Slurm Installation Steps Ubuntu

> This guide will help you create and install a GPU HPC cluster with a job queue and user management. The idea is to have a GPU cluster which allows use of GPUs by many people.

Outline of steps:

```
1. Prepare hardware

2. Install OSs

3. Create slurm/munge users

4. Install Software (Nvidia drivers)

5. Install/configure file sharing (NFS here; if using more than one node/computer
in the cluster)

6. Install munge/SLURM and configure

7. User management
```

Preparing Hardware

For hardware , we have used a hardware setup of

```
1. 1 master node (2 core VM)

2. 2 worker nodes (2 core VM) with atleast 1 worker node (32 core machine)
containing 1 GPU(V100)
```

Installing operating systems

> In our setup we are using Ubuntu 20.04.3 LTS,

> Note: Along the way I used the package manager to update/upgade software many times (sudo apt-get update and sudo apt-get upgrade) followed by reboots. If something is not working, this can be a first step to try to debug it.

Create slurm/munge users

On all machines we need the munge authentication service and slurm installed. First, we want to have the munge and slurm users/groups with the same UIDs and GIDs. In my experience, these are the only GID and UIDs that need synchronization for the cluster to work. On all machines:

```
export MUNGEUSER=966

groupadd -g $MUNGEUSER munge

sudo useradd  -m -c "MUNGE Uid 'N' Gid Emporium" -d /var/lib/munge -u $MUNGEUSER -g munge  -s /sbin/nologin munge

export SLURMUSER=967

groupadd -g $SLURMUSER slurm

useradd  -m -c "SLURM workload manager" -d /var/lib/slurm -u $SLURMUSER -g slurm -s /bin/bash slurm
```

Installing software/drivers

Next you should install SSH. Open a terminal and install:

```
sudo apt install openssh-server -y.
```

Once you have SSH on the machines, you may want to use a parallel SSH utility to execute commands on all machines at once.

Install NVIDIA drivers

```
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt-get update
sudo apt-get install nvidia-driver-470
```

Install NFS (shared storage)

Master Node

On the master server, do:

```
sudo apt install nfs-kernel-server -y
```

Make a storage location:

```
sudo mkdir /storage
```

Change ownership to your administrative username and group:

```
sudo chown vikas:vikas /storage
```

Next we need to add rules for the shared location. This is done with:

```
sudo nano /etc/exports
```

Then adding the line:

```
/storage *(rw,sync,no_root_squash)
```

The * is for IP addresses or hostnames. In this case we allow anything, but you may want to limit it to your IPs/hostnames in the cluster. In fact, it wasn't working for me unless I explicitly set the IPs of the clients here. You have to have a separate entry for each IP. Mine ended up looking like:

/storage 192.168.100.xx(rw,sync,no_root_squash,all_squash,anonuid=999999,anongid=999999) 192.168.100.xx(rw,sync,no_root_squash,all_squash,anonuid=999999,anongid=999999)

where the 'xx's are actual numbers.

Then start the NFS service:

```
sudo systemctl start nfs-kernel-server.service
```

It should start automatically upon restarts.

You should also add a rule to allow for NFS traffic from the workers through port 2049. This is done like so:

```
sudo ufw allow from <ip_addr> to any port nfs
```

Client Nodes

Now we can set up the clients. On all worker servers:

```
sudo apt install nfs-common -y
sudo mkdir /storage
sudo chown vikas:vikas /storage
sudo mount smaster:/storage /storage
```

To make the drive mount upon restarts for the worker nodes, add this to fstab (sudo nano /etc/fstab):

```
smaster:/storage /storage nfs auto,timeo=14,intr 0 0
```

Preparing for SLURM installation

Passwordless SSH from master to all workers

First we need passwordless SSH between the master and compute nodes. We are still using master as the master node hostname and worker as the worker hostname. On the master:

```
ssh-keygen
ssh-copy-id vikas@sworker
```

Install munge on the master:

```
sudo apt-get install libmunge-dev libmunge2 munge -y
sudo systemctl enable munge
sudo systemctl start munge
```

