Consulting Fire Engineers

34 Satara Crescent
Khandallah
Wellington, 6035
New Zealand

Linux Beowulf Cluster
Infiniband Upgrade

for

Fire Dynamics Simulator (FDS)

12 September 2016
Revision 1.4
T. G. O'Brien

DDI: 001164 (0)4 479 3963 Mobile: 001164 (0)27 6413111 fire.aquacoustics.biz

Table of Contents

1 Document Status

The procedures in this document produce a verified FDS 6.5.2 installation on a Beowulf Linux Cluster.

The installation is verified and fully stable for FDS simulation using both OpenMPI and OpenMP.

The speed improvement achieved through the Infiniband upgrade is somewhat model dependant however our internal benchmarks indicate a 10 to 30% reduction in processing time from our earlier Gigabit Ethernet.

Further minor improvements are expected with OpenMPI tuning.

2 Infiniband Upgrade for FDS

A word of warning. I am a 'Mainly Dangerous' when it comes to Linux and networks. The following upgrade procedure may not be efficient, optimized or universal but it was effective for FireNZE's Beowulf Linux cluster.

I am hoping that the 'Elite' Linux and networking guru's out there will find fault with this procedure and send me critical comment so it can be improved for others that don't want to mess with installation and just want to get on with applying FDS to real fire engineering challenges.

This installation note is not about how to build and run Fire Dynamics Simulator (FDS) models, or how to interpret the output. If you are wanting to learn about FDS then you probably don't want to be starting here. The FDS Users Manual might be a better place to start.

Much of the following has been pieced together from .www searches, reading package documentation (and in particular for FDS, ssh, openmpi and Mellanox Infiniband), studying the Linux man (manual) for specific commands, with assistance from the folk at NIST, and often through a process of trial and error.

Nothing here is new or unique but maybe having it all in one place will provide you with a starting point from which you can develop an appropriate procedure to upgrade your cluster hardware and software.

2.1 Syntax

I have tried to be at least self-consistent with syntax in this document.

Parameters in <*italics*> need to be replaced with appropriate input (or read as appropriate output) for your installation without the <>.

Command line input in a Terminal application is shown preceded by an abbreviated command line prompt (:~$ in most cases) and followed by the typed command. The actual prompt will be of the form:

<*user_name*>@<*node_name*>:~$, ie ob1@Master:~$

Some of the input exceeds a single line in this document. I have added line breaks at spaces and indicate the line continuation by tab indenting like this:

```
:~$ This is the command
        and this is a continuation…
        and this is a further continuation
```

Note that input can often be simplified in Terminal by typing the first letter or two of a file or directory in a command line followed by the [Tab] key. Launching Terminal is either from the GUI (Graphical User Interface) task bar, or more conveniently by pressing [Ctrl][Alt][T].

Many Linux files, commands and command options incorporate underscores '_', hyphens '-', and double hyphens '--', and mixing these up is a common source of errors. For clarity I have placed a space between double hyphens in this document '- -'.

I tend to use gedit (Gnome Editor) for editing files but from time to time it may be convenient (or necessary) to using a CLI (Command Line Interface) editor such a nano.

Each of the computers in the cluster is referred to as a node.

When editing files it is a good idea to save a backup copy before you make a mistake (foo-bah).

After you edit a file remember to save it. I have not explicitly described saving files following editing in the following instructions.

In general system files (not located in user's login directory path) will require super user privileges for editing which requires a sudo command prefix.

This document necessarily incorporates a lot of jargon and Linux commands. I have tried to define acronyms and expand on command syntax on first reference, if for no other purpose than for my own education. If I missed something that you found confusing then please let me know.

Each significant step in the installation procedure provides a definitive end point where stuff can (and should) be tested. If stuff isn't working as expected at an end point then either you have made a mistake (this is one of the things that I do best with Linux) or the procedure is not appropriate for your hardware or software environment. You will need to revert to thinking about what you have done and what is (or isn't) occurring to resolve the problem.

3 Purpose

A significant contribution to the FDS processing time of my previous cluster was the speed of the 1 Gb Ethernet LAN (Local Area Network). I have previously estimated this to be about 25% of the total processing time although this is somewhat model and allocated resource dependant.

Experiments with ssh (Secure SHell) and rsh (Remote SHell) showed no significant reduction in the Ethernet communication overhead. The primary issue was therefore attributed almost entirely LAN bandwidth and latency, although there is also a CPU (Central Processing Unit, or core) overhead associated with Ethernet communication.

Infiniband (IB) upgrade options were explored. A new IB switch, IB network cards (called HCAs – Host Channel Adaptors) and cabling were prohibitively expensive. I looked a second-hand options. While perhaps not state of the art, used IB hardware could be purchased for a modest cost and should provide substantial performance gains for my Linux cluster.

I purchased a second hand Mellanox 40 Gb/s 8 IS5022 eight port switch, four Mellanox MHQH19B-XTR 40 Gb/s PCI 2 HCA cards, and four 3 m IB cables.

4 Platform Description

All four nodes are running water cooled Intel I7 quad core processors at 4.4 GHz with 16 GB of DDR3 RAM, Solid State Hard Drives (SSHD), secondary 500 GB Hard Disk Drives (HDD) and 1 Gb/s Ethernet with Internet connectivity. There are no expansion or graphics cards in the nodes. Figure 1 shows the basic cluster configuration with the IB network. Other Ethernet networked components (workstations, printers, scanner, etc) are not shown for clarity.

