Setting Up Cluster Guide

Prerequisites

Ubuntu server or other OS ISO downloaded
VirtualBox or VMWare installed

Install Operating System

- **Create the Master VM (Ubuntu Server Example)**:
  - Open VirtualBox and click **New**.
  - **Name**: `master-node`.
  - **Type**: Linux.
  - **Version**: Ubuntu (64-bit).
  - Allocate at least **1 GB of RAM** (preferably 2 GB) and **20 GB of disk space**.
  - Choose **Create a virtual hard disk** and select **VDI** as the format.
  - Complete the wizard to create the VM.ubuntu

- **Install Ubuntu Server on the Master Node**:
  - Insert the Ubuntu Server ISO by selecting the VM, clicking **Settings > Storage**, and adding the ISO under the **Controller: IDE** section.
  - Start the VM and boot from the ISO.
  - Follow the installation prompts to install **Ubuntu Server**. Set up a user account (e.g., `ubuntu`).
  - Choose **OpenSSH Server** during installation (you can install it later via CLI if you miss it).
  - Make sure you keep the same username for all nodes

Repeat The above process for the worker nodes.

Make sure when creating the usernames, hostnames and passwords that you can remember for example set all the passwords to “123” and set the usernames and hostnames the same and in a consistent manner, such as master, worker1, worker2, etc

If you forget to install OpenSSH, you can install after the operating system is installed here:

```
None
sudo apt install openssh-server openssh-client
```

Network Configuration

To ensure all the VMs can communicate with each other, configure a **host-only network** in VirtualBox.

1. **Create a NAT Network**:
    - Open VirtualBox and click on File ⇒ Preferences ⇒ Network
    - Click on “Add New NAT Network” icon
    - Right Click on newly added NAT Network and edit the settings.
    - Name your NAT Network to something meaningful. Eg “test-cluster”
    - Change the CIDR range per your needs. Eg use 192.168.10.0/24 range.
    - Make sure “supports DHCP” is checked
2. **Configure VM NAT Network Settings**:
    - Select created virtual machine and click on "Settings" button in the right pane. A dialog will open.
    - Click on "Network" option in the left pane.
    - Click on “Adapter 1” tab and ensure “Enable Network Adapter” is checked.
    - Change "Attached to" drop-down to "NAT Network". It should automatically populate Name field with the NAT Network you defined earlier. Eg test-cluster
    - Click Ok
3. **Configure VM Bridged Network Settings**:
    - Click on “Adapter 2” tab and ensure “Enable Network Adapter” is checked.
    - Change "Attached to" drop-down to "Bridged Adapter". It should automatically populate Name field Adapter Type, MAC Address etc. this enables internet access
    - Click Okay
4. **Check Network Configuration**:
    - Start all VMs.
    - Install net-tools using the following command `sudo apt install net-tools`
    - Now run the command `ifconfig` which will should give you a result like this shown on the next page:

```
master@master-node:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.4  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::a00:27ff:fec2:90e6  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:c2:90:e6  txqueuelen 1000  (Ethernet)
        RX packets 28  bytes 3750 (3.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 41  bytes 3943 (3.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 106  bytes 8170 (8.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 106  bytes 8170 (8.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

master@master-node:~$
```

The above shows VMs ip address in red, this is the ip that you use to ssh into the worker nodes, make sure each ip address is different.

5. **Ping Google To Check Internet Connection**:
    - Run the following command:

```
None
ping -c 4 8.8.8.8
```

You should see result like the following to confirm that you pinged google got a response and didn't lose any packets:

```
master@master-node:~$ ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=57 time=21.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=57 time=23.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=57 time=25.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=57 time=28.5 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 21.284/24.775/28.478/2.665 ms
master@master-node:~$ _
```

Check SSH Configuration

Verify that the SSH server on all nodes is configured to allow password authentication.

