Exploring the viability of low powered devices as nodes in cluster computing

Author: Ben Hague (201146260)

Project Supervisor: Dr James Walsh

Project Assessor: Dr Mohammad Hasan

Department of Electrical Engineering and Electronics

29 April 2019

Abstract

Cluster Computing is used widely in the industry for solving complex problems which would take too long to compute on a regular single processor computer. During the last five years, a substantial decrease in price has led to a game-changing increase in the availability of small low power single board computers. Combining the technologies used to process large amounts of data with modern lower cost specialist processors or higher numbers of lower power general use processors can widen the possibility of custom machines being designed purely to crunch massive amounts of data in massively parallel ways. While the scope of this project covers the use of raspberry pi computers, a higher budget approach and prolonged development time could remove bottlenecks from the system through good bespoke design for customized multi-processor computing boards. The outcome of this project is to prove the cost viability and power efficiency for clusters of modern lower power computers rather than higher power workstation grade computers for mathematical based computing solutions. Throughout this project, we discover that there is a clear long-term cost-benefit in opting for a cluster-based setup not only in performance but also in scalability.

Declaration

I confirm that I have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with any other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).

Contents

Table of Figures

Table of Tables

1 Specification

The scope and goals of the project have not changed since the beginning of October. The following extract from the preliminary report [1] is below for reference. The complete report can be found in Appendix C.

"The project has been divided into 4 separate categories, Hardware, Software, Benchmarking, and Paperwork. Each of these categories is a distinct deliverable which must be done to achieve success in the project. Within each of the Deliverables, there is a small number of work packets which relate to the goals which must be completed for the section to succeed.

The outcome of this Project is to create a 6 node 24 Core Cluster out of Raspberry Pi 3 units, the cluster should also be able to run Plasma Simulation code from the Plasma Theory and Simulation Group (PTSG). The cluster should also be able to demonstrate the speed difference between a single node and a full cluster and show how this compares to using a traditional workstation.

Specification and aims for each deliverable:

- *Hardware - The outcome of this deliverable is to create a 6 node 24 Core Cluster out of Raspberry Pi 3 units.*
    - *Construct Hardware Network*
    - *Design Casing*
    - *Construct Casing for Cluster*
- *Software – The outcome of this deliverable is to use an MPI to allow the cluster to work together and execute code, the cluster should also be able to run Plasma Simulation code from the Plasma Theory and Simulation Group (PTSG) [1]*
    - *Build Linux Software for network*
    - *Set up and install MPI software*
    - *Compile Benchmarking code*
- *Benchmarking – The outcome of this deliverable is to Test, Benchmark and improve the cluster built. The cluster should also be able to demonstrate the speed difference between a single node and a full cluster and show how this compares to using a traditional workstation.*
    - *Construct Benchmarking Tests*
    - *Conduct Cluster Benchmarking*
    - *Conduct Single Node Benchmarking*
    - *Conduct Workstation Benchmark "*

2 Introduction

Advancements in computing have long been led through cluster computing principals. We use these principals daily to split tasks into sets of smaller tasks that can be executed concurrently. Due to the concurrent execution, this allows us to scale the performance depending on the number of computers that can be utilized. This can mean that a cluster with more less powerful nodes can rival the performance of a cluster with less more powerful nodes provided the problem fits specific criteria. Exploring this concept and the limitations it imposes allows us to understand reducing the costs of working with high-performance computers in a real-world scenario.

Since the Release of the Raspberry Pi in 2012 and Cox et al.[2] released their high profile cluster computer of the original Pi units, many groups and individuals have explored the possibility of HPC systems constructed for specific uses. Cloud servers [3], Power Grid Monitoring [4] and even DNA sequencing have all been done with clusters of Raspberry Pis. This has shown over time that there is a motivation within research to evaluate the use of ARM based nodes for smaller clustered systems.

While the scope of this project covers the use of Raspberry Pi computers, a higher budget approach with prolonged development time could remove bottlenecks from the system through good bespoke design for customized multi-processor hardware containing many nodes within a smaller package. The methodology behind constructing a cluster for executing plasma simulations and how the cluster can be used are covered in Sections 5 and six respectively. Furthermore, we will discuss the advantages and benefits of using High Performance Computing (HPC) systems on ARM based architectures.

