1. Technical Document (Per Machine)

Machine Hostname				
Interface	IP Address	Subnet Mask	Gateway	Hosts/DNS Alias
Public	. . .			hn/wn01
Private	10.0.0._____	255.255.255.0	10.0.0.1	wn0_____

Partitions	FS Type	FS Size	Recommended
/boot	xfs	1GiB	xfs, 1024MiB
swap	swap	16GiB	64GiB on WN, or 0.5 x RAM others
/	xfs	~GiB	xfs, remaining available space

Mount Points	Mounted From	Read Only	Read Write	Notes
/home			X	
/scratch			X	Fast storage (SSD or NVMe)
/soft				Read only on WNs after soft is installed

Software	Installed	Working	Notes
Firewalld			Only HN, unless WNs have direct internet
Torque Server			HN only
Torque Client			Mostly WNs but can consider on HN
Maui Server			HN only
NFS Server			HN only
NFS Client			WNs only
Ganglia Web-Service & gmetad			HN only
Ganglia gmond client			All nodes
Environment Modules			All nodes
GCC or Chosen Compiler			On /soft
OpenMPI			On /soft

The team leader must ensure that each line item is checked and that each application/service works as expected.
Also, make sure that services are started up after a reboot
Use a naming convention for nodes such as:

wn01, wn02, wn03 or node01, node02, node03

Then use a suffix that is added at the end of the name for the private network, e.g., suffix: -ib

wn01-ib, wn02-ib, wn03-ib or node01-ib, node02-ib, node03-ib

2. Full Design Plan

It could be helpful to use a format like AB & C …. Meaning: member A & B is responsible with C as backup

#	Task Description	On Host	By Member(s)	Notes
03	Install OS	HN	AC & D	
	Install OS	WNs		Set DNS to 8.8.8.8
04	Configure Network	HN		Not required if the network was configured during the installation
	Configure Network	WNs		
05	Configure Firewalld with NAT	HN		
06*	Add node names to the hosts file	HN		
07*	Create SSH keys for the root user	HN		
08*	Configure SSH Service	HN		
	Configure SSH Service	WNs		
09*	Disable SELinux	HN		
	Disable SELinux	WNs		
10*	Create accounts	HN		
11*	Setup sudo	HN		Copy /etc/sudoers to nodes
12*	Auto-generate SSH keys (script)	HN		
13*	Change password if needed	HN		
14	Synchronise files from HN to nodes	HN		
15*	Perform a dnf update	HN		
	Perform a dnf update	WNs		
16*	Configure NFS Server	HN		
17*	Configure FSTab (NFS mount)	WNs		
18	Testing by Team Leader	HN		
19*	Install Environment Modules	HN		
	Install Environment Modules	WNs		
20*	Performance Tuning	HN		
	Performance Tuning	WNs		
21*	Install the Intel Compiler	HN		
22	Reboot Machines	HN		
	Reboot Machines	WNs		
You should have a functional cluster now. If Torque, Maui and Ganglia are not required, continue with installing the Scientific Software				
23*	Install Torque Server	HN		
24*	Install Torque Clients	WNs		

- Add all identified tasks with the responsibilities etc., here
- Tasks numbered the same or marked with * should be executed in parallel to save time
- Task numbers reflect the chapter number in this document
- After each task, the team leader should verify that the specific job has been completed

NOTE: In the following segment, you will see grey code blocks.
Unless stated otherwise, these commands usually have to be executed as the **root** user.
The tool Yellowdog Updater, Modified (**YUM**), was recently replaced by the Dandified YUM (**DNF**) tool.
Following a tutorial referring to yum, you can use both these commands interchangeably.

3. * Install OS & Reboot (HN,SN & WNs)

Each node must be installed according to the design set out in the technical document.
Networking can be configured during the installation process, and it is advised; it will save you time
It is recommended to install all nodes using a Kickstart file (out of this scope); this way, all the configuration is uniform.

Head Node:

- If the HN is used to export software and scratch…. It could be helpful to add an extra HD
- The HN can be installed in the same way as a Compute/Worker node and will most likely be the case in the competition to save money
- If the HN is also going to execute jobs, Torque mom must also be installed – if using Torque

Worker Nodes:

- If a package/library is installed on one node, install the same package(s) on all nodes!!!!

4. * Setup Network Interfaces & ping nodes

If the network interfaces have not been configured during installation, configure them now.
Make sure that the HN can reach the internet and the public network.
Make sure that each IP address of the nodes can be pinged from the HN, e.g.:

Private IPs: ping 10.0.0.1; ping 10.0.0.2; ping 10.0.0.3

A configuration tool such as **nmtui** can set an IP address or edit the NetworkManager files directly.

Example content of **/etc/NetworkManager/system-connections/ens3.nmconnection :**

```
...
[ethernet]

[ipv4]
address1=10.0.0.1/24,10.0.0.100
dns=10.0.0.100;8.8.8.8;
dns-search=examplesdomain.com;
method=manual
...
...
```

The line starting "address1=" specifies the IP address in CIDR notation, followed by the gateway for the network.

Your device names may differ, so your config file, for instance, may be:
/etc/NetworkManager/system-connections/ens8.nmconnection
After modifying the file(s), restart the network service:

```
systemctl restart NetworkManager
```

5. Configure Firewalld - Firewall with NAT enabled (HN only)

Firewalld is a firewall that is widely used on GNU Linux. In recent years, RedHat and SuSE Linux moved over to Firewalld from IPTables. Firewalld should be installed by default, but we can ensure that it is the case:

```
dnf -y install firewalld

systemctl enable firewalld --now
```

Determine which device should be internal and which is external, according to the IP Addresses:

```
#View your current IP address information:
ip a
...

2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP . . .
   inet 192.168.4.32/24 brd 192.168.4.255 scope global dynamic noprefixroute ens3
...
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
   inet 10.0.0.1/24 brd 10.0.0.255 scope global noprefixroute ens4

# Note: ens3 is connected to the "internet" and ens4 is our internal connection.
# YOURS WILL DIFFER, so ensure that you set the following two lines to reflect yours

WAN_INT=ens3          #Specify the WAN interface connected to the Internet
LAN_INT=ens4          #Specify the internal interface

#Add the interfaces to the correct zones:
nmcli con mod $WAN_INT connection.zone external
nmcli con mod $LAN_INT connection.zone internal
nmcli con up  $WAN_INT
nmcli con up  $LAN_INT

#Add the firewall rules to allow NAT through the public interface
firewall-cmd --permanent --new-policy policy_int_to_ext
firewall-cmd --permanent     --policy policy_int_to_ext --add-ingress-zone internal
firewall-cmd --permanent     --policy policy_int_to_ext --add-egress-zone external
firewall-cmd --permanent     --policy policy_int_to_ext --set-priority 100
firewall-cmd --permanent     --policy policy_int_to_ext --set-target ACCEPT
firewall-cmd --permanent --zone=external --add-masquerade
firewall-cmd --reload

#Display some information:
firewall-cmd --get-active-zones
firewall-cmd --list-all
```

The abovementioned rules set up Firewalld, allowing NAT traffic through its public interface.
The text in BOLD needs to be noticed and may need to be modified. In this example, **ens3** is the public interface (192.168.xxx.xxx) with Internet access through its gateway. The other mentioned interface (**ens4)** is the private network interface between the nodes, for example, the interface with a 10.0.0.xxx IP address.

After adding the above commands, or after modifying Firewalld rules, you have to restart the firewall service by executing:

```
firewall-cmd --reload
       # OR
systemctl restart firewalld
```

To test that it works, log into one of the **worker nodes** and ping an external IP:

```
ping 8.8.8.8

#  Also test if DNS-resolving works:
ping www.google.com
```

To set the default gateway, we can specify the default routing interface in **/etc/sysconfig/network** :

```
GATEWAYDEV=ens3
```

Restart the network services:

```
systemctl restart NetworkManager
```

Tip: Whenever you are setting up a service that works through the network, and it seems that it is not working, temporarily stop Firewalld on both machines and try again. If it works, then add the correct entries to the firewall rules. Also, make sure to start Firewalld again afterwards. Another helpful file is **/etc/services** It lists several standard services with the ports on which they run.

6. Modify /etc/hosts to contain all node names and IP addresses (HN)

The **/etc/hosts** file contains the hostnames and local IP addresses of machines you want to refer to that do not use a Domain Name Server (DNS). It is a good idea (if you don't have access to a DNS) to add the IP addresses and different names of the nodes into this file, so the system can resolve them as needed.
The format of the file is simple. Keeping the local and localhost IP addresses in the file is vital.
Here is an example of a **/etc/hosts** file:

```
127.0.0.1                localhost localhost.localdomain
::1                      localhost

#The following line can be uncommented if you use a separate hn, login, storage node
#10.0.0.100              hn login  storage  scratch
10.0.0.1                 wn01  node01  wn01-ib  node01-ib hn login storage scratch
10.0.0.2                 wn02  node02  wn02-ib  node02-ib
10.0.0.3                 wn03  node03  wn03-ib  node02-ib

#If you have a different IP range for storage etc., you should also specify it here:
#10.10.10.100            hn-ib     login-ib    storage-ib  scratch-ib
#10.10.10.1              wn01-ib   node01-ib   hn-ib login-ib storage-ib scratch-ib
#10.10.10.2              wn02-ib   node02-ib
#10.10.10.3              wn03-ib   node03-ib
```

7. Create an SSH Key for root and copy it to WNs (HN)

Using SSH Keys to login onto nodes is very important in a cluster. This allows users to log in to nodes without using a password. It is essential in a cluster because when a job is started from the HN, the job will act as that user and log in to the remote node(s). If it needs a password to log in, the job will fail. This is especially true for MPI jobs (discussed later).

