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 Performance

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 £...

1.2 Typography

This is a computer name...

`node1`

This is a command to type...

```
$ grep
```

This is a command output displayed on your screen...

Listing 1: cat /proc/softirqs

```
                    CPU0       CPU1       CPU2       CPU3
          HI:          1          0          0
1
       TIMER:    3835342    3454143    3431155    3431023
      NET_TX:      36635          0          0
0
      NET_RX:     509189        146        105        121
       BLOCK:      95326       4367       4311       4256
    IRQ_POLL:          0          0          0
0
     TASKLET:       4900          3          4
25
       SCHED:     444569     267214     218701     189120
     HRTIMER:         67          0          0
0
         RCU:     604466     281455     260784     277699
```

This is a file listing...

Listing 2: /etc/hosts

```
1  ##
2  # Host Database
3  #
4  # localhost is used to configure the loopback interface
5  # when the system is booting.  Do not change this entry.
6  ##
7  127.0.0.1 localhost
8  255.255.255.255 broadcasthost
9  ::1             localhost
10 192.168.0.1 node1
11 192.168.0.2 node2
12 192.168.0.3 node3
13 192.168.0.4 node4
14 192.168.0.5 node5
15 192.168.0.6 node6
16 192.168.0.7 node7
```

```
192.168.0.8 node8
192.168.0.9 node9
```

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 3: Example HPL.dat

```
1  HPLinpack benchmark input file
2  Innovative Computing Laboratory, University of Tennessee
3  HPL.out      output file name (if any)
4  0            device out (6=stdout,7=stderr,file)
5  1            # of problems sizes (N)
6  26208        Ns
7  1            # of NBs
8  32           NBs
9  0            PMAP process mapping (0=Row-,1=Column-major)
10 2            # of process grids (P x Q)
11 1 2          Ps
12 8 4          Qs
13 16.0         threshold
14 3            # of panel fact
15 0 1 2        PFACTs (0=left, 1=Crout, 2=Right)
16 2            # of recursive stopping criterium
17 2 4          NBMINs (>= 1)
18 1            # of panels in recursion
19 2            NDIVs
20 3            # of recursive panel fact.
21 0 1 2        RFACTs (0=left, 1=Crout, 2=Right)
22 1            # of broadcast
23 0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
24 1            # of lookahead depth
25 0            DEPTHs (>=0)
26 2            SWAP (0=bin-exch,1=long,2=mix)
27 64           swapping threshold
28 0            L1 in (0=transposed,1=no-transposed) form
29 0            U  in (0=transposed,1=no-transposed) form
30 1            Equilibration (0=no,1=yes)
31 8            memory alignment in double (> 0)
```

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

4.3 HPL.out

Describe HPL.out...

It is very easy to use `grep`to find the lines in HPL.out containing the results.

And to then conduct a general numeric sort, first by P and then by Gflops, to find Rmax for each P and Q pair, squeezing repeated white space down to a single space for readability.

```
$ grep WR HPL.out | sort -g -k 4 -k 7 | tr -s ' ' > HPL.out.sorted
```

