Consulting Fire Engineers

34 Satara Crescent
Khandallah
Wellington, 6035
New Zealand

Linux Beowulf Cluster

FDS 6.7.0 Upgrade

10 August 2018
Revision 1.0
T. G. O'Brien

Table of Contents

Revision Status

R01 Initial description of this upgrade.

1 Purpose

FDS is used for research and commercial fire engineering design by FireNZE. It is a requirement of many New Zealand regulators that FDS should be the latest version for application to commercial fire engineering design projects. Further, FDS 6.7.0 incorporates a number of useful feature enhancements (some in Beta) that build on the program's capability.

Recent updates to Intel compilers, the Intel Maths Kernel Libraries (MKL) and Intel MPI promise improved performance over earlier releases. Note that NIST also uses Intel compilers for their bundled FDS distribution.

The procedures in this document produce a verified FDS 6.7.0 installation on the FireNZE Beowulf Linux Cluster that is stable for commercial simulation using both MPI and OMP.

If you are new to compiling FDS and Infiniband then you might find the FireNZE document 'Linux Beowulf Cluster Infiniband Upgrade for Fire Dynamics Simulator (FDS)', Version 1.4 dated 12 September 2016 a useful primer.

2 Upgrade Path

The path used for implementing this upgrade assumes a previous verified FDS 6.6.0 on a Linux Ubuntu Operating System (OS) cluster with Intel MPI 2018 compiled using Intel Parallel Studio XE 2018.

The upgrade proceeds as follows:

Update and upgrade the OS.

Delete the previous versions of Intel Parallel Studio XE 2018 and Intel MPI 2018 from all nodes.

Install the current release of Intel Parallel Studio XE 2018 and Intel MPI 2018 on the Master Node.

Copy Intel MPI 2018 to all nodes.

Compile FDS 6.7.0 on the Master Node.

Copy FDS 6.7.0 executable to Slave Nodes.

Verify FDS 6.7.0 operation and measure comparative processing speed (which also provides a soak test).

3 Operating System (OS) Update and Upgrade

The Kernel is already locked from the previous installation as the Mellanox OFED installation required for Infiniband (IB) will fail if the OS kernel is too recent. Ubuntu 16.04.1 with kernel 4.4.0-104 works but Ubuntu 16.04.3 with kernel 4.10.x-xxx fails.

Upgrade and update the OS installation on all nodes.

```
sudo apt update && sudo apt upgrade
```

Remove any redundant packages.

```
sudo apt autoremove
```

4 Delete Previous Intel MPI and Intel Parallel Studio XE

On the Master Node:

```
sudo bash /opt/intel/parallel_studio_xe_2018.x.xxx/uninstall.sh
```

use default prompts.

If Intel Parallel Studio XE was a Cluster Edition then this will also delete the Intel Maths Kernel Libraries and Intel MPI.

With the Composer Edition this will delete the Intel Maths Kernel Libraries but Intel MPI will remain. Delete this with:

```
sudo bash /opt/intel/impi/<version.package_number>/ uninstall.sh
```

Delete the Intel libraries on all Slave Nodes:

```
cd /opt/intel

rm –r ./*
```

5 Install Intel MPI and Intel Parallel Studio XE

First obtain updated .tgz files and the associated serial numbers or public keys from Intel. The files versions for this installation were:

```
l_mpi_2108.3.222.tgz

parallel_studio_xe_2018_update3_composer_edition.tgz
```

Copy these files to convenient directories on the Master Node (say ~/PSXE and ~/IntelMPI in the home directory).

Note any serial numbers (you may need them during the installation depending on the selected activation).

5.1 Install Intel Parallel Studio XE

Create a temporary staging area for the Intel Parallel Studio XE installation.

```
sudo mkdir /tmp/psxe_staging_area
```

From ~/PSXE decompress the tar (which stands for tape achieve) installation file to the staging area:

```
sudo tar –xvzf parallel_studio_xe_2018_update3_composer_edition.tgz
         –C /tmp/psxe_staging_area
```

Change directory to the staging area:

```
cd /tmp/psxe_staging_area/
        parallel_studio_xe_2018_update3_composer_edition
```

Execute the installation script from the staging area:

```
sudo bash install.sh
```

and follow the prompts generally deferring to the defaults. The installation process should end with ‘completed successfully’.

Note that you may be prompted for missing 32 bit libraries. On a 64 bit system skip this and continue.

5.2 Install Intel MPI

The procedure is similar to that used to install Intel Parallel Studio XE above.

Create a temporary staging area for the Intel MPI installation.

```
sudo mkdir /tmp/mpi_staging_area
```

From ~/IntelMPI decompress the tar installation file to the staging area:

```
sudo tar –xzf l_mpi_2018.3.222 .tgz –C /tmp/mpi_staging_area
```

Copy the public license file to the staging area:

```
cp ~/IntelMPI/filename /tmp/mpi_staging_area
```

Execute the installation script from the staging area:

```
sudo bash install.sh
```

and follow the prompts generally deferring to the defaults.

