High Performance Computing Cluster Setup Manuel

OS

Convert SDD to non-RAID!!!
When you go into BIOS to change boot order to install in OS, also **set the BIOS to performance mode**

Network Setup Initial

Type “`nmcli d`” command in your terminal for quick list of ethernet cards installed on your machine
Type “`nmtui`” command in your terminal to open Network manager. After opening Network manager chose “Edit connection” and press Enter (Use TAB button for choosing options).
Now choose you network interfaces and click “Edit”
Choose “Automatic” in IPv4 CONFIGURATION and check Automatically connect check box and press OK and quit from Network manager.
Restart network services: `service network restart`

Make sure nmtui is done by now and never do it again. It changes the network scripts
`/etc/sysconfig/network-scripts` for the files for the ethernet ports

Edit the network scripts on all the nodes, for ifcfg-enp8s0f0 and ifcfg-enp8s0f1, found in:

```
      /etc/sysconfig/network-scripts
```

Proxy (for wits, obviously the details will change depending on location):

The following lines should appear in the .bashrc file (these are for wits)

```
export
http_proxy='http://students\12345:password@proxyss.wits.ac.za:80’ in
.bash_profile
export
https_proxy='https://students\12345:password@proxyss.wits.ac.za:80’ in
.bash_profile
export
HTTP_PROXY='http://students\12345:password@proxyss.wits.ac.za:80’ in
.bash_profile
export
HTTPS_PROXY='https://students\12345:password@proxyss.wits.ac.za:80’ in
.bash_profile
```

Craig’s guide:

```
            export HTTP_PROXY=$http_proxy
            export HTTPS_PROXY=$https_proxy
```

```
source .bashrc
```

Stop the firewall on all the nodes:

```
systemctl disable firewalld
systemctl stop firewalld
systemctl status firewalld
```

<u>Adds a static route:</u>

ip route add destination_node_connected_port dev port_connected_to_that_node_from_local_node
Eg:

```
ip route add 10.0.0.92 dev enp8s0f0
```

<u>To add a gateway (CIDR Notation):</u>

ip route add category_of_IP_addresses_to_use_gateway via IP_of_port_on_connected_node_that_local_node_is_sending_traffic_through
Eg:

`ip route add 10.0.0.0/24 via 10.0.0.92` (10.0.0.93 being source node ip sending traffic through 10.0.0.92)

```
systemctl restart network
```

Eg for beyonce to kelly:

```
    ip route add 10.0.0.91 dev enp8s0f1
```

Eg for kelly to beyonce:

```
    ip route add 10.0.0.90 dev enp8s0f0
    ip route add 10.0.0.0/24 via 10.0.0.90
Eg for Kelly to michelle
    ip route add 10.0.0.93 dev enp8s0f1
Eg for michelle to Kelly:
    ip route add 10.0.0.92 dev enp8s0f0
    ip route add 10.0.0.0/24 via 10.0.0.92
    ip route add 10.0.0.90 via 10.0.0.92
Eg beyonce to michelle:
    ip route add 10.0.0.93 via 10.0.0.91
```

We need to setup the following:

-beyonce connect to Kelly
-kelly connect to beyonce
-kelly connect to michelle
-michelle connect to Kelly
-beyonce special gateway to michelle
-michelle special gateway to beyonce

---Explanation of gateway ip rule:

Destination is where we are going, gateway is how you are going to get there, so we'd have to go through kelly to get to michelle from beyonce. You must add all special ip addresses into the iptables.