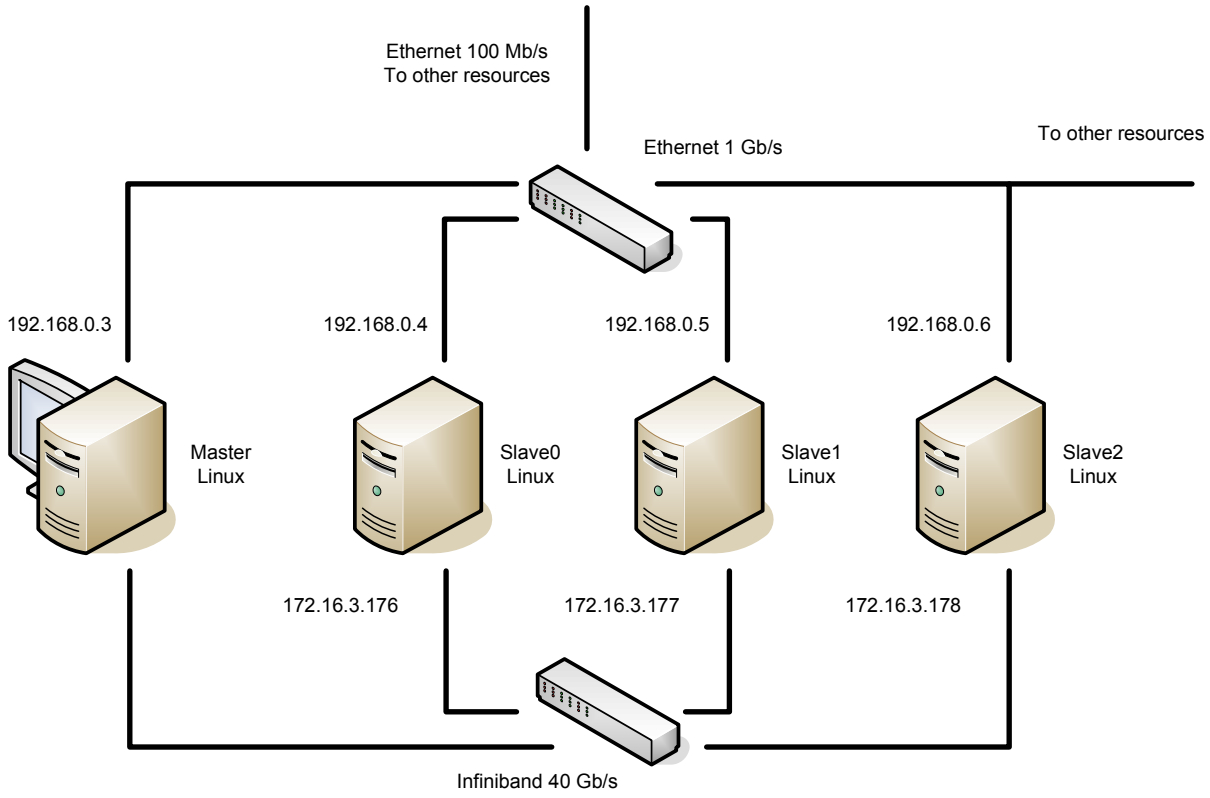

Figure 1. Basic Linux Cluster Configuration

Although the hardware is not server-grade it has demonstrated solid performance and runs Prime95 and other torture tests without error over extended timeframes (weeks) at ambient temperature extremes without failure. While the system is quite capable of over-clocking beyond 5 GHz this pushes the thermal limits of the motherboards and will accelerate component failure. Your hardware needs to be robust for FDS processing as models may take days or weeks to simulate.

5 Upgrade Path

There are two upgrade paths that I considered. One is simply to upgrade a single node, and then clone this to other nodes making minor adjustments as necessary. While this is very efficient it does not allow for progressive testing of the hardware and software so that an error in the installation process might be difficult to solve.

I settled on the second path which is a concurrent installation on two nodes with progressive testing, followed by cloning to other Slave nodes with appropriate minor adjustments.

The upgrade is to be followed by software verification using at least the minimal verification suite described in Appendix B of the FDS Users Manual.

6 Install the Operating System (OS)

The process started with a clean install of Ubuntu 16.04 LTS on two nodes.

http://www.ubuntu.com/download

Note: if you are using Intel compilers Ubuntu 16.04 is not listed as a supported OS (at the time of writing) and you will need to revert to Ubuntu 14.04.

Ubuntu 16.04 is LTS (Long Term Support), it is free, it is compatible with the Mellanox IB software, and the CLI and GUI are almost identical to Ubuntu 14.04.

Note that while Ubuntu is inherently customizable (it can readily be reduced to a server by removing and installing modules) there may be other Linux flavours that provide increased processing speed for FDS.

During the OS installation script use different computer (node) names such as Master, Slave0, but use a common user name and password. This will aid with seamless connectivity between nodes.

6.1 Internet Access

We need internet access to download application software.

My Ethernet connection is already established from the previous installation with DNS (Domain Name Server) Ethernet IP (Internet Protocol) addresses in the range 192.168.0.<*x*>. assigned from a router based on node eth0 MAC (Media Access Control) address during boot.

6.2 Known Hosts

For convenience we will want to address nodes over IB by name (Master, Slave0, …). The use of aliases saves on typing and assists in preventing mistakes.

Edit the ~/etc/hosts file on the Master and Slave0 nodes:

```
:~$ sudo  gedit  /etc/hosts
```

Include IP addresses and names for anticipated future nodes (even those these may not yet exist):

```
127.0.0.1       localhost

172.16.3.175   Master
172.16.3.176   Slave0
172…
```

Note that the IP addresses here are somewhat arbitrary. I have used public allocated address space even though the IB is actually inaccessible from the Internet (it is a private LAN). The example in the Mellanox OFED Linux manual uses IP addresses in the range 11.4.<*x*>.<*x*>.

6.3 Housekeeping and Preferences

With the basic OS installed there are some house keeping and customisation tasks that you may want to perform. I'm running a GUI for development and have some favourite applications that I like to have on hand.

Adjust the screen lock (always on, no login password, no time-out).

Add Terminal and System Monitor to the task bar.

Install Google Chrome (Chromium), my preferred browser.

Remove Firefox (sudo apt-get purge firefox) and some other bundled applications that I never use.

Change the default screen background (I hate the Ubuntu default screen).

6.4 Lock the CPU Frequency

To prevent the OS from throttling the CPU frequency during low demand edit /etc/init.d/ondemand:

```
:~$ sudo gedit /etc/init.d/ondemand
```

Find the script line:

```
echo –n $GOVERNOR > $CPUFREQ
```

and add the following line immediately above it:

```
GOVERNOR=”performance”
```

6.5 Upgrade and Update

Ensure the OS and application installations are up to date.

```
:~$ sudo apt-get update && sudo apt-get upgrade
```

The upgrade may report some redundant packages. These are readily removed with:

```
:~$ sudo apt-get autoremove
```

7 Install the IB Hardware

Power down the nodes.

Install the IB HCAs in the Master and Slave0 nodes in appropriate PCI (Peripheral Component Interconnect) slots on the mother boards noting the HCA MAC addresses and other details from the manufacturer's label (I photographed these because the writing is pretty small). Reading this information after the HCA cards are fitted is at best difficult.

Power up (reboot) and confirm that the OS can find the HCAs:

```
:~$ lspci –v | grep Mellanox
```

You're looking for an ib0 designation. If it isn't there then there is something wrong with the hardware installation. Check that the PCI slots are appropriately rated (PCI slots should be backward compatible) and that the cards are properly seated.

Power up the IB switch and connect it to the HCAs (note the Mellanox switch and HCA manuals advise that hot-plugging is okay).

With no HCA drivers installed the switch should show green LEDs for the status and fan, but with no port LEDs. The HCA LEDs will be not be lit.

8 Lock the Kernel

The IB software (which we are about to install) is OS kernel version specific. If we don't lock the OS kernel modules they will automatically upgrade from time to time and this may cause the IB network to fail (if for no other reason than changes in the kernel name). Kernel updates are generally a good idea as they often contain security fixes but upgrades require specific change management with IB.

```
:~$ sudo -s

:~# for i in $(dpkg -l "*$(uname -r)*" | grep kernel | awk '{print $2}');
        do echo $i hold | dpkg - -set-selections; done

:~# exit
```

9 Install the OFED Support Packages

The IB drivers and software for the Mellanox hardware comes bundled in the OFED (Open Fabrics Enterprise Distribution) available free from Mellanox. This package includes the opensm subnet fabric management, firmware upgrades, and a lot of other functionality.

http://www.mellanox.com/page/products_dyn?product_family=26&mtag=linux_sw_drivers

Also download the Linux Users Manual and Installation Notes for the OFED version of your hardware and OS, and the HCA and switch manuals if you haven't already got them.

The Mellanox OFED.iso was downloaded from the Mellanox website for the OS (Ubuntu 16.04) and the processor environment (x86-64) into the ~/Downloads directory. Note the .iso checksum from the Mellanox downloads page as this is used to validate the download.

