H3ABioNet

Pan African Bioinformatics Network for H3Africa

Basic Beowulf HPC Installation and Configuration Guide with SLURM

Developed by

The H3ABionet Pipelines and Computing Work Package, Computing Infrastructure project team

Prepared for the greater

H3ABioNet and H3Africa Consortium communities

Purpose

The purpose of this document is to guide the reader through the installation and configuration of a basic Beowolf HPC cluster using SLURM as the resource manager.

Document Control

Date	Authorization	Version	Description

Assumptions

This guide makes the below assumptions:

- Familiarity with installing an Ubuntu OS
- The reader is comfortable working at the command line
- There is adequate power and cooling available for the cluster
- A disaster recovery plan has been implemented or at least is being considered.
- Consideration has been given to data and server security
- Server resources such as disk RAID’s, etc have already been setup
- Should the reader be following this guide using virtual machines, the assumption is made that these virtual machines have already been setup and are operational.

How to read this guide

- General text describing each section and commands are written in the Arial font, size 11. This text is to be read to understand which skills will be gained in the following section and to describe the command operation.

- When noting a variation to a default command or to note a point or warning, the default text will be highlighted in yellow.

- When noting a tip, for example, a command that can be run in multiple ways producing the same output, this text will be highlighted in green.

- When giving examples of actual expected output, this text will be highlighted with a grey background.

- When listing a command that is to be run by the reader to produce an output, the command will be in light blue italics.

<Insert-table-of-contents>

Abbreviations

OS Operating System

NFS Network File Server

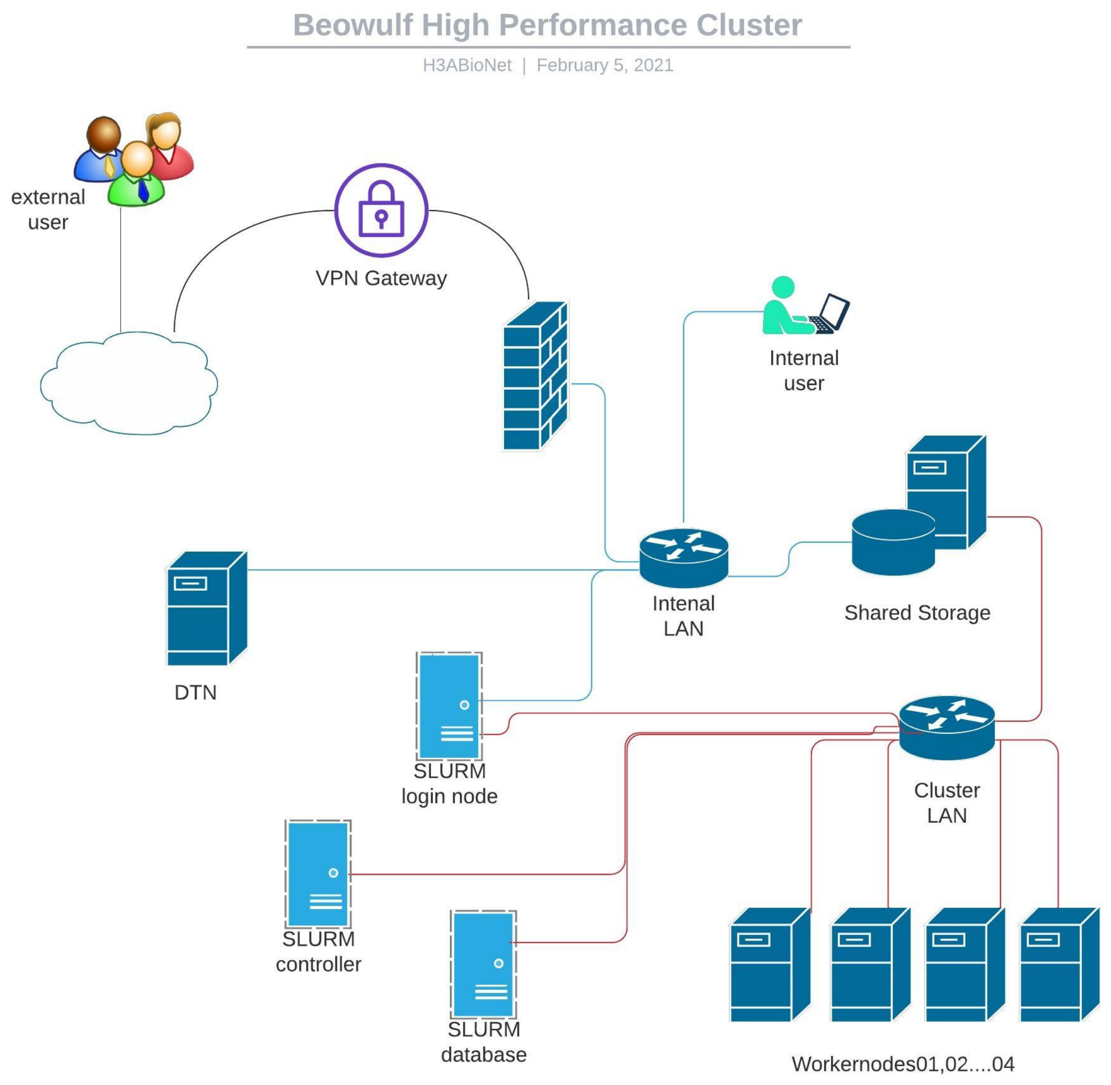
HPC High-Performance Cluster

1. Introduction

Access to big data has drastically increased over the last decade – this is especially true in the Bioinformatics arena. Hardware vendors have made great progress in packing more compute resources into single server – however, even with this advancement in modern technology, single servers still do not have the necessary resource capacity to analyse these large datasets. Too meet this computational need, many organizations turn to High-Performance Computing (HPC). Usually, HPC systems are out of reach for most. To bridge this gap, the Open-Source community has developed various ways of developing small scale HPC systems using every day hardware that is financially viable for smaller organizations to implement. Using open-source tools, one could develop an HPC system across a bunch of desktop machines or smaller, off-the-shelf servers.

The purpose of this guide is to walk the reader through setting up a Beowulf HPC cluster using the SLURM resource manager to manage system resources and job submission. This guide is based on the collective experiences in implementing HPC systems in the H3ABioNet consortium.

Below is a high-level diagram of a Beowulf HPC cluster.



Source: original

2. Server hardware

The cluster can be created using physical or virtual servers. If you are new to Linux, specifically Ubuntu and are not sure how to install the OS, please follow the >> *insert hyperlink to getting started guide* "LINUX - Getting Started". The resources such as RAM, local disk storage, network speeds and number of cores required for the servers are dependent on the expected computational workload of the cluster.