6 Copy Intel MPI to Slave Nodes and Source

Copy the Master Node Intel MPI installation directory and libraries to all Slave Nodes.

This is readily accomplished through the NTFS shared FDS processing directory (~/Projects on the FireNZE cluster).

From the Master Node:

```
cp –r /opt/intel/compilers_and_libraries_2018.3.222 ~/Projects
```

On each Slave Node:

```
sudo cp –R ~/compilers_and_libraries_2018.3.22 /opt/intel
```

We need to source Intel Parallel Studio on the Master Node, and set the path to Intel MPI and assign script variables on all nodes. This is done through .bashrc.

On all nodes edit ~/.bashrc changing all references to the Intel installation and libraries to the current package. On the Slave Nodes this is best done with a non-GUI editor such as nano.

```
sudo nano ~/.bashrc
```

Reboot all Nodes and confirm the new Intel MPI version is accessible with one or more of the following:

```
which mpirun

mpirun - - version, and/or

mpirun
```

7 Compile FDS 6.7.0

7.1 Clone the FDS source files from the GitHub repository

Git should already be installed and configured, but if not:

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.

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

Reboot.

Now we configure git so it knows who we are:

```
:~$ git config - -global  user.name "<username>"

:~$ git config - -global user.email <user@emailserver.domain>

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

Now clone the NIST repository.

```
:~$ git clone https://github.com/firemodels/fds
```

This will produce a directory :~/.git/fds which is a clone (copy) of the repository at the time the clone was made. Note that the ~/.git subdirectory is not visible in the Ubuntu file manager. Press [Ctrl][H] to reveal ~/.git and its contents.

7.2 Compile FDS

Compilation proceeds as follows.

Compiler options are set in the make file in the ~/.git/fds/Build directory. Edit this make file to incorporate desired compiler options for the particular installation flavour.

For this install the –Ofast and –xcore-avz2 flags were set on the impi_intel_linux_64 lines of the makefile.

Identify and navigate to the appropriate directory in ~/fds/Build which contains a description of the intended install options. For this installation head to impi_intel_linux_64:

```
:~$ cd fds/Build/impi_intel_linux_64
```

If you look at the contents of fds/Build/impi_intel_linux_64 you will see that this contains a single script file named make_fds.sh. To compile FDS:

```
cd ~/fds/Build/impi_intel_linux_64

bash ./make_fds.sh
```

Any compilation errors will require research and/or support.

The resulting fds executable file will be in the ~/ fds/Build/impi_intel_linux_64 directory named fds_impi_intel_linux_64.

For compatibility with the pre-installed binary FDS file structure, including the path variables, rename or delete any existing fds executable file in the current FDS installation directory and replace it with the new executable on the Master Node.

```
cd ~/FDS

mv fds fds660

mv ~/fds/Build/impi_intel_linux_64/fds_impi_intel_linux_64 . /fds
```

Copy the fds executable to the FDS installation directory on all Slave Nodes through the NTFS shared FDS processing directory.

From the Master Node:

```
cp ~/FDS/fds ~/Projects
```

On each Slave Node:

```
cp  ~/Projects/fds ~/FDS
```

7.3 Test FDS

On all Nodes start a terminal session and run fds.

```
fds
```

Confirm the version and compilation date.

8 Verification and Testing

With FDS 6.7.0 installed on all nodes the program should be verified in accordance with the FDS Technical Manual, Volume 2, Appendix B. I use a script file that runs the verification suite automatically in under an hour.

I also like to run some speed metrics using both NIST and FireNZE test models to confirm comparative processing performance under both OMP and MPI. This also provides a soak test of the installation to ensure that it will perform robustly on projects. My speed metrics are run automatically using a script file.

9 Local Node Backup

Now is a good time to make a local and/or remote backup of each node. This allows painless restoration of any node to a known state in the event of a computer disaster without having to reinstall everything from scratch. Node backups also facilitate recovery to an earlier version of FDS should the need arise.

I use the Linux dd command to produce an exact image of my installation drive (including formatting) on a 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 (typically /media/<*user_name*>/<*drive_UUID*>). Drive identifiers will be similar to hda, hdb, sda, sdb, ...

Change directory to the target drive mount point:

```
cd /media/<user_name>/<drive_UUID>

eg: cd /media/ob1/Data
```

Now run dd to make the image:

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

eg: /media/ob1/Data$  sudo dd if=/dev/sda of=./Master27Apr17
```

Use something meaningful for the Image Name such as Master27Apr17.

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.

9.1 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*>

eg: /media/ubuntu/Data

Check that you can see your image file:

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

eg: :/media/ubuntu/Data$ dir

Now run the dd command to restore the image:

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

eg: :~$ /media/Ubuntu/Data$ sudo dd if=./Master27Apr17of=/dev/sda

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 and reboot.

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