Check the .iso download integrity:

:~$ md5sum Downloads/MLNX_OFED_Linux… … .iso

The reported checksum should match the Mellanox download page checksum.

The OFED must be installed on all nodes to provide each node with HCA drivers and establish the IB Fabric stack.

9.1 Install Software Packages

The Mellanox OFED install requires a number of other packages as described in the Release Notes. I have written these up in a table to check off the installs.

perl			
dpkg			
autotools-dev			
autoconf			
libtool			
automake1.11			
automake			
m4			
dkms			
debhelper			
tcl			
tcl8.4			
chrpath			
swig			

graphviz			
tcl-dev			
tcl8.4-dev			
tk-dev			
tk8.4-dev			
bison			
flex			
dpatch			
zlib1g-dev			
curl			
libcurl4-gnutls-dev			
python-libxml2			
libvirt-bin			
libvirt0			
libnl-3-dev			
libglib2.0-dev			
libgfortran3			
pkg-config			
libnuma1			
logrotate			
ethtool			
gfortran			

Note: I have added gfortran to the list and updated some of the package versions.

You can use the Ubuntu apt-get (Advanced Packaging Tool) to install these packages from the CLI but I prefer the GUI Synaptic Package Manager which appears to a have more comprehensive repository resource .

```
:~$ sudo apt-get install synaptic

:~$ sudo synaptic
```

The package installation was followed by a further update and upgrade:

```
:~$ sudo apt-get update && sudo apt-get upgrade
```

9.2 Mellanox OFED Installation

I have Mellanox IB hardware so I will be using their IB software for compatibility.

The OFED installation requires specific options for Ubuntu in the install script that are described in the OFED Linux Users Manual. Make sure you have the correct version of the OFED Manual for your OFED version when determining these requirements.

We want Internet Protocol over InfiniBand (IPoIB) with fixed IP addresses on each node so we need to assign IP addresses to the HCA's. We can assign IP addresses manually on each node once the OFED is installed using:

:~$ ipconfig ib0 172.16.3.<*x*> netmask 255.255.0.0

but we want IP addresses established on boot. We can add this requirement to the OFED installation script through the - -net option.

Make a file on each node in the /etc/NetworkManager/system-connections directory named ifcfg_ib0:

:~$ sudo gedit /etc/NetworkManager/system-connections/ ifcfg_ib0

I placed this in the /etc/NetworkManager/system-connections directory which seems appropriate for Ubuntu but the location is not important. The file content should be similar to:

```
# IPoIB Static IP Address Definition
#Change the IPADDR_ib0 parameter for each node
#Note: do not add spaces in the script lines

IPADDR_ib0=172.16.3.<x>
NETMASK_ib0=255.255.0.0
NETWORK_ib0=172.16.0.0
BROADCAST_ib0=172.16.255.255
ONBOOT_ib0=1
```

You will see that I have used an IP address range to match the known hosts (above) rather than those specified in the Mellanox OFED Linux Users Manual.

We need to confirm the kernel version and note if for the OFED install:

:~$ uname –r (or –a)

Mount the Mellanox OFED.iso image from the Downloads directory:

:~$ cd Downloads

:~/Downloads$ sudo mount -o ro,loop MLNX_OFED_LINUX… … .iso
/mnt

Change directory to the mounted .iso:

:~$ cd /mnt

Confirm the mounted file structure:

:/mnt$ dir

or

:/mnt$ ls -al

Now run the OFED installation script from the /mnt directory as a super user, inserting the appropriate kernel version for your OS noted above:

:/mnt$ sudo ./mlnxofedinstall - -without-dkms - -add-kernel-support
- -kernel <*4.4.0-35-generic*> - -without-fw-update
- -net /etc/NetworkManager/system-connections/ifcfg_ib0
- -force

During the installation note the location of the installation log file. You may need to read this if the install fails. There is lots of text output during the installation which is impossible to read in real time.

The OFED software installs binaries and library components by default in the /usr directory in /usr/sbin and /usr/lib. These directories are in the default Ubuntu PATH environment variable.

With Mellanox OFED installed, reboot the nodes.

We can check that the IP address has been assigned to the HCA cards by running:

:~$ ifconfig

To check the install run:

:~$ /etc/infiniband/info

which should report the installed OFED version.

Observe the LEDs on the HCAs and the switch. The HCA LEDs should show solid green indicating a valid hardware connection without traffic. The switch should show solid green for the status and fan LEDs, and the connected port LEDs should show solid yellow (physical connection, no traffic).

9.3 Subnet Manager

The Mellanox IS5022 switch is externally managed and therefore requires external subnet fabric management software which must reside on at least one node in the cluster (note that the cluster is a single subnet).

At this time we are still not running the subnet manager, opensm, which comes with the OFED. The subnet manager searches and configures the IB Fabric for communications. If there are multiple installations of opensm on many nodes in a subnet only one instance will be active. The others will remain idle. opensm can be run from the command prompt on any or all nodes with:

```
:~$ opensm
```

It can also be started as a daemon with:

```
:~$ sudo /etc/init.d/opensmd start
```

or

```
:~$ sudo service opensmd start
```

We want the opensmd daemon to start on the Master node when we boot our cluster without having to launch it manually.

Note that this can be a bit of a challenge for Ubuntu 16.04 because the associated opensmd scripts are configured for SystemV and Ubuntu 16.04 uses Systemd (System Daemon) which, while backwardly compatible with SystemV, doesn't provide an easy way of changing the boot-status of the legacy daemon scripts.

After much experimentation I installed sysv-rc-conf:

```
:~$ sudo apt-get install sysv-rc-conf
```

To view the installed daemon status run:

```
:~$ sudo sysv-rc-conf - -list
```

You will see that opensmd is installed but has no assigned run-levels so it will not run on boot. We can activate it on the Master node with default run levels of 2345 using:

```
:~$ sudo sysv-rc-conf opensmd on
```

Reboot and the Master will now run opensmd at boot and establish the IB Fabric.

With opensm running check the port status LEDs on the switch and the HCAs. The HCAs should shown green and solid yellow. The connected switch port LEDs should be solid green.

9.4 IB Testing

In order to be assured that we are testing the IB network the following tests should be done both with and without the backplane Ethernet connection. Either physically disconnect it or remove power from the Ethernet switch to isolate the IB.

We can now examine the status of the IB network using the following commands from either node (refer to the Mellanox Linux User's Manual for a detailed description of what these commands are actually doing and reporting):

```
:~$ ibstatus
:~$ ibstat
:~$ ibnetdiscover
:~$ ibhosts
:~$ iblinkinfo
```

ibstat returns the GUID (Globally Unique IDentifier) and the LID (sub-net Local IDentifier assigned by the subnet manager). We can use these identifiers to initiate ibping to test the network connectivity.

On each node run:

```
:~$ ibstat
```

to confirm that a fixed IP address has been assigned to ib0. Note the GUID and the LID of the nodes. Be aware that the LID in particular is allocated dynamically and may change for a particular node with many nodes on the subnet depending on the boot order.

Run ibping as a server on a node:

```
:~$ ibping –S
```

Go to the other node (client) and run:

```
:~$ ibping <LID>
```

or

```
:~$ ibping –G <GUID of ibping server>
```

or

```
:~$ ibping  -f <LID>
```

[Ctrl][C] to exit and show statistics.

