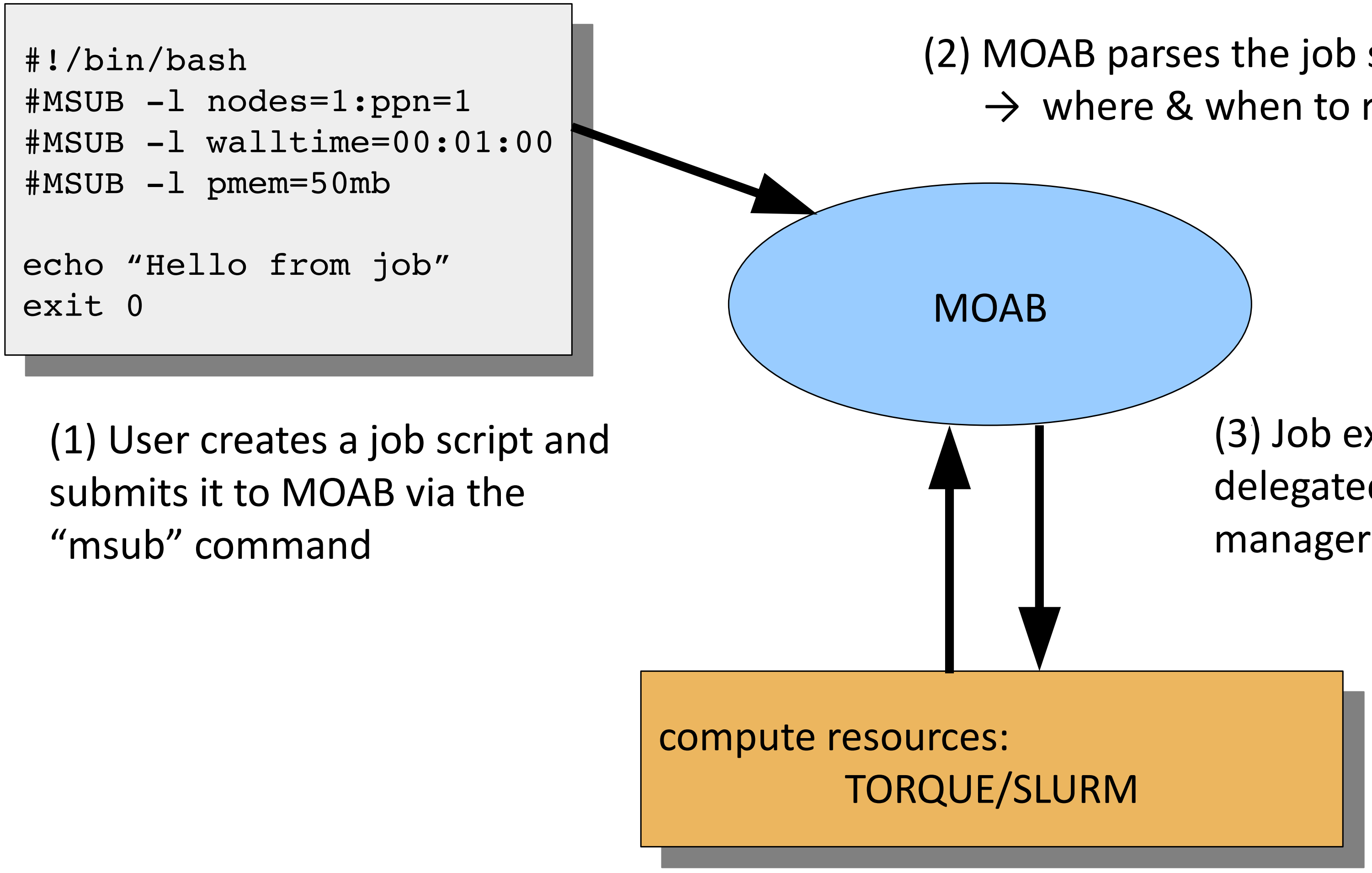
```
$ module unuse $HOME/privatemodules
```

Batch System

Resource management

- Components of management system (Batch System)
  - **resource manager**
    - control over jobs and distributed compute nodes
    - SLURM (bwUniCluster, ForHLR I)
    - TORQUE (bwForCluster, ForHLR II)
  - **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/ForHLR:   job_<uc1,fh1>_<job_ID>.out

