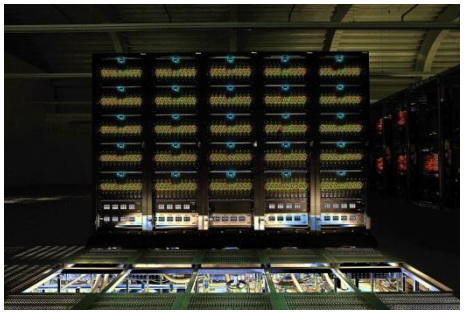
The RWTH Compute Cluster Environment



Source: D. Both, Bull GmbH

Tim Cramer

11.03.2013

How to login

Frontends

cluster.rz.RWTH-Aachen.DE	cluster2.rz.RWTH-Aachen.DE
cluster-x.rz.RWTH-Aachen.DE	cluster-x2.rz.RWTH-Aachen.DE
cluster-linux.rz.RWTH-Aachen.DE	cluster-linux-opteron.rz.RWTH-Aachen.DE
cluster-linux-xeon.rz.RWTH-Aachen.DE	cluster-linux-nehalem.rz.RWTH-Aachen.DE
cluster-linux-tuning.rz.RWTH-Aachen.DE	cluster-copy.rz.RWTH-Aachen.DE

- Use frontends to develop program, compile applications, prepare job scripts or debug programs
- cgroups activated for fair-share
- login / SCP File transfer:
  - `$ ssh [-Y] user@cluster.rz.rwth-aachen.de`

    `$ scp [[user@]host1:]file1 [...] [[user@]host2:]file2`

Why to use the batch system?

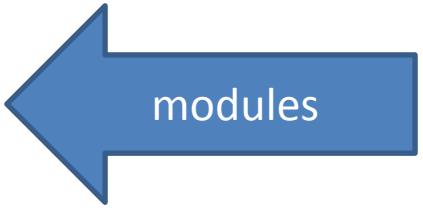
- **Use of backend nodes via our batch system for large calculations**
  - Contra:
    - Jobs sometimes need to wait before they can start
  - Pro:
    - Nodes are not overloaded with too many jobs
    - Jobs with long runtime can be executed
    - Systems are also used at night and on the weekend
    - Fair share of the resources for all users
    - The only possibilty to handle such a big amount of compute nodes

Module System (1/2)

- **Many compilers, MPIs and ISV software**
- **The module system helps to manage all the packages**
  - List loaded modules
    - `$ module list`
  - List available modules
    - `$ module avail`
  - Load / unload a software
    - `$ module load <modulename>`
    - `$ module unload <modulename>`
  - Exchange a module (Some modules depend on each other)
    - `$ module switch <oldmodule> <newmodule>`
  - Reload all modules (May fix your environment, especially with a NX session)
    - `$ module reload`
  - Find out in which category a module is:
    - `$ module apropos <modulename>`

Module System (2/2)

```
$ module avail
------------ /usr/local_rwth/modules/modulefiles/linux/x86-64/DEVELOP ------------
cmake/2.8.5(default)   inteltbb/4.1(default)
cuda/40                intelvtune/XE2013U02(default)
cuda/41                likwid/system-default(default)
cuda/50(default)       nagfor/5.2
ddt/2.6                nagfor/5.3.1(default)
ddt/3.0(default)       openmpi/1.5.3
gcc/4.3                openmpi/1.6.1(default)
gcc/4.5                openmpi/1.6.1mt
gcc/4.6                openmpi/1.6.4
gcc/4.7                openmpi/1.6.4mt
...                                    ...
------------ /usr/local_rwth/modules/modulefiles/GLOBAL ------------
BETA      DEPRECATED GRAPHICS   MATH      TECHNICS   VIHPS
CHEMISTRY  DEVELOP    LIBRARIES  MISC      UNITE
```



LSF: Job Submission (1/3)

How to submit a job

- `$ bsub [options] command [arguments]`

General parameters

Parameter	Description
`-J <name>`	Job name
`-o <path>`	Standard out
`-e <path>`	Standard error
`-B`	Send mail when job starts running
`-N`	Send mail when job is done
`-u <mailaddress>`	Recepient of mails
`-P <projectname>`	Assign the job to the specified project (e.g. jara, integrative hosting costumers)
`-U <reservation>`	Use this for advanced reservations

LSF: Job Submission (2/3)

RWTH AACHEN UNIVERSITY

- **How to submit a job**
  - `$ bsub [options] command [arguments]`
- **Parameters for job limits / resources**

