Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

Hands-on: Introduction to Multiuser Cluster Systems at LRZ

Theory & Practice

Florent Dufour · florent@lrz.de

Big Data & Artificial Intelligence
Leibniz Supercomputing Centre
October 10, 2022

Module description

It has been about 350 years that separate the original 1673 Leibniz mechanical calculator from today's Leibniz Supercomputing Centre's (LRZ) facilities located on the Campus Garching. And yet, the spirit has not changed. To quote the German mathematician:

"It is beneath the dignity of excellent men to waste their time in calculation when any peasant could do the work just as accurately with the aid of a machine."

This course module will allow participants to live up to their dignity by providing a comprehensive walkthrough and usage guide to such contemporary types of these machines that potentially fill whole buildings.

In a general overview, historical and current developments and trends in the space of scientific computing and cluster systems will be presented. This will, amongst others, address the following questions: How do modern cluster systems work and how are they architected? How did we come to High Performance Computing, High Performance Data Analytics and High Performance AI? What makes a system adequate to a specific workload? How are these systems operated and how are they made available to their users? In addition, typical interaction methods and usage patterns will be covered, including various possibilites of setting up user environments (e.g. environment variables and modules, user space package managers, containers) as well as tools for resource allocation (i.e. Slurm Workload Manager) and efficient parallelization (MPI, OpenMP, ...). Finally, an overview of different compute clusters as well as their background storage systems operated by LRZ will be provided. The requirements for acquiring access to these systems will be covered as well.

Participants will gain a good understanding of the characteristics of multiuser cluster systems in general and will practise basic methods of typical interaction. They will familiarize themselves with the landscape of cluster systems available at LRZ and this will allow them to choose the right system for their own compute projects.

This is document revision: 2022.2a
Latest revision available at: https://doku.lrz.de/x/eQBvB

Table of Contents

List of Code snippets

Glossary

SLURM Simple Linux Utility for Resource Management.

VM Virtual Machine.

1 Get into the workbench

We have a workbench ready for you. It is a Virtual Machine (VM) in which we will run your own SLURM cluster. The first step consists in getting into the VM.

```
# ---------------------- #
# On your local machine #
# ---------------------- #

whoami
# florent

uname -a
# Darwin BADWLRZ-AB12345 20.6.0 Darwin Kernel Version 20.6.0: Mon Aug 30 06:12:21 PDT 2021; root:xnu-7195.141.6~3/RELEASE_X86_64 x86_64


ssh <user>@<IP>
# Replace <user> and <IP> with yours.

# The authenticity of host <IP> can't be established.
# ECDSA key fingerprint is SHA256:BZgws5BArCfwE6rDuN5i/aLgZMAKuC4si2D+ZuLN5gw.
# Are you sure you want to continue connecting (yes/no)? yes
# Warning: Permanently added <IP> (ECDSA) to the list of known hosts.
# <user>@<IP> password:

# Type your password when prompted, it's normal if you don't see what you type!

# ------------------------------ #
# On the compute cloud workbench #
# ------------------------------ #

# Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-81-generic x86_64)

whoami
# <user>

uname -a
# Linux course-node 4.19.0-14-cloud-amd64 #1 SMP Debian 4.19.171-2 (2021-01-30) x86_64 GNU/Linux

# Success! We can proceed from here!
```

Code snippet 1: We SSH into the workbench

Once into the VM, you'll be able to start and enter the toy SLURM cluster.

```
# We are in the VM
# The prompt looks like:
# hlrbkurs42@course-node:~$


# Let's start the cluster
slurm-start
# Starting mysql ... done
# Starting slurmdbd ... done
# Starting slurmctld ... done
# Starting c1        ... done
# Starting c2        ... done

# As you can see, we are starting a database,
```

```
# slurm services, as well as 2 compute nodes c1 and c2

# Let's enter the cluster
slurm-enter

# The prompt now looks like
# [root@slurmctld /]#

# Success, you're on the control node
# of the slurnm cluster
```