An SSH Public Key or Certificate can be shared, emailed or copied to other people. However, the Private Key/Certificate should never be shared with other people. If a Private key is shared, it can be used by other people to log into the system using your account without a password. If a Private Key is stolen or shared, that key must no longer be used and should be replaced.

As root, execute the following:

```
# The following line reads; test if a specific file exists
# or else execute:   ssh-keygen -t ed25519
[ -e ~/.ssh/id_ed25519 ] || ssh-keygen -t ed25519

cd ~/.ssh
cat id_ed25519.pub >> authorized_keys
chmod 600 authorized_keys

# Now, we need to copy this key to nodes, but root can't ssh/scp to the nodes yet
# So, we copy the authorized_keys file to the remote hosts as a normal user first:
MyUsername=hpc
MyHome=$(eval echo ~$MyUsername)
nodes=4
for i in $(seq 1 $nodes); do
   ssh-copy-id -i ~/.ssh/id_ed25519 ${MyUsername}@wn0${i}
done

# The following for loop is a bit complex, but in essence; for each node:
# log in as our normal user, create /root/.ssh  and  copy the content of
# the normal user's authorized_keys file into root's authorized_keys file
for i in $(seq 1 $nodes); do
  ssh -n ${MyUsername}@wn0${i} "sudo mkdir /root/.ssh &>/dev/null; \
     sudo bash -c 'cat $MyHome/.ssh/authorized_keys >> /root/.ssh/authorized_keys'"
done

# After the authorized_keys files were copied, we should be able to log into
# the remote nodes as root without being prompted for a password, let's test it:
for i in $(seq 1 $nodes); do
    ssh -n wn0${i} "hostname;uptime;echo;echo"
done
```

Modify the username (**hpc**) above to reflect your local username (not root). This example is tricky since the root user cannot SSH using a password. However, it is crucial that the commands are executed successfully before continuing, or you will have errors when executing later commands.

8. * Configure SSH Service & turn off root login using a password (HN WNs)

Root login from remote machines (Internet) is dangerous, so we disallow the root user to log in from remote locations (over any network interface) using a password.
However:

- The root user will still be able to log in at the physical machine using a password
- Users that have sudo privileges will still be able to become root
- The root user can log in from remote locations via SSH certificates/keys

To-do: Modify **/etc/ssh/sshd_config**

Look for the following parameters and **modify/add** the following values where needed:

```
PubkeyAuthentication yes
PasswordAuthentication yes
PermitRootLogin without-password    #Set this value to yes if you are struggling
UseDNS no                           #Speed up the logins from remote hosts
```

The lines in bold are the important ones; the others should already exist or are the defaults.
After modifying this file, you will have to restart the SSH service:

```
systemctl restart sshd
```

9. * Disable SELinux (HN & WNs)

SELinux is used to harden security on a GNU Linux system (RedHat family). It restricts users and services from accessing files, ports, connection types and devices without pre-approved permission. For instance, Apache is allowed to serve websites from a specific location. Still, if files are placed outside that location and Apache is configured to publish them, an SELinux violation will occur.
It is best practice to keep SELinux running on a production server, but for our purpose here, it is easier for you to disable SELinux than having to "debug" that too.

Modify the **/etc/selinux/config** file and change the SELINUX line to reflect the following:

```
SELINUX=disabled
```

After modifying the SELINUX parameter, you must reboot the system for the changes to take effect. However, if you are not allowed to reboot the system, you can execute the command:

```
setenforce 0
```

10. Create user accounts, add to the admin (wheel) group and give a password (HN)

Adding a group that will become the cluster's software owner is helpful. When regular user accounts are created, those users can then just be added to the group, and they will have the required access to the software:

```
groupadd -r   hpcusers
```

It is good practice not to use the root administrator account unless it becomes essential. A system group exists that already has some root privileges set to it. The group "wheel" is configured in the sudoers file, which allows group members to execute commands as root.

To create a user that is a member of "wheel", the following command can be executed:

```
useradd -G wheel,hpcusers   -m   your_username
```

The above command will create a user called **your_username** (change to the value you want) that belongs to the groups wheel and hpcusers. The user's home directory (-m) will also be created.

You can give an encrypted password for the user when you create the account, or you can just set the password afterwards with the following command:

```
passwd your_username
```

After typing the above command, the user is prompted to enter and confirm the password. Note that nothing is displayed on the screen while the user is typing the password.

11. Setup sudo rights (HN)

A sudo file permits specific users to execute commands requiring root privileges on the system.
The file that controls all the sudo system rights is: **/etc/sudoers**
After adding members to the wheel group, they will, by default, have access to execute commands as root because of the line that reads:

```
%wheel      ALL=(ALL)       ALL
```

You may also opt to add a line in **/etc/sudoers** or simply remove the # in front of the existing line to allow sudo commands without prompting for your password. The line should look as follows:

```
%wheel      ALL=(ALL)       NOPASSWD: ALL
```

It is just mentioned here which file to modify in case you need to change it or if you want to allow specific users to execute some particular commands only. This file can also be copied to other hosts requiring the same permissions, discussed later in Section 14.

12. Each user logs in and Generate their SSH Keys (HN)

In step 7, we discussed creating SSH Keys for the root user. The procedure is similar for other users, except the users' home directories will be shared between the nodes, and thus, the generated keys don't need to be copied over to other nodes. However, one needs to add the public key to an authorised file with a list of public keys that can be used to authenticate as the user.

All the system users should execute the following commands to allow them to connect password-less to the nodes:

```
#!/bin/bash
if ! [ -e $HOME/.ssh/id_ed25519 ]; then
  ssh-keygen -t ed25519
  cd
  cd .ssh
  cat id_ed25519.pub >> authorized_keys
  chmod 600 authorized_keys
fi
```

Very Important:

It will be helpful to add the above code into a script that is executed on the Head Node when a user logs in. If you opt to add it into such as script, place the content in a script in something like **/etc/profile.d/ssh_keys.sh** and remember to make it accessible for all users:
chmod 644 /etc/profile.d/ssh_keys.sh

13. If generic passwords were used, each member must change their password (HN)

If you made use of a generic password or did not set the password for the other members of the team, you may want to do it now because the passwords will be synced to the rest of the nodes in the following steps.

```
passwd  member1
passwd  member2
passwd  member3
passwd  member4
```

14. Sync the /etc/{passwd,hosts,group,shadow,sudoers} files to WNs (HN)

After ensuring that all the team members have set their passwords, are in the wheel group and that the **sudoers** and **hosts** entries are correct; you should copy the configuration files to all the nodes.

Ensure all the nodes are powered on at this stage, and their IP addresses are configured correctly.

The important files can be copied by executing the following command on the HN:

```
NumberOfNodes=4
for i in $(seq 1 $NumberOfNodes); do
       scp /etc/{passwd,shadow,group,sudoers,hosts}   wn0$i:/etc/
done
```

The abovementioned command should be executed whenever a user is added to the system when a host/node is added or its IP address is changed. Note that some applications installed from RPM or through DNF also add users. To be safe, synchronise the files again after installing an RPM on the HN. To be safe on the nodes' side, first sync the files from the HN before installing an RPM.

15. * Perform a dnf update (HN & WNs)

After installing the nodes and if they have internet connectivity, update all the machines:

```
dnf -y update
```

16. Create scratch, soft and home directories and setup NFS (HN or SN)

As mentioned, the scratch, soft, and home directories must be shared over the network between all the nodes. To achieve this, we need to export them using NFS. The files and directories will be physically stored on your Storage Node's hard drive, but users can also access them on the nodes.

Before setting up NFS, I would recommend that the directories that need to be shared are created, and all reside in a logical path. I usually make a /exports or /data directory. Even though we are talking about a directory here, it could (and should) be a different volume or disk than the root (/) filesystem.

