

MSc Scientific Computing Dissertation  
ARM Cluster Benchmarks

John Duffy

September 2020

# Contents

<b>I</b>	<b>Project Report</b>	<b>4</b>
1	Introduction	5
2	ARM Architectures for HP	6
3	The Aerin Cluster	7
<b>II</b>	<b>Benchmarks</b>	<b>8</b>
4	High Performance Linpack (HPL) Benchmark	9
4.1	Introduction . . . . .	9
5	HPC Challenge (HPCC) Benchmark	10
5.1	Introduction . . . . .	10
6	High Performance Conjugate Gradients (HPCG) Benchmark	11
6.1	Introduction . . . . .	11
6.2	Introduction . . . . .	11
6.2.1	Aims . . . . .	11
6.2.2	Typography . . . . .	12

6.3	Raspberry Pi 4 Model B . . . . .	13
6.3.1	Description . . . . .	13
6.3.2	Theoretical Maximum Performance (Gflop/s) . . . . .	13
6.4	Pi Cluster . . . . .	14
6.5	High-Performance Linpack (HPL) Benchmark . . . . .	14
6.5.1	Building and Installing HPL . . . . .	14
6.5.2	HPL.dat . . . . .	15
6.5.3	HPL.out . . . . .	15
6.5.4	Running xhpl . . . . .	16
6.6	OpenMPI . . . . .	17
6.7	OpenMPI Baseline Benchmarks . . . . .	17
6.7.1	OpenBLAS . . . . .	18
6.7.2	BLIS . . . . .	18
6.7.3	1 Core Baseline . . . . .	18
6.7.4	1 Node Baseline . . . . .	18
6.7.5	2 Node Baseline . . . . .	20
6.7.6	8 Node Baseline . . . . .	21
6.7.7	Observations . . . . .	21
6.8	Optimisations . . . . .	24
6.8.1	Single Core Optimisation . . . . .	24
6.8.2	Single Node Optimisation . . . . .	26
6.8.3	Cluster Optimisation . . . . .	28
6.8.4	Kernel TCP Parameters Tuning . . . . .	30

<b>III</b>	<b>Build Instructions</b>	<b>33</b>
<b>7</b>	<b>The Aerin Cluster</b>	<b>34</b>
7.1	Introduction . . . . .	34
7.2	Preliminary Tasks . . . . .	35
7.2.1	Update Raspberry Pi EE-PROMs . . . . .	35
7.2.2	Obtain Raspberry Pi MAC Addresses . . . . .	35
7.2.3	Generate User Key Pair . . . . .	35
7.2.4	Amend macbook /etc/hosts . . . . .	35
7.2.5	Router/Firewall Configuration . . . . .	35
7.3	Ubuntu 20.04 64-bit LTS Installation . . . . .	37
7.3.1	Create the Installation Image . . . . .	37
7.4	Post-Installation Tasks . . . . .	41
7.4.1	Enable No Password Access . . . . .	41
7.4.2	Uninstall unattended-upgrades . . . . .	42
7.4.3	Add Ubuntu Source Repositories . . . . .	42
7.4.4	Create a Project Repository . . . . .	42
7.4.5	Select BLAS Library . . . . .	43
7.5	Install High-Performance Linpack (HPL) . . . . .	45
7.6	Kernel Build Procedure . . . . .	50
7.7	Build Kernel with Jumbo Frames Support . . . . .	52
7.8	Rebuild OpenBLAS . . . . .	53
7.9	Rebuild BLIS . . . . .	55
7.10	Pi Cluster Tools . . . . .	56
7.11	Build OpenMPI from Source . . . . .	58
7.12	Arm Performance Libraries . . . . .	59

## Part I

# Project Report

## Chapter 1

# Introduction

## Chapter 2

# ARM Architectures for HP

## Chapter 3

# The Aerin Cluster



## Part II

# Benchmarks

## Chapter 4

# High Performance Linpack (HPL) Benchmark

### 4.1 Introduction

## Chapter 5

# HPC Challenge (HPCC) Benchmark

### 5.1 Introduction

## Chapter 6

# High Performance Conjugate Gradients (HPCG) Benchmark

### 6.1 Introduction

### 6.2 Introduction

<https://github.com/johnduffymsc/picluster>

#### 6.2.1 Aims

**Investigate Maximum Achievable Linpack Performance**

Efficiency... achieved vs theoretical maximum

**Investigate Gflops/Watt**

Green500 ranking...

## Overview of Competitive Available Gflops/£

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

Plot Gflops vs £...

### 6.2.2 Typography

This is a computer name...

node1

This is a command to type...

```
$ cat /proc/softirqs
```

And this is the command output...