---Explanation of /24 in gateway command:

The /24 indicates the amount of bits variance that is allowed in the IP addresses for them to still use this gateway. So, in this case the first 24 bits are fixed while the last 8 bits may vary (32 bits in an IP). Visually: xxxx.xxxx.xxxx.0000 where x's are fixed and 0's may vary.
IP addresses are broken up into 4 parts of 8 bits representing integers between 0 and 255 (highest number 8 bits can represent). The netmask can be interpreted as the amount of restrictions placed on the IP address. So, 255.255.255.255 means only 1 address can be used, or only one address will pass through that gateway. 0.0.0.0 places no restrictions and thus any range of IP's will pass through the gateway. In finality, a gateway of 10.0.0.0/24 will support a range of IP addresses from 10.0.0.1 to 10.0.0.255 and the netmask should be 255.255.255.0

<u>To delete a gateway:</u>

ip route del connection

Eg:

```
ip route del 10.0.0.93
```

<u>IP Tables and NAT</u>

To show routing table: `ip route` OR `route -n` (net-tools needs to be installed)

Make sure firewalld isn't running:

```
systemctl disable firewalld
systemctl stop firewalld
systemctl status firewalld
```

Flush IP-tables:

```
iptables --flush
iptables --flush -t nat
```

`sysctl -w net.ipv4.ip_forward=1`

`vi /etc/sysctl.conf` and add (only in nodes that need to forward incoming connections):

```
net.ipv4.ip_forward = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.all.disable_ipv6 = 1
sysctl -p /etc/sysctl.conf
```

(Enables the changes)

`vi /etc/resolv.conf` and make sure it says:

```
search ms.wits.ac.za
nameserver 146.141.8.16
nameserver 146.141.21.2
nameserver 146.141.15.210
nameserver 146.141.15.222
```

`scp /etc/resolv.conf root@node1:/etc/`	(This is the DNS server list that was generated on headnode that needs to be copied to compute nodes so that they can access servers by their names)

This will probably change if you're not at wits. It should be generated by dhclient-script

To forward data through nodes (only to be done in nodes actually forwarding data):
iptables -t nat -A POSTROUTING -o port_of_outgoing_connection_to_forward_through -j MASQUERADE
Eg:

```
iptables -t nat -A POSTROUTING -o enp8s0f0 -j MASQUERADE
iptables-save
```

To allow through firewall- only if firewall is running:
iptables -A FORWARD -i port_of_incoming_connection_to_forward_from -n state
Eg:

```
iptables -A FORWARD -i enp8s0f0 (compute node) -n state
```

`iptables --flush`	to clear ip tables
`iptables -t nat –flush`	to clear nat ip tables
`iptables -L -t nat`	to view nat routing tables
`iptables -L`	to view normal routing tables
`iptables-save`	saves routing tables

Copy `/etc/resolv.conf` file from headnode into compute nodes (Instead of doing dns):

```
scp /etc/resolv.conf root@compute_nodes:/etc/
```

<u>If you are restarting one node:</u>

There will be a problem with ssh'ing into the restarted node. The solution is to delete the known hosts entry for the restarted node in the two other nodes:
`vi .ssh/known_hosts` and delete both the nickname entry and actual ip address entry for the restarted node

<u>Changing Hostname:</u>

`hostname "name"` Eg: `hostname beyonce` (without inverted commas)
`vi /etc/sysconfig/network` and add `HOSTNAME="name"` entry (without inverted commas)
`vi /etc/hostname` and type hostname (actual name of node, eg: beyonce in beyonce) by itself and remove the `localhost.localdomain`
`vi /etc/hosts` and add all hosts and all corresponding ports of this cluster into this file (internal and external ports. Some names will be registered to a few ports)
Adding to the `/etc/hosts` file:

```
10.0.0.90 beyonce
```

```
10.0.0.91 kelly
10.0.0.92 kelly
10.0.0.93 michelle
```

<u>To setup passwordless ssh (repeat for every node moving the key to all other nodes):</u>

1)Generate key: `ssh-keygen -t rsa`
2) `ssh-copy-id -i ~/.ssh/id_rsa.pub root@"ip-address"` to copy the key to the other nodes. Thus, you should generate 3 keys and copy 6 times for a 3-node cluster.

