MSc Scientific Computing Dissertation
ARM Cluster Linpack Benchmarks

John Duffy

August 2020

1 Introduction

`https://github.com/johnduffymsc/picluster`

1.1 Aims

1.1.1 Investigate Maximum Achievable Linpack Benchmark

Efficiency... achieved vs theoretical maximum

1.1.2 Investigate Gflops/Watt

Green500 ranking...

1.1.3 Overview of Competitive Available Gflops/£

Buy lots of Pi's, or buy a bigger machine...

Plot Gflops vs £...

2 Raspberry Pi 4 Model B

2.1 Description

Photo...

Description...

Highlights...

Limitations...

Reference data sheet in Appendix...

2.2 Theoretical Maximum Performance (Gflop/s)

The Raspberry Pi 4 Model B uses the Broadcom BCM2711 System on a Chip (Soc).

Block diagram from Cortex-A72 Software Optimisation Guide

4 cores

1.5 GHz

128 bit SIMD

4 GB memory (our chosen model)

Caches...

Pipeline...

Simplistically, ...

This ignores instructions pipelining benefits...

3 Pi Cluster

Photo...

Description...

Ubuntu 20.04 LTS 64-bit Preinstalled Server...

Reference Appendix A for detailed build instructions...

Limitations...

Software/update management...

Next PXE/NFS boot...

Cluster management tools

BLAS libraries...

BLAS library management... update-alternatives –config libblas.so.3-aarch64-linux-gnu

picluster/tools... appendix ?... use from node1...

4 High-Performance Linpack (HPL) Benchmark

Reference Paper...

https://www.netlib.org/benchmark/hpl/...

Describe algorithm...

Terminology R_{peak}, R_{max}..., problem size...

Describe methodology for determining main parameters NB, N, P and Q...

N formula...

Reference http://hpl-calculator.sourceforge.net

4.1 Building and Installing HPL

See Appendix...

4.2 HPL.dat

Describe HPL.dat parameters...

Listing 1: Example HPL.dat

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
0            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
26208        Ns
1            # of NBs
32           NBs
0            PMAP process mapping (0=Row-,1=Column-major)
2            # of process grids (P x Q)
1 2          Ps
8 4          Qs
16.0         threshold
3            # of panel fact
0 1 2        PFACTs (0=left, 1=Crout, 2=Right)
2            # of recursive stopping criterium
2 4          NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
3            # of recursive panel fact.
0 1 2        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

A detailed description of each line of this file is ...

4.3 HPL.out

Describe HPL.out...

4.4 Running xhpl

To run xhpl using the serial version of OpenBLAS...

/picluster/tools/picluster-set-libblas-openblas-serial

cd /picluster/hpl/hpl-2.3/bin/serial

mpirun -host node1:4 -np 4 xhpl

To run xhpl using the OpenMP version of OpenBLAS...

/picluster/tools/picluster-set-libblas-openblas-openmp

export OMPNUMTHREADS=4 TODO: SET GLOBALLY AT INSTALLATION

export BLISNUMTHREADS=4 TODO: SET GLOBALLY AT INSTALLATION

cd /picluster/hpl/hpl-2.3/bin/serial

5 OpenMPI

What is OpenMPI...

6 OpenMP

What is OpenMP...

7 OpenMPI Baseline Benchmarks

Ubuntu 20.04 LTS 64-bit packages, without any tweaks...

1 core... a single ARM Cortex-A72 core...

1 node... a single Raspberry Pi 4 Model B, 4 x ARM Cortex-A72 cores...

Linpack performance scales with problem size... REFERENCE

80% of memory a good initial guess... FAQ REFERENCE...

Methodology...

1 core... to investigate single core performance... caveats... use 1GB of mem-

ory...

1 node... to investigate inter-core performance...

2 nodes... to investigate inter-core and inter-node performance...

1..8 nodes ... to investigate over scaling of performance with node count... with optimal N, NB, P and Q parameters determined from 2 node investigation... caveats...

7.1 1 Core Baseline Benchmark

Problem size restricted to 80% of 1GB...

NB 32 to 256 in increments of 8...