1. Open the SSH configuration file using on the master and worker nodes (might only have to be the worker nodes):

```
None
sudo nano /etc/ssh/sshd_config
```

2. Ensure the following lines are set (they will be commented out using “#” and you will have to uncomment the following lines):

```
None
PermitRootLogin yes
PasswordAuthentication yes
```

3. Save and exit the file (Ctrl + X, then Y, then Enter).
4. Restart the SSH service for the changes to take effect using:

```
None
sudo systemctl restart ssh
```

5. Check if SSH is active using the command:

```
None
sudo systemctl status ssh
```

The command output should state active in bright green, if it does not run the command:

```
None
sudo systemctl start ssh
```

Set Up SSH Key-Based Authentication

To manage the cluster easily, configure SSH key-based authentication between the master node and the worker nodes.

1. **Generate an SSH Key on the Master Node**:
    - On the master node, generate an SSH key:

```
None
ssh-keygen -t rsa -b 2048
```

After the command is run press enter to save the key to the default location (`/home/ubuntu/.ssh/id_rsa`)

2. **Copy the Public Key to the Worker Nodes:**
    - Copy the SSH public key from the master node to both worker nodes using the ssh-copy-id command:

```
None
ssh-copy-id ubuntu@worker-node1-ip
ssh-copy-id ubuntu@worker-node2-ip
```

Replace `worker-node1-ip` and `worker-node2-ip` with the actual IP addresses of your worker nodes, an example would be worker1@192.168.10.5

After entering the above command you will be prompted to enter the password of the node that you are trying to SSH into. This is to copy the public key of the master node to allow of its worker nodes.

3. **Test SSH Access:**
    - Test that you can SSH into both worker nodes without a password:

```
None
ssh ubuntu@worker-node1-ip
ssh ubuntu@worker-node2-ip
```

Example command would be: `ssh worker1@192.168.10.5`

You should also be able to use commands like: `ssh 192.168.10.5`

Now you have successfully used SSH to remotely connect to the worker nodes and can run various commands, such as:

```
None
whoami
```

Or

```
None
ip addr show
```

To confirm that you have remote access to the worker nodes from the master node.

To exit SSH, just type:

```
None
exit
```

This will mean you are now back running the terminal for the master node.

MPI Setup

1. **Install OpenMPI on All Nodes**:
    - On the master node and both worker nodes, install OpenMPI:

```
None
sudo apt install openmpi-bin openmpi-common libopenmpi-dev
```

2. **Verify Openmpi install**:
    - On the master node and both worker nodes, install OpenMPI:

```
None
mpirun --version
```

3. **Create a Hosts File on the Master Node**:
    - On the master node, create a `hosts` file to define the cluster nodes:

```
None
nano ~/mpi-hosts
```

    - Add the IP addresses of the master and worker nodes:

```
None
master-node-ip
worker-node1-ip
worker-node2-ip
```

    - Example:

```
None
192.168.10.8
192.168.10.9
192.168.10.10
```

4. **Test the Cluster with MPI**:
    - Create a simple MPI program to test the cluster. Create a file `hello_world.c`:
    - Using the code from https://mpitutorial.com/tutorials/mpi-hello-world/

```C/C++
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
	MPI_Init(NULL, NULL);

	int world_size;
	MPI_Comm_size(MPI_COMM_WORLD, &world_size);

	int world_rank;
	MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
	printf("Hello from processor %d out of %d processors\n", world_rank,
world_size);

	MPI_Finalize();
	return 0;
}
```

    - Compile the MPI program:

```None
mpicc hello_world.c -o hello_world
```

    - Run the program on one VM:

```None
./hello_world

mpirun --hostfile mpi-hosts -np x ./hello_world
```

NOTE: x represents the number of VMs that you have in the host file example 2, 4, 6, 8 etc

Setting MPI To Run Across Multiple VMs

1. **Check your current working directory:**
    - ./On your master node run the command to get current directory path:

```
None
pwd
```

    - This should return something like “/home/username”, where “username” is the username of the nodes (eg, ubuntu)
2. Copy `hello_world` to the other nodes:
    - You need to copy this file to the exact same location on the other nodes (e.g., `192.168.10.8` and `192.168.10.9`). Use the `scp` (secure copy) command to copy the file to those nodes, repeat this for all nodes:

```
None
scp /home/user/hello_world user@192.168.10.8:/home/user/
scp /home/user/hello_world user@192.168.10.9:/home/user/
```

3. Run the `mpirun` command:
    - Now that the `hello_world` executable is in the same location on all nodes, you can run your `mpirun` command on the master node using:

```
None
mpirun --hostfile mpi-hosts -np 3 ./hello_world
```

Setting Up HPL (High-Performance LAPACK)

You have two options to install OpenBLAS option a (prefered) or option b this will effect the HPL.dat config file later in this document. IMPORTANT: Choose only one and then other steps are the same.