Test munge if you like:

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

Copy the munge key to /storage

```
sudo cp /etc/munge/munge.key /storage/
sudo chown munge /storage/munge.key
sudo chmod 400 /storage/munge.key
```

Install munge on worker nodes:

```
sudo apt-get install libmunge-dev libmunge2 munge
sudo cp /storage/munge.key /etc/munge/munge.key
sudo systemctl enable munge
sudo systemctl start munge
```

If you want, you can test munge:

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

Prepare DB for SLURM

These instructions more or less follow this github repo: https://github.com/mknoxnv/ubuntu-slurm

First we want to clone the repo:

```
cd /storage
git clone https://github.com/mknoxnv/ubuntu-slurm.git
```

Install prereqs:

```
sudo apt-get install git gcc make ruby ruby-dev libpam0g-dev mariadb-server
libmariadbclient-dev libmariadb-dev  mariadb-server build-essential libssl-dev
python3 -y

sudo gem install fpm
```

Next we set up MariaDB for storing SLURM data:

```
sudo systemctl enable mysql
sudo systemctl start mysql
sudo mysql -u root
```

Within mysql:

```
create database slurm_acct_db;
create user 'slurm'@'localhost';
set password for 'slurm'@'localhost' = password('slurmdbpass');
grant usage on *.* to 'slurm'@'localhost';
grant all privileges on slurm_acct_db.* to 'slurm'@'localhost';
flush privileges;
exit
```

Copy the default db config file:

```
cp /storage/ubuntu-slurm/slurmdbd.conf /storage
```

Ideally you want to change the password to something different than slurmdbpass. This must also be set in the config file /storage/slurmdbd.conf.

Install SLURM

Download and install SLURM on Master

```
mkdir slurm_build

cd slurm_build

wget https://download.schedmd.com/slurm/slurm-20.11.8.tar.bz2

tar xvf slurm-20.11.8.tar.bz2

cd slurm-20.11.8

./configure --prefix=/tmp/slurm-build --sysconfdir=/etc/slurm --enable-pam --with-
pam_dir=/lib/x86_64-linux-gnu/security/ --without-shared-libslurm

make -j

make contrib

make install

cd ..
```

Install SLURM

```
fpm -s dir -t deb -v 1.0 -n slurm-20.11.8 --prefix=/usr -C /tmp/slurm-build .
dpkg -i slurm-20.11.8_1.0_amd64.deb
cp ./slurm-20.11.8_1.0_amd64.deb /storage/
```

Make all the directories we need:

```
mkdir /var/spool/slurm
chown slurm:slurm /var/spool/slurm
chmod 755 /var/spool/slurm
mkdir /var/spool/slurm/slurmctld
chown slurm:slurm /var/spool/slurm/slurmctld
chmod 755 /var/spool/slurm/slurmctld
mkdir /var/spool/slurm/cluster_state
chown slurm:slurm /var/spool/slurm/cluster_state
touch /var/log/slurmctld.log
chown slurm:slurm /var/log/slurmctld.log
touch /var/log/slurm_jobacct.log /var/log/slurm_jobcomp.log
chown slurm: /var/log/slurm_jobacct.log /var/log/slurm_jobcomp.log
mkdir -p /etc/slurm/prolog.d /etc/slurm/epilog.
```

copy necessary files

```
sudo cp /storage/ubuntu-slurm/slurmdbd.service /etc/systemd/system/

sudo cp /storage/ubuntu-slurm/slurmctld.service /etc/systemd/system/

The slurmdbd.conf file should be copied before starting the slurm services:

sudo cp /storage/slurmdbd.conf /etc/slurm/

mkdir /var/log/slurm/

touch /var/log/slurm/slurmdbd.log
```

Set permissions:

```
systemctl daemon-reload

chmod 600 /etc/slurm/slurmdbd.conf
chown slurm:slurm /etc/slurm/slurmdbd.conf

systemctl enable slurmdbd
systemctl start slurmdbd
systemctl status slurmdbd

chmod 600 /etc/slurm/slurm*.conf
chown slurm:slurm /etc/slurm/slurm*.conf

systemctl enable slurmctld
systemctl start slurmctld
systemctl status slurmctld
```