3 Industrial Relevance

Complex calculations for modeling and simulations are more and more commonplace as designs get more extensive and detailed. The market for high performance computing solutions is booming as more and more companies are investing to speed up the development cycle of more complex products[5]. Many of the world's leading companies save millions on a yearly basis by employing the use of high performance computing solutions, Tichenor and Reuther[6] suggest that the most significant fault for companies that invest in HPC is the failure to invest enough.

The increased demand for higher performance has put pressure on manufacturers to create more appealing, efficient and stable solutions. Investigating how stability and efficiency can be improved with minimal cost is essential for industries both developing and using HPC systems. An appropriate solution could provide a long-term benefit for industry and research-based implementations. Electricity consumption is one of the most significant costs associated with High Performance Computing with many of the world's most powerful supercomputers costing over $10,000,000 per annum [7]. Not only is this a continual drain on resources but continuing pressure from governments and customers to maintain green energy practices cannot support the future for extreme power consumption. It has been long stated by many that the next major obstacle in supercomputing will be power consumption[8].

Over ten years ago Kogge et al.[9] discusses greatly the challenges and suggests that research focused on low energy logic and memory devices could be implemented with significant effect. Even before this, in his research comparing of efficiency of clusters with lower power nodes against traditional x86 systems Laurenzano et al.[10] found *"performance and energy efficiency of the ARM systems varies by up to an order-of-magnitude and depends on the computational and memory characteristics of the application".* While this shows that arm clusters have the potential to be inefficient provide and provide no benefit, it also indicates that certain applications with high levels of concurrent operations can be considerably more effective.

Reducing system downtime is of paramount importance. Compared to a traditional node, each node within an ARM based system is considerably cheaper. This means that in the event of system failure each node can be treated as almost disposable. Faulty hardware is easily identifiable to a single node which can then be swapped out in minimal time reducing the overall system downtime. This is important for companies as maintaining the overall reliability of the system relates directly to usability. As the number of nodes increases, we also benefit from the boost in redundancy. In the event of a node failure, less performance is lost meaning that each of the remaining nodes has less additional data to process. This makes the system appear more robust to both users and customers in industrial and commercial applications.

A vast number of low cost single board computer systems have become available at nearly all price points. These single board computers offer a distinct value for money when compared to the workstations used in traditional cluster arrays whilst using a fraction of the power. A well-managed compute performance platform as discussed in [11] shows that when applications are

written to specifically make use of customised coprocessors it is possible to achieve more performance without the large increase in power consumption.

The second potential for lower power compute solutions is the education market. With the onset of parallel computing within industry, there is a large demand for engineers and scientists with an understanding of how to utilise these newer technologies[12]. There is a strong lean towards teaching with Raspberry Pi based clusters due to cost, availability, and open documentation[13]. This offers a unique opportunity for students in the sciences field where attempts at creating scalable programs can be tested more effectively on real world hardware. Until more recently this has been impossible to justify time on larger scale HPC systems as the cost of execution is too high. Researchers developing cutting edge applications for new problems often need continual access to their data to ensure that their code scales correctly. Due to the size of smaller low power clusters, it is possible for physical access to be maintained. Several existing projects have shown how over 20 nodes can be contained within the profile of a workstation computer. This allows physical indicators of dataflow can be installed and monitored in real time in ways that would have previously been impractical.

4 Theory

In [14][15], Keipert stated that using a raspberry pi cluster offered a unique angle at cluster computing as the additional hardware enables a cluster of Raspberry Pi computers to more efficiently monitor data from external devices and sensors. He goes on to say that for some applications, monitoring and processing change in Realtime is inconvenient to use shared clusters for. While the cost of Keipert's system would be similar, A similar cluster built with modern hardware would provide a substantial increase in computing power and the speed of networking would result in a much more powerful system. This can offer a different insight as to how the effectiveness of the system can be analysed.

