Local Computing Cluster Resources

Qimin Yan Group

August 4, 2025

Abstract

This article gives an overview of the computational resources of our research group. Currently, it consists of several independent compute nodes, as well as one SLURM computational cluster. Basic usage is discussed along with the neccessary steps to set up new nodes and construct new clusters.

Contents

1 Available Computational Clusters

Currently, we have one operational cluster, which will be referred to as C1 (or simply 'the cluster') hereafter. C1 consists of the basic tree network shown schematically in fig. 1 with current IP addresses. The head node connects to both the internet (through ethernet port enp36s0f1) and the local cluster (port enp36s0f0).

The global IP address of the head node (enp36s0f1) is determined dynamically via DHCP by our network administrators and the local IP address (enp36s0f0) is set statically by us. All other IP addresses in the LAN are set statically. The local connection of the head node connects to the switch which acts as a central hub for all of the other network resources, namely the Network Attached Storage (NAS), and the compute nodes.

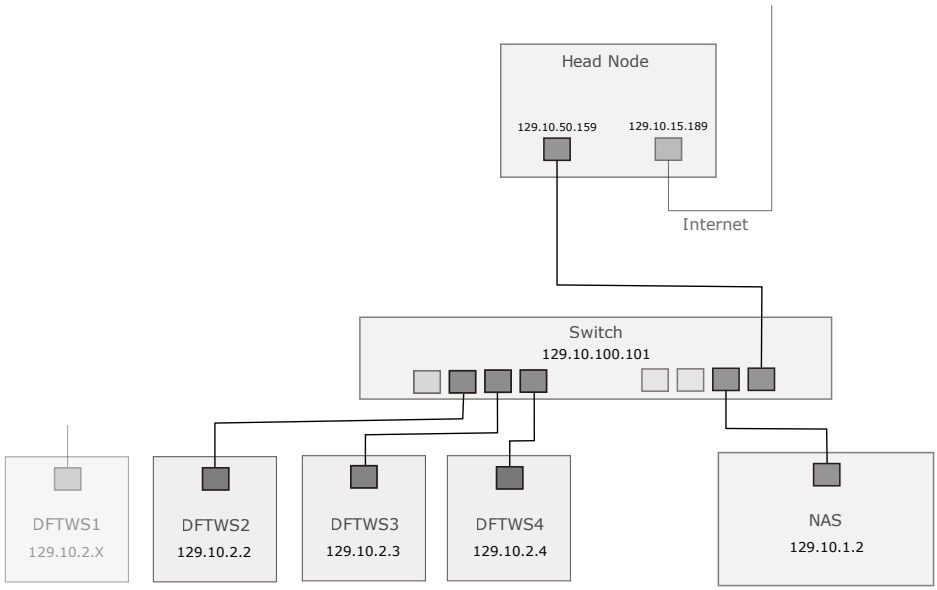

Figure 1: A schematic detailing the basic tree architecture of C1.

The switch is a QNAP QSW-IM1200-8C and may be accessed and configured by connecting locally via an ethernet port and using the associated QNAP QFinder-Pro software. One may also access the switch by navigating to it's IP address in an internet browser on any locally connected device with an IP in it's subnet. To access the switch remotely, one can use SSH tunneling to forward the HTTP traffic to your local browser (see: `https://serverfault.com/questions/581530/https-ssh-tunnel`). This may be accomplished with the command:

```
ssh -L 8443:<SwitchIP>:80 -Nf <user>@<HeadNodeIP>
```

where HTTP traffic (port 80) from the switch is forwarded to localhost:8443 (open in your local browser) through the head node.

The NAS is an ASUSTOR which may be accessed and configured by connecting locally via an ethernet port and using the associated ASUSTOR Control Center software. One may also access the NAS by navigating to it's IP address in an internet browser on any locally connected device on it's subnet. The NAS incorporates a redundant memory scheme to safely store data and is currently configured to the RAID 5 standard. Note that the NAS's OS is not Ubuntu but instead ASUSTOR Data Master (ADM), a proprietary OS. Many basic linux commands and packages are included and useable in it's terminal environment, however.