<u>NFS:</u>

`yum install nfs-utils` on all nodes
`setsebool _p use_nfs_home_dirs 1` (This gives permissions for using nfs folder (run on startup every time))

`systemctl enable and start rpcbind`	(1)
`systemctl enable and start nfs-server`	(2)
`systemctl enable and start nfs-lock`	(3)
`systemctl enable and start nfs-idmap`	(4)

Make a directory for mounting (probably in root, call nfsmount): `mkdir /nfsmount`
`vim /etc/exports` and add the directory for your mount (only on headnode), so add `/nfsmount`:

```
      /nfsmount  *(rw,async,insecure,no_root_squash)
```

`systemctl restart nfs-server`
`chmod 777 /nfsmount` (In headnode give everyone permissions for the directory)

Then go to compute node:

And re-run (1) to (4) on compute node
Make directory to mount (probably also `/nfsmount`)
`vim /etc/fstab` and add the line:

```
   headnode:/nfsmount /nfsmount  nfs   rw,sync,hard,intr    0 0
```

(headnode:/nfsmount is directory of mounted file in head)
(/nfsmount is the directory of the mounted folder on the compute node)
(use tabs between parts except between the two 0's)

Restart `nfs-server` in headnode:

```
systemctl restart nfs-server
```

`mount -a` (mounts everything in `/etc/fstab` (all the directories that are mounted))

`df` (lists all mounted devices).

NFS essentially makes the mounted directory act as a hard drive on the other node (works both ways. So, the mounted directory becomes common on both nodes)

If you get a stale mount:

```
systemctl restart nfs-server  #on headnode
systemctl restart nfs  #on headnode
mount -a        #on compute nodes
```

Mounting an NTFS Drive:

dmesg (Shows the connected usb devices at the end of its printout)

`yum install epel-release` (Install EPEL) (Must be installed before ntfs-3g)

`yum install ntfs-3g` (Install the ntfs-3g driver package)

`mkdir /mnt/win` (create a directory where the NTFS drive shall be mounted)

`df -h` (Shows connected devices)

`mount -t ntfs-3g /dev/sdb1 /mnt/win` (mount the NTFS partition)

Note: In this example the NTFS partition is the device /dev/sdb1, you have to replace that with the device name of your NTFS partition.
The mount point will exist until reboot or until you unmount it

`umount /mnt/win` (Unmount drive, specifies directory where the device is mounted)

To mount the NTFS partition permanently:
vim /etc/fstab
Add the following line to fstab:

```
/dev/sdb1 /mnt/win ntfs-3g defaults 0 0
```

Again changing /dev/sdb1 to the name of your device on the system

Mounting a USB:

`dmesg` (To see list of connected devices)

`mount /dev/sdb1 /mnt` (Change /dev/sdb1 to the name of your device on your system)

`fdisk -l`

`umount /mnt` (unmount a usb)

Creating Users (While as root user)

`adduser` username

`passwd` username (You'll be prompted to enter the password twice)

`usermod -u UID username`	(To change user UID, they must correlate on all nodes per user, `UID` is where you enter the value, 0 to 99 is reserved for the system)
`gpasswd -a username wheel`	(To grant sudo privileges)
`sudo lid -g wheel`	(To show a list of users with sudo privileges)
`userdel username`	(To delete a user)
`userdel -r username`	(To delete a user along with all their files)
`su username`	(switch users)

Installing Intel Compiler (only on headnode, share to compute nodes by nfs sharing `/opt`):

Download from http://registrationcenter-download.intel.com/akdlm/irc_nas/tec/12374/parallel_studio_xe_2018_update1_cluster_edition.tgz
Serial number:

```
cd Intel\ compiler/
```

```
tar -xvzf parallel_studio_xe_2018_cluster_edition.tgz
```

```
cd parallel_studio_xe_2018_cluster_edition
```

`./install.sh` (Will install in `/opt` if you use the default)

`source parallel_studio_xe_2018.0.033/psxevars.sh` (add into .bashrc as well, with its full file path in opt)

```
mpicc
```

Setting up compiler after installation

In compute nodes and on headnode install the following:
```
yum install gcc5 gcc gtk kernel-devel kernel-PAE-devel libgtk-3-dev
libgtk gtk+-devel gtk2-devel wget
```

```
vim .bashrc
```
Add the following to bashrc if it's hasn't already been added:
```
      source /opt/intel/parallel_studio_xe_2018.0.033/psxevars.sh
```

#Run `hpcc-1.5.0/hpl`

HPCC

Helpful Links:

http://wiki.chpc.ac.za/acelab:hpcc	CHPC installation guide

http://icl.cs.utk.edu/hpcc/ HPCC Website
http://hpl-calculator.sourceforge.net/ Theoretical Peak Performance Calculator
https://software.intel.com/en-us/articles/performance-tools-for-software-developers-hpl-application-note THIS ONE IS GOOD!