Assumptions:
Suppose you have a **Solid-State Disk** (or faster, such as m.2 or NVMe) that will be used for the homes, scratch and software storage.
Suppose you have mounted this disk on **/exports** and added the **mount point into your fstab**, which is automatically mounted after the machine is restarted.

```
#Remember, we are assuming /exports is already created/mounted
cd /exports

#Make sure that you are on the correct path and see the data expected in this volume
ls

#Create the scratch and soft directories
mkdir  scratch  soft

#Create the symbolic links:
ln -s   /exports/soft      /
ln -s   /exports/scratch   /

#Install the NFS utilities, which are required later
dnf -y install nfs-utils

#Add firewall rules to allow NFS traffic from WNs:
firewall-cmd --permanent --zone=internal --add-service=nfs
firewall-cmd --permanent --zone=internal --add-service=mountd
firewall-cmd --permanent --zone=internal --add-service=rpc-bind

#Activate the rules
firewall-cmd --reload
```

The NFS server configuration is done on the same storage, scratch, or head node we configured above. Also, edit **/etc/exports** to contain the following (Note, you have to change the IP addresses to reflect yours):

```
/home                    10.0.0.0/24(async,rw,no_root_squash)
/exports/soft            10.0.0.0/24(async,rw,no_root_squash)
/exports/scratch         10.0.0.0/24(async,rw,no_root_squash)
```

The async option is used to cache content in memory before writing it to disk. This allows the node to continue while the server writes the data in the background onto the hard drive. The risk with that is that should the server be restarted in the meantime, data corruption will occur. We can make this trade-off now for better performance….if you want to risk it. If no_root_squash is used, remote root users can change any file on that shared file system. This is okay in our trusted environment but should be removed if you export to untrusted sources.

For the changes to be applied, the following services should be restarted:

```
systemctl restart rpcbind
systemctl restart nfs-server
systemctl enable  rpcbind
systemctl enable  nfs-server
```

17. Mount NFS Exports in the correct paths and modify fstab (WNs)

After the NFS "server" has been configured, you should be able to mount the exported filesystem on all **WNs**. Note that we make use of alias hostnames (**scratch** and **storage**) below. These aliases were set in Section 6 and for the cluster competition, they may all reflect the same node as wn01. However, in a production environment, they will indicate dedicated hosts for those purposes.

```
#Installing the NFS utilities is required to be able to mount an NFS volume
dnf -y install nfs-utils

mkdir /scratch /soft
mount scratch:/exports/scratch   /scratch
mount storage:/exports/soft      /soft
mount storage:/home              /home
```

If those commands were all successful, you should modify the node's**/etc/fstab** file **by adding the following**:

```
scratch:/exports/scratch  /scratch      nfs    rw,tcp,noatime  0   0
storage:/exports/soft     /soft         nfs    rw,tcp,noatime  0   0
storage:/home             /home         nfs    rw,tcp,noatime  0   0
```

After modifying the fstab, you can execute **mount -a** to ensure everything is mounted correctly.
You can also make use of the following options to improve performance…..maybe read up on them:
rw,tcp,noatime,hard,intr,rsize=32768,wsize=32768,retry=60,timeo=60,acl,nfsvers=3

If you have difficulty mounting something, log into the server (scratch/storage in this case) where you are mounting from, look in **/var/log/messages** for messages about why something might fail, and see if the firewall isn't blocking you.

18. Ensure password-less SSH works from HN to WN01 to WN02 back to WN01 back to HN etc.

After NFS is set up on all the nodes, all users (not just root) should be able to SSH to all the nodes without typing in a password. Test this using one of the members' accounts. SSH to wn01, then exit and do the same for wn02….wnXX. Also, make sure to SSH from one of the nodes back to the HN. Here is a command that might be of assistance:

```
Headnode=wn01
Nodes=4
for i in $(seq -w 1 $Nodes); do
        ssh -n wn0$i  "hostname;ssh -n $Headnode 'uptime' "
done
```

You will also notice that you must type in "yes" the first time you connect to a host; it is essential to know this because if a user tries to run an MPI job and hasn't SSH'ed to that node name, the job will hang. Notice I said to that node's name... it can also be wn01-ib etc.

19. *Install Environment Modules (HN & WNs)

The Environment Modules package is beneficial for managing users' environments. It allows you to write a module file for multiple software versions and then lets the user choose which version (s)he would like to use. For instance, you can install four different versions of GCC and then just use the one you require for a specific purpose. It becomes beneficial when installing Scientific Software because a researcher usually uses a particular version for his research, while another researcher needs another version for her study.

We already started using Environment modules in the "Submit and manually start an MPI job that uses all nodes" section. You can install Environment modules by downloading the latest version from the http://modules.sourceforge.net/ website or install the package through dnf. I would recommend the dnf install

method because you have to install this package on all the nodes, and the dnf package available online will suffice for this exercise.

```
#Install the environment-modules package:
dnf -y install environment-modules
```

You may have to log out and in again, to execute the module command after initially installing it.
The above install creates a few modules in **/usr/share/Modules/modulefiles**. They can be helpful to look at.
To see the modules available, execute the following command:

```
module avail
```

Now we want the module command to look for modules in our software directory too. To achieve this, we can create a file called **/etc/profile.d/zhpc.sh**, which is loaded when a user logs in to set the MODULEPATH. We make the filename zhpc.sh because the order of execution in the /etc/profile.d is done alphabetically, and we need the /etc/profile.d/modules.sh to be executed before our script is loaded. The following commands will create the file and make it executable:

```
cat > /etc/profile.d/zhpc.sh <<EOF
#!/bin/bash

export MODULEPATH=\$MODULEPATH:/soft/modules
EOF

#Now create the same file for the C-Shell:
cat > /etc/profile.d/zhpc.csh <<EOF
#!/bin/csh

setenv MODULEPATH "\$MODULEPATH:/soft/modules"
EOF

chmod 755 /etc/profile.d/zhpc.{sh,csh}
```

It is recommended to create a generic module that is **copied to all the nodes** and holds generic variables that nodes can use. Here is what is suggested:

Create a file: **/usr/share/Modules/modulefiles/hpc** with the content:

```
#%Module 1.0
#
#  HPC module for use with the 'environment-modules' package:

set           SOFT                  /soft
set           MODULES               $SOFT/modules

set           username              $::env(USER)
set           tmp_scratch           /scratch/$username

if {[info exists env(PBS_JOBNAME)]} {
    set scratch     $tmp_scratch/$env(PBS_JOBID).$env(PBS_JOBNAME)

} else {
    set scratch     $tmp_scratch
}

setenv        HPC_SOFT              $SOFT
setenv        HPC_MODULE_PATH       $MODULES
setenv        HPC_TMP               /tmp
setenv        HPC_SCRATCH           $scratch
setenv        HPC_OWNER             root
setenv        HPC_GROUP             hpcuser
setenv        TERM                  linux

prepend-path  MODULEPATH            $MODULES
prepend-path  PATH                  $SOFT/hpc

append-path   INCLUDE               /usr/include
append-path   LD_LIBRARY_PATH       /usr/lib64
append-path   PKG_CONFIG_PATH       /usr/lib64/pkgconfig

set-alias     vi                    "/usr/bin/vim"
```

Then add the following line in a file **(on all nodes)** called **/etc/profile.d/zmodules_hpc.sh** :

```
module load hpc
```

This will load the hpc module every time a user is logged in.

The benefit of this module is that the environment will be set up so that modules put in **/soft/modules** will be available to be loaded by users. An entry is made to add **/soft/hpc** as a location where scripts can be put that will be in the user's path. The users will automatically be able to execute scripts in this path and have their executable flag set using **chmod**.

Remember to copy the **/etc/profile.d/zmodules_hpc.sh, /etc/profile.d/zhpc.sh** and the **/usr/share/Modules/modulefiles/hpc** files to all the machines, so you can create it on one node and scp it to the other nodes:

```
#We assume that the files were created on wn01 and are now copied to the others:

for i in $(seq 2 4); do
  scp /etc/profile.d/zmodules_hpc.{sh,csh}   wn0$i:/etc/profile.d/
  scp /etc/profile.d/zhpc.sh   wn0$i:/etc/profile.d/
  scp /usr/share/Modules/modulefiles/hpc    wn0$i:/usr/share/Modules/modulefiles/
done
```

20. *Performance Tuning (HN &WNs)

Some performance tuning can be done within the Linux environment itself. Numerous optimisations will enhance your machines' performance in an HPC environment. However, it is particular to the equipment that is used. For this reason, we will only set up a few important ones, such as the CPU throttling by the kernel.

