U N I V E R S I T Y O F B E R G E N

Scientific Computing, IT department

Introduction to HPC

Part 2

Lóránd Szentannai

Saerda Halifu

Agenda

- UNIX command line interface (CLI)
- Modules
- Compilers
- Batch system

HPC Docs

http://docs.hpc.uib.no/

UNIX command line interface (CLI)

UNIX CLI

- Shell
  - Popular commands
  - Variables
  - Environment
  - Conditionals
  - Loop
  - Redirections and pipe
  - Control operators
- File system
  - Structure
  - Security concept
  - Ownerships and permissions
- Processes

UNIX CLI - Shell

- Text interface to OS
  - Interprets and executes your commands
  - Different shells (i.e. bash, csh, ksh, etc.)
- Bourne-Again shell - bash - most powerful
  - Interactive
  - Scriptable
  - Case-sensitive
  - Name expansion
  - Wildcards
  - Typing shortcuts (TAB, UP, DOWN, Ctrl+R, Esc+.)

```
prompt command param1 param2 …
```

UNIX CLI - Shell

Popular commands

- man, apropos
- pwd, cd, mkdir
- ls, cp, mv, rm
- scp, rsync
- cat, less, grep, tail, head
- awk, sed
- alias
- screen

UNIX CLI - Shell

Variables

- Referenced with $ sign
- Special variables
  - $0 - name of the script
  - $1, $2, …, $9, ${10} - positional arguments
  - $* - all arguments
  - $# - number of arguments
- User variables ($myvar)
- Environment variables ($USER, $HOME, $PATH, etc.)

UNIX CLI - Shell

Environment

- Initialized from:
  - system-wide configuration files
  - user configuration files
- Environment variables (PATH, LD_LIBRARY_PATH, etc.)
- Commands:
  - env
  - export
  - unset

UNIX CLI - Shell

Conditionals

- if

```
if/then/fi
if/then/elif/fi
```

- case

```
case expression in
    case1)
        commands
        ;;
    …
    caseN)
        commands
        ;;
esac
```

UNIX CLI - Shell

Loop

- for - allows repeated execution of the command(s)

```
for variable in list
do
   commands
done
```

- while - executes the command(s) while the expression is true

```
while expression
do
   commands
done
```

- until - similar to while, but code is executed until expression is false

UNIX CLI - Shell

Redirections and pipe

- stdin - standard input
- stdout - standard output
- stderr - standard error

- > - redirect stdout (to a file)
- >> - redirect stdout to a file, w/o overwriting it (append)
- < - stdin (from file)
- 2> - redirect stderr
- 2>&1 - redirect both stdout and stderr
- | - pipe commands

UNIX CLI - Shell

Control operators

- ; - separate sequence of commands
- & - execute command in the background
- && - AND
- || - OR

UNIX CLI - File system

Generic

- /bin - Linux commands
- /etc - Configuration files
- /home - Home directories
- /lib - Libraries
- /sbin - Sysadmin commands
- /tmp - Temporary files (very small)
- /usr - User programs
- /var - Log files, spool files
- /proc, /dev, /sys, /opt, /mnt, etc…

Clusters

Hexagon: hexagon.hpc.uib.no

Fimm: fimm.hpc.uib.no

UNIX CLI - File system

Hexagon specific

- /home
  - enforced quota (10GB/user)
  - backup available
- /work (250TB)
  - no backup
  - automatic cleanup of old files
- /work-common (320TB)
  - no backup
  - semi-permanent storage

UNIX CLI - File system

Fimm specific

- /fimm/home
  - enforced quota
  - backup available
- /fimm/work
  - no backup
- /work-common (320TB)
  - no backup
  - semi-permanent storage

/fimm/home + /fimm/work total quota is 40GB/user

UNIX CLI - File system

Security structure

- each file, directory or device has three permission groups
- first permission group describes owners' permissions
- second one describes (UNIX) groups' permissions
- third group describes permissions related to any other user (not owner and not member of UNIX group)
- set-GID allows a new created file to inherit same group as directory has
- sticky-bit prevents any other user then owner or root from deleting a file within that directory