	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 6.1: /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
17 192.168.0.8 node8
18 192.168.0.9 node9
```

## 6.3 Raspberry Pi 4 Model B

### 6.3.1 Description

Photo...

Description...

Highlights...

Limitations...

Reference data sheet in Appendix...

### 6.3.2 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...

## 6.4 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...`

## 6.5 High-Performance Linpack (HPL) Benchmark

Reference Paper...

[https://www.netlib.org/benchmark/hpl/...](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>

### 6.5.1 Building and Installing HPL

See Appendix...

## 6.5.2 HPL.dat

Describe HPL.dat parameters...

Listing 6.2: 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 ...

## 6.5.3 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 6.3: Example HPL.out.sorted

```
1 WR00C2R2 26208 32 1 8 802.01 1.4965e+01
2 WR00R2C2 26208 32 1 8 799.75 1.5007e+01
3 WR00L2L2 26208 32 1 8 796.04 1.5077e+01
4 WR00C2C2 26208 32 1 8 794.65 1.5103e+01
5 WR00L2C2 26208 32 1 8 793.86 1.5118e+01
6 WR00C2L2 26208 32 1 8 793.67 1.5122e+01
7 WR00R2L2 26208 32 1 8 793.48 1.5126e+01
8 WR00R2R2 26208 32 1 8 790.26 1.5187e+01
9 WR00L2R2 26208 32 1 8 789.16 1.5208e+01
10 WR00R2L4 26208 32 1 8 774.49 1.5497e+01
11 WR00C2R4 26208 32 1 8 773.52 1.5516e+01
12 WR00L2L4 26208 32 1 8 770.20 1.5583e+01
13 WR00R2C4 26208 32 1 8 767.92 1.5629e+01
14 WR00L2C4 26208 32 1 8 763.10 1.5728e+01
15 WR00L2R4 26208 32 1 8 762.43 1.5742e+01
16 WR00R2R4 26208 32 1 8 761.92 1.5752e+01
17 WR00C2C4 26208 32 1 8 761.58 1.5759e+01
18 WR00C2L4 26208 32 1 8 757.87 1.5836e+01
19 WR00R2R2 26208 32 2 4 728.78 1.6468e+01
20 WR00R2C2 26208 32 2 4 728.21 1.6481e+01
21 WR00R2L2 26208 32 2 4 726.55 1.6519e+01
22 WR00C2R2 26208 32 2 4 722.38 1.6614e+01
23 WR00L2C2 26208 32 2 4 721.63 1.6632e+01
24 WR00L2L2 26208 32 2 4 721.54 1.6634e+01
25 WR00C2C2 26208 32 2 4 721.25 1.6640e+01
26 WR00C2L2 26208 32 2 4 720.82 1.6650e+01
27 WR00L2R2 26208 32 2 4 720.80 1.6651e+01
28 WR00L2R4 26208 32 2 4 692.09 1.7341e+01
29 WR00R2C4 26208 32 2 4 690.37 1.7385e+01
30 WR00C2L4 26208 32 2 4 686.69 1.7478e+01
31 WR00C2C4 26208 32 2 4 686.23 1.7489e+01
32 WR00C2R4 26208 32 2 4 686.08 1.7493e+01
33 WR00L2L4 26208 32 2 4 686.02 1.7495e+01
34 WR00L2C4 26208 32 2 4 685.88 1.7498e+01
35 WR00R2L4 26208 32 2 4 685.76 1.7502e+01
36 WR00R2R4 26208 32 2 4 684.45 1.7535e+01
```

#### 6.5.4 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
```

## 6.6 OpenMPI

What is OpenMPI...

## 6.7 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 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...

### 6.7.1 OpenBLAS

### 6.7.2 BLIS

### 6.7.3 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...

### 6.7.4 1 Node Baseline

1x4

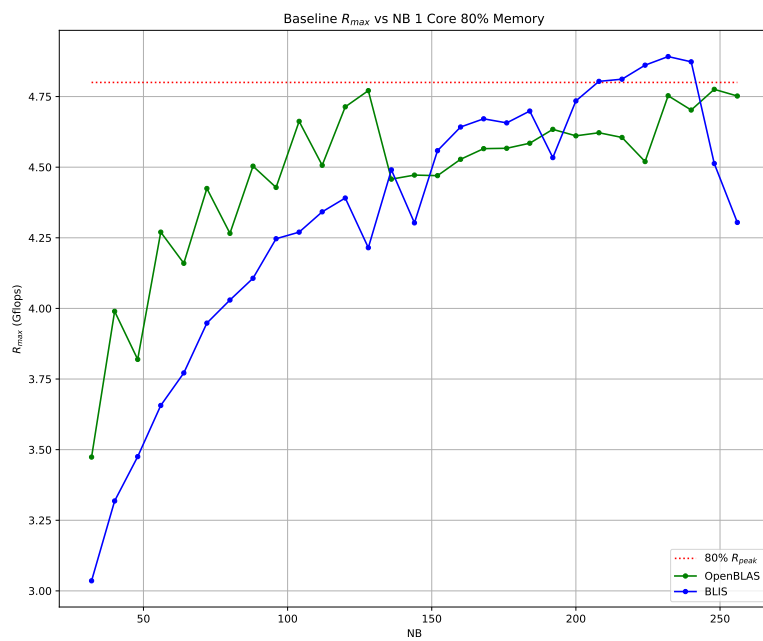


Figure 6.1:  $R_{max}$  vs NB 1 Core using 80% memory.