Downloading the source:

```
    cd ~
    mkdir HPCC
    cd HPCC
    wget http://icl.cs.utk.edu/projectsfiles/hpcc/download/hpcc-
    1.5.0a.tar.gz
    tar -xf hpcc-1.5.0a.tar.gz
    cd hpcc-1.5.0a
```

```
cd HPCC/hpcc-1.5.0a/hpl/setup
```

Use the Makefile which is adapted from the HPL which ships with the Intel MKL package:

```
   cp Makefile.LinuxIntelIA64Itan2_eccMKL ~/HPCC/hpcc-1.5.0a/hpl

   #cp Makefile.intel64 ~/HPCC/hpcc-1.5.0a/hpl
```

Make with:

```
    make arch=LinuxIntelIA64Itan2_eccMKL

    #make arch=intel64
```

`mv Makefile.LinuxIntelIA64Itan2_eccMKL Makefile.destiny` (rename make file)

```
vim Makefile.destiny
```

Don't edit anything until `LAdir`
Then make the following changes:

Point the math library MKL:

```
   LAdir = = /opt/intel/compilers_and_libraries_2018/linux/mkl
   LAinc = = -I$(LAdir)/include
   LAlib = = -L$(LAdir)/lib/intel64 -mkl
```

Change the compiler information to use intel compiler:

```
   CC           = mpiicc
   CCNOOPT      = $(HPL_DEFS)
   CCFLAGS      = = $(HPL_DEFS) -std=c99 -fomit-frame-pointer -O3 -
   funroll-loops -W -Wall
```

(With an o, the capital letter, in O3)

```
   LINKER       = mpiicc
   LINKFLAGS    = $(CCFLAGS)
```

```
mv _hpccinf.txt hpccinf.txt
```
(Remove underscore in front)

```
make arch=destiny clean
make arch=destiny
```

NFS mount the `hpcc-1.5.0` directory across the whole cluster

Benchmarking:

Before running the benchmark, edit the hpccinf.txt file. Set appropriate N, NB, P and Q values (using the calculator given above try get as close to peak performance as possible)

Run the benchmark using:

```
mpirun -np <N> -hostfile <HF> ./hpcc
```

Where <N> is the number of cores available to the system and <HF> is your hostsfile

Eg:

```
mpirun -np 4 hpcc
mpirun -np 24 -hosts node0,node1,node2 ./hpcc
```
to run the full benchmark

Add hostnames to the hostfile:

In hpcc directory (probably in /nfsmount)

```
vim hostfile
```

add name of hosts (beyonce, kelly, michelle) on their own line underneath each other

Formatting the output:

Download the tar file to format the output from the link:
http://wiki.chpc.ac.za/lib/exe/fetch.php?tok=8dd21b&media=http%3A%2F%2Fwiki.chpc.ac.za%2F_media%2Facelab%3Aformat.tar

Run:

```
  ./format.pl -w -f hpccoutf.txt
```

To give the output from `hpccoutf.txt` (the results from the benchmark)in the correct format

HPCC Key:

N=problem size

Ps

Qs

NB = determine block size

P and Q close together (try make the matrix square)

If you get RLIMIT MEMLOCK TOO SMALL:

```
ulimit -l unlimited
vim /etc/security/limits.conf:
```

add the following two lines

```
        *     soft  memlock    unlimited
        *     hard  memlock    unlimited
```
Using tabs in the commands above

Helpful commands:
```
tail -n 200 "filename"| less
```
to see output from files

To check if the benchmark is running on all nodes:

run the benchmark in the background by using `&` at the end of the command to run it

