User interface of the HPC Systems at LRZ

Dr. R. Bader
Leibniz Supercomputing Centre
Logging in
Secure Shell

- ssh provides secure connections via insecure channels (Internet, telephone lines, ...)
- “Secure” means
  - User is authenticated to the system
  - System is authenticated to the user
  - All transmitted data is encrypted
- Only the simplest usage case is given here:
  - Start a shell on the target system
  - after performing password authentication
- Please read the document

http://www.lrz.de/services/compute/ssh/

for further details.
  - In particular, the preparation step of obtaining the target system‘s public keys prior to the first login
Using Secure Shell

**ssh command line:**

```
ssh -Y -l <account name>  <target system>
```

- requires entering your password

**Target system SuperMUC**

- `supermuc.lrz.de`
- shell will be started on one of multiple login nodes

**Notes for SuperMUC:**

- IP address of source system (your desktop) must be registered with LRZ
- The target address will probably change when phase 2 starts operation in summer 2015

**Target system Linux Cluster**

- one of `lxlogin1.lrz.de`
  `lxlogin2.lrz.de`
  `lxlogin3.lrz.de`
  `lxlogin4.lrz.de`

Tunnel X11 traffic. Will only work if you run an X server on your desktop.
Using Secure Shell on Windows

Variant 1: PuTTY
- a graphical user interface that opens a command window
- the SSH entry contains the option to tunnel X11 traffic
- again, this requires that an X server runs locally (e.g., XMing or Exceed)

Variant 2:
- Install Cygwin and use the command line in a terminal
Environment Modules
Environment Variables: some examples

- **Database-like**
  ```bash
  echo $PATH
  /usr/bin:/usr/X11R6/bin:/client/bin
  ```
  - colon (or sometimes other character) used as separator

- **User-dependent settings (e.g., for file systems)**
  ```bash
  echo $SCRATCH
  /gpfs/scratch/uf34x/zhh34ab
  ```

- **Settings for development environment**
  ```bash
  echo $MKL_SHLIB
  -L/lrz/sys/... -lmkl_intel_lp64 ...
  ```
  - to be used for linking against the Intel math kernel library

- **How does one handle these settings?**
  - Large numbers of variable settings are a nuisance to type in
  - For many applications, multiple versions are available
  - the `module` command permits handling environment settings efficiently
Example:

User wants to run Gaussian Quantum Chemistry package

```
$ g09 test001.com
-bash: g09: command not found
$ module load gaussian
$ which g09
$ /usr/local/sys/gaussian/g09.C.02/bin/g09
$ g09 test001.com
$ module unload gaussian
$ which g09
which: no g09 in (...)
```
What does loading the module change?

$ module show gaussian

/usr/local/sys/share/modules/files/applications/gaussian:

module-whatis   Enable usage of Gaussian 09 C.02 QC software package
prereq         tempdir
conflict      gaussian
setenv GAUSS_SCRDIR /lustre/a2832ba
setenv GAUSS_EXEDIR
   /usr/local/sys/gaussian/g09.C.02/bin:/usr/local/sys/gaussian/g09.C.02/exe
setenv GMAIN
    /usr/local/sys/gaussian/g09.C.02/bin:/usr/local/sys/gaussian/g09.C.02/exe
setenv GAUSS_ARCHDIR /usr/local/sys/gaussian/arch
setenv g09BASIS /usr/local/sys/gaussian/g09.C.02/basis
prepend-path PATH
   /usr/local/sys/gaussian/g09.C.02/bin:/usr/local/sys/gaussian/g09.C.02/exe
prepend-path LD_LIBRARY_PATH /usr/local/sys/gaussian/g09.C.02/lib
How do I know what modules exist?

$ module avail gaussian

--- /usr/local/sys/share/modules/files/applications ---
gaussian/03.D.01 gaussian/09.C.02 gaussian/09.D.01 (default)
--- /usr/local/sys/share/modules/files/compilers ---
--- /usr/local/sys/share/modules/files/environment ---
--- /usr/local/sys/share/modules/files/libraries ---
--- /usr/local/sys/share/modules/files/parallel ---
--- /usr/local/sys/share/modules/files/tools ---
--- /usr/local/sys/share/modules/files/grid ---
--- /usr/local/sys/share/modules/extfiles ---

patterns are also possible e.g.,
module avail “gaussi*”

Without an argument, a list of all modules is displayed.
Resolving conflicts

- Cannot have two different versions of Gaussian at the same time
  - LRZ tries to prevent destructive interactions between packages with overlapping functionality or incompatible dependencies

```bash
$ module load gaussian
$ module load gaussian/09.A.02
WARNING: gaussian/09.A.02 cannot be loaded due to a conflict.
HINT: Might try "module unload gaussian" first.
$ module switch gaussian gaussian/09.A.02
```

nearly the same as:

```
module unload gaussian
module load gaussian/09.A.02
```
Which modules are presently loaded?