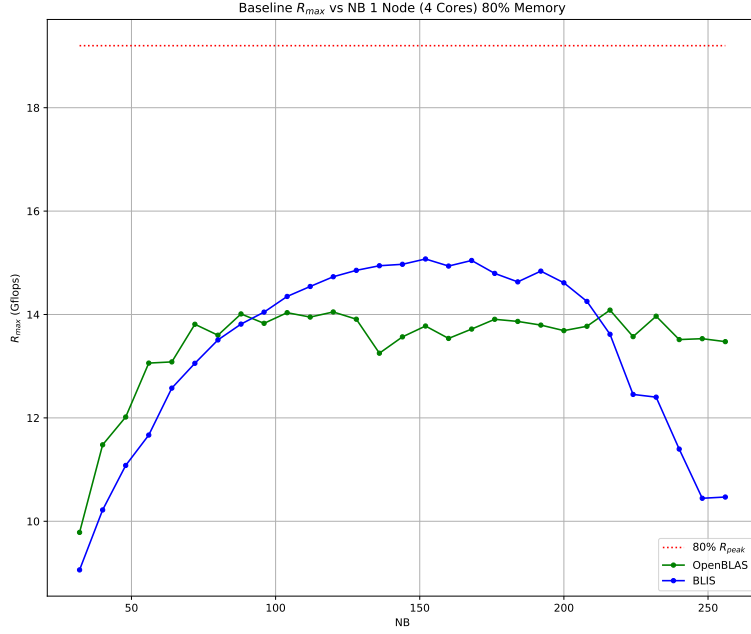


Figure 6.2:  $R_{max}$  vs NB 1 Node (4 cores) using 80% memory.

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  $np \geq 2$

### 6.7.5 2 Node Baseline

P1 x Q8

P2 x Q4

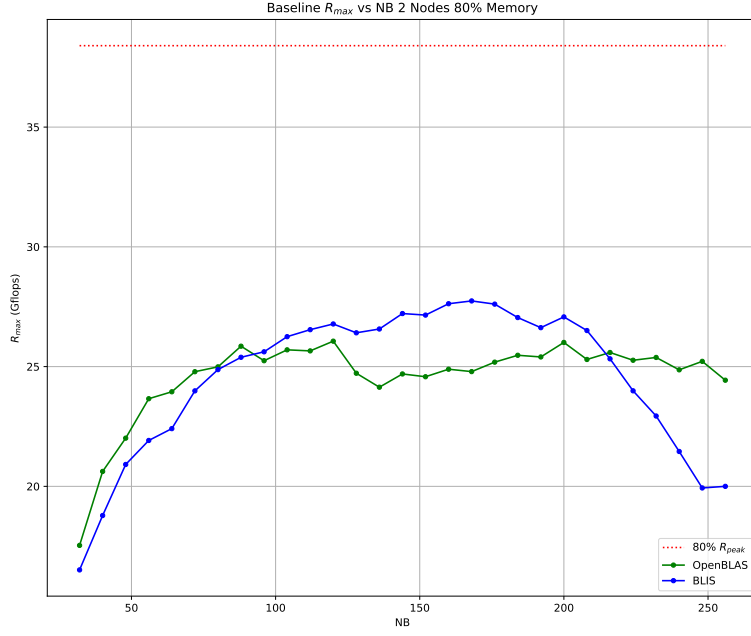


Figure 6.3:  $R_{max}$  vs NB 2 Nodes using 80% memory.

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	-	-

### 6.7.6 8 Node Baseline

1x32 2x16 4x8

### 6.7.7 Observations

Best NB...

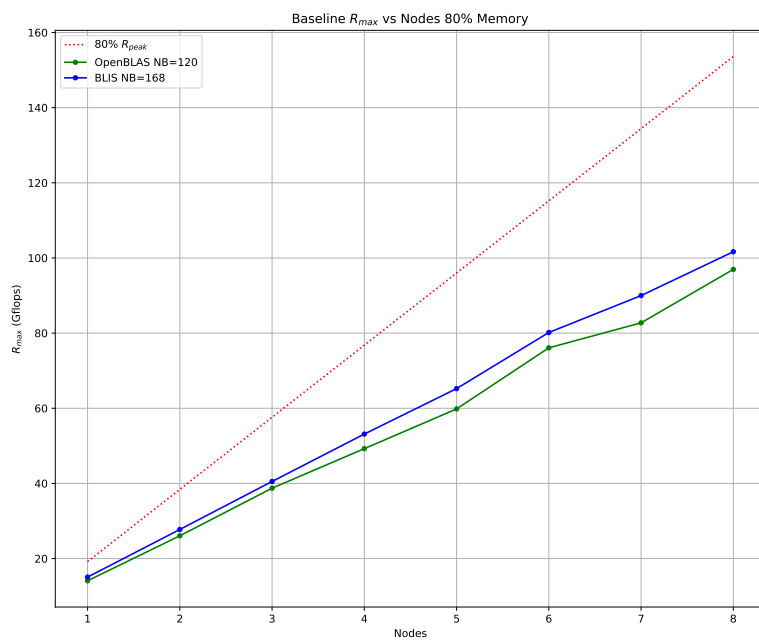


Figure 6.4:  $R_{max}$  vs Nodes using 80% memory.

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

Iperf...

htop...

top...

perf...

cache misses...

software interrupts...

Suggests... improve network efficiency?



## 6.8 Optimisations

### 6.8.1 Single Core Optimisation

#### 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 6.4: 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 6.5: 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...

Rebuild libblis3-serial

## 6.8.2 Single Node Optimisation

### Kernel Preemption Model

The Linux kernel has 3 Preemption Models...

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

As per the Help in the Kernel Configuration...

Listing 6.6: 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...

### 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 interrupts...

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 coalescing...
  - 1.2 Reduce the number of packets - increase MTU...
- 2.1 Share the interrupt servicing activity evenly across the CPUs...