Compute nodes consist of PCs each with a single Intel i9 CPU and four Nvidia RTX 4090 GPUs. Each has Ubuntu installed as the OS.

MAC Addresses for Static IP Addresses

Static IP addresses may be requested from your ISP or local network administrator (in our case, the latter). Generally, the admin will require both the label of the ethernet port (in the wall) the device is connected to,

as well as the MAC address of the ethernet port on the device (if there are multiple, the port connected to the wall/internet). For convenience, when making these requests, the MAC address of the relevant devices is given here, namely for the head node. Currently, the head node is connected to the internet via port `enp36s0f1` with link/ether MAC address `a8:a1:59:c7:0a:34`.

Host Name	MAC Address	IP
MLWS1	c8:7f:54:c5:c9:63	129.10.14.227
MLWS2	08:bf:b8:03:56:c7	129.10.14.209
MLWS3	08:bf:b8:02:4d:e7	129.10.15.138
MLWS4	08:bf:b8:03:55:b7	129.10.14.210
HeadNode	a8:a1:59:c7:0a:34	129.10.15.189
DefectDB	b8:cb:29:f7:97:f5	129.10.14.
(old)WS1	58:11:22:9f:08:1f	129.10.14.

Note that the relevant MAC addresses may change if the ethernet chord is plugged into a different port, since the MAC address technically refers here to the connected ethernet port.

2 Constructing a New Cluster

The construction of a new SLURM computational cluster consists of three basic steps: *(1)* constructing a Local Access Network (LAN) and mounting the Network Attached Storage (NAS) across it; *(2)* homogenizing users and software across nodes; and, *(3)* installing SLURM and initializing it's daemons. Each of these basic steps are detailed in the following sections.

2.1 Local Access Networks

A local access network (LAN) is a set of devices that are locally connected (often physically via ethernet cables) such that each may communicate without access to the internet. For a computational cluster, physical connections (cables) are almost always preferred to reduce network latency (time for communication between devices).

Constructing a LAN is relatively easy: first all devices must be locally connected (of course), and then each must be assigned an IP address on the same subnet. The IP address range for a subnet may be determined by some given IP address (often a global IP address which allows for internet connection and is determined by some ISP or IT staff) and a subnet mask.

The first division of digits of an IP determine it's class. The class of an IP address determines it's subnet mask, and corresponding number of devices it may have on it's subnet. We are often given some IP address from an ISP or network administrator which we may then build a subnet on. At Burlington, our IP range is in 129.10.x.x, so the netmask for all subnets is 255.255.0.0.

2.1.1 Configuring Gateway

In the basic tree architecture used here, only the head node has direct connection to the internet. Thus, other devices in the LAN need to use the head node as a gateway to access the internet. This may be configured with iptables and route commands (from the `net-tools` package).

First pass this command on the local node to set the default gateway:

```
sudo route add default gw 129.10.50.159
```

Then, give the following on the gateway to allow forwarding and receiving on behalf of the subnet devices:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
sudo iptables -t nat -A POSTROUTING -o enp36s0f1 -j MASQUERADE
sudo iptables -A FORWARD -i enp36s0f0 -o enp36s0f1 -j ACCEPT
sudo iptables -A FORWARD -i enp36s0f1 -o enp36s0f0  -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Note that these commands require that the device (ethernet port) labeled enp36s0f0 is outward-facing (to the internet) and the device labeled enp36s0f1 is inward-facing. See `https://unix.stackexchange.com/questions/222054/how-can-i-use-linux-as-a-gateway`. Note that current networking routes may be checked with the `routel` command.