JUSTUS:                STDIN.o<job_ID> or STDIN.e<job_ID>
```

msub options

- 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
-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

→ for workshop: -l advres=workshop.8

→ 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, *pmem*=4000mb	nodes=1	*walltime*=00:30:00, *nodes*=1:*ppn*=16
singlenode	*procs*=1, *pmem*=4000*mb*	walltime=00:30:01, nodes=1	*walltime*=3:00:00:00, *nodes*=1:*ppn*=16
multinode	***procs*=1, pmem=4000mb**	***nodes*=2**	***walltime*=2:00:00:00, *nodes*=16:*ppn*=16**
explicit queue definition			
verylong	*procs*=1, *pmem*=4000mb	*walltime*=3:00:00:01, *nodes*=1	*walltime*=6:00:00:00, *nodes*=*1*:ppn=*16*
fat (fat nodes)	*procs*=1, *pmem*=32000mb	*nodes*=1	*walltime*=3:00:00:00, *nodes*=1:*ppn*=32

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

msub -q *queues* (ForHLR)

- http://www.bwhpc-c5.de/wiki/index.php/Batch_Jobs_-_ForHLR_Phase_I_Features

queue	*default resources*	*MIN resources*	*MAX resources*
	explicit queue choosing		
develop	*procs*=1, *mem*=3200mb, *walltime=00:10:10*	nodes=1	*walltime*=00:30:00, *nodes*=1:*ppn*=20
singlenode	*procs*=1, mem=*3200mb, walltime=00:10:10*	nodes=1	*walltime*=3:00:00:00, *nodes*=1:*ppn*=20
multinode	***procs*=1, *mem*=3200mb, *walltime=00:10:10***	***nodes*=2**	***walltime*=3:00:00:00, *nodes*=128:*ppn*=20**
fat (fat nodes)	*procs*=1, *mem*=160000mb, *walltime=00:10:10*	*nodes*=1	*walltime*=3:00:00:00, *nodes*=1:*ppn*=32

Environment variables

- www.bwhpc-c5.de/wiki/index.php/Batch_Jobs#Environment_Variables_for_Batch_Jobs
- **bwUniCluster + ForHLR + bwForClusters**

```
$ printenv|grep MOAB
```

queue =

Environment variables	Description
MOAB_CLASS	Class name
MOAB_GROUP	Group name
MOAB_JOBID	Job ID
MOAB_JOBNAME	Job name
MOAB_NODECOUNT	Number of nodes allocated to job
MOAB_PARTITION	Partition name the job is running in
MOAB_PROCCOUNT	Number of processors allocated to job
MOAB_SUBMITDIR	Directory of job submission
MOAB_USER	User name

- Using in scripts:

```
## add suffix to job output file
./program > $program_${MOAB_JOBID}.log
```

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 via "`msub -I -V`"
  - Restrictions may apply (shared nodes, single node etc.)

```
$ 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
- **bwForClusters**
  - **see Wiki**

Check/change status of your jobs (1)

- after submission → msub returns <job-ID>

```
$ msub job.sh

659562
```

- **commands:**

$ 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          ///          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 (3)

- **STATE:**

Running	OK, job is running
Idle	Job is waiting for free resources
Deferred	Buffer-state. Job can not run (no free resources or wrong resources)
BatchHold	Job is blocked by scheduler. End-state. Reasons: no resources,limits,failure

```
Idle → Running → Canceling == OK


Idle → Deferred → Idle → Deferred → … → BatchHold → Canceling
```

Check status of your jobs (4)

- 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 (5)
example: MAXNODE limit

- Submitted job (bwUniCluster)

```
$ msub -l nodes=1:ppn=8 -q fat <jobscript>

12345
```

showq:

```
blocked jobs-----------------------
JOBID              USERNAME       STATE PROCS     WCLIMIT            QUEUETIME
12345              ///             Idle     5    00:05:00  Fri Jan 23 15:31:05
```

checkjob 12345:

```
State: Idle
class:fat
...
NodeCount:  1
...

BLOCK MSG: job 12345 violates active
           HARD MAXNODE limit of 2 for class fat user partition ALL
           (Req: 8  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:

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

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) ...
```

* limits for **university_X**
* TODO: only wait!

Change status of your jobs

- **Change commands**

`canceljob <job_ID>`	cancel the job with <job_ID>
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
mjobctl -c -w user=$USER	**cancel ALL your jobs!**

```
$ showq

active jobs------------------------
JOBID        USERNAME     STATE PROCS    REMAINING            STARTTIME
31172        ///          Running   1      00:04:58  Thu Jan 22 19:21:56
…

blocked jobs-----------------------
JOBID        USERNAME     STATE PROCS    WCLIMIT              QUEUETIME
31173        ///             Idle     1    00:05:00   Thu Jan 22 19:21:47
31174        ///          BatchHold   1    00:05:00   Thu Jan 22 19:21:48
```

Example

```
#!/bin/bash
#MSUB -l nodes=2:ppn=16
#MSUB -l walltime=01:00:00
#MSUB -l pmem=2gb
#MSUB -N serial-test

mpirun ./hello
```

→ Is equal to:

```
$ msub -l nodes=2:ppn=16,walltime=01:00:00,pmem=2gb -N serial-test
  <job_script>
```

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

- `# MSUB` instead of `#MSUB` (note the space...)