UNIX CLI - File system

Ownerships and permissions

```
-rwxr-xr-x 1 root root 106120 Apr 16  2015 /bin/ls
```

- change ownership
  - chown USERNAME:GROUP FILE
  - chown USERNAME:GROUP -R DIR
  - chgrp GROUP FILE
  - chgrp GROUP -R DIRECTORY

```
$ chown root:wheel /bin/ls
-rwxr-xr-x 1 root wheel 106120 Apr 16  2015 /bin/ls
```

UNIX CLI - File system

Ownerships and permissions

```
-rwxr-xr-x 1 root root 106120 Apr 16  2015 /bin/ls
```

- change permissions
  - chmod PERM_GROUP+/-PERMISSIONS FILE
  - chmod PERM_GROUP+/-PERMISSIONS -R DIR

```
# chmod g+w /bin/ls
-rwxrwxr-x 1 root root 106120 Apr 16  2015 /bin/ls
```

UNIX CLI - File system

Ownerships and permissions

- permissions have symbolical and octal number representation
  - read “r” or “4”
  - write “w” or “2”
  - execute “x” or “1”

```
# chmod 775 /bin/ls
-rwxrwxr-x 1 root root 106120 Apr 16  2015 /bin/ls
```

UNIX CLI - Processes

- parent/child processes
- priorities
- id

- Commands
  - ps
  - top
  - kill

UNIX CLI - Processes

ps

- ps -fau USER
- ps -ef

i.e.

```
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 Jun16 ?        00:02:43 init [3]
root           2     0  0 Jun16 ?        00:01:43 [kthreadd]
…
root       27227  9009  0 14:59 ?        00:00:00 sshd: lorand [priv]
lorand     27229 27227  0 14:59 ?        00:00:00 sshd: lorand@pts/28
lorand     27230 27229  0 14:59 pts/28   00:00:00 -bash
root       27531     2  0 Jun17 ?        00:12:55 [ldlm_bl_59]
lorand     27677 27230  0 15:01 pts/28   00:00:00 ps -ef
```

- pstree

UNIX CLI - Processes

top

i.e.

```
top - 15:12:14 up 119 days, 20:31, 25 users,  load average: 0.27, 0.37, 0.36
Tasks: 515 total,   1 running, 514 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.1%us,  1.3%sy,  0.2%ni, 98.1%id,  0.1%wa,  0.0%hi,  0.2%si,  0.0%st
Mem:     16146M total,    14909M used,     1237M free,        0M buffers
Swap:        0M total,        0M used,        0M free,    10412M cached

  PID USER      PR  NI  VIRT  RES  SHR S   %CPU %MEM    TIME+  COMMAND
 8466 trondk    25   5 29272 2016 1468 S      5  0.0  41:03.49 wget
28111 trondk    39  19 39840 8000 1064 S      3  0.0 321:42.40 wget
26481 pwe081    25   5 51060 3664 1476 S      1  0.0   2:33.98 aprun
 7627 root       0 -20     0    0    0 S      0  0.0   3317:12 kgnilnd_sd_02
```

UNIX CLI - Processes

kill

- kill PID
  - can be caught and interpreted or ignored by the process
  - allows process to perform clean-up
- kill -9 PID
  - cannot be caught or ignored, thus no clean-up is performed

Modules software

Modules software

Overview

- provides dynamic modification of the user's environment
- each module contains information for adjusting shell for the particular application
- especially helpful when multiple versions of same software must be present on the system
- offers great help for quickly setting up the programming environment

Modules software

Useful commands

- **module help** - short help
- **module avail** - list available modules
- **module list** - list loaded modules
- **module add|load** - load modulefile
- **module rm|unload** - unload modulefile
- **module switch|swap** - switch between two modulefiles
- **module show** - detailed information about modulefile