Listing 4: Example HPL.out.sorted

```
WR00C2R2 26208 32 1 8 802.01 1.4965e+01
WR00R2C2 26208 32 1 8 799.75 1.5007e+01
WR00L2L2 26208 32 1 8 796.04 1.5077e+01
WR00C2C2 26208 32 1 8 794.65 1.5103e+01
WR00L2C2 26208 32 1 8 793.86 1.5118e+01
WR00C2L2 26208 32 1 8 793.67 1.5122e+01
WR00R2L2 26208 32 1 8 793.48 1.5126e+01
WR00R2R2 26208 32 1 8 790.26 1.5187e+01
WR00L2R2 26208 32 1 8 789.16 1.5208e+01
WR00R2L4 26208 32 1 8 774.49 1.5497e+01
WR00C2R4 26208 32 1 8 773.52 1.5516e+01
WR00L2L4 26208 32 1 8 770.20 1.5583e+01
WR00R2C4 26208 32 1 8 767.92 1.5629e+01
WR00L2C4 26208 32 1 8 763.10 1.5728e+01
WR00L2R4 26208 32 1 8 762.43 1.5742e+01
WR00R2R4 26208 32 1 8 761.92 1.5752e+01
WR00C2C4 26208 32 1 8 761.58 1.5759e+01
WR00C2L4 26208 32 1 8 757.87 1.5836e+01
WR00R2R2 26208 32 2 4 728.78 1.6468e+01
WR00R2C2 26208 32 2 4 728.21 1.6481e+01
WR00R2L2 26208 32 2 4 726.55 1.6519e+01
WR00C2R2 26208 32 2 4 722.38 1.6614e+01
WR00L2C2 26208 32 2 4 721.63 1.6632e+01
WR00L2L2 26208 32 2 4 721.54 1.6634e+01
WR00C2C2 26208 32 2 4 721.25 1.6640e+01
WR00C2L2 26208 32 2 4 720.82 1.6650e+01
WR00L2R2 26208 32 2 4 720.80 1.6651e+01
WR00L2R4 26208 32 2 4 692.09 1.7341e+01
WR00R2C4 26208 32 2 4 690.37 1.7385e+01
WR00C2L4 26208 32 2 4 686.69 1.7478e+01
WR00C2C4 26208 32 2 4 686.23 1.7489e+01
WR00C2R4 26208 32 2 4 686.08 1.7493e+01
WR00L2L4 26208 32 2 4 686.02 1.7495e+01
WR00L2C4 26208 32 2 4 685.88 1.7498e+01
WR00R2L4 26208 32 2 4 685.76 1.7502e+01
WR00R2R4 26208 32 2 4 684.45 1.7535e+01
```

4.4 Running xhpl

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

```
$ ~/picluster/tools/picluster-set-libblas-openblas-serial
```

or, with the serial version of BLIS...

```
$ ~/picluster/tools/picluster-set-libblas-blis-serial
```

```
cd ~/picluster/hpl/hpl-2.3/bin/serial
mpirun -np 4 xhpl
```

5 OpenMPI

What is OpenMPI...

6 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 memory...

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...

6.1 OpenBLAS

6.2 BLIS

6.3 1 Core Baseline

Problem size restricted to 80% of memory...

NB 32 to 256 in increments of 8...

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	-	-

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...

6.4 1 Node Baseline

1x4

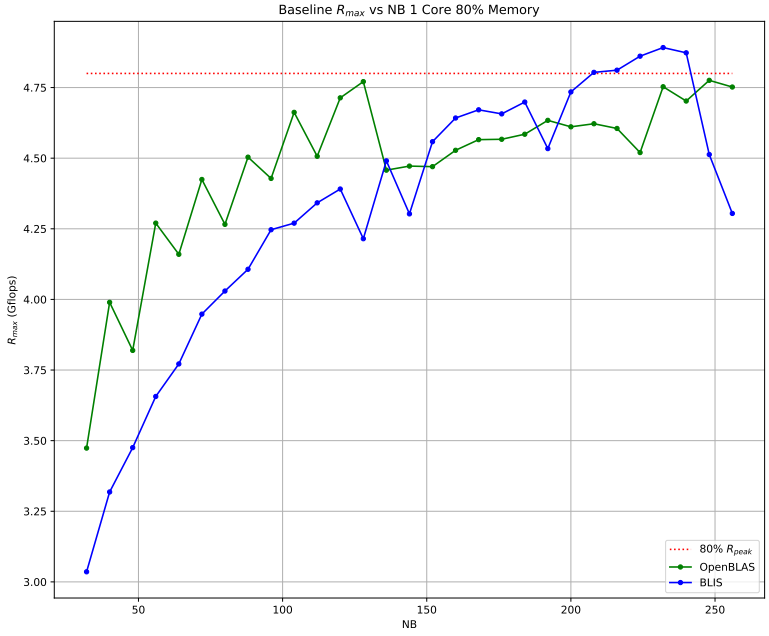

Figure 1: R_{max} vs NB 1 Core using 80% memory.