It is advisable for the worker nodes to have the same hardware and processor types. Example, all Intel or AMD's, etc. While this is advisable, it is not strictly necessary.

3. OS Installation and configuration

This guide does not cover the specifics of an Ubuntu OS installation. If you are not familiar with installing the Ubuntu OS, follow the *<insert-hyper-link>* "Linux: Getting Started" guide for detailed instructions.

Depending on the expected computation work load, some cluster setups have separate servers for user logins and another for the SLURM controller node. The login node is the entry point for user logins. The controller node is where the SLURM daemon that controls and monitors job submissions and cluster resources run. The login and controller functions can reside on the same server if need be.

One of the cluster prerequisites is that all servers in the cluster are able to connect to each other via the network and that there is some form of shared storage available to all server – especially the worker nodes. For this guide, we will setup a basic NFS server to share a few directories across the cluster. Lastly, all worker nodes need to be able to access the internet for OS updates and software installations. The worker nodes themselves do not need to connect directly to the internet, you can configure these servers to proxy through the controller node to access the internet.

With the OS installed, lets first start configuring all servers to talk to each other via the network.

3.1. Networking

One of the first tasks, post OS installation is to configure the local server networking. For this setup, I am using two IP address ranges. The one IP address range is referred to as the "public" IP. this IP has access to the internet and is part of the main organizational IP range. The worker nodes do not have direct access to the internet and are connected to the controller / login node via a second, private switch. Theses' IPs are referred to as the "private" IP address range.

Depending on the reader's needs, the cluster can be setup across a single switch and IP range but best practices suggested that the worker nodes communication over a private network / switch to limit traffic on the production network / switch.

Let us first configure the controller / login node.

3.1.1. SLURM controller machine setup

In Ubuntu 20.04, the networking is managed via netplan and all networking configurations are contained in a .yaml file. A fresh installation of Ubuntu 20.04 will either create a “50-cloud-init.yaml” or “00-installer-config.yaml” file. Whichever file is created, edit the file and place the below content into it.

```
sudo vim /etc/netplan/00-installer-config.yaml
```

```
network:
    ethernets:
  eth0:  # Public IP range
       addresses: []
             dhcp4: true # IP is assigned via a DHCP
       optional: true  # Allows the vm to boot if there are IP related issues
         eth1:  # private IP range
             addresses: [10.1.1.222/24]
             dhcp4: false # IP is statically assigned
         optional: true  # Allows the vm to boot if there are IP related issues
       version: 2
```

eth0 is setup for DHCP and connects to the public network with the eth1 network setup as a static IP and connects to private network. All SLURM worker nodes will communicate over the private network.

The .yaml files are quite finicky and will complain if one character is out by a space. To test that the configuration file has been edited correctly, use the ‘”try” command.

```
sudo netplan try
```

If no errors are encounted, run the below command to activate the changes you just made to the .yaml file

```
sudo netplan apply
```

Now run the “ifconfig” command to confirm that the second network card shows up

If you see IPs assigned, then your controller machine is correctly setup and you are ready to move on.

```
eth0 > <IP-from-public-IP-range>
eth1 > <IP-from-private-IP-range>
```

Next, we will map the private IP ranges to the worker node server’s user-friendly host names.

NOTE:

- The below step needs to be completed on all servers in the cluster so that they can communicate via the IP address or user-friendly host name.

```
sudo vim /etc/hosts
```

Add the below to the hosts file and edit with any additional IPs as required.

```
## SLURM cluster private IP range

# Controller
<private-ip-of-controller> <hostname-of-controller>

# workernodes
<private-ip-of-controller> <hostname-of-controller>
<private-ip-of-controller> <hostname-of-controller>
<private-ip-of-controller> <hostname-of-controller>
```

NOTE:

- For the worker nodes host names, I usually use the below format to identify them:

    <slurm-wrk-110>

slurm = this server is a member of the SLURM cluster
Wrk = identifies the server as a worker node
110 = is a number given to identify the specific server. This can be any number of your choosing

NOTE:

- The cluster worker nodes do not have public IP addresses, as such, they cannot access the internet directly. To allow them to access the internet for software updates and installation, we need to enable NAT and port forwarding on the controller. On the worker node we just need to have the "gateway" line that points to the controller's private IP.

On the controller, first backup the /etc/sysctl.conf file then find the line "net.ipv4.ip_forward=1"

```
sudo cp /etc/sysctl.conf /etc/sysctl.conf.original
sudo vim /etc/sysctl.conf
```

Find the line "net.ipv4.ip_forward=1". Uncomment it, save the file and restart the headnode.

On system reboot, run the below command to ensure that port forward for IPv4 has been enabled

```
sudo sysctl net.ipv4.ip_forward
```

Next, enable NAT in iptables and instruct it to route incoming traffic to the internet via the eth0 NIC / public NIC

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

NOTE:

- eth0 here is the public network card (NIC). The above command enables NAT and allows traffic coming in on the private IP range from the worker nodes which is then passed to eth0 and sent out to the internet now ssh into the worker node and ping www.google.com or update the repositories. You should now be able to access the internet via the controller node.

3.1.2. SLURM worker node machine setup

This section assumes that the OS has already been installed and you just need to configure the IP for networking.

The cluster worker nodes communicate on the local private network. First, we need to assign a static IP address in the private IP range.

Ubuntu 20.04 no longer uses the /etc/networking/interface file to record IP configuration. It uses the new "netplan" which uses ".yaml" style configuration files. When doing a fresh installation of Ubuntu 20.04, the default network .yaml file that is created often differs in the naming. Most of the time however, it is named "50-cloud-init.yaml". Edit this file to configure IP addresses.

```
sudo vim /etc/netplan/50-cloud-init.yaml  # edit the default network configuration file
```

set the static IP on your NIC that goes to the local CBIO network

```
network:
        ethernets:
          eth0:
            addresses: []
            dhcp4: true
            optional: true # when set to true, a server will still boot if there is a problem with the NIC.  This is an optional setting.
          eth1:
            dhcp4: false
            addresses: [<priate-ip>/24]  # type in the unique IP of the worker node.
```

NOTE:

- Don't forget the "/24" subnet bit or you will get errors when applying the changes

```
            gateway4: <private-ip-of-controller>

  version: 2
```

save and exit the file, then apply the changes.

```
sudo netplan apply  # converts the .yaml file into a configuration file readable by the backend network application and applies the changes.
```

Next add your organization's domain for name resolution

```
sudo vim /etc/resolve.conf
```

Add the below line to the bottom of the configuration file

```
search <your-organizations-domain-name>
```

Save and close the file.