`ssh` into each node

`top` (the process should be running in each node)

WRF (Put in home, not in root) (Try option 15 instead of option 20):

Useful Links:

http://www2.mmm.ucar.edu/wrf/OnLineTutorial/compilation_tutorial.php

http://www.hpcadvisorycouncil.com/pdf/WRF_v3.8%20Installation_Best_Practices.pdf

http://wiki.chpc.ac.za/acelab:wrf (CHPC Guide)

The second link includes how to compile with pnetcdf (which may be slower. Using pnetcdf instead of netcdf may or may not be useful. If you have time compile both and compare performance with both and choose whichever performs better.)

Additional packages need to be installed:
```
yum install m4 csh perl perl5 -y
```

Add the following environment variables to .bashrc for WRF:
```
vim ~/.bashrc
export CC=icc
export CXX=icpc
export CFLAGS='-O3 -xHost -ip -no-prec-div -static-intel'
export CXXFLAGS='-O3 -xHost -ip -no-prec-div -static-intel'
export F77=ifort
export FC=ifort
export F90=ifort
export FFLAGS='-O3 -xHost -ip -no-prec-div -static-intel'
export CPP='icc -E'
export CXXCPP='icpc -E'
export DIR=~/WRF/libs
export PATH=$DIR/netcdf/bin:$PATH
export NETCDF=$DIR/netcdf
export LDFLAGS=-L$DIR/grib2/lib
export LD_LIBRARY_PATH=$DIR/grib2/lib:$LD_LIBRARY_PATH
export CPPFLAGS=-I$DIR/grib2/include
export JASPERLIB=$DIR/grib2/lib
export JASPERINC=$DIR/grib2/include
export WRFIO_NCD_LARGE_FILE_SUPPORT=1
```

```
    export PATH=~/WRF/WPS:$PATH
    export PATH=~/WRF/WRFV3/run:$PATH
    export DM_FC=mpiifort
    export DM_CC=mpiicc
```

Make sure the exported path or directory in `.bashrc` is the correct directory of the WRF libs:

```
    Export DIR=/home/$USER/WRF/libs
```

for example, should be used if WRF is in `/root/` this could be different, so you need to check it in relation to the current system

make sure we have libstdc++, gfortran and gcc (can check by running which gcc):

```
    yum install libstdc++ gfortran gcc
    yum install gcc-c++ zlib libpng
```

Set up the directory:

```
    mkdir WRF
    cd WRF
    mkdir libs
    mkdir tars
```

`vim .bashrc` and make sure that `source parallel_studio_xe_2018.0.033/psxevars.sh` appears in it

```
source .bashrc
```

Jasper Setup:

```
    cd ~/WRF/tars

    wget
    http://www2.mmm.ucar.edu/wrf/OnLineTutorial/compile_tutorial/tar_
    files/jasper-1.900.1.tar.gz

    tar -xf jasper-1.900.1.tar.gz

    cd jasper-1.900.1

    ./configure --prefix=/home/$USER/WRF/libs/grib2

    make -j20

    make -j20 install
```

NetCDF Setup:

```
    cd ~/WRF/tars

    wget
    http://www2.mmm.ucar.edu/wrf/OnLineTutorial/compile_tutorial/tar_

    files/netcdf-4.1.3.tar.gz
```

```
tar -xf netcdf-4.1.3.tar.gz

cd netcdf-4.1.3

  ./configure --prefix=/home/$USER/WRF/libs/netcdf --disable-dap
--disable-netcdf-4 --disable-shared
```
(May have to remove shared flag)

```
make -j20
```
(Compiles on all 12 cores, number changes depending on number of cores)

```
make -j20 check

make -j20 install
```

WRF Setup:

```
cd ~/WRF/tars

wget http://www2.mmm.ucar.edu/wrf/src/WRFV3.7.TAR.gz

tar -xf WRFV3.7.TAR.gz

mv WRFV3 ..

cd ../WRFV3

./configure
```