We can also exercise IPoIB using ping.

:~$ ping Slave0 (or Master)

or

:~$ sudo ping –f Slave0 (or Master)

[Ctrl][C] to exit

Everything going well during ping sessions the green switch port LEDs and the yellow HCA LEDs will be flickering indicating IB traffic.

Note that while ping and ibping prove network connectivity they also provide packet metrics I can't find a meaningful way of comparing these. We will use other tests to establish IB fabric network performance shortly.

Our basic IB network is now up and running with native IB and IPoIB connectivity.

9.5 IB Performance Testing

We can test the IB performance using the following utilities that are installed with the Mellanox OFED:

:~$ udaddy on server
:~$ udaddy –s <*server*> on client

:~$ ib_read_bw –a on server
:~$ ib_read_bw –a –F <*server*> on client

:~$ ib_write_bw –a on server
:~$ ib_write_bw –a –F <*server*> on client

:~$ ib_read_lat –a on server
:~$ ib_read_lat –a -F <*server*> on client

:~$ ib_write_lat –a -F on server
:~$ ib_write_lat –a -F <*server*> on client

Note that these commands are all run in client/server pairs. The -a parameter on the ib_... commands will run the command with packet sizes from 2^1 through to 2^23. The -F parameter will suppress frequency alignment warnings between the server and the client. Even though we have locked the CPU frequency at performance there may still be minor speed differences between nodes.

The reported bandwidth (read and write) for my IB Fabric was over 24 Gb/s. While this appears to be somewhat lower than the 40 Gb/s rating of the switch and HCAs, the aggregated throughput of 10 Gb/s QDR (Quad Data Rate) hardware with 8/10 encoding is about 32 GB/s.

Further, the actual performance can undoubtedly be improved by tuning OFED parameters but I won't be going there just now.

If you have previously disconnected the Ethernet network then now is a good time to reconnect it.

10 Install nfs

This step is essential for FDS. The FDS Users Manual states '*The MPI version of FDS requires shared disk access to each computer where cases will be run.*' Failure to complete this step will prevent openmpi running between nodes.

Install nfs-client on both nodes:

```
:~$ sudo apt-get install nfs-common
```

Install nfs server on the Master node (we can install the sever on both nodes but it will only be invoked on the Master):

```
:~$ sudo apt-get install nfs-kernel-server
```

Reboot.

Test for nfs operating system support on both nodes:

```
:~$ cat /proc/filesystems | grep nfs
```

Displays:
```
nodev    nfs
nodev    nfs4
nodev    nfsd
```

Test for portmap listing on both nodes:

```
:~$ rpcinfo –p | grep portmap
```

Displays:
```
100000   4   tcp    111   portmapper
100000   4   udp    111   portmapper
100000   …
```

11 Shared Directory

Now we're going to make a shared directory where our FDS model files will reside, hosted from the Master node.

Make a ~/Projects directory on the Master and Slave0 nodes using either the File Manager tool or from the command line:

```
:~$ mkdir Projects
```

Set the Projects directory permissions to 777.

```
:~$ sudo chmod –R 777 Projects
```

Insert the export directory on the Master node in /etc/exports:

```
:~$ sudo gedit /etc/exports
```

and add the directory to be exported:

```
/home/$USER/Projects 172.16.3.0/24(rw,async)
```

Note: we can be more specific or more general about the export locations by either specifying specific Node names or a universal IP address range by replacing 172/16.3.0/24(rw,async) with:

```
Slave0(rw,async) Slave1(rw,async) …
```

or

```
*(rw,async)
```

Note: there are other nfs options for the export (refer to the associated man pages). The no_root_squash, sync and no_subtree_check options seem to cause problems with SmokeView and FDS causing program termination and segmentation errors. More experimentation and thought is required here to work out what is actually occurring.

On the Slave0 node auto-mount the shared Master Projects directory:

```
:~$ sudo gedit /etc/fstab
```

add:

```
Master:/home/<user_name>/Projects /home/<user_name>/Projects nfs
      defaults 0 0
```

Reboot the nodes.

Test for Master exports on the Master node:

:~$ showmount -e Master

Displays: Export list for Master:
/home/<*user_name*>/Projects *

or

:~$ sudo exportfs -ra

Test that files and directories in the ~/Projects directory are correctly maintained across all nodes on an alteration, addition or deletion on any node. Use File Manager or the command line from a Terminal to insert, append, and delete sub-directories and files on any node and observe that the action is repeated on the other node. Use [Ctrl][R] to refresh the client node Projects directory content display in the GUI File Manager. The Master should automatically refresh the display for changes made on the client nodes.

Note: Ubuntu will place a black box on the client Slave0 node Projects directory icon in the quick launch task bar and in File Manager – but there will be no change to the Master Projects directory icon (it will look just like a normal folder). If this changes then you may have inadvertently invoked Samba or some other Windows network share. This may cause problems down the track.

Note: concurrent access to a single file on more than one node may result in a file system conflict. This can also occur when running FDS if we access an active (simulating) Projects sub-directory from a Slave node which may cause the Master node to crash. Hence viewing running simulation output files or running SmokeView should be restricted to the Master node. This rule is easily applied on my cluster because in normal operation only the Master node has a keyboard and monitor.

We can run some metrics on the file share over IB on client nodes (see http://nfs.sourceforge.net/nfs-howto/ar0s05.html for a description).

:~$ time dd if=/dev/zero of=Projects/testfile bs=16k count=16384

followed by:

:~$ time dd if=Projects/testfile of=/dev/null bs=16k

My metrics with the default bs=16k were build/writes at ~355 MB/s and read/remove at 10 GB/s.

12 Install SSH

We need to install ssh (Secure SHell) on the nodes as this is required by openmpi (Open Message Passing Interface) for setting up and breaking down MPI parallel processes.

ssh connections must be password-less and silent on non-interactive login between all nodes. Failure to establish this is a common cause of error with openmpi installations.

Ubuntu 16.04 comes bundled with openssh-client. Test the package availability on both nodes with:

```
:~$ dpkg –l | grep openssh
```

If you need to install (or reinstall) openssh-client then:

```
:~$ sudo apt-get install openssh-client
```

Now install openssh-server on both nodes:

```
:~$ sudo apt-get install openssh-server
```

12.1 Generate Public/Private Key Pairs

We need to configure ssh to enable password-less access between any two nodes. We do this by generating public/private key pairs on both the Master and the Slave0 nodes.

Note: Ubuntu 16.04 requires the use of rsa as opposed to dsa encryption by default.

```
:~$ ssh-keygen -t rsa
```

Use default file locations.

Use a blank passphrase.

Copy id_rsa.pub to <*node_name*>.pub.

```
:~$ cd .ssh

:~/.ssh$ cp id_rsa.pub  <node-name>.pub
```

Copy Slave0.pub from the Slave0 node to the Master node either through File Manager using the nfs shared Projects directory or using scp (secure copy):

```
:~/.ssh$ scp  Slave0.pub <user_name>@Master:.ssh
```

Build authorized_keys on the Master.

:~/.ssh$ cat Master.pub Slave0.pub id >> authorized_keys

Change file permissions of ~/.ssh/authorized_keys on the Master node.