Now map the IPs to the user-friendly host names.

```
## SLURM cluster private IP range

# Controller
<private-ip-of-controller> <hostname-of-controller>

# workernodes
<private-ip-of-controller> <hostname-of-controller>
<private-ip-of-controller> <hostname-of-controller>
<private-ip-of-controller> <hostname-of-controller>
```

As a quick test, try to ping some of the other servers in the cluster via IP address and hostname. If you do not encounter any errors, you can move on to the next step.

It is important that all servers in the cluster have the same date and time setting. Let's look at the date and time setup on all server to ensure that they are all synchronized.

3.2. Date and time synchronization

For the cluster to work optimally, you need to have the date and time on all the servers in the cluster synchronized.

to see what time zone the server is configured to use, use the “timedatectl” command or type “cat /etc/timezone”

```
timedatectl
```

You should see output similar to the below. Look for the “Time Zone” line to see what time zone the server is configured to use. We are specifically interested in the “bold” text

```
 Local time: Tue 2021-04-20 14:57:58 UTC
 Universal time: Tue 2021-04-20 14:57:58 UTC
 RTC time: Tue 2021-04-20 14:57:58
 Time zone: Etc/UTC (UTC, +0000)
 System clock synchronized: yes
NTP service: active
RTC in local TZ: no
```

Alternate command to view the time zone used

```
cat /etc/timezone
```

output

```
Etc/UTC
```

If the time zone listed in the above outputs is not the time zone for your area, we would need to update the system to use the correct time zone. First, list all the time zones to see if your time zone is listed.

```
timedatectl list-timezones
```

The above command will spit out a long list of time zones, to narrow down the options, use the “grep” command to filter by your country. Example, if you are located in Africa,

```
timedatectl list-timezones | grep Africa
```

Once you have found your time zone, copy the text and past it into the “set-timezone” command as per below. To change the time zone to the “Africa/Johannesburg” time zone, run the below command

```
sudo timedatectl set-timezone Africa/Johannesburg
```

Now use the timedatectl or “timezone” commands to confirm the changes.

You should now see your countries time zone reflected and the date and time should reflect your current date and time. We have one more item to check before we can move on. Ensure that the synchronized setting is set to “yes”. if not, edit using the “set-ntp” command.

If the setting is set to no, enable it with the below command

```
sudo timedatectl set-ntp
```

NOTE:

- The above needs to be repeated on all servers in the cluster

With the date, time and time zone configured. We can proceed to the next step – setting up password-less login.

3.3. SSH keys for password-less login

Next, we need to setup ssh keys for password-less login to connect to the worker-nodes from the control machine. If you have not yet created a SSH keypair, run the below command. If, however, you already have generated a SSH keypair, then simply copy the .pub file over to the worker node.

The “ssh-keygen” command generates two keys. The “id_rsa” is your private key and should never be shared with anyone. This key is used to validate the public key. The second key, “id_rsa.pub” is the public key you copy to the machines that you want to log into without a password.

Generate a ssh keypair if you do not already have one.

```
ssh-keygen # This will create a SSH keypair using the default RSA encryption.
```

NOTE:

- To use a different encryption, add the “-t” option and assign the type of encryption to be used “ssh-keygen -t ed25519”.

We don't set any passphrases so you can accept all defaults by depressing the enter key when prompted.

Once your ssh keypair has been created, copy the .pub key to the machine you want to access without typing in a password. We use the "ssh-copy-id" command to copy the public key to anther server.

NOTE:

- if you only have one ssh keypair, the .pub key will automatically be selected and copied into the ".ssh/known_hosts" or ".ssh/authorized_keys" files on the remote machine.

```
ssh-copy-id <user>@<workernode-ip>
```

If there were no errors with the ssh-copy-id command, you can proceed to test the setup. From the controller node, ssh into the worker nodes and this time you should not be prompted for a password.

If password-less logins are working you can proceed to the next step.

Next, we will move over to setting up NFS for shared storage.

4. Install and Setup NFS shares

Shared storage is an important part of the cluster as all worker nodes need to be able to access the same project directories. Ideally, the NFS server should be an independent server. However, if you are limited to the number of servers you have, this function can be installed on the controller node. NFS is an acronym for Network File Server – the NFS server is a quick and easy way to share centralized storage across multiple servers on the same network.

Let's begin by updating the OS repositories and installing the necessary NFS tools. Log into the server that will be providing the NFS or shared storage function.

```
sudo apt update # updates the files in the Ubuntu repository
sudo apt upgrade # upgrades all relevant packages installed on the server
sudo apt install nfs-kernel-server # installed the NFS tools and all dependencies
```

Confirm that the NFS service has been started and is running

```
sudo systemctl status nfs-server.service
```

The next step is to create a few directories that will be exported / shared with the servers in the cluster.

NOTE:

- These instructions assume that you have already configured the storage in the NFS server. These instructions will only cover installing the NFS tools and exporting the shared storage to servers in the cluster.

The number of directories you decide to share with the cluster is up to you. However, I prefer to create multiple directories to house the various types of data – see the list below:

- Scratch – this space is used as a temporary storage area for data processing
- Projects – is the space used to hold project related data. There is a project directory per project.
- Archives – similar to the projects directory but this space contains just the raw, original data.
- Datasets - these are reference datasets should they be needed for your analysis needs.
- Compute_home - this is the shared home directories and will be mounted over the local server's home directory
- /opt/exp_soft – this space is used to install all the software tools that will be used by users of the cluster.

The above directories can be generated in a single command.

```
sudo mkdir -p /highlevel-share-directory/{scratch,projects,compute_home,datasets,archives}
sudo mkdir /opt/exp_soft/
```

Once all the directories have been created, modify the ownership and permissions. This guide will not discuss the ownership and permissions in details. For details instructions on how to assign permissions and file / directory ownership, look at the first guide in this series "Linux - Getting Started" <*insert-hyper-link-here*>

With the permissions and ownership in place, we need to configure the NFS server and tell it which clients will be accessing the various shares. Only machines listed below will be able to access the shared storage. Client machines able to access the shared storage is listed in the /etc/exports file. Let us edit this file and add our cluster servers to the list of clients.

```
sudo vim /etc/exports
```

Add the machines to the bottom of the file

```
# SLURM cluster

<shared/directory>        <workernode-ip>(rw,fsid=0,nohide,insecure,no_subtree_check)
```

NOTE:

- Add additional rows as needed. You should have one row per server connecting to he shared storage.
- If you are only sharing the high-level directory, the above should suffice. If however, you are sharing each directory individually, then there needs to be a line for each shared directory for each server in the cluster.
- All the rows will be the same except for the IP and shared directory which would change to the IP of each server connecting to the shared storage and the specific shared directory being shared.