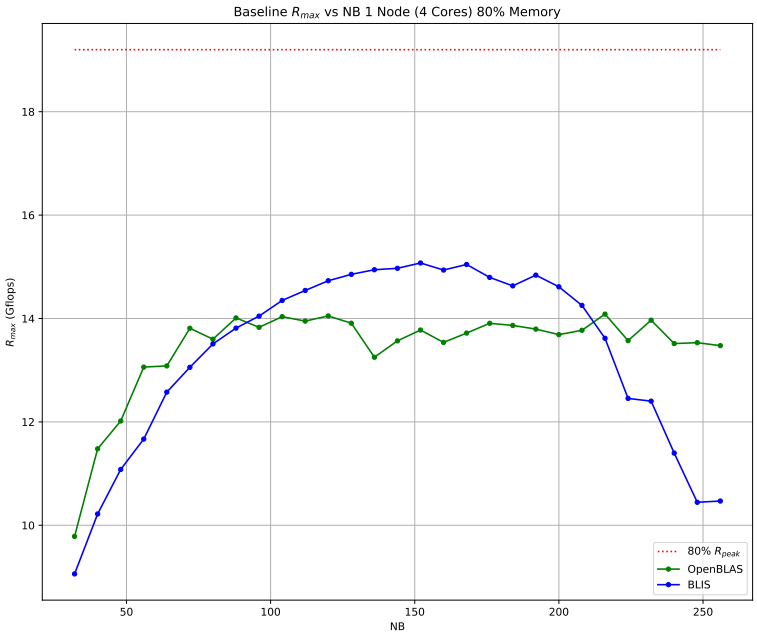

Figure 2: R_{max} vs NB 1 Node (4 cores) using 80% memory.

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 $np \geq 2$

6.5 2 Node Baseline

P1 x Q8

P2 x Q4

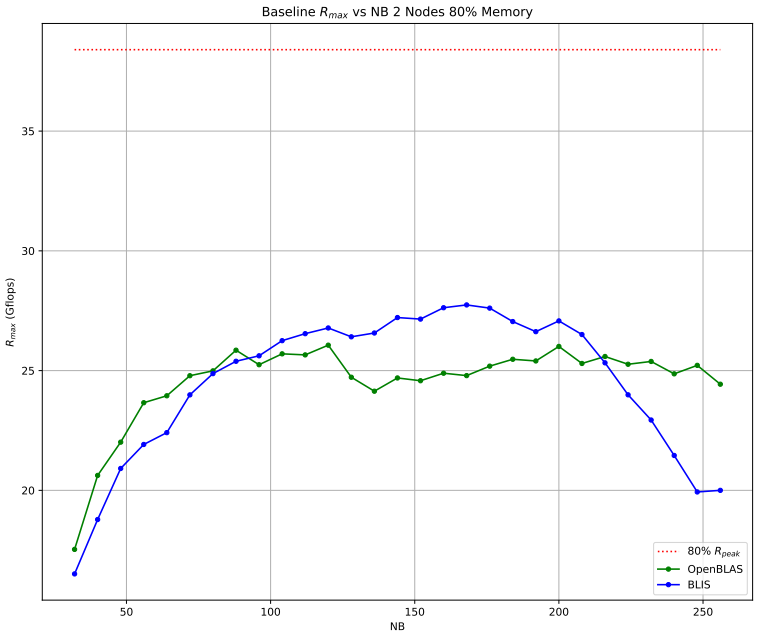

Figure 3: R_{max} vs NB 2 Nodes using 80% memory.