$ module list
Currently Loaded Modulefiles:
  1) admin/1.0  5) ccomp/intel/14.0
  2) tempdir/1.0  6) mpi.intel/4.1
  3) mkl/11.1  7) lrz/default
  4) fortran/intel/14.0

- Default settings:
  - an initial set of modules is automatically loaded at login
Dependencies between modules

- **Automatic resolution**
  - only possible if dependency is unique
  - typically done for application software
  - automatically load required modules

- **Manual resolution**
  - typically needed for development environment
  - user must manually set up module stack according to LRZ web documentation
  - failures unfortunately sometimes possible (start new shell if environment too corrupted)

- **Example for manual resolution:**
  - $ module unload mpi.ibm
  - $ module load gcc
  - $ module load mpi.intel/4.1_gcc

- **Purpose of this specific setup:**
  Use GCC (GNU compiler collection) and an Intel MPI release that uses GCC as underlying compiler

  - incorrect stacking of modules can lead to (sometimes quite nasty) problems
Please help – the module command is not found!

Possible causes that you can fix:

A new shell (or shell script) was started; it did not obtain the necessary function alias

- **therapy**: insert the lines

```bash
typeset -f module > /dev/null
if [ $? != 0 -a -r /etc/profile.d/modules.sh ] ; then
  source /etc/profile.d/modules.sh
fi
```

into your shell initialization file (~/.bashrc for bash, $ENV for ksh)

- this preserves the state of the parent shell

A batch script was started; it does not automatically obtain the necessary function alias

- **therapy**: insert the line

```bash
source /etc/profile.d/modules.sh
```

into your batch script **before** any other commands are executed

otherwise: system problem (rare)
Further module commands and documentation

$ module help
  • provides information about usage of module command

$ module [help|whatis] <name>
  • provides information about module <name>

$ module apropos <string>
  • will run whatis on every module containing <string> in its whatis section

$ module purge
  • removes all loaded modules
  • beware: on LRZ systems, this will cripple your shell. At least the environment module lrz should be loaded after a purge

Web
  http://www.lrz.de/services/software/utilities/modules

UNIX manual pages
  • man module; man modulefile

further topic: how to set up your own modules
Batch Processing
Running HPC Workloads

- **Typical starting point:** a shell script that...

  - sets up working area
  - copies data there
  - starts up the **parallel** program
  - copies still-needed results back to permanent location

- **You cannot use this **without changes** on LRZ's HPC systems**

  - efficient multi-user operation → use of **batch scheduler** is required

```bash
#!/bin/bash

ddir=$(pwd)
wdir=$(mktemp -d $SCRATCH/job.XXXXX)

cp $ddir/... $wdir

mpirun -n 512 $ddir/Prog.exe

stat=$?

if [ $stat -eq 0 ] then
    cp ... $ddir
fi
```

file script.cmd
What does a batch scheduler do?

Scheduler

User

Available Resources

LRZ Policies

User Requirements

© 2015 LRZ

User interface on LRZ HPC Systems

A formal description of these must be supplied.

To be shared out among many users.

Place limitations on resource assignments.

Execution of job script on (usually) dedicated resources … after a waiting time in proportion to queued jobs.
Batch schedulers at LRZ

**SuperMUC uses IBM’s LoadLeveler**
- Web documentation at
  
  http://www.lrz.de/services/compute/supermuc/loadleveler/

**Cluster Systems use SLURM**
(Simple Linux Utility for Resource Management)
- Web documentation for **parallel** processing at
  
  http://www.lrz.de/services/compute/linux-cluster/batch_parallel/

- and for **serial** processing at
  
  http://www.lrz.de/services/compute/linux-cluster/batch_serial/
Scheduler-specific control languages

- Non-executable (i.e. comment) key-value pairs that permit specifications
  - syntax is specific to the scheduler used
  - semantics are often shared/similar across different schedulers