Commands given by `iptables` may not necessarily be persistent upon reboot (by default, no iptables commands will persist). To set persistent rules, first set the desired rules, then use `iptables-save` and `iptables-restore` with the following command:

```
sudo iptables-save | sudo tee /etc/iptables.conf
```

Then, to reload these rules automatically on restart, insert these lines into `/etc/rc.local`:

```
iptables-restore < /etc/iptables.conf
```

See `https://askubuntu.com/questions/66890/how-can-i-make-a-specific-set-of-iptables-rules-permanent`.

2.1.2 Setting DNS Server on Subnodes

A Domain Name System (DNS) is essentially a directory relating web addresses to IP addresses. For example, a DNS provides the map from the web address 'www.google.com' to the IP address '8.8.8.8'. The head node has an IP address provided by our network administrator via a DHCP protocol that also provides a DNS server.

Sub-nodes on the tree architecture used here may need to have their DNS servers updated or set manually, however. To set manually, see the file located at `/etc/resolv.conf` and edit the IP address listed after `nameserver`. For more detail, see `https://www.geeksforgeeks.org/how-to-configure-dns-in-linux/`. One may also use a command, such as:

```
echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf > /dev/null
```

To set a static DNS server on Ubuntu that doesn't get reset on restart, first install `resolveconf` with `apt`. Then, append a line containing the desired DNS server IP address after 'nameserver' in `/etc/resolv.conf` or `/etc/resolvconf/resolv.conf.d/head` and then run `sudo service resolvconf restart`. For more detail, see `https://datawookie.dev/blog/2018/10/dns-on-ubuntu/` or `https://anyware.hp.com/web-help/cas_manager/22.04/troubleshooting/dns_name_resolution/`. Note that 8.8.8.8 is Google's publicly available DNS server, and thus is a good choice due to it's reliability.

2.1.3 Installing Packages on Nodes Without Direct Internet Access

In the basic tree architecture of C1, only the head node has direct internet access. This allows for the head node to act as a global firewall for the network but does hinder some functions of the compute nodes. The following introduces a set of commands that allows for packages to be installed using apt on offline devices through online repositories accessible by the head node.

From the device with internet access, first SSH into the local device without internet access with:

```
ssh -R <selected port>:us.archive.ubuntu.com:80 user@nointernethost
```

We then need to create a file (if it doesn't already exist) '/etc/apt/apt.conf' and add the following two lines:

```
Acquire::http::Proxy "http://localhost:<selected port>";
Acquire::https::Proxy "https://localhost:<selected port>";
```

Running apt on the remote node without internet access should then allow us to download packages as long as the former ssh command is active. In consequent uses, the configuration file should be maintained, so that only the first ssh command is needed. This chain of commands is based on: `https://stackoverflow.com/questions/36353955/apt-get-install-via-tunnel-proxy-but-ssh-only-from-client-side`.

2.1.4 Logging Into Subnode Via SSH Jumping

For some inbound and outbound communications on subnodes without direct internet access, ip addresses must be jumped manually through the gateway. For example, this may be required to ssh into nodes off-LAN with IPs in the LAN subnet. For `ssh`, this is accomplished with the `-J` option which requires an additional argument specifying the gateway node as:

```
ssh -J <gateway-user>@<gateway-ip> <user>@<remote-ip>
```

Similarly, to copy files with `scp`:

```
scp -J <gateway-user>@<gateway-ip> <file-to-copy> <user>@<remote-ip>:<remote-destination>
```

See `https://superuser.com/questions/456438/how-do-i-scp-a-file-through-an-intermediate-server`. Note that one may also update their "~/.ssh/config" file to automatically jump for certain hosts with ProxyJump, as in `https://superuser.com/questions/174160/scp-over-a-proxy-with-one-command-from-local-machine`.

2.1.5 Mounting NAS

The Network Attached Storage (NAS) is a cluster-wide storage node. For the NAS to be available on all other nodes, it must be automatically mounted at startup across the cluster. One popular way to do this for a hard-wired LAN is with NFS (see `https://help.ubuntu.com/community/SettingUpNFSHowTo`; note that another method must be used for wireless/WiFi-based clusters, such as autoFS, see `https://askubuntu.com/questions/884389/auto-mount-nfs-via-autonfs`).