NB	N	NB	N	NB	N	NB	N	NB	N
32	9248	80	9200	128	9216	176	9152	224	9184
40	9240	88	9240	136	9248	184	9200	232	9048
48	9264	96	9216	144	9216	192	9216	240	9120
56	9240	104	9256	152	9120	200	9200	248	9176
64	9216	112	9184	160	9120	208	9152	256	9216
72	9216	120	9240	168	9240	216	9072	-	-

1x1

```
$ mpirun -np 1 xhpl
```

mpirun does bind to core by default for $np \leq 2$

4 x 4.7527e+00 = 19 Gflops

Explain...

Cache misses from peak...

A single core is capable of achieving maximum theoretical performance... CAVEATS whole L2 cache, whole node 4 GB memory, although problem size limited to 80% of 1 GB...

7.2 1 Node Benchmark

1x4

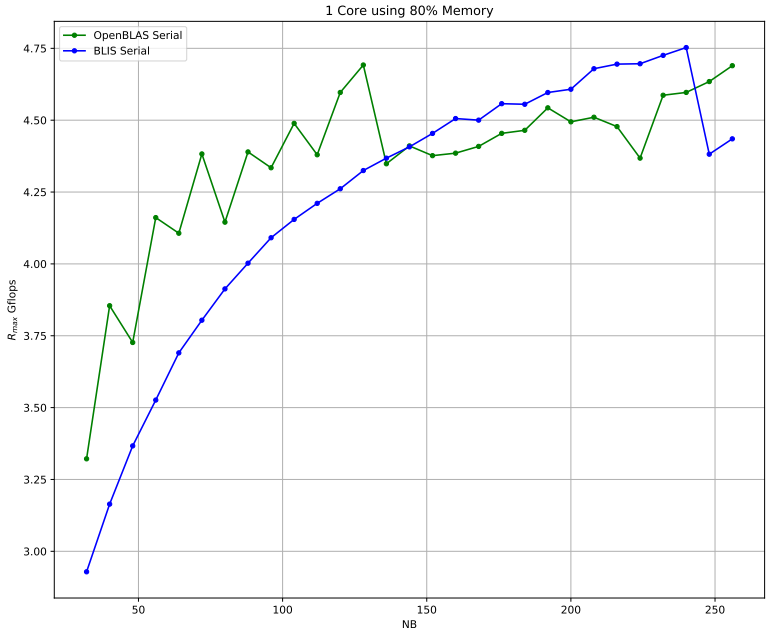

Figure 1: 1 Core R_{max} vs NB using 80% of 1GB memory.

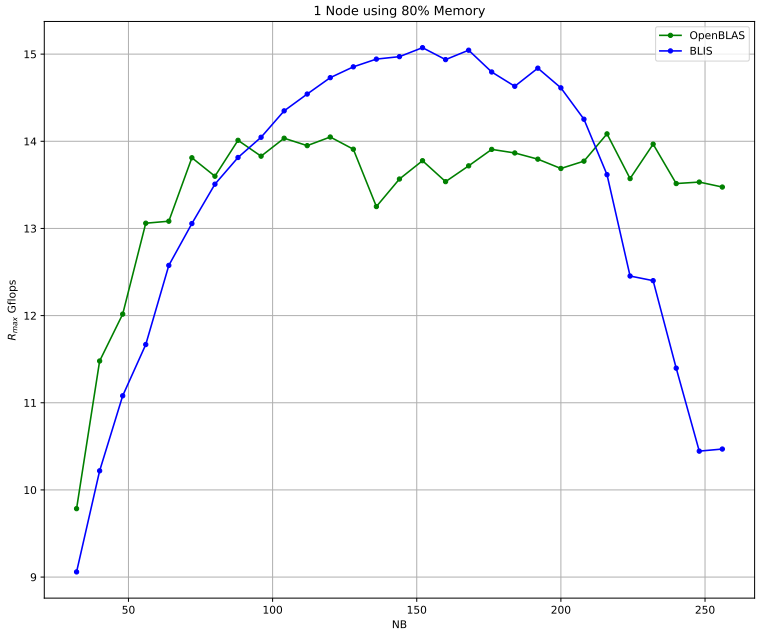

Figure 2: 1 Node R_{max} vs NB using 80% of 4GB memory.