Change or add the following entries in **/etc/security/limits.conf** :

```
* hard memlock unlimited
* soft memlock unlimited
* soft nofile 63488
* hard nofile 63488
```

These settings will only be applied after a system reboot (which will be done in the next section) and can then be viewed with the command:

```
ulimit -a
```

The following script should be executed **on all nodes** to add a new performance-tuning module to your Linux environment:

```
ProfileName=hpc-performance
Vendor=$(lscpu |grep "^Vendor ID" | sed -e "s|.*: *||g")

dnf -y install tuned
systemctl enable tuned --now
if [ "$Vendor" == "GenuineIntel" ]; then
  grep "intel_pstate" /etc/default/grub > /dev/null
  result=$?

  if [ $result -ne 0 ]; then
     sed -i "/^GRUB_CMDLINE_LINUX=/ s|\"$| intel_pstate=disable\"|g" \
             /etc/default/grub
     grub2-mkconfig -o /boot/grub2/grub.cfg
  fi
  cat /proc/cmdline |grep "intel_pstate=disable" > /dev/null
fi

cd /usr/lib/tuned/

[ -e $ProfileName ] || mkdir -p $ProfileName
cd $ProfileName
cat > tuned.conf <<EOF
[main]
summary=Optimize for deterministic performance; increased power consumption
include=throughput-performance

[sysctl]
vm.overcommit_memory = 1
EOF

tuned-adm profile $ProfileName
tuned-adm active
```

Very Important BIOS settings:

Some modifications should also be done in all the nodes' BIOS.
The most important (if they exist in your BIOS) settings are:

Power	Configure to use Max power if there is such an option
P-State	Disabled - This is also enforced by the script executed above
C-State	Disabled
Turbo Mode	Enabled - Specific to Intel CPUs
Hyper-Threading	Disabled - Intel
O Non-Posted Prefetching	Disabled - Intel Haswell/Broadwell and onwards CPUs
CPU Frequency	Set to Max
Memory Speed	Set to Max
Memory channel mode	Set to "independent"
Node Interleaving	Disabled - We need to enable NUMA
Channel Interleaving	Enabled
Thermal Mode	Set to Performance mode
HPC Optimizations	Enabled - AMD Specific

Also, see: https://community.mellanox.com/docs/DOC-2297 for an example of HPC BIOS settings

21. *Install the Intel Compiler (HN)

The Intel Compiler is an optimised compiler that drastically enhances Scientific Software's performance. Intel has a free edition available to students that is valid for a limited time only. Registering for the download can take a few days, so some planning is needed to ensure you have access to the compiler when required. At the time of this writing, a suite known as Intel Parallel Studio XE version 2019 is available. The essential components in this collection are Intel Compiler for C/C++, Intel Compiler for Fortran, Intel MPI SDK,
Intel Math Kernel Library (MKL), Intel Thread Building Blocks (TBB) and optionally, if you are going to run software (such as Pluto) that makes use of Python: Intel Python 2 & 3. Also, make sure you download the Linux versions instead of the Mac or Microsoft versions.

The installation has a graphical or a text wizard that can be run to install the compilers. We will focus on the command line wizard because we use a Linux terminal. To automate the installation procedure, the install.sh script will be called, and a custom silent configuration script will be created that is parsed to the install.sh, script. In the following script, we will install the compiler onto the NFS share, making it available on all the nodes. Execute the next steps on the HN to install the Intel Compilers:

```
#Specify YOUR serial number in the next line
SERIAL_NUMBER=Enter_Your_Serial_Number_Here
INTEL_VERSION=2019
PYTHON_VERSION=2
PACKAGE_NAME=parallel_studio_xe_${INTEL_VERSION}_update1_cluster_edition
TAR_FILE=${PACKAGE_NAME}.tgz
DESTINATION=/soft/intel/$INTEL_VERSION

if ! [ -e $TAR_FILE ]; then
 echo "This should be executed in the directory where the file '$TAR_FILE' resides."
fi

dnf -y groupinstall "Development Tools"
dnf -y install kernel-headers kernel-devel kernel-tools \
               gtk2-devel libstdc++-devel.i686 \
               glibc.i686 libgcc.i686 libstdc++6.i686

tar -xf $TAR_FILE
cd $PACKAGE_NAME

#Create a silent config file:
cat > custom.cfg <<EOF
ACTIVATION_SERIAL_NUMBER=$SERIAL_NUMBER
PSET_INSTALL_DIR=$DESTINATION
ACCEPT_EULA=accept
CONTINUE_WITH_OPTIONAL_ERROR=yes
CONTINUE_WITH_INSTALLDIR_OVERWRITE=yes
PSET_MODE=install
ACTIVATION_TYPE=serial_number
AMPLIFIER_SAMPLING_DRIVER_INSTALL_TYPE=build
AMPLIFIER_DRIVER_ACCESS_GROUP=vtune
AMPLIFIER_DRIVER_PERMISSIONS=666
AMPLIFIER_LOAD_DRIVER=yes
AMPLIFIER_C_COMPILER=auto
AMPLIFIER_KERNEL_SRC_DIR=auto
AMPLIFIER_MAKE_COMMAND=auto
AMPLIFIER_INSTALL_BOOT_SCRIPT=yes
AMPLIFIER_DRIVER_PER_USER_MODE=no
INTEL_SW_IMPROVEMENT_PROGRAM_CONSENT=no
SIGNING_ENABLED=yes
ARCH_SELECTED=INTEL64
COMPONENTS=DEFAULTS
EOF

./install.sh --silent custom.cfg
```

In the above script, type the requested serial number (in bold). Also, ensure you do not have spaces before or after the equal signs.

The previously executed commands should install the most critical Intel Compiler components. The installation comes with scripts (in our case: **/soft/intel/2019/bin/compilervars.sh**) that can be used to set up your environment. In a small cluster such as this one, it would be acceptable to add a small script (e.g. /etc/profile.d/intel.sh) on all the nodes that are executed upon user login that would call this script to set the environment. E.g. one can create a script (/etc/profile.d/intel.sh) with the content:

. **/soft/intel/2019/bin/compilervars.sh intel64**

However, sometimes you want to make use of a different compiler, and in that case, it would cause conflicting library issues. To prevent these issues, we will make use of an environmental modules file that can be used to load the compilers into our environment when needed.

Even though the destination path was set to **/soft/intel/2019**, the installer still copies the license file to a file under /opt/intel/licenses/. This file is required on the other nodes if you want to be able to compile software there too. To copy the license file, execute the following command on the node where the Intel installation wizard was run:

```
Nodes=4
for i in $(seq 1 $Nodes); do
   scp -r /opt/intel wn0$i:/opt/
done
```

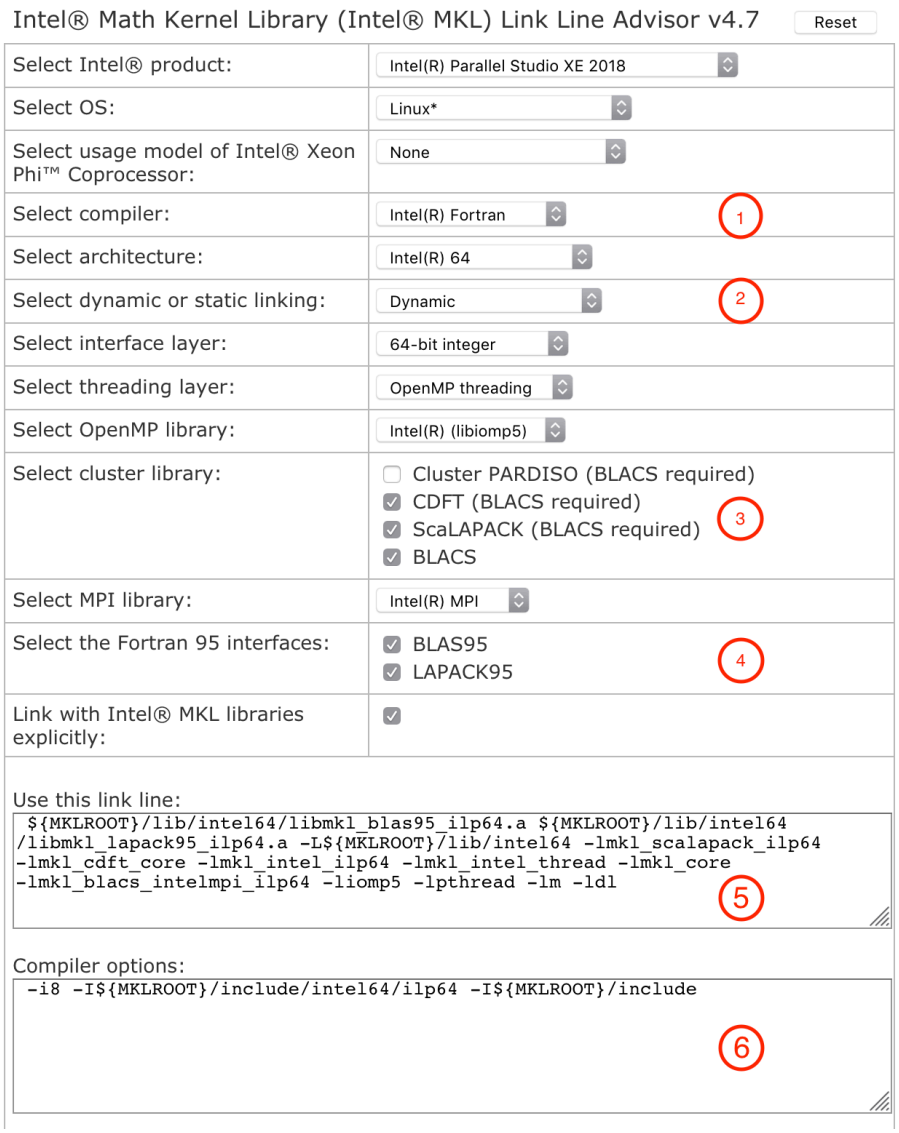
It would also be possible to copy the provided Intel license file to the NFS volume (/soft/intel/2019/licenses) and then set the Environment variable (INTEL_LICENSE_FILE) to the path where the license can be found.