The NAS of C1 uses a proprietary linux-based OS that seems to require us to setup NFS access via an in-browser GUI. For these steps, see `https://www.asustor.com/en/online/College_topic?topic=109#lux3`. Note that this browser applet on port 8000 for HTTP (8001 for HTTPS) may be accessed remotely with the command:

```
ssh -L 9999:localhost:8000 -Nf <user>@<NAS-IP>
```

The client-side nodes of the NAS then must all have the `nfs-common` package installed (with `apt` for instance). The shared directory on the NAS server then may be mounted on the clients with the following command:

```
mount -t nfs <NAS-IP>:/<NAS-Shared-Folder> <Directory-to-Mount-to>
```

This may be done automatically by appending a line of the following form to the file `/etc/fstab`:

```
<NAS-IP>:/<NAS-Shared-Folder>   <Directory-to-Mount-to>    nfs    defaults    0 0 :
```

For more detail on how to update the `fstab` file for automatic mounting at startup, see the following: `https://askubuntu.com/questions/890981/how-to-configure-a-nfs-mounting-in-fstab`. To test a newly updated `fstab` file, the command `mount -a` may be used to load it; however, this is apparently bad practice, and the command `findmnt --verify` should be used first to verify the `fstab` file isn't broken (as specified in `https://serverfault.com/questions/518967/do-i-need-to-restart-my-server-after-editing-fstab-and-mtab`).

2.2 Prerequisites for SLURM Installation

A SLURM cluster requires a homogeneous set of users and specific software across all nodes in the cluster. Specifically, the user names and user ids (UIDs) of the OS as well as the group names and ids (GIDs) must be the same across all the nodes. Furthermore, SLURM requires munge (a linux software package instantiating a set of secure communication protocols) be installed on each device,.which also requires a user named 'munge' on each device which must have it's ids synchronized. Finally, SLURM requires that the devices on the LAN all have synchronized clocks, which may be accomplished with ntp (another linux software package). In the following, we give detailed instructions for each of these processes.

2.2.1 Changing UIDs and GIDs

The UIDs and GIDs can be found in the file '/etc/passwd' and changed with chmod commands. Users may be added with the useradd command. Note that the munge user id need not be synchronized between systems since it's a system user. For an article giving an overiew of the useradd command, see: `https://linuxize.com/post/how-to-create-users-in-linux-using-the-useradd-command/`. If user accounts already exist and ids must be changed, the relevant commands are discussed here.

Current UIDs and GIDs for Group Members

Currently, we used the following set of UIDs for current (and past) members of the group:

```
#!/bin/bash
sudo useradd -m -u 1001 weiyi
sudo useradd -m -u 1002 ajh
sudo useradd -m -u 1003 anoj
sudo useradd -m -u 1004 yuboqi
sudo useradd -m -u 1005 tsai
sudo useradd -m -u 1006 zhenyaof
sudo useradd -m -u 1008 angush
sudo useradd -m -u 1009 ruanyu
sudo useradd -m -u 1500 slurm
```

where we assume 'qimin' already exists as a sudo user with UID and GID 1000.

2.2.2 Installing Munge

Munge is a secure communication protocol that is use by SLURM to communicate between nodes in the cluster. Munge may be installed with apt with the following command:

```
sudo apt install munge  libmunge2 libmunge-dev
```

Note that we need to also install the lib and dev versions of munge, as in the command above, for SLURM to work properly.

After installation, it's daemon must be enabled to start automatically on startup with the following command on Ubuntu:

```
sudo systemctl enable munge.service
sudo systemctl start munge.service
```

Then, install and initiate munge daemons on each compute node. The munge keys should be uniform across the cluster, this is done by copying the key '/etc/munge/munge.key' from the head node to each compute node and restarting the munge daemon.

Testing Munge Installation

The munge status on a single node may be tested with the following command:

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

The munge status between nodes may be tested from one node to another with the following command:

```
munge -n | ssh <NODE_ADDRESS> unmunge
```

For the official installation guide, see: `https://github.com/dun/munge/wiki/Installation-Guide`.