A Cost breakdown for a small Raspberry Pi based Cluster like the one built is shown in Table 1. The cost of this system is one of the key features, a low-end workstation from a reputable manufacturer starts at a similar price. While this does not include the man hours for set up of a system which is more intense than a standard PC, this shows the overall cost (Excluding Casing) shows a very competitive setup cost provided an appropriate application

Table 1, Cost Breakdown of Raspberry Pi Cluster Computer

Component	Cost	Quantity	Total
Raspberry Pi	£ 28.39	6	£ 170.34
SD Card	£ 8.52	6	£ 51.12
Network Switch	£ 20.92	1	£ 20.92
Power Adapter	£ 6.00	6	£ 36.00
Patch Cable	£ 1.92	7	£ 13.44
		Total	£ 291.82

This setup would get a 24-core system with a total of 6gb of RAM shared between nodes and just under 100gb of total solid-state storage. The scalability of this system would allow more computational power to be added for minimal cost and a linear performance increase for additional nodes while executing embarrassingly parallel algorithms. If the cluster can meet the performance of a workstation PC than not only is the cluster-based system cheaper to run on an everyday basis but is cheaper as an initial setup cost. This analysis is only relevant when the workstation has been purchased for the goal of using as a machine for executing long calculations.

The relationship between the speedup of an application and the number of cores assigned to a task can be defined by Amdahl's law [16]. Amdahl's law dictates the relationship between the possible speedup using the percentage of the program which is non-parallelisable (serial). Ahmdal's law is defined by the following Equation 1

$$S = \frac{1}{(1-P) + \frac{P}{C}}$$

Equation 1, Ahmdal's Law

Where S is the theoretical speedup, P is the percentage of the task which is parallelisable, and C is the number of cores available. This can show us that the maximum speed up in a system is equal to the number of cores the system has. This is demonstrated in the graph shown in Figure 4-1.