NB	N	NB	N	NB	N	NB	N	NB	N
32	18528	80	18480	128	18432	176	18480	224	18368
40	18520	88	18480	136	18496	184	18400	232	18328
48	18528	96	18528	144	18432	192	18432	240	18480
56	18536	104	18512	152	18392	200	18400	248	18352
64	18496	112	18480	160	18400	208	18512	256	18432
72	18504	120	18480	168	18480	216	18360	-	-

```
mpirun -np 4 xhpl
```

mpirun does bind to socket by default for $np \geq 2$

7.3 2 Node Baseline Benchmark

P1 x Q8

P2 x Q4

NB	N	NB	N	NB	N	NB	N	NB	N
32	26208	80	26160	128		176		224	
40	26200	88	26136	136		184		232	
48	26208	96		144		192		240	
56	26208	104		152		200		248	
64	26176	112		160		208		256	
72	26208	120		168		216		-	-

7.4 8 Node Baseline Benchmark

1x32 2x16 4x8

7.5 Observations

Best NB...

PxQ discussion... 1x8 vs 2x4... ethernet comment...

Iperf...

htop...

top...

perf...

cache misses...

software interrupts...

Suggests... improve network efficiency?

7.6 Baseline Benchmark

As per software installation from Ubuntu Server 20.04 LTS.

OpenBLAS

OpenMP

OpenMPI

HPL-2.3 compiled with Make.rpi4.baseline

NB = 128 (mid-range guess; to be tuned based on L1 cache size)

N for 80% efficiency (from tool)

Recommended: P×Q as square as possible, with Q > P

4 cores per node 1.5 GHz clock speed 4 GB memory per node 2 instructions per cycle (estimated as NEON is 128 bits)

From tool:

Run:

```
mpirun -host node1:4 -np 4 xhpl
mpirun -host node1:4,node2:4 -np 8 xhpl
mpirun -host node1:4,node2:4,node3:4 -np 12 xhpl
etc
```

TABLE RESULTS - Gflops vs node count, time vs node count GRAPH RESULTS - Gflops vs node count, time vs node count

Discussion...

NB size...

P
Q Ratio...

Node number scaling...

7.7 OpenMPI without OpenMP

Describe processor grid layout.

hosts-with-slots file.

7.8 OpenMPI with OpenMP

Describe processor grid layout.

hosts-no-slots file.

8 Performance Optimisation

8.1 Methodology

1. Measure
2. Study results and propose theory
3. Change something based on 2.
4. Measure
5. Repeat steps 1 - 4

9 Build Kernel with Jumbo Frames Support

Standard MTU is 1500 bytes...

Maximum payload size is 1472 bytes...

NB of 184 (x 8 bytes for Double Precision) = 1472 bytes...

NB > 184 => packet fragmentation => reduced network efficiency...

This causes drop of in performance???...

Max MTU on Raspberry Pi 4 Model B is set at build time to 1500...

Not configurable above 1500...

TODO: EXAMPLE OF ERROR MSG...

Need to build the kernel with higher MTU...

Make source packages available...

```
sudo touch /etc/apt/sources.list.d/picluster.list
sudo vim /etc/apt/sources.list.d/picluster.list...
    deb-src http://archive.ubuntu.com/ubuntu focal main
    deb-src http://archive.ubuntu.com/ubuntu focal-updates main
sudo apt update
```

Create a kernel build directory with the correct access permissions to prevent source download warnings.

```
mkdir kernel
sudo chown _apt:root kernel
cd kernel
```

Install the kernel build dependencies...

```
sudo apt-get build-dep linux linux-image-$(uname -r)
```

Download the kernel source...

```
sudo apt-get source linux-image-$(uname -r)
```

Make the required changes to the source... as per REFERENCE

```
cd linux-raspi-5.4.0

sudo vim include/linux/if_vlan.h...
    #define VLAN_ETH_DATA_LEN   9000
    #define VLAN_ETH_FRAME_LEN  9018

sudo vim include/uapi/linux/if_ether.h...
    #define ETH_DATA_LEN        9000
    #define ETH_FRAME_LEN       9014

sudo vim drivers/net/ethernet/broadcom/genet/bcmgenet.c...
    #define RX_BUF_LENGTH       10240
```