An example module file for the Intel Compilers can be downloaded by executing the following command:

```
[ -e /soft/modules/intel ] || mkdir -p /soft/modules/intel
wget http://grid-ui.ufs.ac.za/chpc/intel.module -O /soft/modules/intel/2019
```

Open the downloaded file **(/soft/modules/intel/2019)** and modify the line that sets the **compiler_flags** to reflect your CPU architecture. See https://software.intel.com/en-us/cpp-compiler-developer-guide-and-reference-march for a list of available options. Also, change the **compiler_ver** value if you are not using the 2019 release.

In the provided module file, a lot of variables are set. The values were determined using the following: https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor as a reference. A screen grab of the selected options follows:



When using this website as a reference, you will notice that Section 1 significantly changes the values of Sections 5 and 6. In this case, Section 1 was set to display the parameters for the Fortran compiler. Section 2 can also be set to compile a static library. You will notice that in Section 3, CDFT, ScaLAPACK and BLACS were set. This will include the mentioned libraries, which supply various scientific constants and functions to the compiled code. The Intel ScaLAPACK etc., are more optimised than most third-party libraries. You may remove some of those libraries, especially if the software requires its own ScaLAPACK, etc. Section 4 is only available, while the Intel Fortran Compiler in Section 1 is selected.

The critical part of this screengrab is Sections 5 and 6. These values were used when compiling the module file referred to. If changing any of the settings as per the above image, Sections 5 and 6 will change, and to implement the changes in the module file, you will have to modify the values of **compiler_additional** and **compiler_link_line** in the module file yourself. Those two variables are used to build the other variables, such as CFLAGS, FFLAGS, and HPC_LINKLINE.

When compiling code against the Intel compiler, modify the Makefile/makefile and replace the values of CPPFLAGS, LIBDIR, CFLAGS, FFLAGS and LLIBS with the values that are set after the Intel module is loaded. E.g. when the Intel module is loaded, you can execute the following to get the "important" values used when compiling code:

```
module load intel
#View the values of the following variables and add those to the makefile
set | grep "^CPPFLAGS\|^LIBDIR\|^CFLAGS\|^FFLAGS\|LLIBS\|^HPC_LINKLINE"

#View the values of the specified compilers and modify the makefile accordingly:
set | grep "^AR=\|^CC=\|^FC=\|^F77=\|^F90=\|^CXX=\|^LD="
```

The above code snippet shows some of the essential variables when compiling code. However, this is only a guideline, and you must replace (and interpret) the correct settings with the required values. If you want to use, e.g., BLAS, you must also add the value of $HPC_LINKLINE_BLAS to the link line (LLIBS or whatever the makefile uses to indicate the link line).

Very Important:

When an application is compiled using a specific compiler, you will also have to set the environment (e.g. **module load intel**) each time before the software application is executed. This is necessary to ensure that the LD_LIBRARY_PATH, the PATH and other environmental variables are correctly set for the application.

22. Reboot all the machines (SN/HN first) and make sure WNs boot up with home & scratch mounted

The bulk of the communication test and configuration are done now. Now you can reboot all the machines to ensure their storage is still mounted. **It is crucial** to make sure that the Storage Node (head node or most likely wn01) is booted up first before booting up the rest of the Worker Nodes; because the WNs won't be able to mount all the mount points (/soft /scratch & /home) if they are not available yet.

23. Install & Configure Torque Server (HN)

Torque can be installed from the EPEL repository or source code. The version on EPEL is relatively new and should contain the functionality required for your purposes. You will also notice that Munge is a dependency for Torque. That is because Munge is used as a method of authentication. For your purposes, you can simply install Munge and copy the key from the HN to the nodes at a later stage.

First of all, we are going to install Torque Server and Munge through dnf. These packages will add users to the system, so a file sync is necessary afterwards.

On the **HN,** execute:

```
dnf -y install epel-release
dnf -y install torque-server  torque-client  torque

#Set the hostname to the short format, if the domain name was specified earlier
hostnamectl set-hostname $(hostname -s)
systemctl restart systemd-hostnamed

#On RedHat systems, it is good practice to enable the service immediately,
# to ensure you don't forget later
systemctl enable pbs_server
systemctl enable munge

#Now try to start munge
systemctl start munge

#It failed, didn't it?
# You can execute journalctl -xeu munge.service to see why

#Now, let's generate a key:
dd if=/dev/urandom   of=/etc/munge/munge.key bs=1 count=1024

chown  munge:     /etc/munge/munge.key
chmod  0400       /etc/munge/munge.key
chmod  0700       /etc/munge

#Now let's try and start munge again:
systemctl start munge

#Test munge:
munge -n |unmunge

#Add firewall rules
firewall-cmd --permanent  --zone=internal --add-port=15001/tcp
firewall-cmd --permanent  --zone=internal --add-port=15002/tcp
firewall-cmd --permanent  --zone=internal --add-port=15003/tcp
firewall-cmd --reload
```

Before continuing, the **/etc/hosts** file must contain all the nodes' names. This should already be the case by now.

On the **HN**, execute:

```
#Set the server name:
PBSServer=wn01

#Create a pbs_server database
pbs_server -t create
kill -9 $(pidof pbs_server)

cat > /var/lib/torque/server_priv/nodes <<EOF
hn       np=1    server no_jobs   all
wn01     np=16                    all  prod
wn02     np=16                    all  prod
wn03     np=16                    all  prod
EOF

echo "$PBSServer" > /etc/torque/server_name

systemctl enable  trqauthd
systemctl restart trqauthd
systemctl restart pbs_server

#Now we need to sync some files to all the nodes again:
for i in $(seq 1 4); do
   scp /etc/{passwd,group,shadow,hosts}    wn0$i:/etc/
done

#Now create a queue called hpc
qmgr -c "create queue hpc queue_type=execution" $PBSServer
qmgr -c "set queue hpc enabled=true" $PBSServer
qmgr -c "set queue hpc started=true" $PBSServer
qmgr -c "set server default_queue = hpc" $PBSServer
```

All the above commands should have been executed without errors.

24. * Install & Configure Torque Client (HN & WNs)

The PBS Server was installed in the previous step and should be running. If that is not the case, the rest will be difficult to debug if something should go wrong. If you struggle to connect to the pbs_server and are sure the service is running, try to ping the DNS/alias name, e.g. wn01. If that works, try switching off the firewalls on both the server and the client. If it finally works, you might have to check your firewall's rules and correct those before continuing.

The torque client was installed but not configured on the HN in the previous section. This was done so that the commands to control the queues etc., are available on the HN. However, no jobs will be sent on the HN at this point. If you want to be able to submit jobs to the HN, you can execute the following instructions on the HN too.

Installing and configuring torque-client to allow jobs to be executed on nodes **(On HN & WNs):**

```
#Set the following Environment variable to the hostname of the HN:
PBS_Server=wn01

#Set the system hostname to a short format:
sudo hostnamectl set-hostname $(hostname -s)
sudo systemctl restart systemd-hostnamed

#Install torque-client
dnf -y install torque-client   torque-mom   torque-devel   torque

#These services must be enabled to start automatically when the server is restarted
systemctl enable pbs_mom
systemctl enable trqauthd
systemctl enable munge

#Munge has been set up on the HN, so we just copy the key from the HN:
[ "$(hostname)" == "$PBS_Server" ] || \
    scp $PBS_Server:/etc/munge/munge.key  /etc/munge/

#Now make the munge user the owner of the munge.key file
chown munge: /etc/munge/munge.key

#Now configure the PBS Client:
cd /var/lib/torque

#Network name, suffix ... the suffix added at the
# end of the hostname to define which network to use:
# You must also have the corresponding entries in the /etc/hosts file.
#E.g., 192.168.xxx.1    wn01-ib
NetSuffix="-ib"

#Get the number of cores on the system:
NumberOfCores=$( lscpu |grep "^CPU(s):" | sed "s|.* ||g")

echo "$PBS_Server" > server_name
cd /var/lib/torque/mom_priv

#The following section, up to the EOF line, has to be executed as a single command:
cat > config <<EOF
\$pbsserver $PBS_Server
\$logevent 255
\$ideal_load $(( $NumberOfCores - 1 ))
\$max_load $NumberOfCores
\$job_exit_wait_time 300
\$nodefile_suffix $NetSuffix
\$source_login_batch true
\$spool_as_final_name true
EOF

echo "$PBSServer" > /etc/torque/server_name

#Now we start all the services
systemctl start munge
systemctl start trqauthd
systemctl start pbs_mom
systemctl restart trqauthd

#On the HN: see if the node is connected to the PBS Server
pbsnodes  $(hostname)
```