- Let's start with relevant non-resource settings:
  - initial path, output and error file
  - messaging via e-mail

```
#!/bin/bash
#@ job_name = myprog
#@ network.MPI = sn_all,not_shared,us
#@ initialdir = $(home)/ddir
#@ output = myprog$(jobid).out
#@ error = myprog$(jobid).err
#@ notification = complete
#@ notify_user = <mail address>
...  LoadLeveler  Obligatory! keys following this one will be ignored
#@ queue
ddir=$(pwd)
...  SLURM
```

```
#!/bin/bash
#SBATCH -J myprog
#SBATCH -D /home/hpc/.../.../ddir
#SBATCH -o myprog.%j.%N.out
#SBATCH -e myprog.%j.%N.err
#SBATCH --mail-type=end
#SBATCH --mail-user=<mail address>
...  SLURM
```

© 2015 LRZ
Setting up the environment

Avoid inheriting the submission environment

- this makes debugging difficult to impossible
- **limited exception**: specific variables that must be propagated between job steps
  → check out documentation; write annotated variable values to stdout in script.

Instead, use the module command within the script

- possibly complemented by your own environment settings

```
#!/bin/bash
...
#@

environment = COPY_ALL
...

source /etc/profile.d/modules.sh
module unload ...
module load ...
export MY_VAR=...

ddir=$(pwd)
...
```

LoadLeveler

```
#!/bin/bash
...

#SBATCH --export=ALL
#SBATCH --export=NONE
#SBATCH --get-user-env
...

source /etc/profile.d/modules.sh
module unload ...
module load ...
export MY_VAR=...

ddir=$(pwd)
...
```

SLURM

run system profile
Resource specifications (focusing on parallel jobs here!)

- **Which ones are relevant?**
  - number of nodes
  - number of tasks (MPI)
  - number of cores assigned to a task (for hybrid parallelism)
  - execution time limit

- **Constrained by architecture**
  - node memory
  - node/socket core count

- **Determined by user program:**
  - memory requirements
  - scalability
  - duration of run → may need to do checkpointing
  - I/O requirements → handled by quota (HOME/WORK) or capacity limit (SCRATCH)

- **Constrained by LRZ policies**
  - job class time limit
  - job class maximum node count

© 2015 LRZ

User interface on LRZ HPC Systems
Illustrating resource determination

- **Speed-Up (model or measurement)**
- **Available Memory (SuperMUC, in GB)**

2 TBytes memory need ~80 nodes

getting near to scaling limit

Number of nodes
Resource specifications cont‘d

MPI program with following assumptions:
- uses IBM PE
- MPP-mode: task count equal to core count

„general“ class:
- up to 512 nodes for up to 48 hours

Smaller simulation on MPP cluster:
- memory ~230 GByte
- uses Intel MPI in MPP-mode

„mpp1“ segment:
- currently up to 32 nodes for up to 48 hours

LoadLeveler

#!/bin/bash
#@ job_type = parallel
#@ class = general
#@ network.MPI = sn_all,not_shared,us
#@ node = 80
#@ total_tasks = 1280
...
mpiexec $ddir/prog.exe

SLURM

#!/bin/bash
#SBATCH --clusters=mpp1
#SBATCH --ntasks=256
#SBATCH --time=08:00:00
...
mpiexec -n 256 $ddir/prog.exe

16 cores per node

16 cores per node

node count (16) is in this case automatically determined
Other MPIs on SuperMUC

Need a variant job type
- example with Intel MPI

```bash
#!/bin/bash
#@ job_type = MPICH
#@ class = general
#@ network.MPI = sn_all,not_shared,us
#@ node = 80
#@ total_tasks = 1280

...
module unload mpi.ibm
module load mpi.intel
mpiexec -n 1280 $ddir/prog.exe
```

- reason: separately maintained node list is needed for MPI-specific startup mechanism
Submitting and querying a job

### LoadLeveler

```bash
$ llsubmit script.cmd
    ...
llsubmit: The job "srv03-ib.297418" has been submitted.
```

Submission returns **job id**: this is a tag by which the job is referred to. 
Note down this information for reporting in case a problem occurs.

```bash
$ llq -u $USER
```

### SLURM

```bash
$ sbatch script.cmd
    Submitted batch job 33343 on cluster mpp1
```

Query returns information about the state of your jobs.
If the `--u` option is omitted, all users' jobs state will be displayed.

```bash
$ llq -s <job ID>
```

Provide reason for job not starting

```bash
$ llq -l <job ID>
```

Return extensive information about job, in particular the hosts it is using.
LoadLeveler job query output