Installing OpenBlas & Dependencies On All Nodes (Option A Prefered)

1. Install these dependencies

```
None
sudo apt install gfortran build-essential libopenblas-dev liblapack-dev
```

Installing OpenBlas & Dependencies On All Nodes (Option B)

1. Install these dependencies

```
None
sudo apt install build-essential hwloc libhwloc-dev libevent-dev gfortran
```

2. Git clone the OpenBLAS packages:

```
None
git clone https://github.com/xianyi/OpenBLAS.git
```

3. Move into the cloned directory:

```
None
cd OpenBLAS
```

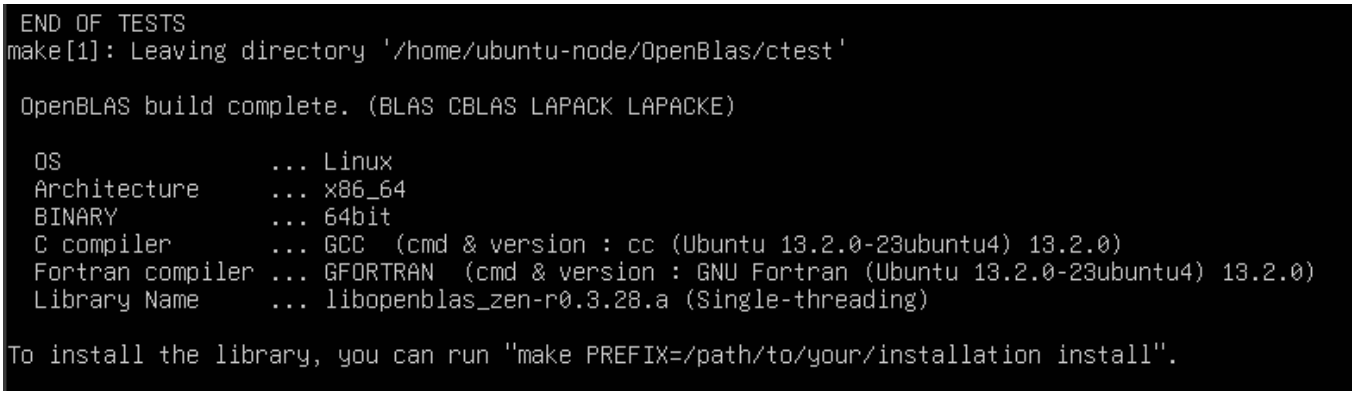
4. Checkout the version you want to make:

```
None
git checkout v0.3.28
```

5. Run the make the command (this will take a while):

```
None
make
```

This should result in the following output:



6. Move OpenBLAS into a folder for example and install (PREFIX means where you want it to be installed):

```
None
make PREFIX=$HOME/opt/OpenBLAS install
```

Compile HPL

1. Download the tar.gz folder using wget:

```
None
wget https://netlib.org/benchmark/hpl/hpl-2.3.tar.gz
```

2. Unzip using gunzip:

```
None
gunzip hpl-2.3.tar.gz
```

3. Extract using the following command:

```
None
tar xvf hpl-2.3.tar
```

4. Remove the tar.gz folder because it is no longer needed:

```
None
rm hpl-2.3.tar
```

5. Move the contents of hpl-2.3 into a hpl folder:

```
None
mv hpl-2.3 ~/hpl
```

Configure HPL

1. Move into hpl/setup directory:

```
None
cd hpl/setup
```

2. Run the make_generic script:

```
None
sh make_generic
```

3. Copy the Make.UNKNOWN file to Make.linux and move it into the hpl directory:

```
None
cp Make.UNKNOWN ../Make.linux
```

4. Move back one directory:

```
None
cd ../
```

5. Open the Make.linux file to configure its paths for OpenMPI and OpenBLAS:

```
None
nano Make.linux
```

Option A (Prefered) Setup

1. Make sure change the following sections to make them look like this before pressing CTRL + X, Y, then Enter to save and exit:

```
None
ARCH = linux
MPdir = /usr
MPinc = -I$(MPdir)/include
MPlib = $(MPdir)/lib/x86_64-linux-gnu/libmpi.so

LAdir = /usr
LAinc = -I$(LAdir)/include
LAlib = -L$(LAdir)/lib/x86_64-linux-gnu -lopenblas -lm
```

NOTE: this is the paths if you installed OpenMPi as described in the previous section. As well as the installation of OpenBLAS installed previously from **OPTION A**, if you have installed these packages differently these paths may be different.