### 6.8.3 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...

## Jumbo Frames

Requires a network switch capable of Jumbo frames...

```
$ ip link show eth0
```

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
    link/ether dc:a6:32:60:7b:cd brd ff:ff:ff:ff:ff:ff
```

```
$ ping -c 1 -s 1500 -M do node2
```

```
PING node2 (192.168.0.2) 1500(1528) bytes of data.
ping: local error: message too long, mtu=1500
```

```
$ ping -c 1 -s 1472 -M do node2
```

```
PING node2 (192.168.0.2) 1472(1500) bytes of data.
1480 bytes from node2 (192.168.0.2): icmp_seq=1 ttl=64 time=0.392 ms
```

Trying to set the MTU to 9000 bytes...

```
$ sudo ip link set eth0 mtu 9000
```

... results with...

```
Error: mtu greater than device maximum.
```

In fact, attempting to set the MTU to anything greater than 1500 bytes...

```
$ sudo ip link set eth0 mtu 1501
```

... results with...

```
Error: mtu greater than device maximum.
```

Need to build a kernel with Jumbo frame support...

See Appendix ?...

```
$ ip link show eth0
```

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP mode DEFAULT gr
    link/ether dc:a6:32:60:7b:cd brd ff:ff:ff:ff:ff:ff
```

```
$ ping -c 1 -s 9000 -M do node2
```

```
PING node2 (192.168.0.2) 9000(9028) bytes of data.
ping: local error: message too long, mtu=9000
```

```
$ ping -c 1 -s 8972 -M do node2
```

```
PING node2 (192.168.0.2) 8972(9000) bytes of data.
8980 bytes from node2 (192.168.0.2): icmp_seq=1 ttl=64 time=0.847 ms
```

On node2 create the Iperf server...

```
$ iperf -s
```

On node1 create and run the Iperf client...

```
$ iperf -i 1 -c node2
```

```
-----
Client connecting to node2, TCP port 5001
TCP window size: 682 KByte (default)
-----
[  3] local 192.168.0.1 port 46216 connected with 192.168.0.2 port 5001
[ ID] Interval           Transfer     Bandwidth
[  3]  0.0-10.0 sec   1.15 GBytes    991 Mbits/sec
```

## 6.8.4 Kernel TCP Parameters Tuning

REFERENCE...

<https://www.open-mpi.org/faq/?category=tcp>

Listing 6.7: /etc/sysctl.d/picluster.conf

```
1 net.core.rmem_max = 16777216
```

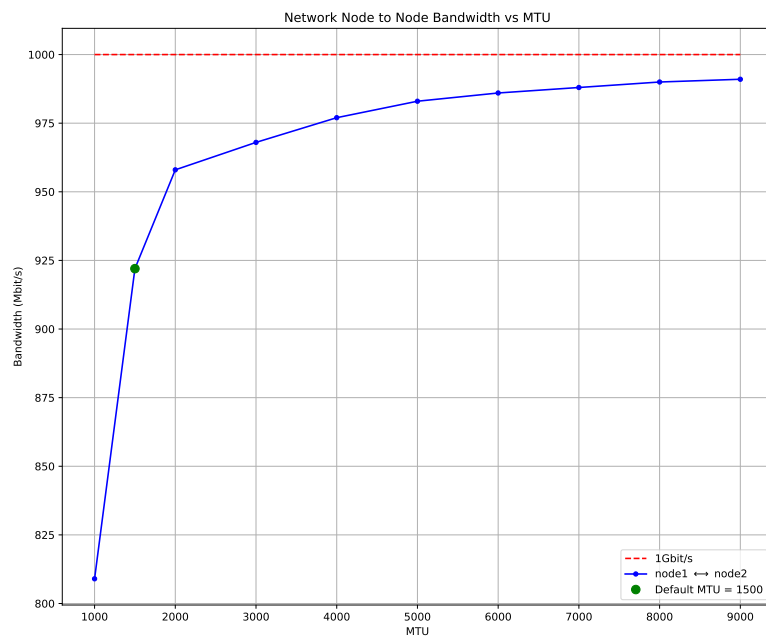


Figure 6.5: Network Node to Node Bandwidth vs MTU.



```
2 net.core.wmem_max = 16777216
3 net.ipv4.tcp_rmem = 4096 87380 16777216
4 net.ipv4.tcp_wmem = 4096 65536 16777216
5 net.core.netdev_max_backlog = 30000
6 net.core.rmem_default = 16777216
7 net.core.wmem_default = 16777216
8 net.ipv4.tcp_mem= 16777216 16777216 16777216
9 net.ipv4.route.flush = 1
```

```
sudo sysctl --system
```

or

```
sudo shutdown -r now
```

```
Aug 11 03:35:40 node5 kernel: [19256.425779] bcmgenet fd580000.ethernet eth0: bcmgene
```

## Part III

# Build Instructions

## Chapter 7

# The Aerin Cluster

### 7.1 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,  
1 x 8 port Gigabit Router/Firewall  
1 x 16 port Gigabit switch with Jumbo Frame support

Items

Photo

## 7.2 Preliminary Tasks

### 7.2.1 Update Raspberry Pi EE-PROMs

### 7.2.2 Obtain Raspberry Pi MAC Addresses

### 7.2.3 Generate User Key Pair

On macbook (no passphrase):