:~$ chmod 600 ~/.ssh/authorized_keys

Note that ssh is very particular about file and directory permissions. If they are not correct then authorized_keys may be ignored and we won't get password-less login. The .ssh directory should have been generated on both nodes with 700 permissions. You can check permissions with the command:

:~$ ls –al

The permission value is a 9 bit mask so 700 corresponds with xrw- - - - - - and 600 corresponds with rx- - - - - - -. If ~/.ssh has the wrong permissions you can modify it with the command:

:~$ chmod 700 ~/.ssh

Now copy authorized_keys from the Master to ~/.ssh on Slave0 either using File Manager through the shared Projects directory or using scp:

:~/.ssh$ scp authorized_keys <*user_name*>@Slave0:.ssh

Reboot both nodes.

12.2 Testing SSH

Note that on first ssh login to a remote node you will be asked to confirm:

> The authenticity of host <*111… ….111*> can't be established. ECDSA key fingerprint is <*fd:fd:... …22:fe*>. Are you sure you want to continue connecting (yes/no)?
>
> Type: yes [Enter]

Test connectivity between nodes through ssh without passwords.

:~$ ssh <*node*>

where <*node*> is Master or Slave0 from either Master or Slave0

:~$ exit or ~$ ~. to close a remote terminal session.

With non-interactive login, test that the ssh sessions are silent.

:~$ ssh <node> echo “Test”

should only return “Test” without login details.

At this stage ssh should be working over the IB Fabric.

13 Install openmpi

openmpi needs to be installed on all nodes. There are no precompiled openmpi binaries for the Linux distribution of FDS with IB support. Therefore the source files need to be compiled from the openmpi source distribution.

For FDS 6.5.2 we need openmpi 1.8.4 (more recent releases of openmpi have compatibility issues with FDS that are being worked on).

FDS expects to find the openmpi installation directory at /shared/openmpi_64 for non-IB installs, and /shared/openmpi_64ib for IB installs. It is recommended that you use the FDS default directories. openmpi needs to be installed in the same location on all nodes.

Install the gnu c++ complier (g++ 4:5.3.1) to the ~/Downloads directory using either Software Centre or Synaptic Package Manager.

Note that NIST uses the Intel compiler suite for both FDS and openmpi, but the openmpi Installation Instructions use the gnu compiler by default. This did not cause problems on the previous Ethernet install and a good reason for using gnu (provided that it works for your installation) is that the Intel compiler is not free for commercial applications.

On both nodes:

Download the openmpi-1.8.4.tar.gz from www.open-mpi.org/software/ompi/v1.81 into the :~/Downloads directory

Extract the tar-ball using File Manager into the ~/Download directory.

Change directory to :~/Downloads/openmpi-1.8.4

```
:~$ cd Downloads/openmpi-1.8.4
```

Set up the configuration file for the installation. We are going to include options for the installation directory and to force IB support.

```
:~/Downloads/openmpi-1.8.4$ ./configure
        - -prefix=/shared/openmpi_64ib
        - -with-verbs= /usr
```

Note that ‘verbs’ is the appreciated (opposite of depreciated) option for ‘openib’. There are other installation options that may affect the performance and operation of openmpi. If there are problems with the install or operation then looking at the installation options may be a good place to start.

There will be lots of screen output and this will take some time.

Next, run the make file (this compiles the binary files for installation)

:~$Downloads/openmpi-1.8.4$ make

More text output and more time (enough for a coffee perhaps).

Now install the compiled binaries:

:~$/Downloads/openmpi-1.8.4$ sudo make install

More text output …

We need to add openmpi and its libraries to the PATH and LD_LIBRARY_PATH environment variables on both nodes.

Note that the precedence and the names of script files that set up environment variables depend on your Linux flavour and whether the shell session is interactive or non-interactive (and other stuff such as the type of login). This aspect of Linux and the reasons behind it are complicated. There are many useful descriptions of the process. http://mywiki.wooledge.org/DotFiles is a good starting point.

A normal ssh login to a remote node (for example :~$ ssh Slave0 from the Master node is an interactive login. But an ssh login with a script file or command (for example :~$ ssh Slave0 echo $PATH) is non-interactive.

For Ubuntu 16.04 we can alter our environment variables in the ~/.bashrc file provided that we place any script lines above the default 5th line which starts with the comment:

If not running interactively don't do anything.

This can result in doubling up on path variables on initial interactive login following boot (a non-interactive logon) but this is not important as the second instance will be ignored. This is far less of an issue than a non-interactive login that doesn't set the environment variables.

On both Master and Slave0 nodes edit :~$ .bashrc:

:~$ gedit .bashrc

Above line 5 add:

PATH = "$PATH: /shared/openmpi_64ib/bin"
LD_LIBRARY_PATH
="$LD_LIBRARY_PATH:/shared/openmpi_64ib/lib"

Reboot both nodes to set the new environment variables.

13.1 Test the Path Environment Variables

This simple step is very important. Path issues are a frequent cause of openmpi and FDS problems.

```
:~$ echo $PATH
:~$ echo $LD_LIBRARY_PATH
```

check for /shared/openmpi_64ib/bin and shared/openmpi_64ib/lib

Now login from the Master to the Master and Slave0 nodes and visa-versa with ssh and try the same commands. For example:

```
:~$ ssh Master
:~$ echo $PATH
:~$ echo $LD_LIBRARY_PATH
:~$exit
```

Now try the non-interactive login from Master to the Master and Slave0 and visa-versa:

```
:~$ ssh Master echo $PATH
:~$ ssh Master echo $ LD_LIBRARY_PATH
```

In each instance the PATH and LD_LIBRARY_PATH environment variables must include /shared/openmpi_64ib/bin and /shared/openmpi_64ib/lib. If they appear twice (and they probably will for the first test) then that's fine but if they don't appear at all then there is a problem that needs to be resolved.

A useful technique for solving path issues is to incorporate:

echo "I did <*Name_of_Script_File*> "

commands in each of the login script files. These files include:

```
:~/.bash_profile
:~/.bash_login
:~/.profile
:~/.bashrc
```

This list is not definitive. Some of these files may not exist in the default OS install and there may be other script files that are run.

13.2 openmpi TCP Tuning

During the development of this procedure FDS initially refused to run under openmpi between nodes. The program hung during openmpi initialization immediately after allocating meshes to processes.

FDS would run fine under openmpi and/or openmp on a single node. By disconnecting the Ethernet it became apparent that openmpi was trying to utilize Ethernet IP nodes that I had not defined.

After much trial and error, research and thinking about what was and wasn't happening I found that the cause of the problem was openmpi's aggressive use of what it considered to be available network resources. The issue is documented at Issue 7 in https://www.open-mpi.org/faq/?category=tcp#tcp-selection which states in part:

> Unless otherwise specified, Open MPI will greedily use all "up" IP networks that it can find and try to connect to all peers *upon demand* (i.e., Open MPI does not open sockets to all of its MPI peers during `MPI_INIT` -- see this FAQ entry for more details). Hence, if you want MPI jobs to not use specific IP networks -- or not use any IP networks at all -- then you need to tell Open MPI.
>
> **NOTE:** Aggressively using all "up" interfaces can cause problems in some cases. For example, if you have a machine with a local-only interface (e.g., the loopback device, or a virtual-machine bridge device that can only be used *on that machine*, and cannot be used to communicate with MPI processes on other machines), you will likely need to tell Open MPI to ignore these networks. Open MPI usually ignores loopback devices by default, but **other local-only devices must be manually ignored.** Users have reported cases where RHEL6 automatically installed a "virbr0" device for Xen virtualization. This interface was automatically given an IP address in the 192.168.1.0/24 subnet and marked as "up". Since Open MPI saw this 192.168.1.0/24 "up" interface in all MPI processes on all nodes, it assumed that that network was usable for MPI communications. This is obviously incorrect, and it led to MPI applications hanging when they tried to send or receive MPI messages.