Once done, save the file and refresh and export the list of directories

```
sudo exportfs -rav
```

r – reexport all directories
a – export all directories
v -verbose

The NFS server is now setup and exporting shared directories. Next, we need to log into the cluster machines and make mount points to mount the above shared directories. We will use the same directory hiarchy as we used in the NFS server installation. On all the servers that will mount the shared storage, create the below list of directories.

```
sudo mkdir -p /highlevel-share-directory/{scratch,projects,compute_home,datasets,archives}
sudo mkdir /opt/exp_soft/
```

NOTE:

- To make things easier, you can also put this into a script to run across all the servers in the cluster to create the directory structure.
- Alternatively, you can use Ansible to update all servers with the above configuration and edits.

Now add the mount points to the login server, controller server and each worker node in the cluster by editing the /etc/fstab file on each of the above servers. By adding the mount points to the /etc/fstab file, all shared directories will be mounted on system reboot.

On each server in the cluster,

```
sudo vim /etc/fstab
```

Add the below lines to the bottom of the file

```
# SLURM cluster mount points

<IP-of-NFS-server>:<spath-to-shared-scratch>      <local-mount-point>    nfs    defaults,async   1 1
<IP-of-NFS-server>:<spath-to-shared-projects>     <local-mount-point>    nfs    defaults,async   1 1
<IP-of-NFS-server>:<spath-to-shared-datasets>     <local-mount-point>   nfs    defaults,async   1 1
<IP-of-NFS-server>:<spath-to-shared-archives>      <local-mount-point>  nfs    defaults,async   1 1
<IP-of-NFS-server>:<spath-to-shared-compute_home>  <local-mount-point>   nfs  defaults,async   1
1
<IP-of-NFS-server>:<s/opt/exp_soft>  <local-mount-point>   nfs  defaults,async   1 1
```

With the mount points added in fstab, mount the NFS shares in the current sessionwith the below command.

```
sudo mount -a
```

NOTE:
If you get an error similar to the below, install the nfs-common packages

mount: /global5/scratch: bad option; for several file systems (e.g. nfs, cifs) you might need a /sbin/mount.<type> helper program.

```
sudo apt install nfs-common
```

You should now be able to mount the directories listed in fstab

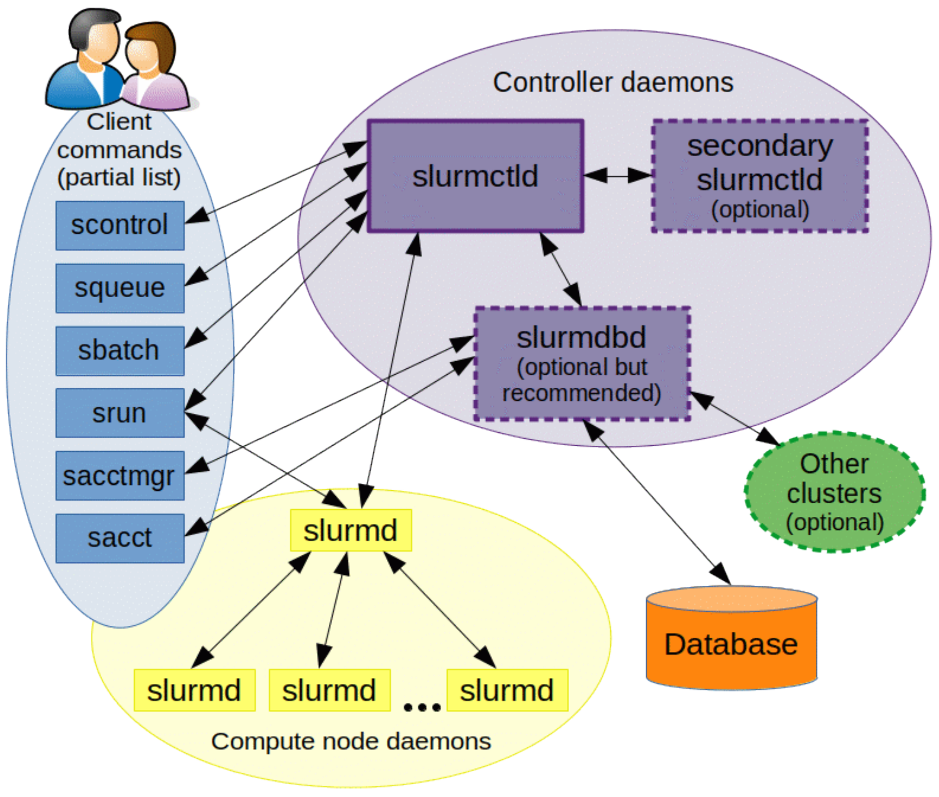
RECAP

Once all the cluster nodes are able to talk to / PING each other via the network, the compute nodes are able to access the internet via the controller node, all the shared NFS directories have been mounted on all machines in the cluster – you can move onto installing and configuring SLURM on the cluster.

5. SLURM – Installation and Configuration

SLURM (**S**imple **L**inux **U**tility for **R**esource **M**anagement) is an open-source workload manager often used on High Performance Clusters (HPC).

The below image highlights the various components of a SLURM setup


Client commands (partial list)
scontrol
squeue
sbatch
srun
sacctmgr
sacct
Controller daemons
slurmctld
secondary slurmctld (optional)
slurmdbd (optional but recommended)
Other clusters (optional)
Database
slurmd
slurmd
slurmd
...
slurmd
Compute node daemons

Source: https://slurm.schedmd.com/quickstart.html

The basic component of a SLURM cluster is a central controller that manages job submission. A database to contain information and worker nodes. The controller and the database can reside on the same server or on separate servers. Munge is used for worker node authentication.

The controller runs the slurmctld process (SLURM) and slurmdbd process (Database). For this setup, we will be using the MariaDB database server.

SLURM and munge is installed on all machines in the cluster. On the controller, we will start and run the slrumctld.service and slurmdbd.service services.

The worker-nodes will run the slurmd.service service

all machines will run the munge service.