Option B Setup

1. Make sure change the following sections to make them look like this before pressing CTRL + X, Y, then Enter to save and exit:

```
None
ARCH = linux
MPdir = /usr
MPinc = -I$(MPdir)/include
MPlib = $(MPdir)/lib/x86_64-linux-gnu/libmpi.so

LAdir = $(HOME)/opt/OpenBLAS
LAinc =
LAlib = $(LAdir)/lib/libopenblas.a
```

NOTE: this is the paths if you installed OpenMPi as described in the previous section. As well as the installation of OpenBLAS installed previously from **OPTION B,** if you have installed these packages differently these paths may be different.

6. Compile the Make.linux file

```
None
make arch=linux
```

This should result in a build success message being displayed in the terminal and an xhpl file created in hpl/bin/linux

7. Move to the hpl/bin/linux directory:

```
None
cd bin/linux
```

8. Edit the HPL.dat file:

```
None
nano HPL.dat
```

Instructions are on the next page.

9. Using this website HPL Tuner enter in your cluster specs and then put these results into the HPL.dat file, before pressing CTRL + X, Y, then Enter to save and exit:

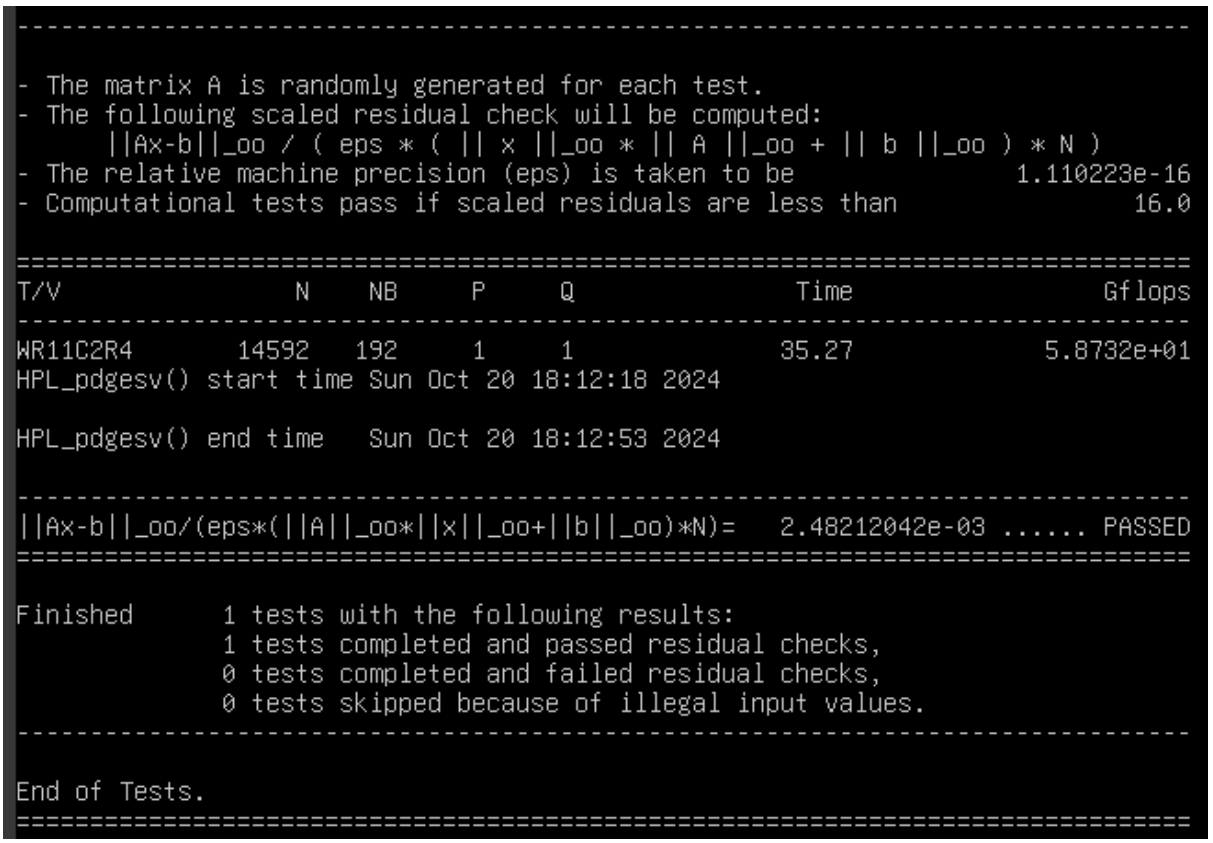
```
None
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
14592        Ns
1            # of NBs
192          NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
1            Ps
1            Qs
16.0         threshold
1            # of panel fact
2            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
4            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
1            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
1            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
1            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
##### This line (no. 32) is ignored (it serves as a separator). ######
0                               Number of additional problem sizes for
PTRANS
1200 10000 30000                values of N
0                               number of additional blocking sizes for
PTRANS
40 9 8 13 13 20 16 32 64        values of NB
```