NB	N	NB	N	NB	N	NB	N	NB	N
32	26208	80	26160	128	26112	176	26048	224	26208
40	26200	88	26136	136	26112	184	26128	232	25984
48	26208	96	26208	144	26208	192	26112	240	26160
56	26208	104	26208	152	26144	200	26200	248	26040
64	26176	112	26208	160	26080	208	26208	256	26112
72	26208	120	26160	168	26208	216	26136	-	-

6.6 8 Node Baseline

1x32 2x16 4x8

6.7 Observations

Best NB...

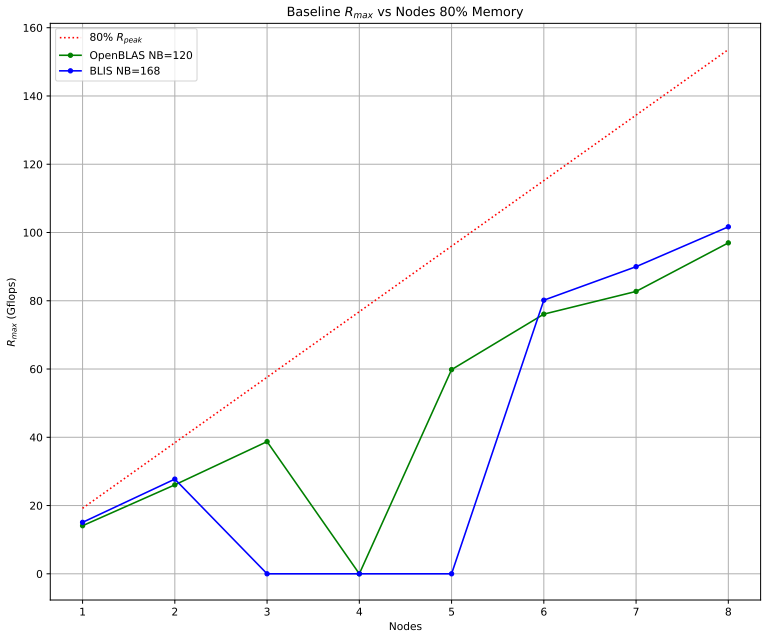

Figure 4: R_{max} vs Nodes using 80% memory.

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

Iperf...

htop...

top...

perf...

cache misses...

software interrupts...

Suggests... improve network efficiency?

7 Optimisations

7.1 Single Core Optimisation

7.1.1 Rebuild libopenblas0-serial

Better BLAS library...

The Debian Science Wiki suggests...

So, following the instructions in /usr/local/share/

Details are in Appendix ?...

Poking around in the OpenBLAS source code, I noticed...

cpuid_arm64.c

in function void get_cpuconfig(void)

Listing 5: cpuid_arm64.c

```
...
  case CPU_CORTEXA57:
  case CPU_CORTEXA72:
  case CPU_CORTEXA73:
    // Common minimum settings for these Arm cores
    // Can change a lot, but we need to be conservative
    // TODO: detect info from /sys if possible
    printf("#define %s\n", cpuname[d]);
    printf("#define L1_CODE_SIZE 49152\n");
    printf("#define L1_CODE_LINESIZE 64\n");
    printf("#define L1_CODE_ASSOCIATIVE 3\n");
    printf("#define L1_DATA_SIZE 32768\n");
    printf("#define L1_DATA_LINESIZE 64\n");
    printf("#define L1_DATA_ASSOCIATIVE 2\n");
    printf("#define L2_SIZE 524288\n");
    printf("#define L2_LINESIZE 64\n");
    printf("#define L2_ASSOCIATIVE 16\n");
    printf("#define DTB_DEFAULT_ENTRIES 64\n");
    printf("#define DTB_SIZE 4096\n");
    break;
...
```

REFERENCE: Arm...