Start the slurm services:

```
sudo systemctl daemon-reload
sudo systemctl enable slurmdbd
sudo systemctl start slurmdbd
sudo systemctl enable slurmctld
sudo systemctl start slurmctld
```

If the master is also going to be a worker/compute node, you should do:

```
sudo cp /storage/ubuntu-slurm/slurmd.service /etc/systemd/system/
sudo systemctl enable slurmd
sudo systemctl start slurmd
```

Worker nodes

slurmd.service

```
sudo cp /storage/ubuntu-slurm/slurmd.service /etc/systemd/system/
sudo systemctl enable slurmd
sudo systemctl start slurmd
```

Now install SLURM on worker nodes:

```
cd /storage
sudo dpkg -i slurm-20.11.8_1.0_amd64.deb
sudo systemctl enable slurmd
sudo systemctl start slurmd
```

Configuring SLURM

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.

configure your systems specs into this file (slurm.conf)

visit : https://slurm.schedmd.com/configurator.html

```
cp "Path to  Downloaded conf. File"/slurm.conf /etc/slurm/slurm.conf
cp /storage/ubuntu-slurm/slurm.conf /storage/slurm.conf
```

Once SLURM is installed on all nodes, we can use the command

```
sudo slurmd -C
```

to print out the machine specs. Then we can copy this line into the config file and modify it slightly. To modify it, we need to add the number of GPUs we have in the system (and remove the last part which show UpTime). Here is an example of a config line:

```
NodeName=sworker CPUs=2 Boards=1 SocketsPerBoard=2 CoresPerSocket=1
ThreadsPerCore=1 RealMemory=3903
```

Take this line and put it at the bottom of slurm.conf.

Next, setup the gres.conf file. Lines in gres.conf should look like:

```
NodeName=jarvis Name=gpu File=/dev/nvidia0
NodeName=jarvis Name=gpu File=/dev/nvidia1
```

If you have multiple GPUs, keep adding lines for each node and increment the last number after nvidia.

Finally, we need to copy .conf files on all machines. This includes the slurm.conf file, gres.conf, cgroup.conf , and cgroup_allowed_devices_file.conf. Without these files it seems like things don't work.

```
sudo cp /storage/ubuntu-slurm/cgroup* /etc/slurm/
sudo cp /storage/slurm.conf /etc/slurm/
sudo cp /storage/gres.conf /etc/slurm/
```

This directory should also be created on workers:

```
sudo mkdir -p /var/spool/slurmd
sudo chown slurm /var/spool/slurmd
```

After the conf files have been copied to all workers and the master node, you may want to reboot the computers, or at least restart the slurm services:

Workers: `sudo systemctl restart slurmd Master`:

```
sudo systemctl restart slurmctld
sudo systemctl restart slurmd
```

Next we just create a cluster: `sudo sacctmgr add cluster compute-cluster`

check gpu node :

> sinfo -N -o "%N %G"

you will see an output like this :

```
NODELIST GRES
jarvis gpu:1
sworker (null)
```

Troubleshooting

When in doubt, first try updating software with sudo apt update;

```
sudo apt upgrade -y and rebooting (sudo reboot).
```

Log files

When in doubt, you can check the log files. The locations are set in the slurm.conf file, and are /var/log/slurmd.log and /var/log/slurmctld.log by default. Open them with sudo nano /var/log/slurmctld.log. To go to the bottom of the file, use ctrl+_ and ctrl+v. I also changed the log paths to /var/log/slurm/slurmd.log and so on, and changed the permissions of the folder to be owner by slurm: sudo chown slurm:slurm /var/log/slurm.