Modules software

module avail

```
$ module av cray-netcdf

---------------------------------- /opt/cray/modulefiles ----------------------------------
cray-netcdf/4.2.1.1                        cray-netcdf-hdf5parallel/4.2.1.1
cray-netcdf/4.3.0                          cray-netcdf-hdf5parallel/4.3.0
cray-netcdf/4.3.1                          cray-netcdf-hdf5parallel/4.3.1
cray-netcdf/4.3.2                          cray-netcdf-hdf5parallel/4.3.2
cray-netcdf/4.3.3.1(default)               cray-netcdf-hdf5parallel/4.3.3.1(default)
```

module swap

```
$ module swap pgi pgi/14.9.0
```


Modules software

module list

```
$ module li
Currently Loaded Modulefiles:
  1) mam/8.1.1.2-1438126669_9ceadf49(default)
  2) python/2.7.9-dso(default)
  3) rbh/2.5.4-l25(default)
  4) modules/3.2.10.3
  5) nodestat/2.2-1.0502.53712.3.109.gem
  6) sdb/1.0-1.0502.55976.5.27.gem
  7) job/1.5.5-0.1_2.0502.54585.3.66.gem
  8) alps/5.2.1-2.0502.9041.11.6.gem
  9) lustre-cray_gem_s/2.5_3.0.101_…
 10) udreg/2.3.2-1.0502.9275.1.25.gem
 11) ugni/5.0-1.0502.9685.4.24.gem
 12) gni-headers/3.0-1.0502.9684.5.2.gem
 13) dmapp/7.0.1-1.0502.9501.5.211.gem
 14) xpmem/0.1-2.0502.55507.3.2.gem
 15) hss-llm/7.2.0
 16) Base-opts/1.0.2-1.0502.53325.1.2.gem
 17) craype-network-gemini
 18) craype/2.4.2
 19) cce/8.4.0
 20) totalview-support/1.2.0.6
 21) totalview/8.15.7
 22) cray-libsci/13.2.0
 23) pmi/5.0.9-1.0000.10911…
 24) rca/1.0.0-2.0502.53711.3…
 25) atp/1.8.3
 26) PrgEnv-cray/5.2.40
 27) xtpe-interlagos
 28) cray-mpich/7.2.5
 29) moab/8.1.1.2-201508…
 30) torque/5.1.1.2
 31) hpnssh/6.6p1
 32) notur/0.1
 33) WRF/3.6.1
```

Modules software

module show

```
$ module show gcc
-------------------------------------------------------------------
/opt/modulefiles/gcc/5.1.0:

conflict   gcc
prepend-path         PATH /opt/gcc/5.1.0/bin
prepend-path         MANPATH /opt/gcc/5.1.0/snos/share/man
prepend-path         INFOPATH /opt/gcc/5.1.0/snos/share/info
prepend-path         LD_LIBRARY_PATH /opt/gcc/5.1.0/snos/lib64
setenv               GCC_PATH /opt/gcc/5.1.0
setenv               GCC_VERSION 5.1.0
setenv               GNU_VERSION 5.1.0
-------------------------------------------------------------------
```


Compilers

Compilers

Programming Environments

- programming environments are loaded using modules
- named as PrgEnv-COMPILER

- COMPILER:
  - cray (only on Hexagon)
  - gnu
  - pgi
  - intel

Compilers

Programming Environments

- check loaded compiler
  - module list
  - echo $PE_ENV
- switch between compiler versions
  - module swap COMPILER COMPILER/required_version

- please pay attention to dependencies between libraries and compiler
- try to use default versions (usually latest) whenever possible

Questions?

Batch system

Batch system

- ensures fair use of the clusters
- manages the queuing, scheduling, starting and stopping of jobs

Topics

- Machine architecture
- Job management

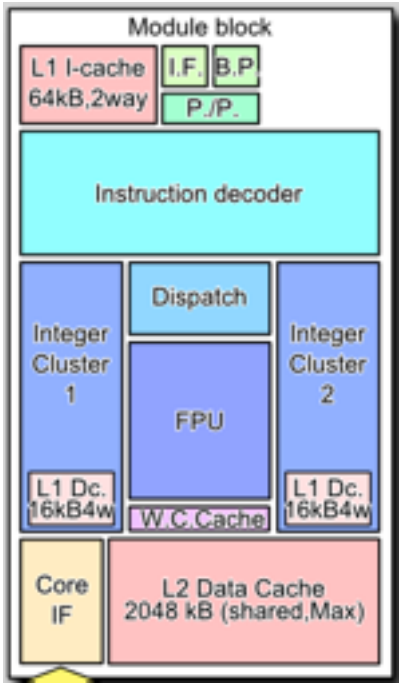
Resources

- Hexagon (Torque & Moab)
- Fimm (Slurm)