After executing the **pbsnodes $(hostname)** command, you may notice the node is offline. This is because the NUMA configuration has not been done. New versions of PBS Torque consider NUMA configuration. This means that you have to specify the layout of your CPU cores regarding the memory layout.
You can have a look at: http://docs.adaptivecomputing.com/9-0-3/MWM/Content/topics/torque/1-installConfig/buildingWithNUMA.htm

On the HN, also have a look at the Torque server logfiles in **/var/lib/torque/server_logs/**

The following will give you an idea of the layout **(On all HN & WN, execute to get layout.....AFTER pbs_mom HAS BEEN INSTALLED ON THAT MACHINE)**:

```
cd /sys/devices/system/node
ls -l
-r--r--r--. 1 root root 4096 Jun 11 14:43 has_cpu
-r--r--r--. 1 root root 4096 Jun 11 14:43 has_memory
-r--r--r--. 1 root root 4096 Jun 11 14:43 has_normal_memory
drwxr-xr-x. 4 root root    0 May 29 13:03 node0
drwxr-xr-x. 4 root root    0 May 29 13:03 node1
-r--r--r--. 1 root root 4096 Jun 11 14:43 online
-r--r--r--. 1 root root 4096 Jun 11 14:43 possible
drwxr-xr-x. 2 root root    0 Jun 11 14:43 power
-rw-r--r--. 1 root root 4096 May 29 13:03 uevent

#From this file listing, we see that there are two nodes: node0 and node1
#This tells us that the num_node_boards must be set to 2

#Here is a probable 'easier' way to configure the NUMA settings:
NumNUMA=$( lscpu |grep "^NUMA node(s)"|sed "s|.*: *||g" )
NumCores=$( lscpu |grep "^CPU(s)"|sed "s|.*: *||g" )

#Clear the content of the mom.layout file and then add the entries:
>  /var/lib/torque/mom_priv/mom.layout
for i in $(seq 0 $(( $NumNUMA - 1 )) ); do
    echo "nodes=$i"  >> /var/lib/torque/mom_priv/mom.layout
done

echo "Now you have to make the following entry on the HN."
echo "In the file:  /var/lib/torque/server_priv/nodes"
echo "$(hostname)      np=$NumCores num_node_boards=$NumNUMA  all prod"
```

By the way, a **lscpu** and **numactl -H** should also show you these results. When performing a **lscpu**, you need to take note of the lines starting with "NUMA node(s)" and "CPU(s): "those are the values required for the **/var/lib/server_priv/nodes** file and the **/var/lib/mom_priv/mom.layout** file on the WN. For a graphical representation of the NUMA layout, execute the command: **lstopo --output-format png -v --no-io > cpu.png** More information regarding your hardware can also be obtained using the **dmidecode** command.

The /var/lib/torque/server_priv/nodes must have the NUMA layout of the nodes. Otherwise, the command **pbsnodes** will state that all the nodes are down. The nodes file will now have to be changed to reflect a configuration such as the following:

```
hn       np=1  server no_jobs      all
wn01     np=16 num_node_boards=1   all prod
wn02     np=16 num_node_boards=1   all prod
wn03     np=16 num_node_boards=1   all prod
```

25. Submit and manually start a single node test job (HN)

Torque should be operational now, and you can submit a simple test job to the queue.
You have to be a "normal" user to execute the following and submit a job....root can't!
Create a file **test.pbs** with the content:

```
#!/bin/bash
#PBS -l walltime=00:01:00
#PBS -l nice=19
#PBS -l nodes=1:ppn=16:prod
#PBS -q hpc
echo "I am running on $(hostname)"
date
sleep 10
date
```

Now submit the test job:

```
qsub test.pbs
```

26. Make sure that job executes correctly (HN)

The job should be queued at this point, **then as root**, you can force run the job:

```
qrun xxxx      # xxxx should be replaced with the job id
```

All should be well if the job runs according to the queue (**qstat -a**).
If the job fails to run, check the log files:
On the HN: **/var/lib/torque/server_logs/$(date +%Y%m%d) and /var/log/messages**
On the node: **/var/lib/torque/mom_logs/$(date +%Y%m%d) and /var/log/messages**
To see on which node the job is running, execute **qstat -n**

If the job fails, chances are that the user doesn't have an SSH Key, the home directory is not mounted on the node, or SELinux is enabled on either the node or the HN.
Also, check that pbs_mom is running on the node and pbs_server is running on the HN. On both HN and WN, execute:

```
netstat -nap |grep pbs
```

Check if the WN is available:

```
pbsnodes  wn01
```

27. Submit and manually start an MPI job that uses all nodes (HN)

An MPI job is started on a node, which becomes the primary node for the job. This primary node then forks (starts up) the jobs on the other nodes. You should note that multiple MPI jobs can run on a single node. You should also note that an MPI job can use OpenMP too. This means that a single MPI job can use all the node cores, or multiple MPI jobs can use one or more cores per job. The SSH keys must be working (See section 18) before trying to execute the next section. It must work for your standard users too.

For this test, you should become a normal user again. Here is a simple MPI Code snippet to test your MPI execution. Create a file **my_mpi.c** and add the following content:

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
/* Initialise the MPI environment */
MPI_Init(NULL, NULL);

/* Get the number of processes */
int world_size;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);

/* Get the rank of the process */
int world_rank;
MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

/* Get the name of the Processor */
char processor_name[MPI_MAX_PROCESSOR_NAME];
int name_len;
MPI_Get_processor_name(processor_name, &name_len);

/* Print off a hello world message */
printf("Hello from host %s, rank %d" " out of %d processors\n", processor_name,
world_rank, world_size);

/* Finalise the MPI environment. */
MPI_Finalize();
}
```

Now the code has to be compiled with an MPI compiler. Let's assume you installed the OpenMPI packages (openmpi and openmpi-devel) on all the nodes and HN, which installs a modules file as **/etc/modulefiles/mpi/openmpi-x86_64**. The installation of Environment modules is discussed later and should be followed if you want to use the OpenMPI packages here efficiently. For now, let's assume environment-modules is are already installed:

```
module load mpi

mpicc my_mpi.c -o my_mpi

#Now let's create a hosts file called worker nodes:
cat >  workernodes  <<EOF
wn01
wn02
EOF

#Now we can execute the code:
$(which mpirun) --hostfile workernodes -np 30 my_mpi
```

Again, it is crucial that OpenMPI is installed on all the nodes and that the user has an SSH key installed and can SSH password less to all the nodes you are using. **Also, note that the root user can't execute the mpi commands**. If the above commands work, you can continue submitting a PBS MPI job using the following template:

```
#!/bin/bash
#PBS -l walltime=00:01:00
#PBS -l nodes=2:ppn=16
#PBS -q hpc

module load mpi
cd $PBS_O_WORKDIR
cat $PBS_NODEFILE > nodes
$(which mpirun) --hostfile nodes -np 32 my_mpi
```

28. Make sure that the job executes correctly (HN)

After submitting the above test MPI job, you have to start the job (**qrun <job-id>**) and monitor the queue (**qstat -a** and **qstat -n**). Make sure that the job is running on all the nodes. If the job is not running, you can monitor the log files as described in section 23. To further debug, make sure the user has an SSH key installed. Ensure the user's home directory is accessible on all the nodes. Ensure the user can SSH to all the nodes without a password. Also, ensure the user can SSH to all the node names with the "-ib" or similar postfix chosen while configuring the cluster.

29. Install, Configure & Test Maui (HN)

Maui is used to execute jobs in the PBS Torque queues. Maui probes the queues at a set interval and checks if enough resources are available. If the available resources fulfil the job's requirements, Maui will kick off the job. The Maui License does not strictly comply with GNU and GPL licenses; therefore, you won't find it in the EPEL or CentOS repositories. We are going to install it from the source code. You will need a free registration account to be able to download Maui from the Adaptive Computing's website:
https://support.adaptivecomputing.com/wp-content/uploads/filebase/maui-downloads/maui-3.3.1.tar.gz

Here is the installation procedure to install Maui **on HN**:

```
#We need the PBS development package to build Maui
dnf install torque-devel wget tar gcc sed
dnf -y groupinstall "Development Tools" "Legacy UNIX Compatibility"

cd /tmp

#Get the Source Code:
wget http://grid.ufs.ac.za/public/maui-3.3.1.tar.gz
tar -xf maui-3.3.1.tar.gz
cd maui-3.3.1
./configure --prefix=/usr/lib/maui
make -j  4                                  #Executes the make command using 4 cores
make install
cd ..
rm -rf maui-3.3.1
echo "export PATH=\$PATH:/usr/lib/maui/bin" > /etc/profile.d/maui.sh
chmod 644 /etc/profile.d/maui.sh
```

The Maui configuration file is located at **/usr/local/maui/maui.cfg**
The default configuration file will work fine, but you can look at the file for future reference.

Now we need to make a systemd unit file so the service can startup when the service is restarted.