The key issue here is highlighted in bold in the NOTE. Ubuntu 14.04 and 16.04 both install a virbr0 device by default. It is a Virtual Bridge interface used for NAT (Network Address Translation). It is provided by the libvirt library and used by virtual environments to connect to an outside network.

You can confirm the presence of the virtr0 device by running:

```
:~$ ifconfig
```

So we must instruct openmpi not to use this device. This can be done at run time by setting the following MCA (modular Component Architecture) BTL (Byte Transfer Layer) parameter in the mpirun command:

```
:~/Projects$ mpirun - - mca btl_tcp_if_exclude virbr0 …
```

This leads to very long mpirun commands which are prone to error however we can set this parameter in ~/.bashrc:

```
:~$ gedit .bashrc
```

Above # If not running interactively don't do anything. Insert the following two lines:

OMPI_MCA_btl_tcp_if_exclude="virbr0"
export OMPI_MCA_btl_tcp_if_exclude

13.3 Test openmpi

If the path environment variables are good then we can test the openmpi install from the command line in Terminal. Start with:

:~$ mpirun

reports that there is nothing to do, and

:~$ mpirun - -version

reports the openmpi version.

Now compile the connectivity_c, hello_c and ring_c programs in the unpacked /Downloads/openmpi-1.8.4/examples directory.

:~$ cd Downloads/openmpi-1.8.4/examples

:~/Downloads/openmpi-1.8.4/examples$ make

Then run the connectivity_c, hello_c, and ring_c test cases on each node:

:~/Downloads/openmpi-1.8.4/examples$ mpirun –np 4 <*test_case*>

Try running them between nodes:

:~/Downloads/openmpi-1.8.4/examples$ mpirun –np 4
–host <*other_node*> <*test_case*>

Try running them on both nodes:

:~/Downloads/openmpi-1.8.4/examples$ mpirun -np 8
-host Master,Slave0 <*test_case*>

Output should conform to the descriptions at https://www.open-mpi.org/community/lists/users/2012/03/18846.php. There are other tests that you might want to run too, but if we have IB connectivity between nodes, can launch openmpi processes on all nodes from any node through ssh, and communicate openmpi processes between nodes then the openmpi installation should be complete.

14 Local Node Backup

If you have got to this stage you should have a working IB network, a ~/Projects directory shared between nodes through nfs over IB, ssh password-less logon between nodes over IPoIB, and openmpi running over IB.

Now is a good time to make a backup of the Master and Slave0 nodes. This allows painlessly restoration of either node to a known state without having to reinstall everything from scratch.

There are a number of backup tools that you might consider such as Clonezilla. I found the user interface of this program to be somewhat user unfriendly and unintuitive.

I used the Linux dd command to produce an exact image of my installation drive (including formatting) on the second internal hard drive. You can also save the image to a portable USB HDD (Hard Disk Drive). Note that the target drive must be at least the same size or larger than the source drive.

On each node first identify the drives:

```
:~$ sudo lsblk
```

Identify the OS installation disk (the source), the location where you want to store the disk image (the target), and the target mount directory (usually in /media/<*user_name*>/<*drive_UUID*>). Drive identifiers will be similar to hda, hdb, sda, sdb, ...

Change directory to the target drive mount point:

```
:~$ /media/<user_name>/<drive_UUID>
```

Now run dd to make the image:

```
:~$ /media/<user_name>/<drive_UUID>$  sudo dd
        if=/dev/<Drive_Identifier> of= ./<Image_Name>
```

Use something meaningful for the Image Name such as MasterImage4Aug16.

The image process will take some time depending on the size of the source drive as it is copying every Byte on the drive, whether or not it is actually being used. On completion dd will report the number of Bytes transferred and the average transfer speed.

14.1 Restore/Recovery

A critical aspect of any backup regime is the ability to restore in a crisis. Many folk make backups and never test them, only to find that, after a disaster, they never really had a backup at all. So test your restore.

Power down the node. Connect a Ubuntu live USB stick (the one that you used for the OS install is fine) and the USB HDD if you saved the image to an external drive.

During the POST (Power On Self Test) press [Delete] to access the BIOS. Change the boot priority in the BIOS to the USB stick. Save [F10] and reboot [Enter].

From the Ubuntu screen prompt select the 'Try Ubuntu without installing' option.

Open a Terminal application [Ctrl][Alt][T].

Identify the source and target drives:

```
:~$ sudo lsblk
```

Change directory to the source drive (where your image is):

```
:~$ /media/ubuntu/<drive_UUID>
```

Check that you can see your image file:

```
:/media/ubuntu/<drive_UUID>$  dir
```

Now run the dd command to restore the image:

```
:~$ /media/Ubuntu/<drive_name>$  sudo dd if=./<Image_Name>
        of=/dev/<Drive_Identifier>
```

At the completion of the restore your screen output may be just a plain prompt or a Grub2 rescue menu. Don't panic quite yet!

Power down, remove the Ubuntu USB boot stick and any external drives.

Reboot to the BIOS. During the POST (Power On Self Test) press [Delete].

Change the boot priority to the Ubuntu HDD, save [F10] and reboot [Enter].

All going well you will have recovered your system from the image, probably in less than 20 minutes.

15 Install FDS

FDS needs to be installed on all nodes in the same directory. We need a Linux version with openmpi support, and perhaps openmp too.

The install can be completed on every node but it is more efficient to install FDS on a single node and then copy the resulting installation directories and path script files to other nodes.

It is recommended that, before the FDS installation, you should read:

> the FDS Users Manual, and in particular the sections on installation and running FDS. FDS users will be familiar with this document.
> the Installing and Running FDS on a Linux Cluster document: https://github.com/firemodels/fds-smv/wiki/Installing-and-Running-FDS-on-a-Linux-Cluster
>
> the Installing Open MPI on a Linux Cluster document: https://github.com/firemodels/fds-smv/wiki/Installing-Open-MPI-on-a-Linux-Cluster
>
> the FDS Compilation document: https://github.com/firemodels/fds-smv/wiki/FDS-Compilation
>
> and perhaps browse through the associated Issues pages and release notes.

There are two approaches to the FDS install:

> a precompiled bundle (note that the link and name will change as FDS and SmokeView are upgraded over time).
> https://github.com/firemodels/fds-smv/releases/download/Git-r16/FDS_6.5.2-SMV_6.3.12_linux64.sh
>
> or as a system-specific compilation from source code.

The precompiled bundle installation is relatively straight forward and is arguably preferable assuming that FDS runs without error. However the precompiled binaries will not always result in stable FDS performance because they were compiled on NIST platforms with specific hardware and software resources that are unlikely to be identical to yours.