Batch system - Hexagon

Architecture - Node configuration

- 696 nodes
- 2 x 16 core AMD Opteron 6276
- 32 cores per node
- 32GB RAM per node
- 1GB RAM/core
- IPU/core
- shared FPU/2 cores



Batch system - Hexagon

Generic information

- Torque - resource manager
  - low level functionality: start, monitor, cancel jobs
- Moab - scheduler
  - optimally use resources
  - policy enforcement
- MAM - allocation manager
  - usage tracking
  - accounting charges
  - allocation enforcements

Batch system - Hexagon

Generic information

Queues

Name	*Description*	*Limitations*
batch	general routing queue	job limitations
normal	queue for normal jobs	job limitations
small	small jobs, higher priority	512 CPUs, 1h walltime
debug	debugging jobs, higher priority	64 CPUs, 20min walltime

- automatic job placement based on job parameters

Job submission

- Batch

```
$ qsub my_batch_script.pbs
```

- Interactive

```
$ qsub -I -l walltime=00:10:00,mppwidth=32 -A my_accnt
```

Batch system - Hexagon

Limitations

Job limitations

- if not specified default values are:
  - 1 CPU with 1000mb of memory
  - 60 minutes

Maximum amount of resources per user

- 4096 CPU cores
- 4-8 running (active) jobs (depending on load)
- 2 idle jobs

Batch system - Hexagon

Job submission

- Batch mode
  - batch script is a shell script with #PBS arguments
  - script specifies resources, executables, etc.
  - can be created with any text editor
- Interactive
  - permits interaction with allocated resources
  - useful for debugging purposes
  - account charged for all the time the interactive session is open

Batch system - Hexagon

Job submission

qsub options

- -walltime* - maximum allowed wall-clock time for execution of the job
- -mppwidth* - number of processing elements
- -mppnppn - number of processing elements per node
- -mppdepth - number of OpenMP threads/core
- -mppmem - an upper memory usage limit per process
- -o, -e - stdout, stderr

- man qsub
- man pbs_resources_linux

Batch system - Hexagon

Job submission

aprun

- resources requested in PBS has to match the arguments for aprun
- -B use PBS parameters (mppwidth,mppnppn,mppmem,mppdepth)
- -N processors per node
- -n processing elements
- -d number of threads
- -m memory per element suffix

Batch system - Hexagon

Job submission

qsub examples

- minimal

```
#!/bin/bash
#
#PBS -N “job_name”
#PBS -A “account”
#PBS -l mppwidth=32
#PBS -l walltime=00:20:00
cd /work/$USER/my_job

aprun -B ./my_executable
```

- save stderr and stdout

```
#PBS -e error.log
#PBS -o output.log
```

- send email on abort, begin and end

```
#PBS -m abe
#PBS -M my_email@hpc.uib.no
```

Batch system - Hexagon

Job submission

qsub examples

- redirect output of running application to a file

```
aprun -B ./my_executable 2>&1 out_and_err.log
```

- save stderr and stdout

```
#PBS -e error.log
#PBS -o output.log
```

Batch system - Hexagon

Job management

Useful commands

- qstat - job status

```
$ qstat -f <jobid>      # full status for job, reported by resource manager
$ qstat -Q              # display queue status
```

- qdel - cancel a job
- showq - display job ordering of the scheduler

```
$ showq -r              # display running jobs
$ showq -b -u <username> # display blocked jobs for user
```

- showstart - display estimated start time of the job

```
$ showstart <jobid>
```

- checkjob - display status for job

```
$ checkjob <jobid>       # display job status, reported by scheduler
$ checkjob -v <jobid>    # same as above, be verbose
```

Batch system - Hexagon

CPU quota and hour accounting

- each user has at least one CPU account (project)
- each account has allocated CPU hours (or quota)
- quota is the maximum number of CPU hours that all the users connected to the project can consume
- CPU hour usage is walltime of the user's job multiplied with number of processors used