Parameter	Description
`-W <runlimit>`	Sets the hard runtime limit in the format [hour:]minute [default: 15]
`-M <memlimit>`	Sets a **per-process** memory limit in MB [default: 512]
`-S <stacklimit>`	Set a **per-process** stack size limit in MB [default: 10]
`-x`	Request node(s) exclusive
`-R "select[hpcwork]"`	ALWAYS set if you using the HPCWORK (Lustre file system)

LSF: Job Submission (3/3)

- **How to submit a job**
  - `$ bsub [options] command [arguments]`
- **Parameters parallel jobs**

Parameter	Description
`-n <min_proc>[,max_proc]`	Submits a parallel job and specifies the number of processors required [default: 1]
`-a openmp`	Use this to submit a shared memory job (e.g. OpenMP)
`-a {open|intel}mpi`	Specify the MPI (remember to switch the module for Intel MPI)
`-R „span[hosts=1]“`	Request the compute slots on the same node
`-R „span[ptile=n]“`	Will span *n* proceses per node (hybrid)

  - `$MPIEXEC $FLAGS_MPI_BATCH a.out`

LSF: Using the Magic Cookie

- **You can use the magic cookie `#BSUB` for a batch script `job.sh`**

```
#!/bin/zsh
#BSUB -J TESTJOB                    #Job name
#BSUB -o TESTJOB.o%J                #STDOUT, the %J is the job id
#BSUB -e TESTJOB.e%J                #STDERR, the %J is the job id
#BSUB -We 80                        #Request 80 minutes
#BSUB -W 100                        #Will run max 100 minutes
#BSUB -M 1024                       #Request 1024 MB virtual mem
#BSUB -u user@rwth-aachen.de        #Specify your mail
#BSUB -N                            #Send a mail when job is done
cd /home/user/workdirectory         #Change to the work directory
a.out                               #Execute your application
```

- **Submit this job**
  - `$ bsub < job.sh`
  - **Please note the <, with SGE this was not needed, with LSF it is**

LSF: Environment Variables

- **For your submission script some environment variables might be useful**
  - **Note:** These variables are interpreted within your script,

    but not in combination with the magic cookie **`#BSUB`**

Variable	Description
`LSB_JOBID`	The job ID assigned by LSF
`LSB_JOBINDEX`	The job array index
`LSB_JOBNAME`	The name of the job
`LSB_HOSTS`	The list of hosts selected by LSF to run the job
`LSB_MCPU_HOSTS`	The list of the hosts and the number of CPUs used
`LSB_DJOB_NUMPROC`	The number of slots allocated to the job

LSF: Job Status (1/2)

Use `bjobs` to display information about LSF jobs

- `$ bjobs [options][jobid]`

```
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
3324    tc53084 RUN   serial     linuxtc02   ib_bull     BURN_CPU_1 Jun 17 18:14
3325    tc53084 PEND  serial     linuxtc02   ib_bull     BURN_CPU_1 Jun 17 18:14
3326    tc53084 RUN   parallel   linuxtc02   12*ib_bull  *RN_CPU_12 Jun 17 18:14
3327    tc53084 PEND  parallel   linuxtc02   12*ib_bull  *RN_CPU_12 Jun 17 18:14
```

Option	Description
-l	Long format – displays detailed information for each job
-w	Wide format - displays job information without truncating fields
-r	Displays running jobs
-p	Displays pending job and the **pending reasons**
-s	Displays suspended jobs and the suspending reason

- LSF can display the reasons for a pending job

LSF: Job Status (2/2)

- **Use `bpeek` to display stdout and stderr of an running LSF job**
  - `$ bpeek [options][jobid]`

```
<< output from stdout >>
Allocating 512 MB of RAM per process
Writing to 512 MB of RAM per process
PROCESS 0: Hello World!
PROCESS 1: Hello World!
[ application output ]
<< output from stderr >>
```

- **Remove a job from the queue**
  - `$ bkill [jobid]`
- **Remove all jobs from the queue**
  - `$ bkill 0`

Feedback and Documentation

RWTH Compute Cluster Environment

- HPC Users‘s Guide:

  http://www.rz.rwth-aachen.de/hpc/primer

- Online documentation (including example scripts):

  https://wiki2.rz.rwth-aachen.de/

- Full LSF documentation:

  http://www1.rz.rwth-aachen.de/manuals/LSF/8.0/index.html

- Man-Pages for all commands available

- In case of errors / problems let us know:

  servicedesk@rz.rwth-aachen.de

We provide laptops exercises

- Login to the laptops with the local „hpclab“ account (PC pool accounts might also work)
- Use PUTTY, the NX Client or X-Win32 to login to the cluster (use “hpclabXY” or your own account)
- Feel free to **ask questions** or prepare your own job scripts
- We prepared many exercises, skip those which are not relevant for you

Source: D. Both, Bull GmbH