You will be prompted to select config options for WRF. NetCDF should be found from environment variables set above:

for Linux x86_64 options: select 20 (may be 15, hardware dependent),
then option 1 for compile nesting

After the configure step is complete, build the code with:

```
./compile -j20 em_real(space or no space it does not matter
for j20)
```

WPS Setup:

```
cd ~/WRF/tars

wget http://www2.mmm.ucar.edu/wrf/src/WPSV3.7.TAR.gz

tar -xf WPSV3.7.TAR.gz

mv WPS ..

cd ../WPS

./configure
```

Select option 17: Linux, with Intel, serial /w grib2

```
./compile
```

```
        ln -sf ungrib/Variable_Tables/Vtable.GFS Vtable
```

Benchmarking:

```
        cd ~/WRF/tars
```

Copy the input data from:

http://www.ace.chpc.ac.za/tars/GEOG.tar.gz

```
        wget http://www.ace.chpc.ac.za/tars/GEOG.tar.gz

        tar -xf GEOG.tar.gz

        mv GEOG ..

        cd ../GEOG

        cp namelist.wps ../WPS

        cd ~/WRF/tars
```

Copy more input data from

http://www.ace.chpc.ac.za/tars/DATA.tar.gz

wget http://www.ace.chpc.ac.za/tars/GEOG.tar.gz

```
        tar -xf DATA.tar.gz

        mv DATA ..

        cd ../WPS

        ./link_grib.csh ../DATA/
```

Build the model from the geographical data:

```
                ./geogrid.exe
                ./ungrib.exe
                ./metgrid.exe
```

Set up the benchmark:

```
                cd ~/WRF

                cp GEOG/namelist.input WRFV3/run

                cd WRFV3/run
```

```
ln -sf ../../WPS/met_em.d01.2015* .
```
(With the full stop at the end)

Final setup step:

```
ulimit -s unlimited
```
(makes stack size unlimited, stops segfaults)

If setting unlimited stack size above doesn't fix segfaults, add the following to .bashrc:
(This is optional, don't try initially with this)

```
vim ~/.bashrc
export OMP_STACKSIZE=64000000
```

It may not have made a difference but if problems persist then comment out all the environment variables with "`static`" in them in .bashrc

```
./real.exe
```

Run the benchmark:

```
mpirun -np <N> -hostfile <HF> ./wrf.exe
```

Where <N> = Number of cores to run the benchmark on
<HF> = hostfile (can also use `-hosts node0,node1,node2` in place of hostfile)

If the model segfaults at the very beginning of the run it is likely that there is a problem with the input data.

If you run into problems try:

```
./clean -a
./configure
```

Choose 15 (as opposed to 20)

```
./compile em_real
```

Recompile WPS

If program doesn't build properly:

```
cd WRFV3
```

`vi configure.wrf` and take out the word "time" in `FC=`

Note:

The dx and dy in WPS and WRFV3 have to correlate/match
Try removing all optimization flags and possibly using gcc instead of the Intel compiler if all else fails.

LAMMPS

`yum install libjpeg-turbo-devel` (must be performed on all nodes)

Setup directory:

```
    mkdir LAMMPS
    cd LAMMPS
    mkdir tars
```

`vim .bashrc` and make sure that `source parallel_studio_xe_2018.0.033/psxevars.sh` appears in it

```
source ~/.bashrc

cd tar
```

Building the benchmark:

Download the LAMMPS source from: http://www.ace.chpc.ac.za/tars/lammps-stable.tar.gz

```
wget http://www.ace.chpc.ac.za/tars/lammps-stable.tar.gz
```

`yum install libjpeg-turbo-devel` (If it says we need a .h file we need the devel)

```
tar -xf lammps.stable.tar.gz

mv lammps-15May15 ..

cd ../lammps-15May15/
```

Edit the makefile:

```
cd src
cp MAKE/OPTIONS/Makefile.intel_cpu MAKE/Makefile.intel_cpu
vim MAKE/Makefile.intel_cpu
```

Edit `FFT_INC` to be:

```
FFT_INC =         -DFFT_MKL
```

Also edit `LINKFLAGS` and `CCFLAGS`, make `openmp` into `qopenmp` in the top few lines.