5.1. SLURM controller setup

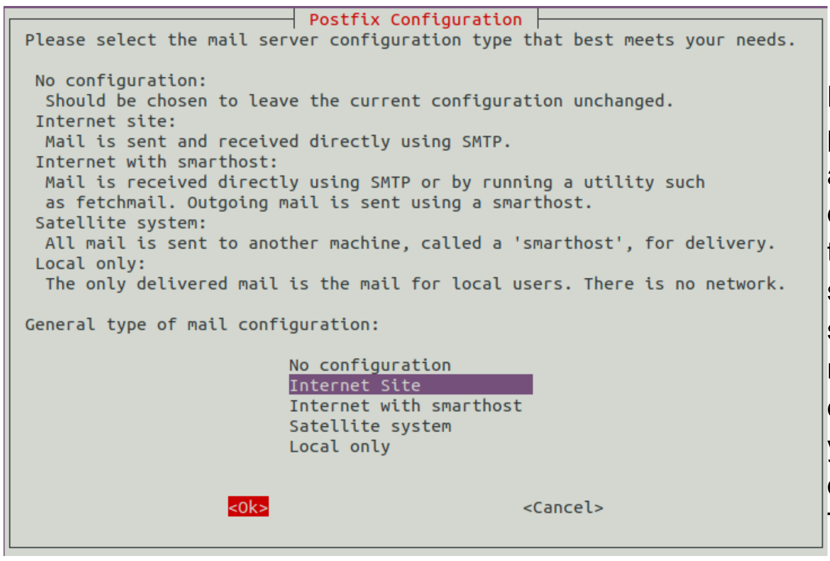
First, install all the required SLURM tools onto the controller

sudo apt install -y mailutils slurm-wlm slurm-wlm-doc sview slurmdbd mariadb-server

- mailutils # send email from the cli
- slurm-wlm # main slurm package
- sview # GUI to view and modify SLURM state
- slurm-wlm-doc # SLURM documentation
- mariadb-server # Mariadb database server (based on MySQL)
- slurmdbd # Secure enterprise-wide interface to a database for SLURM. This service does not have to be installed on the same server as the SLURM controller. In fact, best practices suggests that this service is installed on a separate machine.

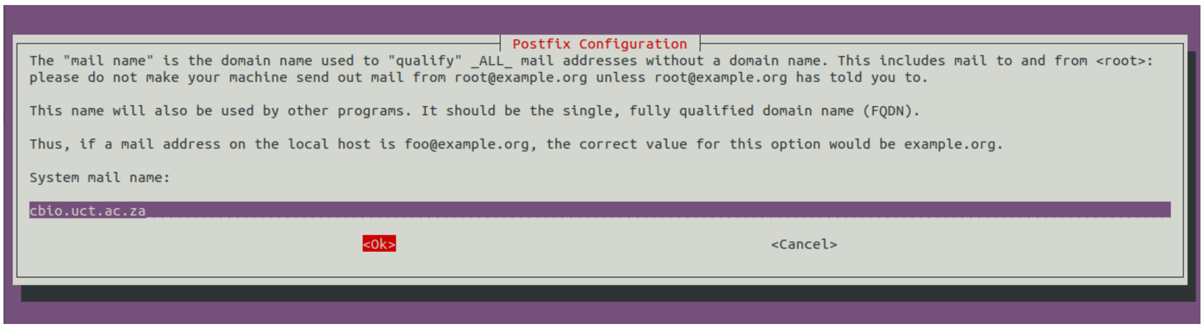
In the above command we are installing "*mailutils*" which will allow you to send email from the cli. This program will install postfix as a dependency. You will be prompted to configure postfix during the installation.

Postfix initiates first. You will be prompted to select the installation type. I choose "*internet site*" as this server will send email directly via smtp. We will configure postfix to be more secure once the server is setup.



```
Postfix Configuration
Please select the mail server configuration type that best meets your needs.

 No configuration:
  Should be chosen to leave the current configuration unchanged.
 Internet site:
  Mail is sent and received directly using SMTP.
 Internet with smarthost:
  Mail is received directly using SMTP or by running a utility such
  as fetchmail. Outgoing mail is sent using a smarthost.
 Satellite system:
  All mail is sent to another machine, called a 'smarthost', for delivery.
 Local only:
  The only delivered mail is the mail for local users. There is no network.

General type of mail configuration:

                          No configuration
                          Internet Site
                          Internet with smarthost
                          Satellite system
                          Local only

                    <Ok>                         <Cancel>
```

Next you are prompted for an email domain. In the below screen set the system mail name to the domain of your organization. This can be changed post installation in the "*/etc/postfix/main.cf*" file if need be.



```
Postfix Configuration
The "mail name" is the domain name used to "qualify" _ALL_ mail addresses without a domain name. This includes mail to and from <root>:
please do not make your machine send out mail from root@example.org unless root@example.org has told you to.

This name will also be used by other programs. It should be the single, fully qualified domain name (FQDN).

Thus, if a mail address on the local host is foo@example.org, the correct value for this option would be example.org.

System mail name:

cbio.uct.ac.za

                    <Ok>                                          <Cancel>
```

The remainder of the postfix installation should not prompt you for any other information. The other applications in the command should also complete without any further prompting.

5.1.1. Database installation and configuration

SLURM best practices suggest installing the SLUM database daemon on a separate server with high-speed disks. This is to reduce performance related issues when running multiple large jobs. If this installation does not expect multiple large jobs to be run, it is fine to install the database daemon on the same server as the SLURM controller.

For the purposes of this guide, I've installed the database daemon on the same host as the SLURM controller.

NOTE:

- when installing the database daemon on a separate machine, be sure to allow the MySQL or MariaDB port "3306" through the firewall for connectivity.

if installing the SLURM database daemon on a separate node, the below database setup will need to be done on the database server and not on the SLURM controller node.

We will first setup the database. This setup will make use of two databases. One database will contain the SLURM accounting information while the second database will contain the SLURM job information.

In the initial command we installed the MariaDB database server "*mariadb-server*" to be used as the SLURM database backend.

In this installation, the SLURM controller and db are on the same machines so once we have secured the default fresh installation of MariaDB, we bind the MariaDB address to the local host and create the SLURM databases

NOTE:

- if this is a fresh installation of MariaDB then you need to access the MySQL cli with the "*sudo mariadb*" login command. This will allow you to login without a password as the later versions of MariaDB use the unix.socket authentication plug-in. This plugin makes use of a local Linux account authentication.
- The account used in the above login command needs to have sudo rights.

Seeing as the SLURM database will run on the same vm as the SLURM controller, we edit the mariadb configuration file and set it to listen at localhost for connections. If the database was installed on a separate vm, we would then set the controllers IP here.

```
sudo vim /etc/mysql/mariadb.conf.d/50-server.cnf
```

look for "*bind-address*", comment out 127.0.0.1 and replace it with "localhost". Close and save the file.

If this is a fresh installation of MariaDB, then our first task is to secure the database instance. We do this by running the "*mysql_secure_installation*" script. Part of securing the database, we set a root password + remove anonymous users + test database + disallow remote root login.

```
sudo mysql_secure_installation  # set a root password and accept the defaults.
```

Ok, now we can login using the root user and begin creating the SLURM databases. SLURM requires two databases: accounting and job database. Create the two databases and a user that will own both databases.