Add a Jumbo Frames identifier, "+jf", to the new kernel name...

```
sudo vim debian.raspi/changelog...
    linux (5.4.0-1013.13+jf) focal; urgency=medium
```

Build the kernel...

```
sudo LANG=C fakeroot debian/rules clean
sudo LANG=C fakeroot debian/rules binary
```

Install the new kernel...

```
sudo sudo dpkg -i linux*5.4????????.deb
```

10 Single Core Optimisation

10.1 Block Size Optimisation

The block size, NB tuning parameter, is used for matrix calculations and also for network transport.

The most efficient block size is related to the L1 cache size. Describe...

11 Single Node Optimisation

12 Cluster Optimisation

12.1 Recompile HPL for Cortex-A72

Block size!

RESULTS

12.2 Recompile OpenBLAS with OpenMP Support

As advised by the DebianScience/LinearAlgebraLibraries, OpenBLAS should be recompiled from source for best performance.??? See comflicting statement below.

The Ubuntu/Debian OpenBLAS package is an ARM64 multi-architecture build of OpenBLAS which includes the ARM Cortex-A72. As stated in the README.Debian, performance improvements will be minimal by compiling from source.

However, Ubuntu/Debian build does not include support for OpenMP. So, to test the combination of OpenMPI/OpenMP it is necessary to recompile OpenBLAS from source.

To download the same version of OpenBLAS as Ubuntu 20.04 Server LTS (v0.3.8):

```
cd ~/phas0077/downloads
wget https://github.com/xianyi/OpenBLAS/archive/v0.3.8.zip -O OpenBLAS-0.3.8.zip
cp OpenBLAS-0.3.8.zip ~/phas0077/projects
```

```
cd ~/phas0077/projects
unzip OpenBLAS-0.3.8.zip
rm OpenBLAS-0.3.8.zip
cd OpenBLAS-0.3.8
cp Makefile.rule Makefile.rule.original
vim Makefile.rule
make
make install
make clean
```

PROCESSOR AFFINITY - include later.

Need to edit 'Makefile.rule'.

The file 'Makefile.arm64' already has the following so there is no need to add specific architecture flags to 'Makefile.rule'.

```
ifeq ($(CORE), CORTEXA72)
CCOMMON_OPT += -march=armv8-a -mtune=cortex-a72
FCOMMON_OPT += -march=armv8-a -mtune=cortex-a72
endif
```

Block size!

TODO: Read DebianScience howto recompile/package

TODO: Check ARM gcc options; -mune, -march for ARM Cortex-A72 USE LD_PRELOAD trick to use recompiled libopenblas.so for testing before packaging as a Debian package. WHAT ABOUT THE OTHER NODES - How do they know about the re-compiled package on node1?

LD_PRELOAD for testing on node1.

ONCE installed as Debian package, prevent it being 'updated/upgraded'.

RESULTS

12.3 Recompile OpenMPI for Cortex-A72

Block size!

TODO: Check ARM gcc options; -mune, -march for ARM Cortex-A72

RESULTS

12.4 Network Tuning

Is is possible to improve performance looking at network parameter? MTU?

TODO: Read OpenMPI docs

RESULTS

12.5 Perf/Dtrace Linpack

See if there are any performance bottlenecks

New package OpenBLASRPI4? for RPI4 specific optimisations?

RESULTS

13 Appendix A - Raspberry Pi Cluster Build

13.1 Introduction

This appendix is intended to be a complete and self contained guide for building a Raspberry Pi Cluster. With the caveat that the cluster has the bare minimum software/functionality necessary to compile and run the High Performance Linpack (HPL) benchmark, namely the build-essential package, two BLAS libraries (OpenBLAS and BLIS), and Open-MPI. A number of performance measurement tools are also installed, such as perf and iperf. The latest version of HPL is downloaded and built from source.

It would be a relatively simple task to add... SLIRM or...

The cluster consists of the following components...