The following two lines are incorrect for the Arm Cortex-A72:

```
    printf("#define L2_SIZE 524288\n");
    printf("#define DTB_DEFAULT_ENTRIES 64\n");
```

To reflect the 1MB of L2 cache of the BCM??????, and the 32 entry L1 Data TLB, they should be:

```
    printf("#define L2_SIZE 1048576\n");
    printf("#define DTB_DEFAULT_ENTRIES 32\n");
```

Having changed these to the correct values, the build process now accurately reflects the 1MB of L2 cache on line 18 of 0-serial/config.h from which the libopenblas0-serial package is built:

Listing 6: 0-serial/config.h

```
1  #define OS_LINUX 1
2  #define ARCH_ARM64 1
3  #define C_GCC 1
4  #define __64BIT__ 1
5  #define PTHREAD_CREATE_FUNC pthread_create
6  #define BUNDERSCORE _
7  #define NEEDBUNDERSCORE 1
8  #define ARMV8
9  #define HAVE_NEON
10 #define HAVE_VFPV4
11 #define CORTEXA72
12 #define L1_CODE_SIZE 49152
13 #define L1_CODE_LINESIZE 64
14 #define L1_CODE_ASSOCIATIVE 3
15 #define L1_DATA_SIZE 32768
16 #define L1_DATA_LINESIZE 64
17 #define L1_DATA_ASSOCIATIVE 2
18 #define L2_SIZE 1048576
19 #define L2_LINESIZE 64
20 #define L2_ASSOCIATIVE 16
21 #define DTB_DEFAULT_ENTRIES 64
22 #define DTB_SIZE 4096
23 #define NUM_CORES 4
24 #define CHAR_CORENAME "CORTEXA72"
25 #define GEMM_MULTITHREAD_THRESHOLD 4
```

On completion of the build process, and after uninstalling the original libopenblas0-serial package and installing the new one...

Discussion...

7.1.2 Rebuild libblis3-serial

7.2 Single Node Optimisation

7.2.1 Kernel Preemption Model

The Linux kernel has 3 Preemption Models...

1... 2... The default 3...

As per the Help in the Kernel Configuration...

Listing 7: Kernel Configuration Preemption Model Help

```
CONFIG_PREEMPT_NONE:

This is the traditional Linux preemption model, geared towards
throughput. It will still provide good latencies most of the
time, but there are no guarantees and occasional longer delays
are possible.

Select this option if you are building a kernel for a server or
scientific/computation system, or if you want to maximize the
raw processing power of the kernel, irrespective of scheduling
latencies.
```

So, kernel rebuilt with CONFIG_PREEMPT_NONE=y

See Appendix ? on how to rebuild the kernel...

Installed on each node...

So, although this optimisation applies to single node, the benefits of applying this optimisation may not be apparent until the kernel has to juggle networking etc...

RESULTS...

7.2.2 Recieve Queues

```
$ sudo perf record mpirun -allow-run-as-root -np 4 xhpl
```

Running xhpl on 8 nodes using OpenBLAS...

```
$ mpirun -host node1:4 ... node8:4 -np 32 xhpl
```

SHORTLY AFTER PROGRAM START...

On node1,... where we initiated...

top...

```
top - 20:33:15 up 8 days,  6:02,  1 user,  load average: 4.02, 4.03, 4.00
Tasks: 140 total,   5 running, 135 sleeping,   0 stopped,
0 zombie
%Cpu(s): 72.5 us, 21.7 sy,  0.0 ni,  0.0 id,  0.0 wa,
0.0 hi,  5.8 si,  0.0 st
MiB Mem :   3793.3 total,    330.1 free,   3034.9 used,
428.3 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.
698.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU
%MEM     TIME+ COMMAND
  34884 john      20   0  932964 732156   7980 R 100.3
18.8 106:40.29 xhpl
  34881 john      20   0  933692 732272   7916 R 100.0
18.9 107:29.75 xhpl
  34883 john      20   0  932932 731720   8136 R  99.3
18.8 107:33.25 xhpl
  34882 john      20   0  932932 731784   8208 R  97.7
18.8 107:33.64 xhpl
```

SOFTIRQS...

NODE 2 - 2 NODES ONLY TO SEE EFFECT...

IPERF!!!