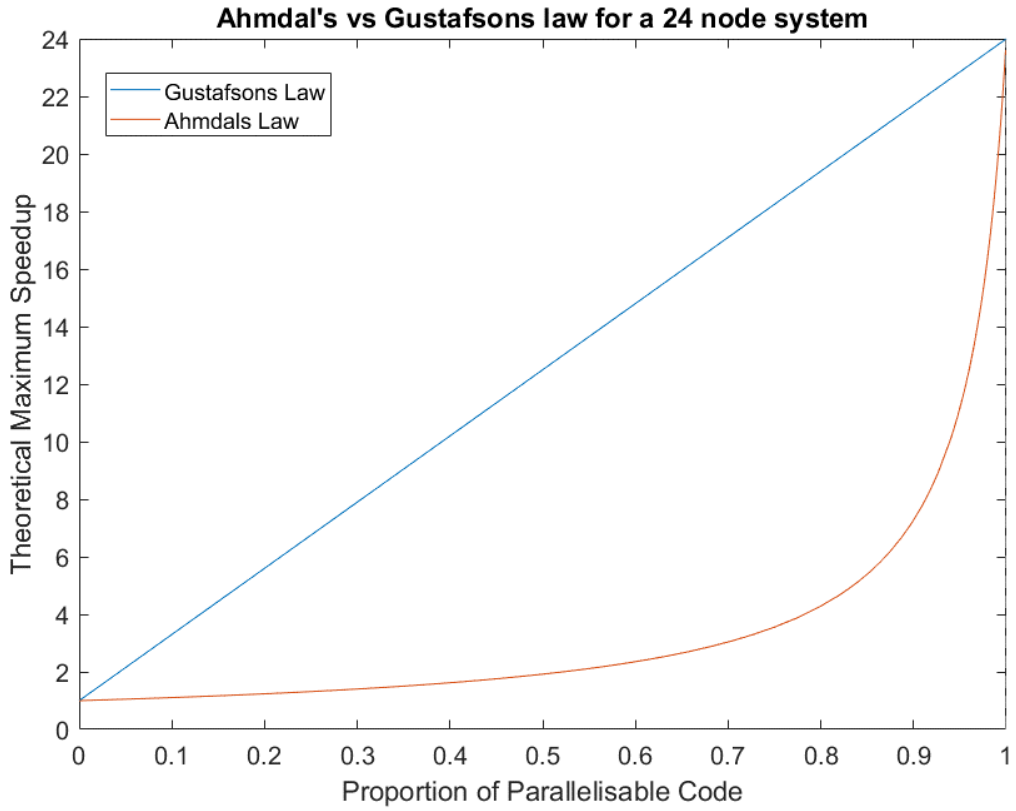


Figure 4-1, Amdahl's law vs Gustafson's law

For pure computational purposes, the Pi cluster constructed provides an insight as to what could be achieved with more bespoke hardware providing a more efficient sharing of resources.

This differs from the more optimistic Gustafson's law as introduced in the article *"Reevaluating Amdahl's law"* in 1998 [17] which characterises speedup using Equation 2

$$S = 1 - P + CP$$

Equation 2, Gustafson's Law

Where S, C, and P are the same as characterised in Ahmdal's law. It is clear from Figure 4-1 that while there is debate about the amount speedup of adding processing cores to a problem, there is no debate as to an effect being made due to the combining of resources.

5 Design

5.1 Hardware

The hardware needed to implement the cluster is simple, but each component was selected based upon its merits, availability, and price. Table 2 shows each of the product specification, selection, and justification. Much of the hardware selected is similar to that used by Parker Mitchell et al. [18]; however the system scale has been reduced to increase computing power due to the smaller management requirements. The scale of the project is reduced to equal performance equivalent to a single node of a traditional cluster.

Table 2, Hardware Selection

Product Specification	Product Selection	Justification
Single Board computer preference for low cost high core count and fast networking	Raspberry Pi 3B	This is the most Readily available small low-cost computer based on the ARM architecture.
High Speed Storage Medium for Cluster	Transcend 16 GB MicroSD Card Class 10, UHS-1 U1	Fast (minimum write speed of 10Mb/s) and each Pi can have one resulting in just under 1000Gb of storage when combined. Typically for larger calculations, a larger hard drive would be ideal to be included
Network Switch	D-Link Unmanaged Gigabit Ethernet Switch 8 port	Network Switch must be equal or faster than the speed of any individual node.
Network Patch Cables	RS PRO Yellow Cat6 Cable S/FTP PVC, 500mm	The cables must be able to sustain the same speed connection between nodes and Network switch.
Power Supply	RS PRO, 12W Plug in Power Supply 5V $\mathrm{dc}, 0 \rightarrow 2.4\mathrm{A},$	Power supplies must be able to sustain each node with 100% utilisation.

This shows us that each of the components is suitable for its chosen function. There are possible alternatives for each component, for different goals it is possible to tailor the cluster to respond better to the demand. For example, a cluster where the nodes require access to large amounts of data would benefit from a dedicated storage node in addition to the compute nodes.