Checking SLURM states

Some helpful commands:

scontrol ping -- this checks if the controller node can be reached. If this isn't working (i.e. the command returns 'DOWN' and not 'UP'), you might need to allow connections to the slurmctrlport (in the slurm.conf file). This is set to 6817 in the config file. To allow connections with the firewall, execute on all nodes :

```
sudo ufw allow from any to any port 6817
```

and

```
sudo ufw reload
```

Node is stuck draining (drng from sinfo)

This has happened due to the memory size in slurm.conf being higher than actual memory size. Double check the memory from free -m or sudo slurmd -C and update slurm.conf on all machines in the cluster. Then run

```
sudo scontrol update NodeName=worker1 State=RESUME
```

Nodes are not visible upon restart

After restarting the master node, sometimes the workers aren't there. I've found I often have to do.

```
sudo scontrol update NodeName=worker1 State=RESUME
```

to get them working/available.

Taking a node offline

The best way to take a node offline for maintenance is to drain it:

```
sudo scontrol update NodeName=sworker State=DRAIN Reason='Maintenance'
```

Users can see the reason with sinfo -R

Changing IPs

If the IP addresses of your machines change, you will need to update these in the file /etc/hosts on all machines and /etc/exports on the master node. It's best to restart after making these changes.

NFS directory not showing up

Check the service is running on the master node: sudo systemctl status nfs-kernel-server.service

If it is not working, you may have a syntax error in your /etc/exports file. Rebooting after getting this working is a good idea. Not a bad idea to reboot the client computers as well.

Once you have the service running on the master node, then see if you can manually mount the drive on the clients:

```
sudo mount master:/storage /storage
```

If it is hanging here, try mounting on the master server:

```
sudo mkdir /test sudo mount master:/storage /test
```

If this works, you might have an issue with ports being blocked or other connection issues between the master and clients.

You should check your firewall status with sudo ufw status. You should see a rule allowing port 2049 access from your worker nodes. If you don't have it, be sure to add it with sudo ufw allow from <ip_addr> to any port nfs then sudo ufw reload. You should use the IP and not the hostname. A reference for this is here.

Node not able to connect to slurmctld

If a node isn't able to connnect to the controller (server/master), first check that time is properly synced. Try using the date command to see if the times are synced across the servers.

Running an interactive job with gpu:

```
root@smaster:~# srun -n 1 -t 2:00:00  --gres=gpu:1 --pty /bin/bash -l
root@jarvis:~# nvidia-smi
Wed Jan 19 10:09:54 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.86       Driver Version: 470.86       CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-PCIE...  Off  | 00000000:AF:00.0 Off |                    0 |
| N/A   45C    P0    25W / 250W |    309MiB / 16160MiB |      9%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+


+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1201      G   /usr/lib/xorg/Xorg                 84MiB |
|    0   N/A  N/A      9882    C+G   ...lib/vmware/bin/mksSandbox       18MiB |
|    0   N/A  N/A      9977    C+G   ...lib/vmware/bin/mksSandbox       18MiB |
|    0   N/A  N/A     10088    C+G   ...lib/vmware/bin/mksSandbox       18MiB |
|    0   N/A  N/A     10358    C+G   ...lib/vmware/bin/mksSandbox       18MiB |
|    0   N/A  N/A     10501    C+G   ...lib/vmware/bin/mksSandbox       20MiB |
|    0   N/A  N/A     10597    C+G   ...lib/vmware/bin/mksSandbox       18MiB |
|    0   N/A  N/A     10693    C+G   ...lib/vmware/bin/mksSandbox       18MiB |
|    0   N/A  N/A     10792    C+G   ...lib/vmware/bin/mksSandbox       16MiB |
|    0   N/A  N/A     10915    C+G   ...lib/vmware/bin/mksSandbox       18MiB |
|    0   N/A  N/A     11916    C+G   ...lib/vmware/bin/mksSandbox       16MiB |
|    0   N/A  N/A     12020    C+G   ...lib/vmware/bin/mksSandbox       16MiB |
|    0   N/A  N/A    106403    C+G   ...lib/vmware/bin/mksSandbox       16MiB |
+-----------------------------------------------------------------------------+
```

File missing accounting_storage_mysql.so

> accounting_storage_mysql.so is missing because you forgot to install the mariadb-devel deb before building Slurm debs. You must install the mariadb-devel deb and rebuild and reinstall Slurm debs as shown above.

User Management