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

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

### 7.2.4 Amend macbook /etc/hosts

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

```
1 192.168.0.1 node1
2 192.168.0.2 node2
3 192.168.0.3 node3
4 192.168.0.4 node4
5 192.168.0.5 node5
6 192.168.0.6 node6
7 192.168.0.7 node7
8 192.168.0.8 node8
9 192.168.0.9 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.

### 7.2.5 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.

## 7.3 Ubuntu 20.04 64-bit LTS Installation

The idea is to have a single (modified) Ubuntu 20.04 image which can be used to install Ubuntu 20.04 on all of the nodes...

### 7.3.1 Create the Installation Image

The instructions below are for MacOS but should be straightforward to adjust for other operating systems.

On macbook...

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

Double click the compressed the .xz file to extract the .img file.

Double click the .img file to mount the image in the macbook filesystem as:

/Volumes/system-boot

We now need to edit the user-data file which stores the cloud-init configuration. The user-data file used to create the Aerin Cluster is at Listing 7.

Listing 7.1: /Volumes/system-boot/user-data

```
1 #cloud-config
2
3 # This is the user-data configuration file for cloud-init. By default this sets
4 # up an initial user called "ubuntu" with password "ubuntu", which must be
5 # changed at first login. However, many additional actions can be initiated on
6 # first boot from this file. The cloud-init documentation has more details:
7 #
8 # https://cloudinit.readthedocs.io/
9
10 # On first boot, set the (default) ubuntu user's password to "ubuntu" and
11 # expire user passwords
12 chpasswd:
13   expire: true
14   list:
15     - ubuntu:ubuntu
16     - john:john
17
18 # Enable password authentication with the SSH daemon
19 ssh_pwauth: true
20
21 ## Add users and groups to the system, and import keys with the ssh-import-id
```

```

22 groups:
23 - john: [john]
24
25 users:
26 - default
27 - name: john
28   gecos: John Duffy
29   primary_group: john
30   sudo: ALL=(ALL) NOPASSWD:ALL
31   shell: /bin/bash
32   ssh_authorized_keys:
33   - ssh-rsa ...= john
34
35 ## Update apt database and upgrade packages on first boot
36 package_update: true
37 package_upgrade: true
38
39 ## Install additional packages on first boot
40 packages:
41 - git
42 - tree
43 - unzip
44 - iperf
45 - net-tools
46 - linux-tools-common
47 - linux-tools-raspi
48 - build-essential
49 - gfortran
50 - gdb
51 - fakeroot
52 - devscripts
53 - openmpi-bin
54 - libblis3-serial
55 - libblis3-openmp
56 - libopenblas0-serial
57 - libopenblas0-openmp
58
59 ## Write arbitrary files to the file-system (including binaries!)
60 write_files:
61 - path: /etc/hosts
62   content: |
63     127.0.0.1 localhost
64     192.168.0.1 node1
65     192.168.0.2 node2
66     192.168.0.3 node3
67     192.168.0.4 node4
68     192.168.0.5 node5
69     192.168.0.6 node6
70     192.168.0.7 node7
71     192.168.0.8 node8

```



Figure 7.1: Using Raspberry Pi Imager to erase and format a MicroSD card.

```

72     192.168.0.9 node9
73     permissions: '0644'
74     owner: root:root
75
76     ## Run arbitrary commands at rc.local like time
77     runcmd:
78     - hostnamectl set-hostname --static node$(hostname -i | cut -d ' ' -f 1 | cut -d '.'
79     - reboot

```

Eject/unmount the .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 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.





Figure 7.2: Using Raspberry Pi Imager to write the server image to a MicroSD card.

## 7.4 Post-Installation Tasks

### 7.4.1 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 node1
```

And then from node1, for node2 to node9...

```
$ ssh node2
```

This will generate will generate a message similar to...

```
The authenticity of host 'node2 (192.168.0.2)' can't be established.  
ECDSA key fingerprint is SHA256:5VgsnN2nPvpfbJmALh3aJd0eT/NvDXqN8TCreQyNaFA.  
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Respond yes to this, which imports the host key into the ~/.ssh/knownhosts file of node1.

And then exit from the connected node...

```
$ exit
```

Repeat the above for node2 to node9.

The above is only required to be done once (unless the host keys on node2 to node9 change).

### 7.4.2 Uninstall unattended-upgrades

The `unattended-upgrades` package is installed automatically...

This can potentially interfere with long running benchmarks...

Remove...

From macbook:

```
$ ssh node1
$ ~/picluster/tools/do "sudo apt remove unattended-upgrades"
```

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

```
$ ssh node1
$ ~/picluster/tools/upgrade
```

### 7.4.3 Add Ubuntu 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 7.2: `/etc/apt/sources.list.d/picluster.list`

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

... and then update the repository cache...

```
$ sudo apt update
```

### 7.4.4 Create a Project Repository

Xpand upon...

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

Ensure you do push your repository to a remote repository at regular intervals...

### 7.4.5 Select BLAS Library

The cloud-init process will have installed four BLAS libraries, namely...

libopenblas0-serial

libopenblas0-openmp

libblis0-serial

libblis0-openmp

To query the BLAS library currently in use on each node we can use one of our Pi Cluster tools...