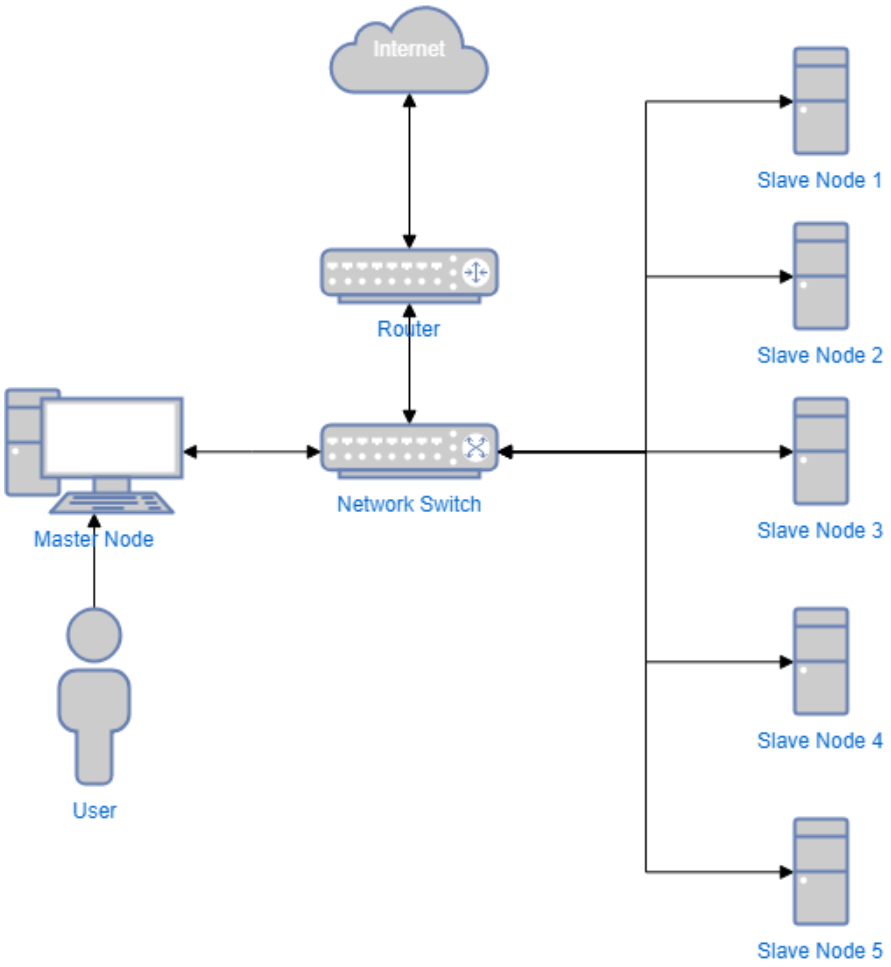


Figure 5-1, Network diagram of Cluster

The hardware is connected as shown in Figure 5-1. This shows that the user is only given access to the master node from which they can execute all the necessary code without making any changes to the overall cluster structure. All data is also stored on the Master Node and all the work compiling applications for execution on the cluster is stored in the shared storage space in this master node. Due to the limitation and bottleneck currently being the network speed, the nodes all have local installations of operating systems and critical dependencies. In an ideal set up each node would have a wider bandwidth and would boot from the Shared storage allowing system updates and software installs to configured with less system downtime. While appropriate for a small cluster, as the number of nodes on the network increase the demand will cause latency on the network which is used additionally to the cluster. The most effective way to overcome this would be to route all network traffic for the cluster through the master node which in turn is the only node with immediate access to the extended network.

5.2 Software Component

There are a few different reasons to form a cluster. The main two are redundancy and performance. The focus of the project is to achieve a high level of performance.

For the operating system, Raspbian [19] has been chosen as it is the most appropriate optimised software available for the Raspberry Pi ecosystem.

The key components of a successful cluster computer with single processor nodes are shared file system fast networking and light message passing [20]. There have long been alternatives which attempt to combine all of this into a package indistinguishable from using a workstation [21]. While this sounds ideal, for a system designed to make the most of performance it seems inappropriate to forgo the performance increase by managing all aspects of the system centrally rather than allowing each node to manage their tasks as appropriate for maximum performance.

Since the introduction of Message Passing Interface (MPI) as a standard, there has been a level of uniformity between the different applications and how they can be used[22]. As this is a standard, it is unimportant which model is used as both share the same commands such as "mpirun" for executing an MPI program[23]. There are many benefits of using MPI above other message handling interfaces such as documentation, reliability, and simplicity. The two major competitors in the open source field for MPI installations, MPICH, the original and OpenMPI. While MPICH has more backing and documentation, OpenMPI is the installation the cluster will use as when compiled and installed it is known to run faster with fewer compatibility errors.

For the shared filesystem a robust assessable network is needed. Multiple nodes must have access to the storage at the same time. There are a few different options for this; however, the two most appropriate options would be to use a Networked File System (NFS) share or to implement a parallel data structure. NFS Shares are high efficiency and require a minimal amount of data to be transferred into and out of the nodes as there is a central storage location. Parallel data structures require all data to exist on all nodes and therefore the all the new data must be transferred from each of the nodes to all the others. This characteristic makes the storage more robust however places considerably more load onto the network. As Cox et al. showed, the performance of the cluster has historically been limited by the speed of networking[2] consequently this is not appropriate for this implementation as it places additional load on the existing finite resources.