Ilq –u

<table>
<thead>
<tr>
<th>Id</th>
<th>Owner</th>
<th>Submitted</th>
<th>ST</th>
<th>PRI</th>
<th>Class</th>
<th>Running On</th>
</tr>
</thead>
<tbody>
<tr>
<td>srv04-ib.233136.0</td>
<td>xxxxyyy</td>
<td>2/11 12:31</td>
<td>R</td>
<td>50</td>
<td>micro</td>
<td>i18r05a38-ib</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>srv04-ib.233148.0</td>
<td>xxxxyyy</td>
<td>2/11 12:31</td>
<td>NQ</td>
<td>50</td>
<td>micro</td>
<td></td>
</tr>
<tr>
<td>srv03-ib.297418.0</td>
<td>xxxxyyy</td>
<td>2/14  08:25</td>
<td>I</td>
<td>50</td>
<td>general</td>
<td></td>
</tr>
</tbody>
</table>

job states:
R – running, I – idle, C – completing, D – deferred, NQ – not queued, H – user or system hold

Ilq –s on non-queued job

===== EVALUATIONS FOR JOB STEP srv04-ib.233148.0 =====
Step state : Not Queued
User = xxxxyy has reached the maximum number of job steps allowed in the queue for class .

remains in job list, but is currently not considered for scheduling
SLURM job query output

```
- squeue -M mpp1
```

<table>
<thead>
<tr>
<th>CLUSTER: mpp1</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBID</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>32982</td>
</tr>
<tr>
<td>32803</td>
</tr>
</tbody>
</table>

**job states:** for the most part self-explaining; the same applies for the reason given for a job not running. In particular:

(Resourses) – the scheduler must free nodes to start the job
(Priority) – other jobs come earlier in the queue
LRZ-provided add-ons on SuperMUC

- **llx**
  - nicely formatted text output (with PBS-like style)

- **llj**
  - graphical overview of system state
  - jobs coded by letters

- **llrun**
  - pseudo-interactive program runs (IBM PE only)
### SLURM add-ons

#### sview GUI

- Resource hungry → please don’t run permanently

#### salloc
- Reserve interactive resources on a small scale for a short time
Canceling a job

**LoadLeveler**

$ llcancel <job id>

...  
llcancel: Cancel command has been sent to the central manager.

- sometimes, repetition of llcancel appears to be needed

**SLURM**

$ scancel -M <cluster> <job id>
Energy Aware Jobs on SuperMUC

- Sandy Bridge nominal frequency is **2.7 GHz**
- To save energy, the default frequency used for jobs is **2.3 GHz**
  - turbo mode is disabled altogether

Loadleveler EAS feature permits changing the frequency indirectly

- specifications needed:
- first run (at least 10 min!) measures at 2.3 GHz
- subsequent runs execute at **energy-optimal** frequency, or at **increased** frequency subject to model prediction and following policy:

```
#!/bin/bash
#@ energy_policy_tag = myprog_tag
#@ minimize_time_to_solution = yes
```

<table>
<thead>
<tr>
<th>delivered frequency (GHz)</th>
<th>condition on run time change (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>-2.5</td>
</tr>
<tr>
<td>2.5</td>
<td>-5.0</td>
</tr>
<tr>
<td>2.6</td>
<td>-8.5</td>
</tr>
<tr>
<td>2.7</td>
<td>-12.0</td>
</tr>
</tbody>
</table>
Handling Energy Policy Tags

Additional LoadLeveler commands

- inspection of a tag that has been set up by a measurement run:

```bash
$ llrqetag -e my_prog_tag
...
Frequency(GHZ)  EstEngCons(KWH)  EngVar(%)  EstTime(Sec)  TimeVar(%)  Power(W)
2.70           0.001380         18.10       61            -14.08      81.43
2.60           0.001317         12.75       63            -11.27      75.27
2.50           0.001266         8.39        65            -8.45       70.14
2.40           0.001241         6.26        68            -4.23       65.72
2.30           0.001168         0.00        71            0.00        59.24
2.20           0.001183         1.26        73            2.82        58.34
2.10           0.001159         -0.82       77            8.45        54.17
2.00           0.001155         -1.15       80            12.68       51.97
1.90           0.001148         -1.72       85            19.72       48.63
1.80           0.001158         -0.88       90            26.76       46.32
1.70           0.001163         -0.42       95            33.80       44.09
1.60           0.001189         1.78        101           42.25       42.38
...
```

- deletion of a tag

```bash
$ llrrmetag -e my_prog_tag
```
Thank you for your attention!

Any questions?