In any case some folk install the precompiled bundle, as a fast method of setting up default directories and script files, even if they intend to compile FDS.

15.1 Precompiled Bundle Installation

I recommend that you try the precompiled FDS-SMV precompiled bundle first. The NIST instructions are clear (even I could follow them).

Note that this will install SmokeView, and FDS with openmp and openmpi support (assuming the later is installed and the expected default directory).

Download the precompiled bundle using your favourite browser from:

https://github.com/firemodels/fds-smv/releases/download/Git-r16/FDS_6.5.2-SMV_6.3.12_linux64.sh

Go to the :~/Downloads directory:

```
:~$ cd Downloads
```

Run the installation shell script using bash:

```
:~/Downloads$ bash FDS_6.5.2-SMV_6.3.12_linux64.sh
```

You will be prompted to confirm the install [Enter] and the installation directory [1][Enter] where upon FDS should be installed in the :~/FDS/FDS6 directory.

FDS will append some scripts at the end of ~/.bashrc which, in conjunction with :~/.bashrc_fds, will create a number of environment variables and adjust the PATH and LD_LIBRARY_PATH to incorporate the necessary directories for both FDS and openmpi.

The installation will detect the presence of openmpi if it has been installed in either :/shared/openmpi_64 for non-IB installs, or :/shared/openmpi_64ib for IB installs, and adjust the path variables for this.

Reboot to set the environment variables and try running FDS from the command line on the installation node

```
:~$ fds
```

Reports version, openmp and openmpi status.

Press [Enter] to exit.

While FDS isn't actually doing anything useful and is only operating on a single node, if the FDS executable file isn't found then there was either an installation problem or (more likely) there is a problem with the PATH and LD_LIBRARY_PATH. You may need to edit :~/.bashrc and :~/.bashrc_fds to correct this.

We need to add some additional environment variables to the ~/.bashrc_fds file for openmp and openmpi:

```
:~$ gedit .bashrc_fds
```

At the end of the file you will find:

OMP_NUM_THREADS= 4

Adjust this line if you need to change the default number of openmp threads. openmpi gives better performance than openmp so a smaller number might such as 2 might be a better default. We can always change this at run time through the openmpi command line –x directive which passes environment variables to the executable program running under openmpi.

Immediately following OMP_NUM_THREADS= 4 add:

```
OMP_STACKSIZE=200M
ulimit –s unlimited
```

Note: While it is possible to set the application memory size software limit (ulimit) in /etc/security/limits.conf this does not ensure that appropriate memory limits are set for non-interactive login through ssh resulting in SEGSIGV error 174 for modest sized meshes. The reason is that this file is only read by PAM (Pluggable Authentication Modules for Linux) which is not accessed for ssh login.

We can also tidy up the PATH and LD_LIBRARY_PATH variables by removing the openmpi PATH and LD_LIBRARY_PATH assignments that we made earlier in ~/.bashrc file. Note that these variables will now be set in the ~/.bashrc_fds script file.

:~$ gedit .bashrc

Above line 5 remove (or comment out with #):

```
PATH = “$PATH:/shared/openmpi_64ib/bin”
LD_LIBRARY_PATH =”$LD_LIBRARY_PATH:
        /shared/openmpi_64ib/lib”
```

Copy the :~/FDS/FDS6 directory (and sub-directories), the :~/.bashrc and the :~/.bashrc_fds files to the same locations on Slave0 either using the shared :~/Projects file through File Manager, or from the command line.

:~$ scp –r FDS <*user*>@Slave0:

:~$ scp .bashrc <*user*>@Slave0:

:~$ scp –r .bashrc_fds <*user*>@Slave0:

15.1.1 Test FDS

Reboot and set about testing a multi-mesh fds model under openmpi on the Slave0 node and on both nodes assigning multiple node resources.

To reiterate, this installation note is not about how to build and run FDS models, or how to interpret the output.

However the test model needs to have at least as many meshes as the number of openmpi processes that you wish to run. Typically this will be at least the number of physical cores on both nodes.

Note that the model must reside in or under the shared ~/Projects directory and either the model path must be explicitly stated, or the openmpi command should be run from the model directory. The general form of the openmpi command line is:

mpirun –x OMP_NUM_THREADS=<*X*> -np <*Y*> -host
<*node_name*>,<*node_name*>,… fds <*model_name*>.fds

where *X* is the number of openmp threads and *Y* is the number of openmpi processes.

While testing you should be specifically looking for numerical instability, segmentation errors and other program stability errors such as excessive processing time or stalling/hanging.

If you run System Monitor in the GUI or a command line utility such as htop on a separate Terminal application you will see how systems resources are being used during the simulation.

If the test models runs to completion then you should be comparing the output with a proven model. The FDS verification suite contains a number of such models complete with documented output and tolerances. See https://github.com/firemodels/fds-smv/wiki/FDS-Verification-Process

15.2 Compile and Install

On my cluster the precompiled FDS bundle worked reliably once I had resolved the MCA TCP tuning issue (see 13.2 above).

So far we have used the gnu compiler suite because some of our existing software instructions recommend this, and because it is free. NIST uses Intel compilers for their packaged FDS binaries, although they provide gnu source code for FDS with openmpi (but without IB).

In a perfect world all compilers would produce equivalent code but this is not a perfect world.

We have the gnu compiler suite installed already. FDS has software version control and development managed by GIT. GIT is apparently a very powerful system but it is challenging for folk migrating from other software management systems, and some of us do not even have this step-up. Hence we must revert

to following the instructions provided by NIST somewhat blindly and think, research and seek help if stuff falls over.

Start by reading https://en.wikipedia.org/wiki/Git and https://github.com/firemodels/fds-smv/wiki/Git-Notes-Getting-Started

Install Git on the Master node. If you fail to complete this step the you will not be able to compile the FDS source code even if you have copied it from the GitHub repository.

```
:~$ sudo apt-get install git-all
```

Reboot.

Now we configure git so it knows who we are:

```
:~$ git config –global <user.<name>
```

```
:~$ git config –global <user>.<email>
```

```
:~$ git init /home/<user>/.git/
```

Now 'clone' the NIST repository. We can also 'fork' the repository but as you might expect these operations are different. Fork produces a dynamically updated local repository linked to the master repository. This is of particular interest to developers because it allows the modifications to be returned to the master repository through a process referred to as a pull request.

We need a local clone in order to compile FDS. This can be completed with:

```
:~$ git clone http://github.com/fdsmodels/fds-smv
```

This will produce a directory :~/fds-smv which is a clone (copy) of the repository at the time the clone was made.

15.2.1 Compile FDS

Compilation proceeds as follows. Identify and navigate to the appropriate directory in ~/fds-smv/FDS_Compilation which contains a description of the intended install options. For this install we head to mpi_gnu_linux_64:

```
:~$ cd fds-smv/FDS_Compile/mpi_gnu_linux_64
```

If you look at the contents you will see that this contains a single script file named make_fds.sh. To compile FDS:

```
:~/ fds-smv/FDS_Compile/mpi_gnu_linux_64$ bash ./make_fds.sh
```

Note that this particular compilation script there is no native IB support and openmp is not installed. This is not necessarily a problem because all of our communications will be over IB and openmpi will almost always provide faster processing than openmp provided that we have a sufficient number of meshes in the FDS model.