  NR_OF_NODES * 32 * CPU_TIME (even 1 CPU will be counted as 32 CPUs)

Batch system - Hexagon

CPU quota and hour accounting

Monitoring Tools

- cost
- gstatement
- glsusage

Monitoring Examples

- cost

```
$ cost                  # show used CPU hours for all accounts (current alloc. period)
$ cost -p <accountname> # show used CPU hours for specified account
$ cost -p <accountname> --detail      # same as above, plus usage detailed by user
```

- gstatement

```
$ gstatement -s <date> -e <date> -a <accountname>
```

Batch system - Hexagon

CPU quota and hour accounting

Monitoring Examples

- showq -i

```
eligible jobs----------------------
JOBID                 PRIORITY  XFACTOR  Q  USERNAME    ACCNT  PROCS     WCLIMIT     CLASS      SYSTEMQUEUETIME

1822998*                482305      1.8 UI     aoltu       vd   8192     1:30:00    normal   Sun Nov  8 18:46:38
1822969                 130621      1.0 NF     matsc  nn2834k   3072  6:00:00:00    normal   Sun Nov  8 15:18:39
1822985                  32206      1.0 IM     jonal      imr    768 25:00:00:00    normal   Sun Nov  8 17:15:21
1823036                  31922      1.0 IM     jonal      imr    768 16:16:00:00    normal   Sun Nov  8 19:37:27

4 eligible jobs

Total jobs:  4
```

Questions?

Batch system - Fimm

fimm.hpc.uib.no

- Linux Cluster
- Running CentOS 6.7
- Lustre File system

Batch system - Fimm

Architecture

- 3 login nodes (login1~login3)
- 1 master node
- 97 compute nodes
  (compute-0-[0~21],compute-6-[0~31],
  compute-3-[0~31], compute-7-[0-11])
- 3 Lustre file server 110TB
- 3 10GB switches
- 2 1GB switches

Batch system - Fimm

Architecture

Nr of Nodes	CPU Type	Cores per Node	Memory per Node GB
2	Quad-Core Intel(R) Xeon(R) CPU E5420 @ 2.50GHz	8	32 c3
30	Quad-Core Intel(R) Xeon(R) CPU E5420 @ 2.50GHz	8	16 c3
32	Quad-Core Intel(R) Xeon(R) CPU L5430 @ 2.66GHz	8	16 c6
12	Six-Core AMD Opteron(tm) Processor 2431	12	32 c7
21	Quad-Core Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz	32	128 c0

Batch system - Fimm

Generic information

SLURM - Simple Linux Utility for Resource Management

https://docs.hpc.uib.no
http://slurm.schedmd.com/

Batch system - Fimm

Slurm partitions

- Idle where all user have access BUT……
- t1 only physics (alice) group member have access
- c6 only physics (alice) group member have access
- hpc default partition for all user
- kjem only kjem group member have access
- login not for running jobs.

Batch system - Fimm

Simple sbatch script

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=1000MB
#SBATCH --time=30:00
#SBATCH --output=my.stdout
#SBATCH –mail-user=saerda@uib.no
#SBATCH --mail-type=ALL
#SBATCH –job-name="slurm_test"

hostname
env|grep SLURM
```

```
#!/bin/bash
#
#PBS –N “PBS_test”
#PBS –A “account”
#PBS –l procs=1
#PBS –l mem=500mb
#PBS –l walltime=00:30:00
cd /work/$USER/my_job

./my_executable
```

Batch system - Fimm

Sbatch options

-N, —nodes=<number of nodes >
Request that number of nodes to be allocated to this job.

-n, —ntasks=<number>
Controls the number of tasks to be created for the job

--ntasks-per-node
Controls the maximum number of tasks per allocated node

--mem-per-cpu
specify the the memory need per allocated CPU in MB

Batch system - Fimm

Job management

- submission