```
cd /etc/systemd/system
cat > maui.service <<EOF
[Unit]
Description=Maui Scheduler
Requires=network.target
After=network.target remote-fs.target

[Service]
Type=forking
User=root
PIDFILE=/run/maui.pid
ExecStart=/usr/lib/maui/sbin/maui

[Install]
WantedBy=multi-user.target
EOF

chmod 664 maui.service

systemctl daemon-reload
systemctl enable maui
systemctl start maui
```

30. Well done….. you have a simple, functional cluster…From here on, multiple tasks can be executed in unison

At this point, you should have a fully functional cluster. The "hardest" part is done now. Now you can start installing and benchmarking Scientific Software.

31. Install & Configure Apache (HN) - Optional

Apache is the most widely used Linux webserver and can easily be installed on a GNU Linux distribution. On RedHat-derived distributions such as CentOS, Scientific Linux and Fedora Core, it is installed by installing the httpd or httpd2 packages. This section can be skipped for now because ganglia-web installs the httpd package too. If you are not installing ganglia-web on the HN, you can install httpd as follows:

```
#Here is the quick and easy Apache installation method:
dnf -y install httpd
systemctl enable httpd
systemctl start  httpd
```

Remember to configure your firewall to ensure the server is accessible using port 80 and, optionally, port 443. After executing the above commands (and configuring your firewall to allow TCP connections to port 80), you should be able to see a test webpage at: http://< The_HN's_IP_address> for example, http://10.0.0.1 or on the HN itself: http://127.0.0.1

32. Install Ganglia gmetad & ganglia-web-interface (HN) - Optional

Ganglia is used to see an overall view of the whole cluster and how busy the cluster is. It is helpful to see how many resources are used per machine visually and can indicate that a node is full or that a node could use more RAM. You can install the latest version of Ganglia from their website, but following the dnf install method might be more manageable.

Execute the following to install the Ganglia Web interface on the HN:

```
dnf install ganglia-web
systemctl enable httpd
systemctl start  httpd
```

The critical config files for Ganglia-web are:

/etc/ganglia/conf.php
/etc/httpd/conf.d/ganglia.conf

In the ganglia.conf file, you will see a line like this:

```
Require local
```

This restricts the website to be only accessible from the local host. Thus you can't use a different machine to access the web interface. To enable other hosts to connect to the Ganglia web interface, remove the Require local line and change it to:

```
Require all granted
```

After modifying the configuration file(s), you have to restart the httpd service:

```
systemctl restart httpd
```

You can use a web browser and access: **http://IP-address_of_HN/ganglia**
At this point, you will see an error when you try to access the website. The error will read something like this:
There was an error collecting ganglia data (127.0.0.1:8652): fsockopen error: Connection refused
This is because the gmetad service is not configured and running on the HN.
Start the gmetad service and make it start automatically after a system reboot:

```
systemctl start  gmetad
systemctl enable gmetad
```

After executing the above, you should be able to access the website, and some NULL values should be displayed.

33. Install Ganglia gmond (HN & WNs) – Optional, required if Ganglia was installed

The Ganglia gmond service is the service that monitors and adds the system resources used to the Ganglia gmetad service. The gmetad service is probed from the Ganglia web interface and displays the resource maps to the user. Each node that has to be monitored will have to install the Ganglia gmond service:

```
dnf -y install ganglia-gmond
systemctl enable gmond
systemctl start  gmond
```

After installing the packages on the HN, I would remind you to perform a file sync first and then install it on the Worker Nodes.

After executing the above, you can refresh the web browser, and you should start to see the resources on the website: http://10.0.0.1/ganglia

34. Configure Firewalld to allow HTTP traffic and gmond traffic (HN) – Optional, required if Ganglia was installed

Remember to configure Firewalld to allow HTTP traffic from the HN to other machines and configure the Apache ganglia.conf file to give access to the IPs that can see the website. Ganglia gmond uses the multicast address of **239.2.11.71**

It may be necessary to add the following rules in your Firewalld configuration file to allow multicast packets:

```
-A FW -s 224.0.0.0/4   -j ACCEPT
-A FW -d 224.0.0.0/4   -j ACCEPT
-A FW -m pkttype --pkt-type multicast -j ACCEPT
```

35. *(In parallel) Install some Scientific Software (WNs):

From here on, you can install and configure the Scientific Software. After installing a package, create an environment module file that users load to set the path. I would recommend that you install a node where it is going to be executed. This will ensure that if an auto wizard is run and the hardware is detected, the hardware reflects the actual hardware on which the software will be executed.

The following software packages will be an excellent exercise to install before the competition. You should familiarise yourself with the installation procedures to ensure you can perform it under pressure. You can also take notes and write installation scripts that could become helpful during the competition.

a. First, install GCC; the rest of the software needs to be installed using it

This is the dirty install for gcc.
Make sure the following dependencies have already been installed:

```
dnf -y install epel-release
dnf -y groupinstall "Development Tools" "Legacy UNIX Compatibility"

dnf -y install \
  libgcc.i686  cmake fftw-devel glibc-devel.i686 \
  hwloc-devel  hwloc
```

```
Install_Version=7.2.0
Install_Destination=/soft/gcc/$Install_Version
cd /tmp

wget "http://mirror.ufs.ac.za/gnu/gcc/gcc-$Install_Version/  \
gcc-${Install_Version}.tar.gz" -O gcc-${Install_Version}.tar.gz

tar -xvf gcc-${Install_Version}.tar.gz

cd gcc-${Install_Version}
./configure \
         --prefix=$Install_Destination \
         --enable-threads \
        --enable-languages="c,c++,fortran,objc,obj-c++ " \
        --disable-multilib
make -j  &&  sudo make install

mkdir -p /soft/modules/gcc
cd /soft/modules/gcc

sudo cat > $Install_Version <<EOF
#%Module1.0
## gcc modulefile
##
proc ModulesHelp { } {
        puts stderr "\tAdds GCC C/C++ compilers ($Install_Version) to your
environment."
}
module-whatis "Sets the environment for using GCC C/C++ compilers
($Install_Version)"

set             GCC_VERSION     $Install_Version
set             GCC_DIR         $Install_Destination

prepend-path    PATH            \$GCC_DIR/include
prepend-path    PATH            \$GCC_DIR/bin
prepend-path    MANPATH         \$GCC_DIR/man
prepend-path    LD_LIBRARY_PATH \$GCC_DIR/lib
prepend-path    LD_LIBRARY_PATH \$GCC_DIR/lib64

setenv          GCC_VER         \$GCC_VERSION
setenv          CC              \$GCC_DIR/bin/gcc
setenv          GCC             \$GCC_DIR/bin/gcc
setenv          FC              \$GCC_DIR/bin/gfortran
setenv          F77             \$GCC_DIR/bin/gfortran
setenv          F90             \$GCC_DIR/bin/gfortran

#For CFLAGS, see:   https://gcc.gnu.org/onlinedocs/gcc/x86-Options.html
setenv          CFLAGS "-march=broadwell -m64"
EOF

chown -R :hpcusers    $Install_Destination  /soft/modules
```

The above commands will install GCC version 7.2.0 in **/soft/gcc/7.2.0** and create a module **gcc/7.2.0** that is loadable by the **module load** command. Follow the steps in Section 32 to make the gcc/7.2.0 available for users to load from **/soft/modules/gcc.**

b. Install OpenMPI

OpenMPI is a message parser that allows jobs to be processed on multiple nodes through the network. Although OpenMPI can be installed via dnf or other binary methods, you will need to install OpenMPI from the source code for better performance on your system. The following method can be used to install OpenMPI from Source Code:

```
module load gcc || exit 1

APP_NAME=openmpi
APP_VER=2.1.2
APP_DEST=/soft/$APP_NAME/$APP_VER
APP_TMP=/tmp/$APP_NAME/$APP_VER

[ -e $APP_TMP ] && rm -rf $APP_TMP
mkdir /soft/modules/$APP_NAME
mkdir -p $APP_TMP
cd $APP_TMP
rpm -q torque-devel > /dev/null || dnf -y install torque-devel

cd $APP_TMP
[ -e $APP_NAME-${APP_VER}.tar.gz ] || wget http://www.open-
mpi.org/software/ompi/v${APP_VER::${#APP_VER}-2}/downloads/$APP_NAME-
${APP_VER}.tar.gz
tar -zxvf $APP_NAME-${APP_VER}.tar.gz
cd $APP_TMP/$APP_NAME-${APP_VER}
./configure  \
        --prefix=$APP_DEST \
        --with-tm \
        --enable-mpi-thread-multiple

make -j && make install

cat > /soft/modules/$APP_NAME/$APP_VER <<EOF
#%Module 1.0
#
#  OpenMPI module for use with the 'environment-modules' package:
#

proc ModulesHelp { } {
        puts stderr "\\tAdds OpenMPI ($APP_VER) to your environment."
}

module-whatis "Sets the environment for using OpenMPI ($APP_VER)"

set               app_name                $APP_NAME
set               app_ver                 $APP_VER
set               prefix                  $APP_DEST
set               arch                    $(uname -m)

module load gcc

prepend-path      PATH                    \$prefix/bin
prepend-path      LD_LIBRARY_PATH         \$prefix/lib
prepend-path      INCLUDE                 \$prefix/include
prepend-path      MANPATH                 \$prefix/share/man
prepend-path      PKG_CONFIG_PATH         \$prefix/lib/pkgconfig
prepend-path      PYTHONPATH              /usr/lib64/python2.7/site-
packages/\$app_name

setenv            MPI_VER                 \$app_ver
setenv            MPI_HOME                \$prefix
setenv            MPI_SYSCONFIG           \$prefix/etc
setenv            MPI_BIN                 \$prefix/bin
setenv            MPI_INCLUDE             \$prefix/include
setenv            MPI_LIB                 \$prefix/lib
setenv            MPI_MAN                 \$prefix/share/man
setenv            MPI_FORTRAN_MOD_DIR     /usr/lib64/gfortran/modules/\${app_name}-
\$arch
setenv            MPI_PYTHON_SITEARCH     /usr/lib64/python2.7/site-
packages/\$app_name
setenv            MPI_COMPILER            \${app_name}-\$arch
setenv            MPI_SUFFIX              _\${app_name}

EOF
```

c. Installing FFTW3

FFTW is a scientific library used by multiple software packages such as Gromacs, LAMMPS etc. We must compile a single precision version of FFTW to get optimal performance. By default, a double-precision version is compiled that does not support MPI threads either. The following setup will compile FFTW3 for us, with thread support via OpenMP and multi-host execution via OpenMPI.