Code snippet 2: We start and enter the SLURM cluster

Let's explore some SLURM commands. You'll be using: `sinfo` and `squeue` quite a lot.

```
# Let's gather the information on the cluster
# Let's add the --cluster=all option

sinfo
# CLUSTER: linux
# PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
# normal*      up 5-00:00:00      2   idle c[1-2]

# We have access to 1 cluster named linux
# The cluster contains 1 slurm partition composed of 2 compute nodes
# Both nodes are idle at the moment, nothing is running

# Let's see if there is jobs queued
squeue
#JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)

# It's empty!

# The cluster is yours
```

Code snippet 3: We get familiar with our SLURM cluster

Question

Can you SSH into the Linux cluster or datalab login nodes and compare the slurm partitions and nodes available there?

2 Interactive Jobs

Alright, it's about time we start using out cluster. Let's start interacting interactively with the cluster before submitting batch jobs.

```
# Let's run a job interactively

srun --pty bash # We keep the pty open and ask to run bash
hostname # The prompt changed, let's confirm we are on compute node c1

# From there I can do computation
# For example, python is available on the compute node

python
# Python 3.6.8 (default, Sep 13 2022, 09:02:49)
# [GCC 8.5.0 20210514 (Red Hat 8.5.0-10)] on linux
# Type "help", "copyright", "credits" or "license" for more information.
# >>> 2+2
# 4 # That's HPC (!)

exit()
exit

# We are back into the login node
```

Code snippet 4: We run our first interactive job

Cool, let's dive a bit more how job are scheduled now

```
# Let's keep c1 busy for a while
# We use "&" to run the job in the background
srun sleep infinity &

sinfo
# PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
# normal*      up 5-00:00:00      1  alloc c1
# normal*      up 5-00:00:00      1   idle c2

# c1 is allocated, c2 is free

# Let's check the slurm queue
squeue
# JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
#     8    normal    sleep     root  R       2:11      1 c1

# What happens if we request more resources=
# Let's request a node and run bash interactively on it
srun --pty bash
hostname
# c2
# I am c2 because c1 is allocated

exit

# Let's now keep c2 busy a 30 secs
srun --pty sleep 30 & # make c2 busy 30 secs

# What happens if we submit a job while both nodes are allocated
srun --pty sleep 30 &
# srun: job 11 queued and waiting for resources
# We are queued
```

```
33
34 squeue
35 # JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
36 #    11    normal    sleep     root PD       0:00      1 (Resources)
37 #     8    normal    sleep     root  R      14:37      1 c1
38 #    10    normal    sleep     root  R       2:21      1 c2
39
40 sacancel 8 # Let's cancel job on c1
41 # srun: job 11 has been allocated resources
42 # Since c is freed, the next job is starting
43
44 sacct # Let's check the accounting of what we have been doing
45 # JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
46 # ------------ ---------- ---------- ---------- ---------- ---------- --------
47 # 7                 bash     normal       root          1     FAILED      1:0
48 # 8                sleep     normal       root          1 CANCELLED+      0:0
49 # 9                 bash     normal       root          1  COMPLETED      0:0
50 # 10               sleep     normal       root          1    RUNNING      0:0
51 # 11               sleep     normal       root          1    RUNNING      0:0
```

Code snippet 5: We explore how jobs are scheduled on a SLURM cluster

3 Batch Jobs

I hope SLURM starts to make sense to you! This might be the right time now to introduce you to batch jobs. This is how you would usually run your jobs. For this, you would write a batch script and submit it with the SLURM scheduler.

```
#!/bin/bash
#SBATCH --cluster=linux
#SBATCH --nodes=2
#SBATCH --output=batch.stdout
#SBATCH --error=batch.stderr

echo "Hello, I am a script run on node: `hostname`"
```

Code snippet 6: A simple batch script

And this is how you would run this job:

```
sbatch batch.sh
```

Code snippet 7: We run a simple batch script

Question

Can you submit 2 jobs and try submitting a third? Do you observe the same behaviour as when we did it interactively?