```
$ sbatch test_slum.sh
Submitted batch job <jobid>
```

slurm starts your job from where your submission script is, PBS starts from your home directory.

When you get <jobid> you can check you job status with scontrol show job <jobid>

Batch system - Fimm

Squeue status

Status	Meaning
PD	Pending
R	Running
S	Suspended
CG	Completing
CD	Completed
CF	Configuring
F	Failed
TO	Timeout
PR	Preempted
NF	Node failed

Batch system - Fimm

Job management

- list all current jobs for user

```
$ squeue –u <username>
$ squeue –u <username> –t RUNNING
$ squeue –u <username> –t PENDING
$ squeue –u <username> –p kjem
```

- check your job

```
$ scontrol show jobid <jobid>
$ scontrol show jobid –dd <jobid>
```

Batch system - Fimm

Job management

- to cancel one job

```
$ scancel <jobid>
$ scancel –u <username>
$ scancel –t PENDING –u <username>
$ scancel ––name <myJobName>
```

- Show available partition

```
$ scontrol show partition
```

Batch system - Fimm

Job management

- sinfo - reports the state of partitions and nodes managed by SLURM.

```
$ sinfo –p hpc –l
```

- srun is used to submit a job for execution or initiate job steps in real time

```
$ srun –n 2 ––tasks–per–node=1 ––exclusive  my_program
$ srun –N2 –n1 –l /bin/hostname
```

(you will get warning)

Batch system - Fimm

SLURM MPI examples

```
#!/bin/bash
#SBATCH --job-name=test_mpi
#SBATCH --output=res_mpi.txt
#SBATCH --ntasks=4
#SBATCH --time=10:00
#SBATCH --mem-per-cpu=100

module load openmpi
mpirun hello.mpi
```

Batch system - Fimm

SLURM interactive job

```
$ srun --pty $SHELL

$ srun -p idle -I -N 1 -c 1 -pty -t 0-00:05 $SHELL

$ srun --nodes=1 --ntasks-per-node=4 --mem-per-cpu=1024 --pty $SHELL

IF you want x11 then -Y option
```

Batch system - Fimm

Other useful information

- Convert your pbs script to slurm script

  ```
  $ pbs2slurm < orig-file.pbs > new-file.slum
  ```

  only tested for small scripts

- Use PBS script and PBS commands directly on fimm

  Not recommended, better run slurm script

PBS	SLURM	Comment
`#PBS -N JobName`	`#SBATCH --job-name=JobName`	Give the job a name.
`#PBS -A VR9999`	`#SBATCH --account=VR9999`	Account to charge quota.
`#PBS -q batch`	`#SBATCH -p main`	PBS default scheduler queue is called 'batch'. In SLURM a queue is called a partition, and the default is 'main'.
`#PBS -l procs=128`	`#SBATCH --ntasks=128`	The number of cores.
`#PBS -l pvmem=24GB`	`#SBATCH --mem-per-cpu=24576`	Per core memory. SLURM must be in MB.
`#PBS -l walltime=30:23:59:59`	`#SBATCH --time=30-23:59:59`	Walltime. Note '-' between day(s) and hours for SLURM.
`#PBS -m a`	`#SBATCH --mail-type=FAIL`	Send email notification when: job fails.
`#PBS -m b`	`#SBATCH --mail-type=BEGIN`	job start running.
`#PBS -m e`	`#SBATCH --mail-type=END`	job stops.
`#PBS -M name@email.address`	`#SBATCH --mail-user name@email.address`	e-mail address to send information to. Best to not use this, and the system will use your known e-mail address.
`#PBS -o path`	`#SBATCH --output path/file-%j.ext1`	Redirect output to this path (PBS)/ file (SLURM). For SLURM, %j is replaced by the job number
`#PBS -e path`	`#SBATCH --error path/file-%j.ext2`	Redirect error to this path (PBS)/ file (SLURM). For SLURM, %j is replaced by the job number.
`cd $PBS_O_WORKDIR`		Change to the directory that the script was launched from. This is the default for SLURM.

Questions?

HPC Docs

http://docs.hpc.uib.no/

Getting Help

support-uib@notur.no

hpc-support@hpc.uib.no

HPC News

http://syslog.hpc.uib.no/

@uib_vd

Thank you!

UNIVERSITY OF BERGEN

Scientific Computing, IT department