Any compilation errors will require research and/or support. Note that the compilation script calls the makefile in the ~/fds-smv/FDS_Compilation directory which sets the compilation options. You can examine these by opening the makefile and looking for the lines with the prefix:

mpi_gnu_linux-64 :

and

mpi_gnu_linux-64_db :

The resulting fds executable file will be in the :~/ fds-smv/FDS_Compile/mpi_gnu_linux_64 directory, but note that it will be named fds_mpi_gnu_linux_64 and not fds.

For compatibility with the pre-installed binary FDS file structure (including the path variables) rename or delete any existing fds executable file in the :~/FDS/FDS6/bin directory:

```
:~$ cd FDS/FDS6/bin

:~/FDS/FDS6/bin$ rm fds
```

Now copy the newly compiled executable file to this directory and rename it fds:

```
:~$ cd ~/fds-smv/FDS_Compile/mpi_gnu_linux_64

:~/fds-smv/FDS_Compile/mpi_gnu_linux_64$
      mv fds_mpi_gnu_linux_64 ~/FDS/FDS6/bin/fds
```

or

```
:~/fds-smv/FDS_Compile/mpi_gnu_linux_64$ cp –f
       fds_mpi_gnu_linux_64 ~/FDS/FDS6/bin/fds
```

Note: The compiled version of FDS uses static libraries. This means that the executable file does not access the associated FDS libraries at run time so there will be no need to incorporate the fds libraries in the LD_LIBRARY_PATH environment variable. The use of dynamic compilation does not result in a significant fds runtime burden because this only occurs when an fds process is launched.

Now copy the :~/FDS directory to the same location on the Slave0 node using File Manager through the shared :~/Projects directory, or from the command line.

:~$ scp –r FDS <*user_name*>@Slave0:

Now we need to test the FDS install. The procedure is as previously stated for the precompiled binary installation at Section 15.1.1 above.

15.3 FDS Installation Problems

All going well FDS will now be running reliably on a node and between nodes under openmpi. If this is not the case then here are some things that you might try.

- Check that your fds model actually runs on a single computer installation (or use one of the NIST example models).
- Make sure that you have defined at least as many meshes as allocated openmpi processes.
- Confirm that your model is in the shared directory and has appropriate file permissions.
- Check you path variables to make sure that the openmpi and fds executables can be found on all nodes with interactive and non-interactive login.
- Search the GitHub Issues https://github.com/firemodels/fds-smv/issues for related problems and solutions.
- Try changing the compiler optimization level reducing and increasing it.
- Try a different compiler.
- Review the MCA parameters that may be applicable to your openmpi installation and the associated trouble-shooting notes at https://www.open-mpi.org/faq/?category=troubleshooting.

16 Cloning

At this stage we should have FDS working under openmpi across IB between just two nodes (Master and Slave0). While we could proceed with a new installation on our other Slave nodes this is clearly inefficient.

Provided that our hardware is compatible we can simply clone our Slave0 node to our other Slave nodes and make relatively minor adjustments to provide the additional nodes with a unique identity.

Start by installing the HCA cards in the additional Slave nodes as described above under Installing the IB Hardware, but do not connect these to the IB switch.

Make a mirror of the Slave0 boot drive to an external HDD as described in Local Node Backup above.

Install the drive mirror on the additional Slave nodes using the Restore/Recovery procedure above.

We should now have configured our additional Slave nodes as bootable copies of the Slave0 node.

16.1 Rename Node and Change IB IP Address

Note: the new nodes may not boot cleanly due to the nfs file shares and media mounts. These should eventually time out (after 90 seconds or so) to a CLI interface. If this occurs you will need to use a CLI editor such as nano to modify the following files.

Rename the new Slave nodes by modifying the entry in /etc/hostname to the corresponding names in the /etc/hosts file (in my case Slave1 and Slave2 as described in Known Hosts procedure above):

```
:~$ sudo gedit /etc/hostname
```

Change the IB assigned IP addresses by editing the IPADDR_ib0=172.16.3.<*x*> line in /etc/network/interfaces to the corresponding IP address in the /etc/hosts file:

```
:~$ sudo gedit /etc/network/interfaces
```

16.2 Auto-Mounts

If you also have other internal SSHD or HDD you will need to add the auto-mount settings for these drives. Ubuntu will have recognised that the hardware specific UUID (Universal Unique IDentifier) of the drive in Slave0 is not present in the new nodes and removed the auto-mount lines from /etc/fstab.

Identify the new node UUID's, file system type and name for other internal drives:

:~$ sudo blkid

Edit the auto-mount :/etc/ fstab (File System Table) file on the new nodes to reflect the actual hardware:

:~$ sudo gedit /etc/fstab

and add a new line

UUID=<*UUID*> /media/<*user-name*>/Data ext4 defaults 0 0

Reboot the new nodes.

Connect the new nodes to the IB switch.

16.3 Testing

Test ssh connectivity between nodes using the procedure above for Testing SSH. As a consequence of the cloning operation the ssh secure keys for the new nodes should be identical to the Slave0 public/private pairs, but these will need to be exercised to ensure that they operating without prompts (ie silent login).

If there is any problem here then new key pairs should be generated on every node, the file authorized_keys should be rebuilt on the Master node and then copied back to the Slave nodes using the procedure described under Generate Public/Private Key Pairs in Section 12.1 above.

The new nodes should now be tested using FDS under openmpi.

17 Project Benchmarks

The purpose of this upgrade was to improve the processing speed of my dedicated FDS cluster and update FDS. Sure, the cluster does other stuff but its main function is processing FDS fire models.

A good question to ask at this time is what was the speed improvement? In my experience many improvement projects (and not just IT) are completed without a valid measure of before and after performance so improvement (and cost benefit) are never actually quantified.

Thanks to an earlier metrics study using FDS 6.2.0 I have a base from which I can measure overall speed improvements with the upgrade.

The transition from FDS 6.2.0 to FDS 6.5.0 results in about a 7% increase in processing time (a loss in performance). This was measured by running identical FDS models under Intel compiled FDS 6.2.0 and 6.5.0 on identical hardware. Note that software improvements often come with a processing time penalty.

I also anticipate that there may be additional performance loss associated with the relative computational efficiency of the compilers. Intel advertises a 30% increase in efficiency over gnu for MPI (openmpi) processing. I cannot measure this without actually purchasing the Intel compiler but I figure Intel have some basis for this claim.

The actual speed improvement depends on the FDS model, and in particular the computation/communication ratio, and on the core commitment of each node. Model computation is less efficient with a higher computation/communication ratio and if node cores are fully allocated.

For my cluster the IB upgrade has resulted in:

- a minimum speed improvement of approximately 10% for fully committed cores with a large computational/communications ratio,

- a typical improvement of 30% (exceeding the anticipated 25% improvement), and

- a maximum speed improvement of 42% for models with a low computational/communications ratio and under-utilized cores on all nodes.

Some of my recent projects have required extensive modelling with individual models taking almost a week to complete the simulation. The improvement would have resulted in saving of at least one day per model, and probably closer to two.

While the actual cost of hardware was minimal (less than $1,000) the time required to work through the installation and resolve issues was significant. A portion of this time was associated with the FDS upgrade from 6.2.0 to 6.5.2, upgrading the OS, and running verification.