bw | HPC – C5

bwUniCluster Tutorial

Access, Data Transfer, Compiling, Modulefiles, Batch Job Scripting

Mehmet Soysal

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

www.bwhpc-c5.de

Login

- Username `<username>`
  - Same username as your user account at university.
  - Users from other universities than KIT have to prefix their username by the organization's token, e.g. ho_anfuchs, xy1234
- Host `<host>`
  - bwUniCluster: `uc1.scc.kit.edu`

- Linux / OS X
  - open terminal:
    ```
    > ssh <username>@<host>
    ```

- Windows
  - use SSH-Client, e.g. PuTTY
  - connect to <host>:
    ```
    > Login as:  <username>
    ```

Basic commands

`$ pwd`	show path of working directory
`$ mkdir <dirname>`	make directory
`$ cp <sourcefile> <targetfile>`	copy file
`$ mv <sourcefile> <targetfile>`	move file
`$ rm <filename>`	remove file
`$ man <command>`	show command's manual

Data Transfer

- From localhost to cluster:
  - use `scp` (secure copy) or `sftp` (secure file transfer program)
  - Read manual for options/syntax questions (`man scp`, `man sftp`)
  - From Linux to Linux Systems you can also use rsync
- Linux / OS X
  - Open terminal at your computer:

```
$ scp <sourcefile> <username>@<host>:<targetfile>
```

or

```
$ sftp <username>@<host>:<targetdir>
$ put <sourcefile>
```

- Windows
  - use SCP/SFTP-Client, e.g. WinSCP
  - connect to `<username>@<host>`
  - copy data by drag&drop mechanism

Module Environment

- Users require different software in different versions.
- Software is installed and can be used by loading corresponding modules.

`> module avail`	show all installed software packages
`> module avail compiler`	show all available compilers
`> module load <modulepath>`	load a module in list
`> module unload <modulepath>`	remove a module from list
`> module list`	show all loaded modules
`> module show <modulepath>`	show environment variables of module
`> module help <modulepath>`	show usage information of module

From $HOME to $WORK

- Compute nodes read&write in $WORK very much faster than in $HOME directory.
- **DO NOT COMPUTE IN $HOME !!**
- $HOME:
  - Source code
- $WORK:
  - Program input (e.g. initial and boundary conditions)
  - Program output

If lifetime of $WORK is too short, create a workspace.
But **never** compute in $HOME!

Exercise

- Download source code from indigo

  https://indico.scc.kit.edu/indico/event/132/material/0/
  /opt/bwhpc/kit/workshop/

- Copy source code to bwUniCluster

- Log on bwUniCluster

- Load module file corresponding to the compiler of choice

- Compile the source code, e.g. sequential version with Intel-Compiler:

```
$ module load compiler/intel
$ icc -o hello hello.c
$ ./hello
```

- Move your binary in `$WORK`

Source code is written in C and Fortran90 and provided in a sequential version or with OpenMP, MPI or hybrid parallelization.

Submitting jobs via script

- Example: requesting one CPU and 3000 MB of main memory for 5 hours to run the sequential program `hello`

```
#!/bin/bash
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=5:00:00
#MSUB -l mem=3000mb
#MSUB -q singlenode
#MSUB -N serial-test
#MSUB -m abe

./hello
#sleep 60
```

Interpreter

Header with `msub` options
- resource requirements
- queue definition
- notification options,...

Execution part

- Submitting the script **`jobuc.sh`** with MOAB:

```
> msub jobuc.sh
```

Environment variables in job scripts

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

- MOAB variables and own environment variables

	Using MOAB Variables	Defining own variables
Header	`#MSUB -o $(JOBNAME).o$(JOBID)`	`#MSUB -v EXEC=./hello`
Execution Part	`echo „Job ${MOAB_JOBNAME} is running (ID=${MOAB_JOBID})“`	`export EXEC=./hello`

Parallel Jobs (MPI)

- `mpicc -o hello_mpi hello_mpi.c`

```
#!/bin/bash
#MSUB -l nodes=2:ppn=4
#MSUB -l walltime=05:00
#MSUB -l pmem=1000mb
#MSUB -q multinode
#MSUB -N hello_mpi

module load mpi/impi



mpirun hello_mpi
#sleep 60
```

- For computations on more than 1 node use queue `multinode`.
- The corresponding MPI module has to be loaded on the compute nodes.
- Use `mpirun` to execute the binary.

Parallel Jobs (OpenMP)

icc -openmp hello_omp.c -o hello_omp

```
#!/bin/bash
#MSUB -l nodes=1:ppn=8
#MSUB -l walltime=05:00:00
#MSUB -l pmem=1000mb
#MSUB -q singlenode
#MSUB -N hello_omp

EXECUTABLE=./hello_omp
export OMP_NUM_THREADS=${MOAB_PROCCOUNT}

echo "Executable ${EXECUTABLE} running
with ${OMP_NUM_THREADS} threads"

./hello_omp
```

- Shared memory restricts to 1 node.
- Do not define number of threads explicitely. Use MOAB variables.

Keep track of a job

- Submit job script

```
$ msub <jobscript>
```

  - If a job (script) is accepted the <jobid> appears at screen.

`$ checkjob <jobid>`	show job details
`$ showq`	list all my running, idling and blocked jobs by <jobid>
`$ showq -n`	list all my running, idling and blocked jobs by <jobname>
`$ showq -c`	list my completed jobs
`$ canceljob <jobid>`	cancel job