Run HPL On A Single VM

1. Run the following command:

```
None
./xhpl
```

You should see an output similar to the below:



Run HPL On Multiple VMs

1. Go back to the home directory:

```
None
cd $HOME
```

2. Make sure the HPL executable and HPL.dat file are available on all nodes. You can copy them over using SCP:

```
None
scp -r hpl/ node1:/path/to/hpl/
```

Example:

```
None
scp -r hpl/ 192.168.10.8:/home/ubuntu-node/
scp -r hpl/ 192.168.10.9:/home/ubuntu-node/
```

3. Confirm it has been copied to the other nodes by running the command in the home scp directory:

```
None
ls
```

You should see a hpl directory in here as shown below:

Figure x: Confirmed HPL Folder On Slave1

4. On the master cd back to the /hpl/bin/linux, using the command:

```
None
cd /hpl/bin/linux
```

5. Run the XHPL Binary using MPIRUN:

```
None
mpirun -np 3 --hostfile /home/ubuntu-node/mpi_hosts ./xhpl
```

NOTE: we need to include the path to the hostfile for this to work correctly, since we are in the same folder as the XHPL binary or you can move the hostfile into the same directory.

6. After running the command you should see a result similar to this:

```
--------------------------------------------------------------------------------

- The matrix A is randomly generated for each test.
- The following scaled residual check will be computed:
      ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo ) * N )
- The relative machine precision (eps) is taken to be               1.110223e-16
- Computational tests pass if scaled residuals are less than                16.0

================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR11C2R4       25344   192     1     3              78.65             1.3799e+02
HPL_pdgesv() start time Sun Oct 20 18:15:26 2024

HPL_pdgesv() end time   Sun Oct 20 18:16:44 2024

--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=   2.25647678e-03 ...... PASSED
================================================================================

Finished      1 tests with the following results:
              1 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
```

Optional How To Change Username:

You can watch this video from 1.32 onwards:
https://www.youtube.com/watch?v=K0pVw0Ebijo&t=92s

Or follow the below:

1. First add a new user:

```
None
sudo adduser tempuser
```

2. Give new user sudo privileges:

```
None
sudo adduser tempuser sudo
```

3. Logout
4. Signin with the new created “tempuser”
5. Now that you have signed with the temp user you can change the name of your other user, change the home directory and login directory to reflect that name:

```
None
sudo usermod -l newUsername -m -d /home/newUsername oldusername
```

6. log out then log in as the modified user
7. Delete the temporary user using the command:

```
None
sudo userdel tempuser
```

Useful Resource

https://www.linkedin.com/pulse/how-create-three-nodes-linux-cluster-using-virtualbox-naushahi

https://www.mgaillard.fr/2022/08/27/benchmark-with-hpl.html