For `CCFLAGS`, change `-no-offload` to `-pno-offload` and might need to edit `override` to `qoverride`

```
vim MAKE/Makefile.intel_cpu
```

Set build parameters:

```
Make yes-opt
make yes-user-intel
make yes-user-omp
make intel_cpu
```

or use `make intel_cpu -j12` to compile in parallel with 12 cores

The lmp_intel_cpu binary should be produced. (That's an L in lmp)

```
cp lmp_intel_cpu ../bench
```

Add into .bashrc:

`Export OMP_NUM_THREADS=12` (This may break the other benchmarks)

Benchmarking:

The stock “3d Lennard-Jones melt” test problem is used as a benchmark for this code. A fixed number of particles/core is used for the problem size.

```
cd ~/LAMMPS/lammps-15May15/bench
```

Get the input script from
http://wiki.chpc.ac.za/lib/exe/fetch.php?tok=35559e&media=http%3A%2F%2Fwiki.chpc.ac.za%2F_media%2Facelab%3Acpu.tar.gz

```
wget
http://wiki.chpc.ac.za/lib/exe/fetch.php?tok=35559e&media=http%3A%2F%2Fwiki.chpc.ac.za%2F_media%2Facelab%3Acpu.tar.gz
```

For x86 CPU benchmarks, this is **500K particles per core.** Therefore, if you with to run the benchmark on 24 x86 cores, a total of 12,000K (500×24) particles is required.

To run the benchmark use:

```
mpirun -np <N> -hostfile <HF> ./lmp_intel_cpu -sf intel -v
x <X> -v y <Y> -v z <Z> -v t 100 < in.lj 2>&1 | tee
output.txt
```
(have to submit output.txt)

where: <N> is the number of cores, <HF> is the hostfile amd <X>, <Y> and <Z> are the problem scaling factors - used to reach 500K particles/core. The benchmark has been pre-configured to operate using 500K particles. Therefore, the X,Y and Z values are using to scale the number of particles up - for more cores. In order to run the benchmark on more cores simply scale X,Y and Z accordingly, such that their product equals the number of cores desired.

For example, running on 4 nodes with 24 cores each, the run command would be:

```
    mpirun -np 96 -hostfile hosts ./lmp_intel_cpu -sf
    intel -v x 6 -v y 4 -v z 4 -v t 100 < in.lj
```

Giving a total particle count of 500K * (6*4*4) = 48×10^6. Which conforms to the 500K particles per core, (48×10^6 / 96 = 500K).

If you get : “Error: rlimit memlock too small”:

in each node go to `/etc/security/limits.conf` and append

```
*       hard      memlock           unlimited
*       soft      memlock           unlimited

*       hard      stack           unlimited
*       soft      stack           unlimited
```

```
ulimit -a          (Displays current limits)
ulimit -l unlimited     (That's an L for -l)
ulimit -s unlimited
ulimit -l          (To check size of -l limit)
ulimit -s          (To check the size of the limit of the stack)
```

TMUX

start new:
```
tmux
```
start new session with name:
```
tmux new –s myname
```
attach:
```
tmux a
```
attach to named:
```
tmux a –t myname
```
list sessions:
```
tmux ls
```
kill session:
```
tmux kill-session –t myname
```

Create new windows:
```
Ctrl-b  c
```
Shift to next window:
```
Ctrl-b n
```
Shift to previous window:
```
Ctrl-b p
```
List windows:
```
Ctrl-b w
```
Split screen vertically:
```
Ctrl-b %
```
Split screen horizontally:
```
Ctrl-b "
```
Make scrollable:
```
Ctrl-b [
```

Useful Commands:

For when we are copying or removing a file or directory and we do not wish to keep pressing “y”

Use: `rm –r –f` … (what we are trying to remove)

-r Flag = Directory

-f Flag = Force

To copy from directory to directory:
```
scp “file” “host”:”file”
```