8 x Raspberry Pi 4 Model B 4GB compute nodes, node1 to node8 1 x software development and build node, node9 9 x Official Raspberry Pi 4 Model B power supplies 9 x 32GB Class 10 MicroSD cards 1 x *workstation*, in my case my MacBook Pro, macbook 1 x 8 port Gigabit Router/Firewall 1 x 16 port Gigabit switch with Jumbo Frame support

Items

Photo

13.2 Preliminary Tasks

1. Update the EE-PROM
2. Get MAC address
3. Generate keys
4. Amend macbook /etc/hosts file...

13.2.1 Update Raspberry Pi EE-PROMs

13.2.2 Get Raspberry Pi MAC Addresses

13.2.3 Generate User Key Pair

On macbook (no passphrase):

```
ssh-genkey -t rsa -C ucapjbj
```

This will create two files... in ...

13.2.4 Amend macbook /etc/hosts

On macbook, using your favourite editor, add the following to /etc/hosts:

```
192.168.0.1 node1
192.168.0.2 node2
192.168.0.3 node3
192.168.0.4 node4
192.168.0.5 node5
192.168.0.6 node6
192.168.0.7 node7
192.168.0.8 node8
192.168.0.9 node9
```

This enables...

```
ssh john@node1
```

or, the abbreviated...

```
ssh node1
```

provided the user name on the macbook is the same as the Linux user created by cloud-init.

13.2.5 Router/Firewall Configuration

Local network behind firewall/switch: 192.168.0.254

WAN address LAN address

Firewall/Switch (Netgear FVS318G)

Describe DHCP reservations mapping IP to MAC addresses.

Describe ssh access

Add relevant PDFs.

13.2.6 Create the Raspberry Pi Ubuntu Server Image

On macbook...

Download Ubuntu 20.04 LTS 64-bit pre-installed server image for the Raspberry Pi 4...

Double click to uncompress the .xz file which leaves the .img file.

Double click to mount the .img in the filesystem...

Amend /Volumes/system-boot/user-data...

```
#cloud-config

# This is the user-data configuration file for cloud-init. By default this sets
# up an initial user called "ubuntu" with password "ubuntu", which must be
# changed at first login. However, many additional actions can be initiated on
# first boot from this file. The cloud-init documentation has more details:
#
# https://cloudinit.readthedocs.io/
#
# Some additional examples are provided in comments below the default
# configuration.

# On first boot, set the (default) ubuntu user's password to "ubuntu" and
# expire user passwords
chpasswd:
  expire: false
  list:
  - ubuntu:ubuntu
  - john:john

# Enable password authentication with the SSH daemon
ssh_pwauth: true
```

```
## On first boot, use ssh-import-id to give the specific users SSH access to
## the default user
#ssh_import_id:
#- lp:my_launchpad_username
#- gh:my_github_username

## Add users and groups to the system, and import keys with the ssh-import-id
## utility
#groups:
#- robot: [robot]
#- robotics: [robot]
#- pi
#
groups:
- john: [john]

#users:
#- default
#- name: robot
#  gecos: Mr. Robot
#  primary_group: robot
#  groups: users
#  ssh_import_id: foobar
#  lock_passwd: false
#  passwd: $5$hkui88$nvZgIle31cNpryjRfO9uArF7DYiBcWEnjqq7L1AQNN3
users:
- default
- name: john
  gecos: John Duffy
  primary_group: john
  sudo: ALL=(ALL) NOPASSWD:ALL
  shell: /bin/bash
  ssh_authorized_keys:
  - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDGsnzP+1Q6NgeeKFTd/+Mom+UCYJTL/wzIiS9nL

## Update apt database and upgrade packages on first boot
#package_update: true
#package_upgrade: true
package_update: true
package_upgrade: true

## Install additional packages on first boot
#packages:
#- pwgen
#- pastebinit
#- [libpython2.7, 2.7.3-0ubuntu3.1]
```

```
packages:
- git
- tree
- unzip
- iperf
- net-tools
- linux-tools-common
- linux-tools-raspi
- build-essential
- gdb
- openmpi-common
- openmpi-bin
- libblis3-serial
- libblis3-openmp
- libopenblas0-serial
- libopenblas0-openmp

## Write arbitrary files to the file-system (including binaries!)
#write_files:
#- path: /etc/default/keyboard
#  content: |
#    # KEYBOARD configuration file
#    # Consult the keyboard(5) manual page.
#    XKBMODEL="pc105"
#    XKBLAYOUT="gb"
#    XKBVARIANT=""
#    XKBOPTIONS="ctrl: nocaps"
#  permissions: '0644'
#  owner: root:root
#- encoding: gzip
#  path: /usr/bin/hello
#  content: !!binary |
#    H4sIAIDb/U8C/1NW1E/KzNMvzuBKTc7IV8hIzcnJVyjPL8pJ4QIA6N+MVxsAAAA=
#  owner: root:root
#  permissions: '0755'
write_files:
- path: /etc/hosts
  content: |
    127.0.0.1 localhost
    192.168.0.1 node1
    192.168.0.2 node2
    192.168.0.3 node3
    192.168.0.4 node4
    192.168.0.5 node5
    192.168.0.6 node6
    192.168.0.7 node7
```