NOTE:

- If you are not familiar with the MariaDB cli, note that all commands end with a semicolon. Failure to add the semicolon will result in a command run erro.
- While not strictly necessary, to separate MySQL commands from our input, we normally write the MySQL commands in uppercase. Should you write the full command in lowercase characters, the command will still be valid if the context is correct.

```
sudo mariadb    # type in your sudo password when prompted.
```

Once at the mysql cli, create the databases and user. In the below example, I have named the databases "slurm_acc_db" and slurm_job_db". You may change this to any name of your choosing.

```
CREATE DATABASE slurm_acc_db;  # creates the slurm_acc_db
CREATE DATABASE slurm_job_db;  # creates the slurm_job_db
```

now create a user and grant the user ownership of both databases. In the below example, I created a database user with the name "slurm_usr"

```
CREATE USER 'slurm_usr'@localhost IDENTIFIED BY '<secret-password>';
```

NOTE:

- The password should not include any special characters as this creates issues when logging into the db via the cli.

Grant full privileges to the "slurm_usr" for both SLURM databases.

```
GRANT ALL PRIVILEGES ON slurm_acc_db.* TO ‘slurm_usr’@localhost;
GRANT ALL PRIVILEGES ON slurm_job_db.* TO ‘slurm_usr’@localhost;
```

Now flush the privileges to save the above changes and bring them into effect in the current instance.

```
FLUSH PRIVILEGES;
```

Good. Now exit out of the mariadb cli and login as the user you just created to make sure there are no login issues. Once done, exit the MariaDB cli using the "exit" command.

```
EXIT;
```

The database side of things are done. Before we move no, lets just make sure that MariaDB has been started and is running.

to see if the MariaDB instance is running:

```
sudo systemctl status mariadb
```

look for the following:

first line, look for the word "***enabled***". This tells you that the service is set to start on system boot.

Second, line – look for "***active (running)***". Shows that MariaDB is currently running

In the process output, look for

ubuntu-mariadb /etc/mysql/debian-start[22596]: OK

If MariaDB is not running, you can start it with

```
sudo systemctl start mariadb
```

If there have been no errors to this point, we can move onto configuring munge.

5.1.2. Munge setup

With MariaDB installed and required DB created, we can move onto configuring munge which SLURM uses for authentication between the slurmd and slurmctl services.

Munge is automatically installed as part of the slurm-wlm installation so no need to install munge as a separate process.

You need to generate a munge key which is copied to all nodes in the cluster. This key is used to authenticate nodes in the SLURM cluster.

Make sure munge is installed

```
which munge  # shows where the munge executable is located
```

or

```
whereis munge  # shows where all the instances of munge can be found
```

or

> *sudo systemctl status munge* # looks at the status of the munge service

Next edit the munge default settings to set the default location for the munge key location

> *sudo vim /etc/default/munge* # add the below line to this file
>
> *OPTIONS="--syslog --key-file /etc/munge/munge.key"* # munge key will reside in /etc/munge/

Next, make a backup the old key and generate a new munge key for the cluster setup that will be used on all the nodes.

> *sudo cp /etc/munge/munge.key /etc/munge/munge.key.original* # backups the key to "munge.key.original"
>
> *sudo /usr/sbin/create-munge-key* # generates a new munge key and saves it in the /etc/munge/ directory

```
The munge key /etc/munge/munge.key already exists
Do you want to overwrite it? (y/N) y
Generating a pseudo-random key using /dev/urandom completed.
```

5.1.3. slurmdbd setup

Now we setup the SLURM accounting database configuration file to connect to the slurm_acc_db we created. If the slurmdb.conf file does not exist, manually create the /etc/slurm-llnl/slurmdbd.conf file and add the below information:

```
# Authentication info
AuthType=auth/munge
AuthInfo=/var/run/munge/munge.socket.2

# SlrumDBD info
DbdAddr=<IP-of-DB server>
DbdHost=localhost
DebugLevel=4
LogFile=/var/log/slurm-llnl/slurmdbd.log
PidFile=/var/run/slurm-llnl/slurmdbd.pid
SlurmUser=slurm

# Accounting database info
StorageType=accounting_storage/mysql
StorageHost=localhost
StoragePort=3306
StorageUser=slurm_usr # this is the DB user which owns the database
StoragePass=<slurm_usr-database-password>
StorageLoc=slurm_acc_db
```

SLURM expects reasonable sizes to be defined for the database innodb buffer_pool_size and lock_wait_timeout setting. To determine the current size allocated to these settings, log into the MariaDB instance and run the below commands:

There are three ways of determining what the above variables are set to.

METHOD 1: You could run the slurmdbd daemon in debug and verbose mode and review the output

```
sudo -u slurm slurmdbd -Dvvv
```

METHOD 2: You could view these variables from within the MariaDB database server

```
sudo mariadb
```

```
SHOW VARIABLES LIKE 'innodb_buffer_pool_size';
 SHOW VARIABLES LIKE 'innodb_log_file_size';
        SHOW VARIABLES LIKE 'innodb_lock_wait_timeout';
```

METHOD 3: View the limits set in the my.cnf file

```
sudo vim "/etc/mysql/my.cnf
```

If the limits set are not favourable to SLURM, you can edit them in the my.cnf file. If these variables are not listed in the my.cnf file, you can add them. The suggested defaults are listed below:

```
[mysqld]
innodb_buffer_pool_size=1024M
 innodb_log_file_size=64M
   innodb_lock_wait_timeout=900
```

To configure these variables, first stop the MariaDB server

```
sudo systemctl stop mariadb
```

Edit the my.cf file and increase the innodb_buffer_pool_size and innodb_lock_wait_timeout settings. If these settings are not listed, add them.

```
sudo vim /etc/mysql/my.cnf
```

edit or add the below lines:

```
 [mysqld]
 innodb_buffer_pool_size=1024M
innodb_log_file_size=64M
innodb_lock_wait_timeout=900
```

TIP:

The innodb_buffer_pool_size can be configured to be up to ~50% to 80% of the server's RAM.

Now start the mariadb server's

```
sudo systemctl start mariadb
```

to view the status of the SLURM database:

```
mysqlshow -u slurm_usr -p --status slurm_acc_db
```

the above command uses the slurm user to access the slurm_acc_db database. When prompted for a password, use the slurm_usr's password.

To list the contents of the databases

```
mysqlshow -u slurm_usr -p slurm_acc_db
```

To list the tables of the database, login as the root or database owner and run the show tables command

```
mariadb -u slurm_usr -p

USE slurm_acc_db

SHOW tables;
```

To see the headers of a specific table

```
DESCRIBE user_table;
```

To view a specific record in the user's table

```
SELECT * FROM user_table WHERE name="a-name-of-a-row"
```

Once the checks have completed, we can start the slurmdbd service. If the service starts and runs without error, we can move onto setting up the slurmctld service.

```
sudo systemctl restart slurmdbd.service
sudo systemctl status slurmdbd.service
```

5.1.4. slurmctld setup

The slurmctld is the service that monitors and controls the SLURM resources like job queues, allocates resources to compute nodes.