2.2.3 Synchronizing Clocks

Synchronized clocks are particularly important for munge protocols, since they incorporate time stamps in credentials for finite length-of-time authentification. Time synchronization is often achieved simply by installing the ntp package on linux with apt (or some other package manager).

Generally speaking, if ntp is installed on each device, and munge doesn't return an error communicating between devices, the clocks should be sufficiently synchronized.

2.3 SLURM Installation

To install SLURM, we then need to download the desired version of SLURM and then decompress and build it. Note that while SLURM packages exist on many linux repositories, none are (currently) directly supported by SchedMD (the company that manages SLURM). After compiling SLURM, a config file must be made specifically for the cluster, and loaded by SLURM. After configuration the SLURM daemons must be initialized and then SLURM commands should be available for scheduling and submitting jobs. Each of these steps is covered in more detail in the following.

2.3.1 Download and Compiling

The most current stable version of SLURM may be downloaded as a compressed tar-ball at: `https://www.schedmd.com/download-slurm/`. This tar-ball `slurm.version.tar.bz2` may be decompressed with the following command:

```
tar -xf slurm.version.tar.bz2
```

Then, change directory to the uncompressed source and execute `./configure` with the appropriate options. An important argument here is `--prefix=<install_dir>`, which allows us to specify the install directory of the slurm program. To aid in bookeeping of slurm builds, install with a prefix (as recommended https://groups.google.com/g/slurm-users/c/9Qvs8q1Uc-A):

```
sudo bash ./configure --prefix=\opt\slurm-<version_number>-build-<build_number>
```

After running the configure command, run the `make` command in the same directory to compile the source and then execute `make install`. If `make install` returns an error of the sort "make: *** [Makefile:613: install-recursive] Error 1", try executing `sudo make install-recursive` instead.

2.3.2 Configuring SLURM

Manual configuration is particularly important for SLURM since every cluster is different and has a totality of hardware not local to any one node. The easiest way to create a new configuration file is to use the SchedMD endorsed html configuration tool which can be found at `https://slurm.schedmd.com/configurator.html`. Make sure the version number of the configurator matches the version of SLURM being installed.

The SLURM configurator returns a long list of text in the browser. This may be copied into a file 'slurm.conf' where SLURM will see it. On Ubuntu an appropriate location for this file is in '/etc/slurm' (which may need to be created if it doesn't already exist) if created with root access or '/usr/local/etc' otherwise. This same configuration file must be on each node of the SLURM cluster.

2.3.3 Giving SLURM Access to Files

By default, the SLURM user will be ineligible to read or write in directories it needs access to to perform logging, load configurations, etc. In this case we need to add these privileges to the 'slurm' user. On C1, the PID and log files are written in '/etc/slurm', with the following command being neccessary for the control node:

```
sudo chown -R slurm:slurm /etc/slurm
sudo chown -R slurm:slurm /var/spool/slurmctld
```

Otherwise, for compute nodes:

```
sudo chown -R slurm:slurm /etc/slurm
sudo chown -R slurm:slurm /var/spool/slurmd
```

To test run SLURM before starting the daemons, you can run it first on controller nodes with the command:

```
slurmctld -D -vvvvvv
```

and on compute nodes with the command:

```
slurmd -D -vvvvvv
```

where the -D option specifies we want outputs written to the terminal and the excessive number of v's indicates we want a very verbose output (useful for debugging). Note that slurmctld and slurmd may need to be run with sudo (at least) the first time to create certain files, etc. Note also that some directories may need to first be made manually with sudo and the mkdir command; such necessities are usually apparent from the errors returned when running the '* -D -v' commands above.

2.3.4 Starting SLURM Daemons

If the only errors returned from the test runs above are failure to communicate between nodes, we now need to start the daemons for the SLURM control and compute functions.