On node8, running the top command...

```
$ top
```

We can see...

```
top - 18:58:44 up 8 days,  4:29,  1 user,  load average: 4.00, 3.75, 2.35
Tasks: 133 total,   5 running, 128 sleeping,   0 stopped,
0 zombie
%Cpu(s): 50.7 us, 47.8 sy,  0.0 ni,  0.0 id,  0.0 wa,
0.0 hi,  1.4 si,  0.0 st
MiB Mem :   3793.3 total,    392.7 free,   2832.6 used,
568.0 buff/cache
```

```
MiB Swap:      0.0 total,      0.0 free,      0.0 used.
901.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU
%MEM     TIME+ COMMAND
  23928 john      20   0  883880 682456   8200 R 100.0
17.6  13:14.17 xhpl
  23927 john      20   0  883988 682432   7932 R  99.7
17.6  13:12.58 xhpl
  23930 john      20   0  883912 682664   7832 R  99.7
17.6  13:17.01 xhpl
  23929 john      20   0  883880 682640   8376 R  99.3
17.6  13:16.25 xhpl
```

Indicates that only 50.7% of CPU time is being utilised by user programs (us), Linpack/OpenMPI...

I hypothesise that the 1.4% of software interrupts (si) is responsible 47.8% of CPU time in the kernel (sy) servicing these interupts...

Lets have a look at the software interrupts on the system...

```
$ watch -n 1 cat /proc/softirqs
```

```
Every 1.0s: cat /proc/softirqs

                    CPU0       CPU1       CPU2       CPU3
          HI:           0          1          0
1
       TIMER:  122234556   86872295   85904119   85646345
      NET_TX:  222717797     228381     147690     144396
      NET_RX: 1505715680       1132       1294       1048
       BLOCK:      63160      11906      13148      11223
    IRQ_POLL:          0          0          0
0
     TASKLET:   58902273         33          2
6
       SCHED:    3239933    3988327    2243001    2084571
     HRTIMER:       8116         55         53
50
         RCU:    6277982    4069531    4080009    3994395
```

As can be seen...

1. the majority of software interrupts are being generated by network receive (NET_RX) activity, followed by network transmit activity (NET_TX)...

2. these interrupts are being almost exclusively handled by CPU0...

What is there to be done?...

1. Reduce the numbers of interrupts...

1.1 Each packet produces an interrupt - interrupt coalesing...

1.2 Reduce the number of packets - increase MTU...

2.1 Share the interrupt servicing activity evenly across the CPUs...

7.3 Cluster Optimisation

On node2 start the Iperf server...

```
$ iperf -s
```

On node1 start the Iperf client...

```
$ iperf -c
```

ping tests of MTU...

iperf network speed...

7.3.1 Jumbo Frames

Requires a network switch capable of Jumbo frames...

Appendix ? - Pi Cluster Build Instructions

7.4 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

7.5 Preliminary Tasks

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

7.5.1 Update Raspberry Pi EE-PROMs

7.5.2 Get Raspberry Pi MAC Addresses

7.5.3 Generate User Key Pair

On macbook (no passphrase):

```
$ ssh-genkey -t rsa -C john
```

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

7.5.4 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.

7.5.5 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.

7.5.6 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 s
# up an initial user called "ubuntu" with password "ubuntu", which must be
# changed at first login. However, many additional actions can be initiated
# 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-i
## 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/wzI

## 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
    192.168.0.8 node8
    192.168.0.9 node9
```

Figure 5: Using Raspberry Pi Imager to erase and format a MicroSD card.

```
  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 | cu
- 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.

The cloud-init configuration process will now start. The Raspberry Pi will ac-

Figure 6: Using Raspberry Pi Imager to write the server image to a MicroSD card.

quire 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.

7.6 Post-Installation Tasks

7.6.1 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)

7.6.2 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...

7.6.3 Add Source Repositories

We are going to be rebuilding some packages from source...

```
ssh node1
sudo touch /etc/apt/sources/list.d/picluster.list
sudo vim /etc/apt/sources/list.d/picluster.list
```

... and add the following source repositories...