There is a way to generate the slurm.conf file using a browser. I however found that it's easier to just create the /etc/slurm-lln/slurm.conf file and paste the below contents into the file. This way you don't have to install a cli browser on your server.

```
sudo vim /etc/slurm-ll/slurm.conf
```

```
# slurm.conf file generated by configurator easy.html.
# Put this file on all nodes of your cluster.
# See the slurm.conf man page for more information.
#

# General
```

```
ControlMachine=nucleus
#ControlAddr=ip-of-controller>
AuthType=auth/munge
CacheGroups=0
CryptoType=crypto/munge
JobCheckpointDir=/var/lib/slurm-llnl/checkpoint
KillOnBadExit=01
MpiDefault=pmi2
MailProg=/usr/bin/mail
PrivateData=usage,users,accounts
ProctrackType=proctrack/cgroup
PrologFlags=Alloc,Contain
PropagateResourceLimits=NONE
RebootProgram=/sbin/reboot
ReturnToService=1
SlurmctldPidFile=/var/run/slurm-llnl/slurmctld.pid
SlurmctldPort=6817
SlurmdPidFile=/var/run/slurm-llnl/slurmd.pid
SlurmdPort=6818
SlurmdSpoolDir=/var/lib/slurm-llnl/slurmd
SlurmUser=slurm
StateSaveLocation=/var/lib/slurm-llnl/slurmctld
SwitchType=switch/none
TaskPlugin=task/cgroup
#MpiParams=ports=#-#
#SlurmUser=slurm_usr
#StateSaveLocation=/var/spool/slurm-llnl
#TaskPlugin=task/none

# Timers
InactiveLimit=0
KillWait=30
MinJobAge=300
SlurmctldTimeout=120
SlurmdTimeout=300
Waittime=0

# SCHEDULING
FastSchedule=1
SchedulerType=sched/backfill
SchedulerPort=7321
SelectType=select/cons_res
SelectTypeParameters=CR_CPU_Memory
#SelectType=select/linear
#SelectTypeParameters=

# Preemptions
PreemptType=preempt/partition_prio
PreemptMode=REQUEUE
```

```
# LOGGING AND ACCOUNTING
AccountingStorageType=accounting_storage/slurmdbd
AccountingStoreJobComment=YES
ClusterName=cbio
JobAcctGatherFrequency=30
JobAcctGatherType=jobacct_gather/linux
SlurmctldDebug=3
SlurmctldLogFile=/var/log/slurm-llnl/slurmctld.log
SlurmdDebug=3
SlurmdLogFile=/var/log/slurm-llnl/slurmd.log
SlurmSchedLogFile=/var/log/slurm-llnl/slurmschd.log
SlurmSchedLogLevel=3

# COMPUTE NODES
NodeName=slurm-wrk-110 Procs=2 Sockets=1 CoresPerSocket=2 ThreadsPerCore=1
        RealMemory=14000 Weight=4
NodeName=slurm-wrk-111 Procs=2 Sockets=1 CoresPerSocket=2 ThreadsPerCore=1
        RealMemory=14000 Weight=3

PartitionName=base Nodes=slurm-wrk-110,slurm-wrk-111 Default=YES MaxTime=72:00:00 State=UP
PartitionName=long Nodes=slurm-wrk-110,slurm-wrk-111 Default=No MaxTime=UNLIMITED
        Priority=1 State=UP AllowGroups=long
#PartitionName=debug Nodes=slurm-wrk-110,slurm-wrk-111 Default=No MaxTime=INFINITE
        State=UP
```

NOTE:

- The # COMPUTE NODES section is where you will list all your worker nodes and the associated system resources per server and which queues, they belong to.
- When referreing to the worker nodes, I use the user-friendly host names instead of the IP addresses
- If you do not yes know what system resources your worker nodes have, you can add them later by edting this file.

Once done, save and close the file then start the slurmctld service

```
sudo systemct restart slurmctld.service
sudo systemctl status slurmctld.service
```

NOTE:

- if there are any errors when starting the slurmdbd.service or the slurmctld.service. You can troubleshoot by looking at the "sudo systemctl status slurmdbd.service" and log files in "/var/log/slurm-llnl/".

If the services start without error. You are done setting up the SLURM controller. You can move onto setting up the worker nodes.

5.2. SLURM worker node setup

Slurm worker nodes are the servers that will execute the actual job once initiated from the SLURM controller.

All worker nodes need to run the “slurmd” and “munge” services. Once the OS is setup and configured, the following packages are to be installed on each worker node:

- slurm-wlm
- slurm-wlm-basic-plugins
- munge

NOTE:

- “slurm-wlm” installs “munge” + “slurm-wlm-basic-plugins” as part of the overall installation so you only really need to install the “slurm-wlm” application

  sudo apt update # update Ubuntu repositories
  sudo apt upgrade # upgrade outdated applications and dependencies
  sudo apt install slurm-wlm # installs SLURM

Once slurm-wlm is installed, all we need to do is copy some files from the controller to each worker node.

- slurm.conf
- Munge.key, and
- create the following file /etc/slurm-llnl/cgroup.conf

slurm.conf
Copy the slurm.conf file from the controller node to each worker node

sudo scp /etc/slurm-llnl/slurm.conf <workernode>:path-to-remote-directory

on the worker node, move the slurm.conf file from the remote directory to the local /etc/slurm-llnl/ directory

sudo mv slurm.conf /etc/slurm-llnl/

munge.key
This is not strictly necessary, but I prefer to backup old files before replacing before editing them. On the worker node, backup the default munge key.

sudo cp /etc/munge/munge.key /etc/munge/mung.key.original

Now copy the munge key from the controller to each worker node

sudo scp /etc/munge/munge.key <user>@<ip-of-worker-node:/etc/munge/munge.key

Good, now restart munge and test that it’s operational

```
sudo systemctl restart munge   # start the munge service
sudo systemctl status munge    # checks the status of the munge service
```

NOTE:

- If you get an error similar to 01 and 02 below, make sure that the munge.key file is owned by the munge user.

01 >> Job for munge.service failed because the control process exited with error code.

See "systemctl status munge.service" and "journalctl -xe" for details.