Within the cluster, the nodes will follow a strict naming system where the master node is designated *"ClusterNode0",* and each of the slave nodes is named *"ClusterNode1"* to *"ClusterNode5"* as appropriate. Dependencies will be installed to all nodes to reduce demand; however, the executable files and input files for the cluster will be installed to the shared file system. This will increase the reliability of the system and ensure that the current working directory of all the nodes is identical; this is important to maintain the system integrity.

A simple GUI software run on the same network will allow remote execution of profiles. This GUI written with the Tkinter framework in python [24] is shown in Figure 5-2. The GUI

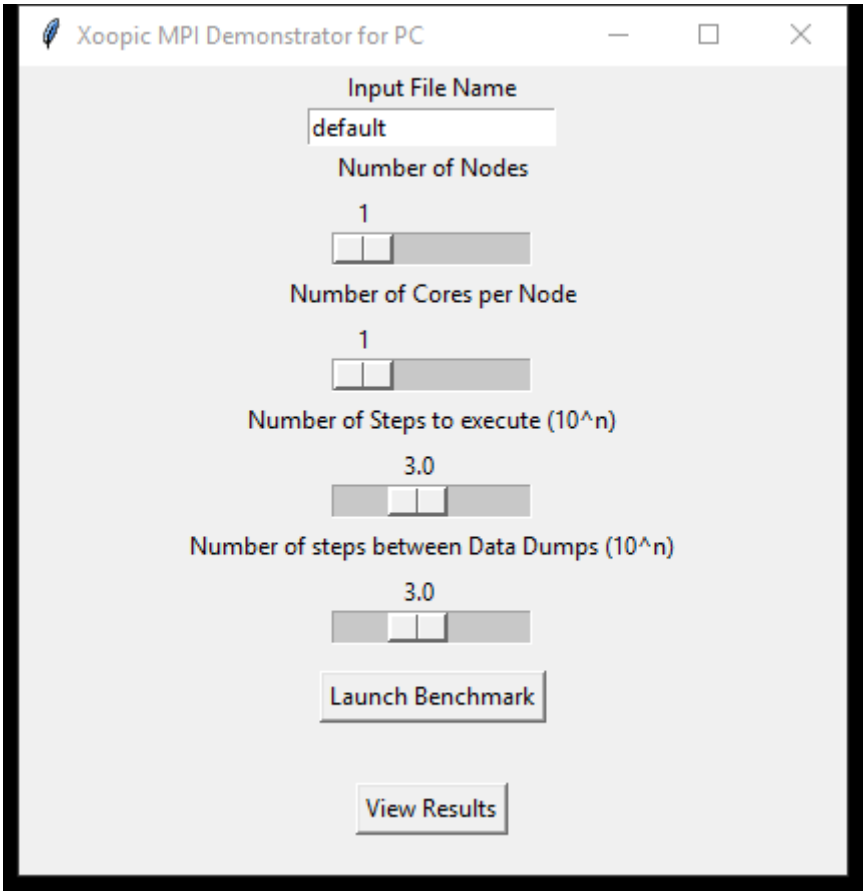


Figure 5-2, GUI interface

simplifies the process of running a simulation on the cluster. To effectively use this software the clusters NFS share must be mounted on the local PC and a local install of xoopic is needed to view the results.

When the launch benchmark button is clicked the program executes the following function

```
def launchXoopic():
   Hostgen()
   system("ssh ClusterNode0 mpirun -n " + str(Nodes.get()*Cores.get()) + " --hostfile hosts
xoopic -nox -i " + Benchmark +INPFILE.get()+"  -dp "+ str(10**Dumpfreq.get()) +" -s " +
str(10**Iterations.get()))
```

Hostgen is a function which generates a host file. The host file dictates the number of nodes and the cores per node. This is then transferred to the cluster. Following this, an SSH command is sent to execute the command following the parameters dictated by reading the slider positions and input file name.