```
APP_NAME=fftw
APP_VER=3.3.7
APP_DEST=/soft/$APP_NAME/$APP_VER
APP_MODULE=/soft/modules/$APP_NAME/$APP_VER

module load gcc
module load openmpi

export MPICC=$(which mpicc)
export LDFLAGS="-L$MPI_LIB"
export CPPFLAGS="-I$MPI_INCLUDE"
export MPILIBS="-lmpi"

cd /tmp
wget http://www.fftw.org/fftw-3.3.7.tar.gz
tar -xf $APP_SRC
cd /tmp/$APP_NAME-$APP_VER
make clean

CONFIG_OPTIONS=" \
        --prefix=$APP_DEST \
        --enable-shared \
        --enable-openmp \
        --enable-mpi \
        --enable-fma \
        --enable-sse2 \
        --enable-avx \
        --enable-avx2 \
        --enable-avx-128-fma"

#Configure and compile Double precision version
./configure \
        $CONFIG_OPTIONS

make -j 16   &&   make install

#Configure and compile the single-precision version
make clean
./configure \
        $CONFIG_OPTIONS \
        --enable-sse \
        --enable-single
make -j 16   &&   make install

mkdir $(dirname $APP_MODULE)
```

The next section will generate the required module file. You must execute this section in the same shell as the previous steps.

```
cat > $APP_MODULE <<EOF
#%Module1.0
## $APP_NAME modulefile
##
proc ModulesHelp { } {
        puts stderr "\tAdds $APP_NAME ($APP_VER) to your environment."
}
module-whatis "Sets the environment for using $APP_NAME ($APP_VER)"

module load openmpi
module load gcc

set             ver              $APP_VER
set             dir              $APP_DEST

setenv          FFTW_DIR        \$dir
setenv          FFTW_VER        \$ver

prepend-path    PATH            \$dir/bin
prepend-path    MANPATH         \$dir/share/man
prepend-path    LD_LIBRARY_PATH \$dir/lib
prepend-path    PKG_CONFIG_PATH \$dir/lib/pkgconfig

EOF
```

d. Installing Gromacs

Have a look at: http://www.gromacs.org/Documentation/Performance_checklist

In this example, you will see that we use the GCC and OpenMPI modules. It is essential to first install GCC like discussed above, and then install openmpi-devel before continuing.

The code section below will allow you to install a basic compiled version of Gromacs. Note that there are some optimisation options, such as the -DGMX_SIMD option, that need to be set according to your CPU's optimal optimisation flags:
cat /proc/cpuinfo | grep flags | tail -n 1
If your Processor supports a higher level of optimisation, such as AVX, AVX2, AVX128, AVX256, AVX512 etc., make sure to use those.

As mentioned before, I would compile my own GCC, LAPACK and FFTW; create and load their module files before continuing.

```
Install_Ver=2016.4
Install_Dest=/soft/gromacs/$Install_Ver
Install_Src= gromacs-${Install_Ver}


module load mpi
module load fftw

cd /tmp
wget ftp://ftp.gromacs.org/pub/gromacs/${Install_Src}.tar.gz \
       -O ${Install_Src}.tar.gz

tar -xvf ${Install_Src}.tar.gz
cd /tmp/${Install_Src}
[ -e my_build ] && rm -rf my_build
mkdir my_build && cd my_build
cmake ../ \
         -DCMAKE_INSTALL_PREFIX=$Install_Dest \
         -DGMX_MPI=ON \
         -DGMX_OPENMP=ON \
         -DGMX_THREAD_MPI=OFF \
         -DGMX_OPENMP_MAX_THREADS=32 \
         -DFFTWF_INCLUDE_DIR=$FFTW_DIR/include \
         -DFFTWF_LIBRARY=$FFTW_DIR/lib/libfftw3f.so \
         -DCMAKE_C_FLAGS="-march=broadwell" \
         -DGMX_SIMD=AVX2_256

make -j  &&   sudo make install

#Now we create the module file for Gromacs
sudo mkdir -p /soft/modules/gromacs

sudo cat > /soft/modules/gromacs/$Install_Ver <<EOF
#%Module1.0
## gromacs modulefile
##
proc ModulesHelp { } {
        puts stderr "\tAdds Gromacs($Install_Ver) to your environment."
}
module-whatis "Sets the environment for using Gromacs($Install_Ver)"

module load mpi
module load fftw

set             GMX_VERSION     $Install_Ver
set             GMX_DIR         $Install_Dest

setenv          GMX_SUFFIX      _mpi
setenv          GMXLDLIB        \$GMX_DIR/lib
setenv          GMXBIN          \$GMX_DIR/bin
setenv          GMXMAN          \$GMX_DIR/share/man
setenv          GMXDATA         \$GMX_DIR/share
setenv          GMXLIB          \$GMX_DIR/share/gromacs/top

prepend-path    PATH            \$GMX_DIR/bin
prepend-path    MANPATH         \$GMX_DIR/share/man
prepend-path    LD_LIBRARY_PATH \$GMX_DIR/lib64
EOF

chown -R :hpcusers    $Install_Dest  /soft/modules
```

After installing Gromacs, you can download and test the installation using the following method:

```
#Execute the following as a regular user:

module load gromacs

#Go to your home directory
cd

#Download the example:
wget http://grid.ufs.ac.za/public/examples/gromacs_test.tar.gz

tar -xvf gromacs_test.tar.gz
cd gromacs_test

#There are two scripts 00prepare_input.sh and 01submit.pbs
#The first script that generates the input files can be executed as:
#     ./00prepare_input.sh
#After the script ran (about 1.5 hours on a VM), the input files should be
#generated
#You can submit the script 01submit.pbs


#Modify the script to reflect the number of nodes etc., before submitting.
#On three virtual machines with 16 cores each, this runs for about 25 mins.
qsub 01submit.pbs
```

Although the above **qsub** command will most likely submit the job to the queue, you must ensure the job is running effectively. Check the generated log files and see if more optimisation can be done.

e. LAMMPS + It's environment module

f. WRF + It's environment module

g. HPC Challenge

h. OpenFoam + It's environment module

i. Bonus: Install HPC Challenge Benchmark + environment module

36. *Submit test jobs for each Scientific/benchmarking Software stack and ensure it runs on multiple nodes (HN)

Summary

The following packages can be installed on the WNs and HN/SN from the get-go. This will allow you to perform multiple configuration options discussed above with more ease:

```
#You first have to install the EPEL Release package for other content
dnf -y install epel-release

#Now you can install the following Groups of software:
dnf -y groupinstall "Development Tools" "Legacy UNIX Compatibility"

#If the above command failed, type:
dnf -y group install --setopt=group_command=objects  "Development Tools" \
    "Legacy UNIX Compatibility"

#Now, some tools that we might want to use later on
dnf -y install \
        nfs-utils      firewalld            net-tools       \
        vim            nano                 htop            numactl \
        torque-client  torque-mom           munge           ganglia-gmond \
        openmpi-devel  environment-modules  wget            elinks \
        dos2unix       unix2dos             hwloc           hwloc-devel


#Finally, some development packages:
dnf -y install \
        libxml2-devel   boost-devel          kernel-devel   gtk2-devel \
        libgcc.i686     fftw-devel           glibc-devel.i686          \
        glibc.i686      libgcc.i686          libstdc++-devel.i686
```

If you install Scientific Software, many will use libraries from Boost, LAPACK, ScaLAPACK, FFTW etc. It is usually better to install your own versions of these packages and configure your environment (Setting LD_LIBRARY_PATH, PATH, INCLUDES, LIBDIR etc.) before compiling the Scientific Software. After the software is compiled, you must set the environment each time a user wants to use the software. Therefore, I recommend using the Environment Modules package and setting the environment there.