02 after running journalctl -xe >> munged: Error: Keyfile is insecure: "/etc/munge/munge.key" should be owned by UID 112

```
sudo chown munge:munge /etc/munge/munge.key
```

The service should start without error now. Test that munge is working. From the cli, type:

```
munge -n | unmunge | grep STATUS
```

you should see the following output:

```
STATUS:  Success (0)
```

Next create the cgroup.conf file in /etc/slurm-llnl/ directory and add the below content to the file.

```
sudo vim /etc/slurm-llnl/cgroup.conf
```

add the below

```
######################################################
#
# Slurm cgroup support configuration file
#
# See man slurm.conf and man cgroup.conf for further
# information on cgroup configuration parameters
######################################################

CgroupAutomount=yes
ConstrainCores=yes
```

You can now start the slurmd service

```
sudo systmectl restart slurmd.service
sudo systmectl status slurmd.service
```

NOTE:

- If you have not already done so, please add the SLURM cluster queue and worker node details to the slurm.conf file on the controller.

- If you modifiy the slurm.conf file on the controller, you need to recopy it to all worker nodes as this file needs to be sync'd across all machines in the cluster.

NOTE:

- If the slurmd.service fails to start. Make sure the "cgroup.conf" file has been created and the above text has been copied into it.

If the worker node and queue information has been added to the slurm.conf file, then you can run the "sinfo" command to see if all the nodes show up. If you get an output similar to the below, then your base cluster is operational. If not, review all the steps for setting up the controller and worker nodes.

```
sinfo
```

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
base*       up 3-00:00:00      2   idle slurm-wrk-[110-111]
```

6. Synchronized UID and GID's across cluster

For a cluster to be functional, beside the physical hardware and setting up the queue manager with worker-nodes. All user's, including the munge and default slurm user need to have their UID's and GID's syc'd across the cluster. Failure to synchronize all UID's and GID's, will result in permission issues later on and jobs will fail.

6.1. Manually sync UID and GID of users

It would be easier to add these instructions to a script or use a local identity manager, like LDAP to manage user account, create. This guide however does not go into detail about developing a script to generate user accounts acorss the cluster. For the purposes of this guide, I will go over the manual commands to change the UID and GID of user accounts and associated files / directories.

First, we will sync the munge and default slurm users UID's and GID's. Check what UID and GID's have been assigned to these users on the controller node.

```
id munge
```

```
uid=112(munge) gid=114(munge) groups=114(munge)
```

```
id slurm
```

```
uid=64030(slurm) gid=64030(slurm) groups=64030(slurm)
```

On the worker nodes, check the UID and GID of the slurm and munge users. If they differ from the controller node then we need to sync them to that of the controller's UID and GID. The same would apply to all user account generated.

Log into each worker node.

id munge # the munge UID and GID were different to the controller
id slurm # the slurm user's UID and GID matched the controllers

To change the UID and GID of the munge user, first stop the munge service

sudo systemctl stop munge

Change the UID and GID to match the controller

sudo usermod -u 112 munge # change the munge UID from 111 to 112
sudo groupmod -g 114 munge # change the munge GID from 113 to 114

Now find all the files and directories that are owned by the old GID and change them to the new GID. In the below command, we just change it to the munge group.

sudo find / -group 113 -exec chgrp -h munge '{}' \; # find all the files owned by GID 113 and change it to munge

run the same find and replace command for the UID

sudo find / -user 111 -exec chown -h munge '{}' \; # find all the files owned by UID 111 and change it to munge.

If you have reached this stage without error, you have successfully setup your basic SLUM Beowulf cluster. The next step is to test out your new toy by submitting a test job via SLURM.

7. Submitting a job on the SLURM cluster

There are a few ways of running jobs on the SLURM cluster.

7.1. Batch Job

A batch job or batch file will allow you to script a job to be run unattended for some time over the cluster. The below instruction will provide the basic instruction on how to write a batch script and how to execute it on the cluster.

Firs we create a SLURM script. Usually, we will end the file with a ".batch" extension

sudo vim first_run.batch

To run the slurm script, use the "sbatch" commands

sbatch first_run.batch

Basic layout of general slurm script:

```
#!/bin/bash
#SBATCH –job-name=first_run.batch # name of script
#SBATCH –output=/path-to-output-file_first_run-%j.out # output dir
#SBATCH –error=/path-to-error-file_first_run-%j.err # error out dir
#SBATCH –nodes=2  # number of nodes
#SBATCH –ntasks=8  # number of cores
```

```
#SBATCH –time=72:00:00  # wall time requested
#SBATCH –mem=1G  # memtbin:$PATH

WORKDIR=path-to-working-directory

cd $WORKDIR

./test.sh

srun hostname

srun sleep 60
```

7.2. Interactive job

When you run an interactive job, you are effectively logging onto a worker node to run your job locally. This is a two-step process. First you allocate the resources that you want. Once on the worker node, you may run your job locally

salloc --cpus-per-task=1 --mem=100MB # salloc request one core with 100MB or memory to be allocated to your interactive session

srun –pty /bin/bash # will drop you into the bash shell on the worker node

You may now run your commands. Once done, type “exit” to close the interactive job.

That is it, you have reached the end of this guide, below are a few useful SLURM commands to get you going and some thoughts on future work.

Happy computing....

8. Useful SLURM commands

Useful SLURM commands to manage resources

Command	Description
sinfo	Will print information about the queue’s and the state of your workernodes
sacct	Displays the accounting data for all jobs – past and present
squeue	Shows the status of the queues
sstat -j <jobid>	Shows the status of a running jobs tasks

smap	Is similar to the “squeue” command.
smap -c	Provides additional information
sview	Is a graphical interface to interact with your jobs. You need the X session for this to work
salloc	Allows you to allocated resources for an interactive job. Works with “srun”
srun	Will execute a interactive job and works with “salloc”
scancel [job_id]	Will cancel the specified job at job_id
sbatch	Will submit a batch script to SLURM
scontrol	Used to view and modify SLURM’s configuration and state. Accepetal optoins are - show job - show job [job_id] - show node [node-name] - show config - shutdown
sudo scontrol update NodeName=slurm-wrk-110 State=RESUME	Would change node “slurm-wrk-110” from a downed to an “idle” state.
scontrol show node slurm-wrk-112	Will print SLURM related information about the node “slurm-wrk112”
scontrol show config | grep SchedulerType	Will list the scheduler type

9. Proposed future work

The above guide walks the reader through developing a basic Beowulf HPC cluster using the SLURM resource manager. Once you have your cluster operational, it would be worth looking into the below topics to secure and simplify management of the cluster.

- Review security best practices, particularly on the controller and login node.
- Shared storage usage
- Local server health and resource monitoring
- Ansible or bash scripting to keep worker nodes in sync and to minimize deployment time

For more information on future work or assistance with any content in this guide, please communication with the H3ABioNet consortium via the H3ABioNet helpdesk https://helpdesk.h3abionet.org

OO-END-Oo