```
$ ~/picluster/tools/libblas-query
```

```
node8... /usr/lib/aarch64-linux-gnu/openblas-openmp/libblas.so.3
node7... /usr/lib/aarch64-linux-gnu/openblas-openmp/libblas.so.3
node6... /usr/lib/aarch64-linux-gnu/openblas-openmp/libblas.so.3
node5... /usr/lib/aarch64-linux-gnu/openblas-openmp/libblas.so.3
node4... /usr/lib/aarch64-linux-gnu/openblas-openmp/libblas.so.3
node3... /usr/lib/aarch64-linux-gnu/openblas-openmp/libblas.so.3
node2... /usr/lib/aarch64-linux-gnu/openblas-openmp/libblas.so.3
node1... /usr/lib/aarch64-linux-gnu/openblas-openmp/libblas.so.3
```

To select an alternative library we can use another of our Pi Cluster tools...

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

```
node8... done
node7... done
node6... done
node5... done
node4... done
node3... done
node2... done
node1... done
```

```
$ ~/picluster/tools/libblas-query
```

```
node8... /usr/lib/aarch64-linux-gnu/blis-serial/libblas.so.3
node7... /usr/lib/aarch64-linux-gnu/blis-serial/libblas.so.3
node6... /usr/lib/aarch64-linux-gnu/blis-serial/libblas.so.3
```

```
node5... /usr/lib/aarch64-linux-gnu/blis-serial/libblas.so.3
node4... /usr/lib/aarch64-linux-gnu/blis-serial/libblas.so.3
node3... /usr/lib/aarch64-linux-gnu/blis-serial/libblas.so.3
node2... /usr/lib/aarch64-linux-gnu/blis-serial/libblas.so.3
node1... /usr/lib/aarch64-linux-gnu/blis-serial/libblas.so.3
```

## 7.5 Install High-Performance Linpack (HPL)

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 a Make.picluster file...

```
$ cd setup
$ bash make_generic
$ cp Make.UNKNOWN ../Make.picluster
$ cd ..
```

Amend Make.picluster as per listing ???.

Listing 7.3: /picluster/hpl/hpl-2.3/Make.picluster

```
1 #
2 # -- High Performance Computing Linpack Benchmark (HPL)
3 #   HPL - 2.3 - December 2, 2018
4 #   Antoine P. Petitet
5 #   University of Tennessee, Knoxville
6 #   Innovative Computing Laboratory
7 #   (C) Copyright 2000-2008 All Rights Reserved
8 #
9 # -- Copyright notice and Licensing terms:
10 #
11 # Redistribution and use in source and binary forms, with or without
12 # modification, are permitted provided that the following
13 # conditions
14 # are met:
15 #
16 # 1. Redistributions of source code must retain the above copyright
17 # notice, this list of conditions and the following disclaimer.
18 #
19 # 2. Redistributions in binary form must reproduce the above copyright
20 # notice, this list of conditions, and the following disclaimer in the
21 # documentation and/or other materials provided with the distribution.
22 #
23 # 3. All advertising materials mentioning features or
24 # use of this
```

```

23 # software must display the following acknowledgement:
24 # This product includes software developed at the University
    of
25 # Tennessee, Knoxville, Innovative Computing Laboratory.
26 #
27 # 4. The name of the University, the name of the Laboratory,
    or the
28 # names of its contributors may not be used to endorse or promote
29 # products derived from this software without specific
    written
30 # permission.
31 #
32 # -- Disclaimer:
33 #
34 # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
35 # 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
    BUT NOT
36 # LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
37 # A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE UNIVERSITY
38 # OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
    INCIDENTAL,
39 # SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
    BUT NOT
40 # LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
41 # DATA OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
42 # THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
    OR TORT
43 # (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
44 # OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
45 # #####
46 #
47 # -----
48 # - shell -----
49 # -----
50 #
51 SHELL          = /usr/bin/bash
52 #
53 CD              = cd
54 CP              = cp
55 LN_S            = ln -s
56 MKDIR           = mkdir -p
57 RM              = rm -f
58 TOUCH           = touch
59 #
60 # -----
61 # - Platform identifier -----
62 # -----
63 #
64 ARCH            = picluster
65 #

```

```

66 # -----
67 # - HPL Directory Structure / HPL library -----
68 # -----
69 #
70 TOPdir      = $(HOME)/picluster/hpl/hpl-2.3
71 INCdir      = $(TOPdir)/include
72 BINDir      = $(TOPdir)/bin/$(ARCH)
73 LIBdir      = $(TOPdir)/lib/$(ARCH)
74 #
75 HPLlib      = $(LIBdir)/libhpl.a
76 #
77 # -----
78 # - Message Passing library (MPI) -----
79 # -----
80 # MPinc tells the C compiler where to find the Message Passing library
81 # header files, MPlib is defined to be the name of the library to be
82 # used. The variable MPdir is only used for defining MPinc and MPlib.
83 #
84 MPdir       = /usr/lib/aarch64-linux-gnu/openmpi
85 MPinc       = $(MPdir)/include
86 MPlib       = $(MPdir)/lib/libmpi.so
87 #
88 # -----
89 # - Linear Algebra library (BLAS or VSIPL) -----
90 # -----
91 # LAinc tells the C compiler where to find the Linear Algebra
92 # library
93 # header files, LAlib is defined to be the name of the library to be
94 # used. The variable LAdir is only used for defining LAinc and LAlib.
95 #
96 LAdir       = /usr/lib/aarch64-linux-gnu
97 LAinc       =
98 LAlib       = $(LAdir)/libblas.so.3
99 #
100 # -----
101 # - F77 / C interface -----
102 # -----
103 # You can skip this section if and only if you are not planning to use
104 # a BLAS library featuring a Fortran 77 interface. Otherwise,
105 # it is
106 # necessary to fill out the F2CDEFS variable with the
107 # appropriate
108 # options. **One and only one** option should be chosen in **each** of
109 # the 3 following categories:
110 #
111 # 1) name space (How C calls a Fortran 77 routine)
112 #
113 # -DAdd_      : all lower case and a suffixed underscore
114 # (Suns,
115 #
116 # Intel, ...),

```