Listing 8: /etc/apt/sources.list.d/picluster.list

```
deb-src http://archive.ubuntu.com/ubuntu focal main universe
deb-src http://archive.ubuntu.com/ubuntu focal-updates main universe
```

... and then update the repository cache?

```
sudo apt update
```

7.6.4 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...

7.6.5 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.

Appendix ? - 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
```

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
```

Appendix ? - 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
```

Appendix ? - Rebuild OpenBLAS

```
$ ssh node1
$ mkdir -p build/openblas
$ chown -R _apt:root build
$ cd build/openblas
$ sudo apt-get source openblas
$ sudo apt-get build-dep openblas
$ cd openblas-0.3.8+ds
```

Edit cpuid_arm64.c...

```
$ sudo cp cpuid_arm64.c cpuid_arm64.c.original
$ sudo vim cpuid_arm64.c
```

```
$ diff cpuid_arm64.c cpuid_arm64.c.original
```

```
275c275
<       printf("#define L2_SIZE 1048576\n");
---
>       printf("#define L2_SIZE 524288\n");
278c278
<       printf("#define DTB_DEFAULT_ENTRIES 32\n");
---
>       printf("#define DTB_DEFAULT_ENTRIES 64\n");
```

And, then following the instructions in debian/README.Debian

```
$ DEB_BUILD_OPTIONS=custom dpkg-buildpackage -uc -b
```

Once the build is complete..

```
cd ..
$ sudo apt remove libopenblas0-serial
$ sudo dpkg -i libopenblas0-serial\_0.3.8+ds-1\_arm64.deb
```

Ensure the correct BLAS library is being used...

```
$ sudo update-alternatives --config libblas.so.3-aarch64-linux-gnu
```

copy to other nodes remove old... install new...

If more than one BLAS library is installed, check update-alternatives!!!

ssh node2 .. node8

```
$ ssh node2 sudo apt remove libblas0-serial
$ scp libopenblas0-serial\_0.3.8+ds-1\_arm64.deb node2:~
$ ssh sudo dpkg -i libopenblas0-serial\_0.3.8+ds-1\_arm64.deb
$ ssh sudo update-alternatives --config libblas.so.3-aarch64-linux-gnu
```

Appendix ? - Rebuild libblis3-serial

```
ssh node1
mkdir -p picluster/build/blis
cd picluster/build/blis
apt-get source blis
sudo apt-get build-dep blis
cd blis-0.6.1
```

Appendix ? - Hints

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

7.7 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
```

7.8 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.

7.9 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.

7.10 git

`git` is your best friend!

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

Appendix ? - cloud-init user-data

Listing 9: picluster/cloudinit/user-data

```
#cloud-config

# This is the user-data configuration file for cloud-init. By default this s
# up an initial user called "ubuntu" with password "ubuntu", which must be
# changed at first login. However, many additional actions can be initiated
# 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-i
## 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/wzI

## 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
    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 | cu
- reboot
```

Appendix ? - Pi Cluster Tools

Listing 10: picluster/tools/picluster-update

```
#!/usr/bin/bash

# A simple bash script to upgrade the cluster.

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
```

Listing 11: picluster/tools/picluster-reboot

```
#!/usr/bin/bash

# A simple bash script to reboot the cluster.

NODES=9

for (( i=$NODES; i>0; i-- ))
do
  echo "Rebooting node$i..."
  ssh node$i sudo shutdown -r now
done
```

Listing 12: picluster/tools/picluster-shutdown

```
#!/usr/bin/bash

# A simple bash script to shutdown the cluster.

NODES=9

for (( i=$NODES; i>0; i-- ))
do
  echo "Shutting down node$i..."
  ssh node$i sudo shutdown -h now
done
```

Listing 13: picluster/tools/picluster-libblas-query

```
#!/usr/bin/bash

# A simple bash script to query the current alternative for libblas.

NODES=9

for (( i=1; i<=$NODES; i++ ))
do
  printf "node$i: "
  ssh node$i update-alternatives --query libblas.so.3-aarch64-linux-gnu \
    | grep Value: \
    | gawk '{print $2}'
done
```

