MSc Scientific Computing Dissertation
ARM Cluster Linpack Benchmark

John Duffy

August 2020

1 Introduction

Objectives.

1. Best achievable High-Performance Linpack performance.

2. An entry in 'The Green500' list!?

3.

etc

This is also intended to be a complete guide for reproducing the build (hardware and software) of the ARM cluster and also for reproducing the results.

Top Gflops.

Describe setup, referencing appendices for detail.

Diagram of setup.

node1 is *master* node.

Code at:

```
https://github.com/johnduffymsc/phas0077.git
```

2 Theoretical Maximum Performance (Gflop/s)

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

4 cores

1.5 GHz

NEON 128 bit SIMD

4 GB memory (our chosen model)

Caches...

Pipeline...

Simplistically, ...

This ignores instructions pipelining benefits...

3 High-Performance Linpack (HPL)

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

3.1 Installation

Reference Appendix... with specifics.

3.2 Tuning

Describe methodology for configuration file parameters

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

3.3 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...

3.4 OpenMPI without OpenMP

Describe processor grid layout.

hosts-with-slots file.

3.5 OpenMPI with OpenMP

Describe processor grid layout.

hosts-no-slots file.

4 Performance Optimisation

4.1 Methodology

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

5 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
```

6 Single Core Optimisation

6.1 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...

7 Single Node Optimisation

8 Cluster Optimisation

8.1 Recompile HPL for Cortex-A72

Block size!

RESULTS

8.2 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

8.3 Recompile OpenMPI for Cortex-A72

Block size!

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

RESULTS

8.4 Network Tuning

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

TODO: Read OpenMPI docs

RESULTS

8.5 Perf/Dtrace Linpack

See if there are any performance bottlenecks

New package OpenBLASRPI4? for RPI4 specific optimisations?

RESULTS

9 Appendix A - Raspberry Pi Cluster Build

9.1 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

9.2 Preliminary Tasks

1. Update the EE-PROM

2. Get MAC address

10 Router/Firewall Configuration

Local network behind firewall/switch: 192.168.0.255

10.1 Firewall/Switch (Netgear FVS318G)

Describe DHCP reservations mapping IP to MAC addresses.

Add relevant PDFs.

10.2 Nodes /etc/hosts

The /etc/hosts file on each node is created by Cloud-Init:

```
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
```

10.3 Workstation /etc/hosts

On workstation 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 reduces the typing of IP addresses, e.g.

```
    ssh ucapjbj@node1
```

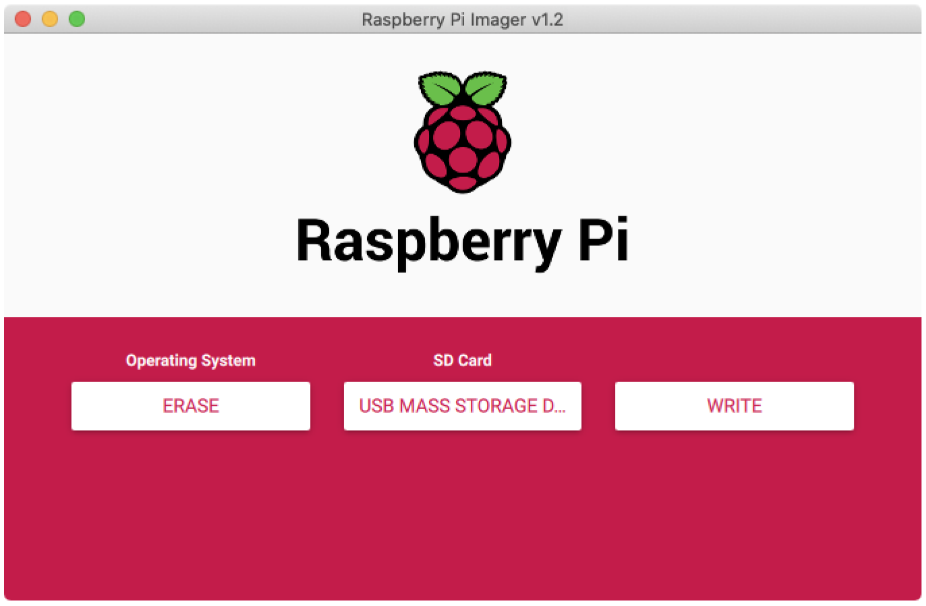

Figure 1: Using Raspberry Pi Imager to erase and format a MicroSD card.

11 Appendix C - Software Configuration

11.1 RPI 4 EEPROM Update

11.2 Ubuntu Server 20.04 LTS

Ubuntu is a popular Linux distribtion... ... Ubuntu 20.04 LTS 64-bit pre-installed server image for the Raspberry Pi 4...

On my MacBook Pro

Download as .img.xz file...

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

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

Amend user-data, vim /Volumes/system-boot/user-data TODO: CHECK THIS!

Eject/unmount .img file

Use Raspberry Pi Imager to erase...

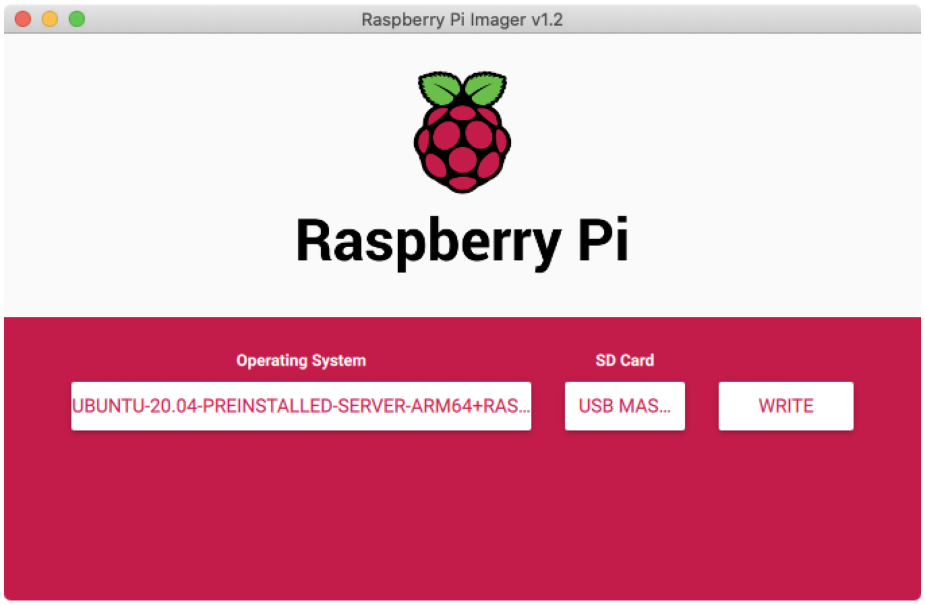

Figure 2: Using Raspberry Pi Imager to write the server image to a MicroSD card.

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.

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.

11.3 Post-Installation Tasks

11.3.1 Copy Private Key to node1

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), so that openmpi can access node2 .. node8 without a password.

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

11.3.2 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
```