Before we can start the daemons, we need to be sure the system daemon, systemd can see their service files, which can be accomplished by copying the service unit files to systemd's path directory on both control and all compute nodes as:

```
sudo cp slurmd.service slurmctld.service slurmdbd.service /etc/systemd/system
sudo cp slurmd.service slurmctld.service slurmdbd.service /etc/systemd/user
```

On the control node, we then need to enable the service on startup and start the service with the commands:

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

On the compute node we perform, similarly

```
systemctl enable slurmd
systemctl start slurmd
```

The status of the slurm service then may be tested with the `sinfo` command.

2.4 Installing Cluster-Wide Environments & Software with Ansible

To avoid manually installing a list of the same software and environments on each device of the cluster, software provisioning tools may be used to install cluster-wide in one command from one device. Many options exist, here we focus on accomplishing this task with Ansible, an open source IT automation engine that can be use to automate provisioning as well as many other things. For a brief overview of some Ansible commands, see `https://spacelift.io/blog/ansible-cheat-sheet`.

Note that before using Ansible on remote devices, the current user must be able to SSH into the same remote devices without being prompted for a password. This may be accomplished by saving the SSH pair's public key, as shown in the following section (2.4.1). After this, an overview of the directory of current ansible playbooks for C1 is given; as well as a template for typical user downloads and conda environment tools.

2.4.1 Generating SSH Keys For Automatic (Password-less) Login

If they do not already exist on the head node, first, generate RSA key pairs for the head node's SSH client with the following command:

```
ssh-keygen -t rsa
```

Then, copy all necessary public keys into `~/.ssh/authorized-keys` for each subnode using `ssh-copy-id` with the following command (see here):

```
ssh-copy-id <user>@<subnode-identifier>
```

which will prompt you to log in as `<user>` on the specified subnode. After this, no password should be requested in following ssh interactions (as current user on local device to `<user>` on remote device, for each subnode this command was used on). This is a necessary prerequisite for every node Ansible is intended to run on.

2.4.2 Cluster Hosts

The hosts available to Ansible are defined (by default) in the file `/etc/ansible/hosts`, which requires sudo access to edit. Hosts may also be specified locally in a file `inventory.ini` in the folder of ansible playbooks (see here).

2.4.3 Cluster-wide Playbooks

Ansible runs *playbooks*, specified in YAML files, across sets of nodes.

Installing Packages Cluster-wide

First, and perhaps most commonly, we generally install packages cluster-wide with the following playbook:

```
---
- hosts: localhost
tasks:
- name: install packages
become: true
become_user: root
apt:
state: present
name:
- python
- conda
```

See `https://opensource.com/article/20/9/install-packages-ansible`.

Note that for tasks that require root privileges (such as the example above with `become_user: root`), the option `--ask-become-pass` must be passed as:

```
ansible-playbook --ask-become-pass ./playbook.yml
```

For generic shell commands across the cluster using ansible, see `https://kodekloud.com/blog/ansible-shell-module/`.

3 Basic Usage

This section details basis usage of the cluster, namely the submission of jobs to the scheduler through SLURM, and the execution of cluster-wide commands through Ansible.

3.1 Submitting Jobs (SLURM)

SLURM jobs may be run across the cluster by submitting `sbatch` commands.

3.2 Cluster-wide Commands & Installation (Ansible)

Ansible runs *playbooks* on sets of nodes specified in YAML files.

Installing Packages Cluster-wide

First, and perhaps most commonly, we generally install packages cluster-wide with the following playbook:

```
---
- hosts: localhost
tasks:
- name: install packages
become: true
become_user: root
apt:
state: present
name:
- python
- conda
```

See `https://opensource.com/article/20/9/install-packages-ansible`.

Note that for tasks that require root privileges (such as the example above with `become_user: root`), the option `--ask-become-pass` must be passed as:

```
ansible-playbook --ask-become-pass ./playbook.yml
```

For generic shell commands across the cluster using ansible, see `https://kodekloud.com/blog/ansible-shell-module/`.