Figure 3: Using Raspberry Pi Imager to erase and format a MicroSD card.

```
    192.168.0.8 node8
    192.168.0.9 node9
  permissions: '0644'
  owner: root:root

## Run arbitrary commands at rc.local like time
#runcmd:
#- [ ls, -l, / ]
#- [ sh, -xc, "echo $(date) ': hello world!'" ]
#- [ wget, "http://ubuntu.com", -O, /run/mydir/index.html ]
runcmd:
- hostnamectl set-hostname --static node$(hostname -i | cut -d ' ' -f 1 | cut -d
- reboot
```

Eject/unmount .img file

Use Raspberry Pi Imager to erase...

Then use the Raspberry Pi Imager to write preinstalled server image to the MicroSD card...

When complete, remove the MicroSD card from the card reader, place it the Raspberry Pi and plug in the power cable.

Figure 4: Using Raspberry Pi Imager to write the server image to a MicroSD card.

The cloud-init configuration process will now start. The Raspberry Pi will acquire its IP address from the router, setup users, update apt, upgrade the system, download software packages, set the hostname (based on the IP address), and finally the system will reboot.

13.3 Post-Installation Tasks

13.3.1 Enable No Password Access

This is required for Open-MPI...

Our public key was installed on each node by cloud-init. So, we can ssh into each node without a password, and use the abbreviated ssh node1, instead of ssh john@node1 (assuming john is the user name on the workstation).

We need to copy our private key to node1 (only node1)...

```
scp ~/.ssh/id_rsa node1:~/.ssh
```

Then to enable access to nodes node2 to node9 without a password from node1, we need to import the ... keys into the node1 knownhosts file...

This is easily done...

From macbook, ssh into node1...

ssh node1

and then from node1, for each of the nodes node2 to node9:

ssh node2

This will generate...

```
The authenticity of host 'node2 (192.168.0.2)' can't be established.
ECDSA key fingerprint is SHA256:5VgsnN2nPvpfbJmALh3aJdOeT/NvDXqN8TCreQyNaFA.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

responding yes, imports the key into the node1 knownhosts file...

```
exit
```

Next node...

This is only required to be done on intial contact with nodes node2 to node9 (unless the keys on these nodes change)

13.3.2 Uninstall unattended-upgrades

The package unattended-upgrades is installed automatically...

Can potentially interferer with long running benchmarks...

Remove...

From macbookpro:

```
ssh node1 sudo apt remove unattended-upgrades --yes
ssh node2 sudo apt remove unattended-upgrades --yes
ssh node3 sudo apt remove unattended-upgrades --yes
ssh node4 sudo apt remove unattended-upgrades --yes
ssh node5 sudo apt remove unattended-upgrades --yes
ssh node6 sudo apt remove unattended-upgrades --yes
ssh node7 sudo apt remove unattended-upgrades --yes
ssh node8 sudo apt remove unattended-upgrades --yes
ssh node9 sudo apt remove unattended-upgrades --yes
```

Don't forget to update your cluster regularly at convenient times...

See update/upgrade script below...

13.3.3 Create a Project Repository

Xpand upon...