Listing 14: picluster/tools/picluster-libblas-set-openblas-serial

```
#!/usr/bin/bash

# A simple bash script to query the current alternative for libblas.

NODES=9

for (( i=1; i<=$NODES; i++ ))
do
  printf "node$i: "
  ssh node$i update-alternatives --query libblas.so.3-aarch64-linux-gnu \
    | grep Value: \
    | gawk '{print $2}'
done
```

Listing 15: picluster/tools/picluster-libblas-set-openblas-openmp

```
#!/usr/bin/bash

# A simple bash script to query the current alternative for libblas.

NODES=9

for (( i=1; i<=$NODES; i++ ))
do
  printf "node$i: "
  ssh node$i update-alternatives --query libblas.so.3-aarch64-linux-gnu \
    | grep Value: \
    | gawk '{print $2}'
done
```

Listing 16: picluster/tools/picluster-libblas-set-blis-serial

```
#!/usr/bin/bash

# A simple bash script to query the current alternative for libblas.

NODES=9

for (( i=1; i<=$NODES; i++ ))
do
  printf "node$i: "
  ssh node$i update-alternatives --query libblas.so.3-aarch64-linux-gnu \
    | grep Value: \
    | gawk '{print $2}'
done
```

Listing 17: picluster/tools/picluster-libblas-set-blis-openmp

```
#!/usr/bin/bash

# A simple bash script to query the current alternative for libblas.

NODES=9

for (( i=1; i<=$NODES; i++ ))
do
  printf "node$i: "
  ssh node$i update-alternatives --query libblas.so.3-aarch64-linux-gnu \
    | grep Value: \
    | gawk '{print $2}'
done
```

Appendix ? - Arm Performance Libraries

This does not work yet! HPL will build, but raises an illegal instruction error at runtime. At the time of writing, Arm Performance Libraries release 20.2.0 require a minimum of armv8.1-a. Unfortunately, the Raspberry Pi's Cortex-A72 cores are armv8.0-a. The next release will support armv8.0-a. Appendix included for future reference.

"Arm Performance Libraries provides optimized standard core math libraries for high-performance computing applications on Arm processors. This free version of the libraries provides optimized libraries for Arm® Neoverse™ N1-based Armv8 AArch64 implementations that are compatible with various versions of GCC. You do not require a license for this version of the libraries."

Downloaded Arm Performance Libraries 20.2.0 with GCC 9.3 for Ubuntu 16.04+.

```
$ ssh node1
$ sudo apt install environment-modules
$ mkdir picluster/armpl
$ cd picluster/armpl
$ tar xvf arm-performance-libraries_20.2_Ubuntu-16.04_gcc-9.3.tar
$ rm arm-performance-libraries_20.2_Ubuntu-16.04_gcc-9.3.tar
$ sudo ./arm-performance-libraries_20.2_Ubuntu-16.04.sh
```

The default installation directory is /opt/arm...

TODO: CHANGE TO /usr/local + ldconfig

Compile HPL with armpl...

```
$ cd ~/picluster/hpl/hpl-2.3
$ cp Make.serial Make.armpl-serial
```

Edit Make.armpl-serial...

Listing 18: Make.armpl-serial extract

```
1 # ----------------------------------------------------------------
2 # - Linear Algebra library (BLAS or VSIPL) ----------------------
3 # ----------------------------------------------------------------
4 # LAinc tells the  C  compiler where to find the Linear Algebra
  library
5 # header files,  LAlib  is defined  to be the name of
  the library to be
6 # used. The variable LAdir is only used for defining LAinc and LAlib.
7 #
8 LAdir        = /opt/arm/armpl_20.2_gcc-9.3
```

```
9  LAinc        =
10 LAlib        = -L$(LAdir)/lib -larmpl -lgfortran -lamath -lm
```

Compile HPL...

```
$ make arch=armpl-serial
```