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

See update/upgrade script below...

11.3.3 Create a Project Repository

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

11.4 Create an System Update/Upgrade Script

```
ssh node1 sudo apt update
ssh node1 sudo apt full-upgrade --yes
ssh node1 sudo autoremove --yes
ssh node1 sudo shutdown -r now
```

We have installed two 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.

11.5 Cloud-Init

user-data

11.6 MPI Node Authentication

MPI requires 'no-user-interaction' to access nodes. This is achieved using public key authentication.

Generate public and private keys on workstation (no passphrase):

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

1. Copy public key for user 'ucapjbj' into cloud-init 'user-data'

2. Copy private key to node1 once up and running.

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

The user ucapjbj can now log into the other nodes without a password. Likewise OpenMPI, running under the user ucapjbj, can now access the other nodes from node1.

12 High-Performance Linpack Installation

Describe specifics of installation procedure on following pages...

```
cd ~/picluster
mkdir dir 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.rpi4 in top directory

Include copy of Make.rpi4 in appendix.

To build:

```
    make arch=rpi4
```

This places the 'xhpl' executable in

```
.../hpl-2.3/bin/rpi4
```

To clean up:

```
    make arch=rpi4 clean
```

Copy 'xhpl' executable to all nodes...

```
    ssh node2 (etc)
    mkdir -p ~/phas0077/projects/hpl-2.3/bin/rpi4
    exit
    scp bin/rpi4/xhpl ucapjbj@node2:~/phas0077/projects/hpl-2.3/bin/rpi4
```

The above should be capable of being achieved using the mpirun –preload-binary option. To be further investigated. May need xhpl to be built with static OpenMPI and OpenBLAS libraries.

To run:

```
cd bin/rpi4
mpirun -host node1:4 -np 4 xhpl
mpirun -host node1:4,node2:4 -np 8 xhpl
etc
```

or

```
cd bin/rpi4
mpirun -hostfile nodes -np 32 xhpl
```

13 High-Performance Linpack Tuning

Describe specifics of tuning procedure on following pages...

14 Appendix D - Hints and Tips

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

14.1 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
```

14.2 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.

14.3 Package 'unattended-upgrades'

Unattended upgrades can interfere/stop long running jobs. So, remove the package 'unattended-upgrades' from all nodes:

```
sudo apt remove unattended-upgrades
```

But don't forget to do regular manual upgrades!

14.4 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.

14.5 git

`git` is your best friend!

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