```
ssh node1
mkdir picluster
cd picluster
git init
```

Ensure you do

git add git commit git push

at regular intervals...

13.3.4 Create an System Update/Upgrade Script

Automate...

```
#!/usr/bin/bash

NODES=9

for (( i=$NODES; i>0; i-- ))
do
        echo ""
        echo "UPGRADING node$i..."
        ssh node$i sudo apt update
        ssh node$i sudo apt full-upgrade --yes
        ssh node$i sudo apt autoremove --yes
        ssh node$i sudo shutdown -r now
done
```

13.3.5 Select BLAS Library

We have installed four BLAS libraries...

Confirm all nodes are using the same one initially...

ssh node1 sudo update-alternatives –config libblas.so.3-aarch64-linux-gnu

TODO screen output...

Confirm option 0, OpenBLAS, is selected. Press return to keep this option and then exit.

14 Appendix B - High-Performance Linpack (HPL) Installation

Download and install the latest version of HPL on node1...

```
ssh node1
cd picluster
mkdir hpl
cd hpl
wget https://www.netlib.org/benchmark/hpl/hpl-2.3.tar.gz
gunzip hpl-2.3.tar.gz
tar xvf hpl-2.3.tar
rm hpl-2.3.tar
cd hpl-2.3
```

Create Make.serial file...

```
cd setup
bash make_generic
cd ..
cp setup/Make.UNKNOWN Make.serial
```

Amend Make.serial as per...

Build...

```
make arch=serial
```

This creates xhpl and HPL.dat in bin/serial

Copy xhpl to all nodes (only xhpl, and not HPL.dat)...

```
ssh node2 mkdir -p picluster/hpl/hpl-2.3/bin/serial
ssh node3 mkdir -p picluster/hpl/hpl-2.3/bin/serial
ssh node4 mkdir -p picluster/hpl/hpl-2.3/bin/serial
ssh node5 mkdir -p picluster/hpl/hpl-2.3/bin/serial
ssh node6 mkdir -p picluster/hpl/hpl-2.3/bin/serial
ssh node7 mkdir -p picluster/hpl/hpl-2.3/bin/serial
ssh node8 mkdir -p picluster/hpl/hpl-2.3/bin/serial
ssh node9 mkdir -p picluster/hpl/hpl-2.3/bin/serial

scp bin/serial/xhpl node2:~picluster/hpl/hpl-2.3/bin/serial
scp bin/serial/xhpl node3:~picluster/hpl/hpl-2.3/bin/serial
scp bin/serial/xhpl node4:~picluster/hpl/hpl-2.3/bin/serial
scp bin/serial/xhpl node5:~picluster/hpl/hpl-2.3/bin/serial
```

```
scp bin/serial/xhpl node6:~picluster/hpl/hpl-2.3/bin/serial
scp bin/serial/xhpl node7:~picluster/hpl/hpl-2.3/bin/serial
scp bin/serial/xhpl node8:~picluster/hpl/hpl-2.3/bin/serial
scp bin/serial/xhpl node9:~picluster/hpl/hpl-2.3/bin/serial
```

15 Appendix D - Hints and Tips

Hints from experience... and time savers... for building a development cluster on a local network.

15.1 IP/MAC Addresses

If IP/MAC address assignments get confused, which is easily done during initial build, view IP address assignments on the local network with:

```
arp -a
```

Then delete *incomplete* IP addresses with:

```
sudo arp -d incomplete-ip-address
```

15.2 SSH known_hosts

If *ssh* reports differing keys in 'known-hosts', and warns of a potential 'man-in-the-middle-attack', then just delete 'known-hosts':

```
sudo rm ~/.ssh/known_hosts
```

'known_hosts' will be re-populated as you log into each node.

15.3 tmux

`tmux` is your friend!

Monitoring long running jobs from a workstation, which goes to sleep after a period of no activity, for example, may interfere with the running of the jobs if a SSH connection is broken.

Use a `tmux` session to start long running jobs, and then detach from the `tmux` session. The job will quite happily run in the background on the cluster. Turn the workstation off and go to bed. In the morning, turn the workstation on and 'attach' to the `tmux` session. All will be well.

15.4 git

`git` is your best friend!

During your cluster build you will accidentally delete files, results etc. After every significant...