```

[default]
112 # -DNoChange           : all lower case (IBM RS6000),
113 # -DUpCase            : all upper case (Cray),
114 # -DAdd_              : the FORTRAN compiler in use is f2c.
115 #
116 # 2) C and Fortran 77 integer mapping
117 #
118 # -DF77_INTEGER=int    : Fortran 77 INTEGER is a C int,
[default]
119 # -DF77_INTEGER=long   : Fortran 77 INTEGER is a C long,
120 # -DF77_INTEGER=short  : Fortran 77 INTEGER is a C short.
121 #
122 # 3) Fortran 77 string handling
123 #
124 # -DStringSunStyle     : The string address is passed at the string loca-
125 #                       : tion on the stack, and the string length is then
126 #                       : passed as an F77_INTEGER after all
explicit
127 #                       : stack arguments,
[default]
128 # -DStringStructPtr    : The address of a structure is passed
by a
129 #                       : Fortran 77 string, and the structure is of the
130 #                       : form: struct {char *cp; F77_INTEGER len;},
131 # -DStringStructVal     : A structure is passed by value for each
Fortran
132 #                       : 77 string, and the structure is of the form:
133 #                       : struct {char *cp; F77_INTEGER len;},
134 # -DStringCrayStyle     : Special option for Cray machines,
which uses
135 #                       : Cray fcd (fortran character descriptor)
for
136 #                       : interoperation.
137 #
138 F2CDEFS      = -DAdd_ -DF77_INTEGER=int -DStringSunStyle
139 #
140 # -----
141 # - HPL includes / libraries / specifics -----
142 # -----
143 #
144 HPL_INCLUDES = -I$(INCdir) -I$(INCdir)/$(ARCH) -I$(MPinc)
145 HPL_LIBS     = $(HPLlib) $(LALib) $(MPLib)
146 #
147 # - Compile time options -----
148 #
149 # -DHPL_COPY_L          force the copy of the panel L before bcast;
150 # -DHPL_CALL_CBLAS      call the cblas interface;
151 # -DHPL_CALL_VSIPL      call the vsip library;
152 # -DHPL_DETAILED_TIMING enable detailed timers;
153 #

```

```

154 # By default HPL will:
155 # *) not copy L before broadcast,
156 # *) call the BLAS Fortran 77 interface,
157 # *) not display detailed timing information.
158 #
159 # HPL_OPTS      =
160 #
161 # -----
162 #
163 HPL_DEFS        = $(F2CDEFS) $(HPL_OPTS) $(HPL_INCLUDES)
164 #
165 # -----
166 # - Compilers / linkers - Optimization flags -----
167 # -----
168 #
169 CC              = mpicc
170 CCNOOPT         = $(HPL_DEFS)
171 CCFLAGS         = $(HPL_DEFS) -O3 -march=armv8-a -mtune=cortex-a72
172 #
173 LINKER          = $(CC)
174 LINKFLAGS       = $(CCFLAGS)
175 #
176 ARCHIVER        = ar
177 ARFLAGS         = r
178 RANLIB          = echo
179 #
180 # -----

```

Build HPL...

```
$ make arch=picluster
```

This creates the executable `xhpl` and input file `HPL.dat` in `bin/picluster`

The `xhpl` executable has to exist in the same location on each node, so copy `xhpl` to node2 to node8 (only `xhpl`, and not `HPL.dat`)...

```

$ cd bin/picluster
$ ~/picluster/tools/do "mkdir -p picluster/hpl/hpl-2.3/bin/picluster"
$ scp xhpl node2:~picluster/hpl/hpl-2.3/bin/picluster
$ scp xhpl node3:~picluster/hpl/hpl-2.3/bin/picluster
$ scp xhpl node4:~picluster/hpl/hpl-2.3/bin/picluster
$ scp xhpl node5:~picluster/hpl/hpl-2.3/bin/picluster
$ scp xhpl node6:~picluster/hpl/hpl-2.3/bin/picluster
$ scp xhpl node7:~picluster/hpl/hpl-2.3/bin/picluster
$ scp xhpl node8:~picluster/hpl/hpl-2.3/bin/picluster

```

## 7.6 Kernel Build Procedure

This procedure is derived from the Ubuntu Wiki BuildYourOwnKernel document...

Make sure you have made the source code repositories available as per...

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

```
$ ssh node1
$ mkdir -p ~/picluster/build/kernel
$ sudo chown _apt:root ~/picluster/build/kernel
$ cd ~/picluster/build/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)
$ cd linux-raspi-5.4.0
```

This bit is a fix for the subsequent `editconfigs` step of the build procedure...

```
$ cd debian.raspi/etc
$ sudo cp kernelconfig kernelconfig.original
$ sudo vim kernelconfig
```

And make the following change...

Listing 7.4: `diff kernelconfig kernelconfig.original`

```
5c5
< archs="arm64"
---
> archs="armhf arm64"
```

Then move back up to the kernel source top level directory...

```
$ cd ../../
```

Prepare the build scripts...

```
$ sudo chmod a+x debian/rules
$ sudo chmod a+x debian/scripts/*
$ sudo chmod a+x debian/scripts/misc/*
```

SOURCE CHANGES AND/OR verb—editconfigs— AT THIS POINT

```
$ sudo apt install libncurses-dev
$ sudo LANG=C fakeroot debian/rules clean
$ sudo LANG=C fakeroot debian/rules editconfigs
```

Tweak the kernel name for identification...

```
$ cd debian.raspi
$ sudo cp changelog changelog.original
$ sudo vim changelog
```

And make the following change, where `+piccluster0` is our kernel identifier...

Listing 7.5: diff changelog changelog.original

```
1c1
< linux-raspi (5.4.0-1015.15+piccluster0) focal; urgency=medium
---
> linux-raspi (5.4.0-1015.15) focal; urgency=medium
```

Move up to the top level kernel source directory...

```
$ cd ..
```

And build the kernel...

```
$ sudo LANG=C fakeroot debian/rules clean
$ sudo LANG=C fakeroot debian/rules binary-arch
cd ..
```

Install the new kernel...

```
$ sudo dpkg -i linux*piccluster0*.deb
$ sudo shutdown -r now
```

Another build procedure fix...

After each kernel build delete the `linux-libc-dev` directory...

```
$ cd ~/piccluster/build/kernel/linux-raspi-5.4.0/debian
$ rm -rf linux-libc-dev
$ cd ..
```

## 7.7 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 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, "+jff", to the new kernel name...

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

## 7.8 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
```

## 7.9 Rebuild BLIS

```
$ 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
```



## 7.10 Pi Cluster Tools

Listing 7.6: picluster/tools/upgrade

```
1 #!/usr/bin/bash
2
3 NODES=9
4
5 for (( i=$NODES; i>0; i-- ))
6 do
7     echo "Upgrading node$i..."
8     ssh node$i sudo apt update
9     ssh node$i sudo apt full-upgrade --yes
10    ssh node$i sudo apt autoremove --yes
11    ssh node$i sudo shutdown -r now
12 done
```

Listing 7.7: picluster/tools/reboot

```
1 #!/usr/bin/bash
2
3 NODES=8
4
5 for (( i=$NODES; i>0; i-- ))
6 do
7     echo "Rebooting node$i..."
8     ssh node$i sudo shutdown -r now
9 done
```

Listing 7.8: picluster/tools/shutdown

```
1 #!/usr/bin/bash
2
3 NODES=8
4
5 for (( i=$NODES; i>0; i-- ))
6 do
7     echo "Shutting down node$i..."
8     ssh node$i sudo shutdown -h now
9 done
```

Listing 7.9: picluster/tools/libblas-query

```
1 #!/usr/bin/bash
2
3 NODES=8
4
5 for (( i=$NODES; i>0; i-- ))
6 do
7     printf "node$i... "
```

```

8      ssh node$i update-alternatives --query libblas.so.3-aarch64-linux-gnu \
9      | grep Value: \
10     | gawk '{print $2}'
11 done

```

Listing 7.10: picluster/tools/libblas-set

```

1  #!/usr/bin/bash
2
3  NODES=8
4
5  case $1 in
6      "openblas-serial" | "openblas-openmp" | "blis-serial" | "blis-openmp")
7      for (( i=$NODES; i>0; i-- ))
8      do
9          printf "node$i... "
10         ssh node$i sudo update-alternatives --quiet --set \
11             libblas.so.3-aarch64-linux-gnu \
12             /usr/lib/aarch64-linux-gnu/$1/libblas.so.3
13         printf "done\n"
14     done
15     exit
16     ;;
17  esac
18
19  echo "Usage: libblas-set {openblas-serial|openblas-openmp|blis-serial|blis-openmp}"

```

## 7.11 Build OpenMPI from Source

Do all of this on node1...

```
$ ssh node1
```

We want to avoid collisions with multiple OpenMPI installations, so remove original installed version...

```
$ sudo apt remove openmpi-common
$ sudo apt remove openmpi-bin
$ sudo apt autoremove
```

OpenMPI requires the libevent-dev package...

```
$ sudo apt install libevent-dev
```

Create a build directory, and download and, and and following BLAH, BLAH build OpenMPI...

```
$ mkdir -p ~/picluster/build/openmpi
$ cd ~/picluster/build/openmpi
$ wget https://download.open-mpi.org/release/open-mpi/v4.0/openmpi-4.0.4.tar.gz
$ gunzip openmpi-4.0.4.tar.gz
$ tar xvf openmpi-4.0.4.tar
$ rm openmpi-4.0.4.tar
$ cd openmpi-4.0.4
$ mkdir build
$ cd build
$ ../configure CFLAGS="-O3 -march=armv8-a -mtune=cortex-a72"
$ make all
$ sudo make install
$ sudo ldconfig
```

OpenMPI will installed to /usr/local

EXTRACT FROM HPL.dat

TODO: HOW TO COPY TO ALL NODES!

## 7.12 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 7.11: Make.armpl-serial extract

```
1 # -----
2 # - Linear Algebra library (BLAS or VSIBL) -----
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
```