5.3 Casing

There were key features required by the casing. It must provide a protected enclosure for all nodes with appropriate cable management while maintaining a suitable level of airflow to allow proper cooling. A simple expandable way to meet these specifications is to mount each node on an identical mounting plate. The design for this plate is shown in Figure 5-3

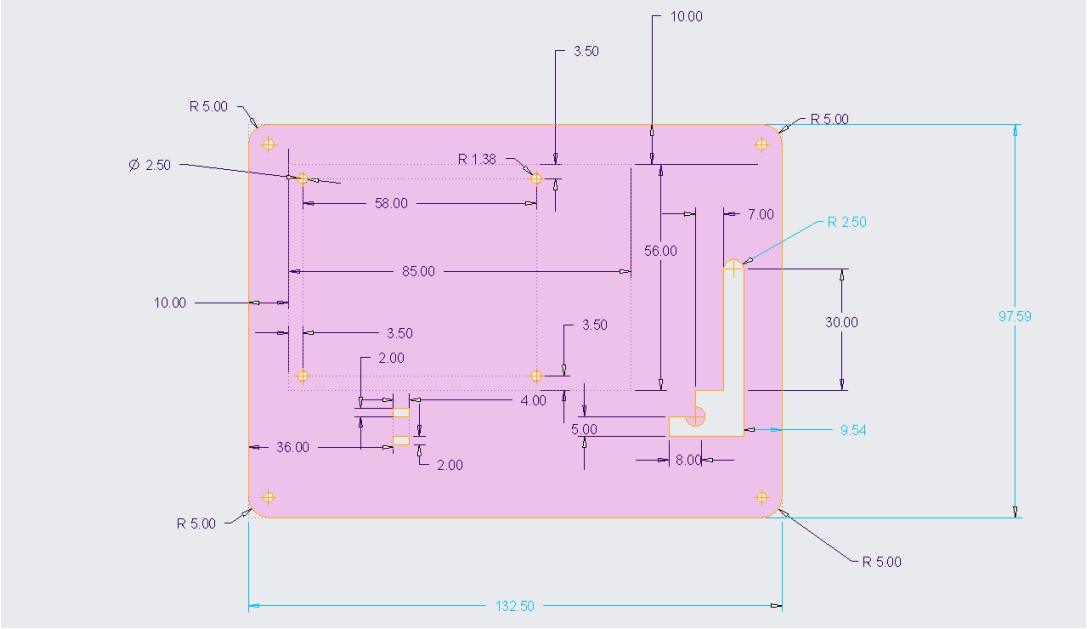


Figure 5-3, Technical Drawing of mounting plate

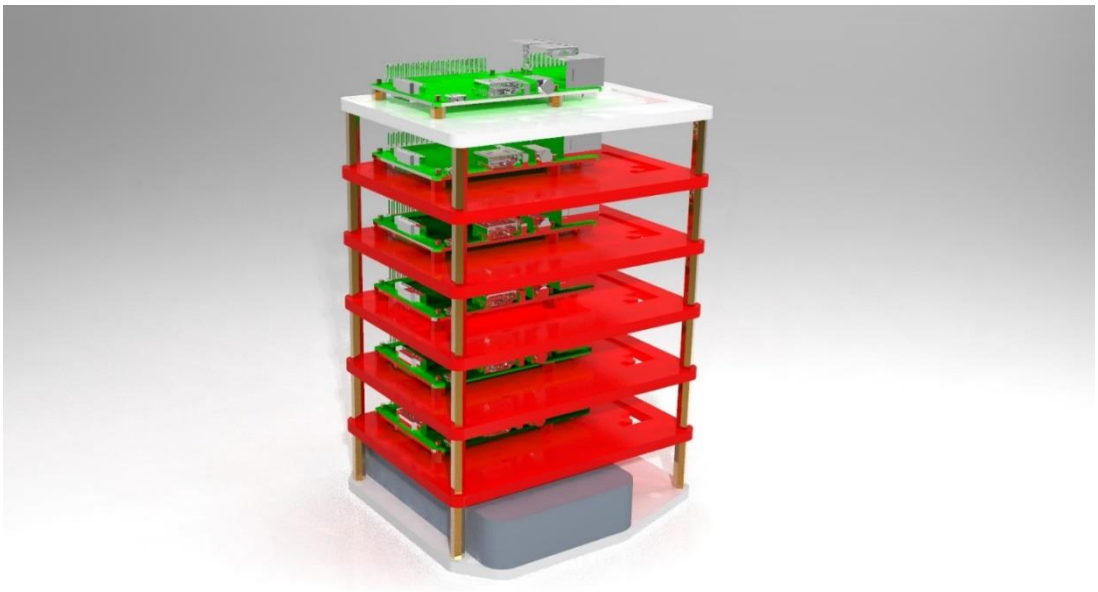


Figure 5-4, Render of final casing design

The mounting plates are then stacked, and cables fed through the appropriate holes. Figure 5-4 shows a render of the final design while Figure 5-5 shows the constructed cluster.

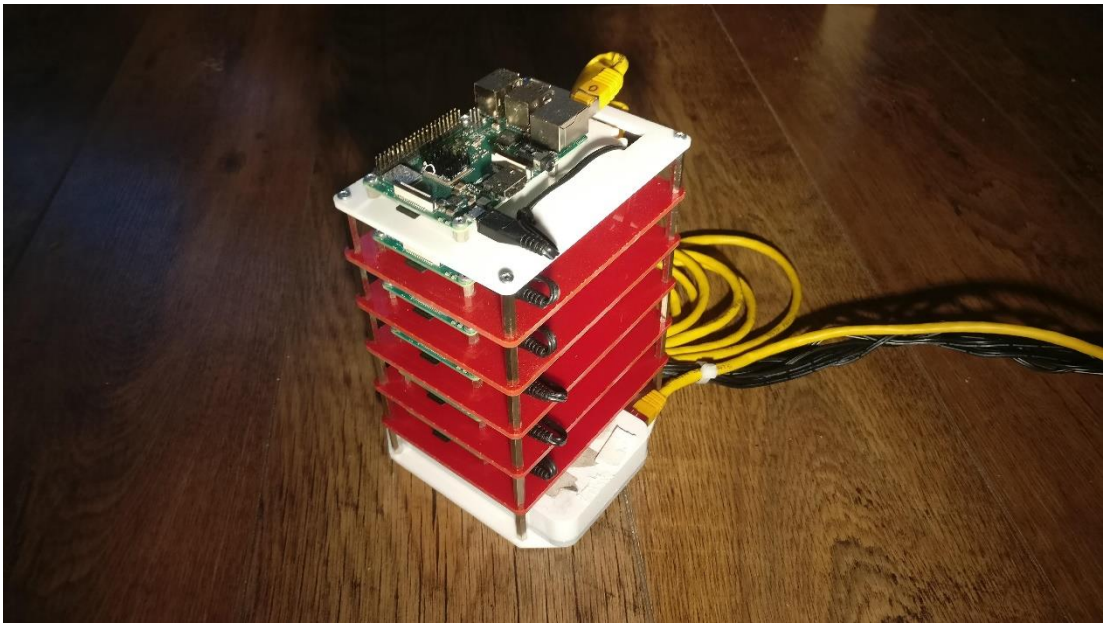


Figure 5-5, Final Cluster Casing Constructed

6 Method

Hardware should be constructed following the design outlined above in section 5.1

6.1 Software

Once the hardware has been set up the system needs to be set up as a static entity in the network. This means that the hostname and IP address must not change at any point. Changes need to be made to 3 main files to achieve this:

The contents of "/etc/hostname" are replaced with the chosen hostname.

The contents of "/etc/hosts" are changed to contain the IP address of all nodes on the system by hostname

The content of "/etc/dhcpcd.conf" is changed to declare the static IPV4 address of the current node

These changes are shown and highlighted in Appendix A.

The dependencies shown in Table 3 must all be installed and configured on all nodes for xoopic and xgrafix to run correctly. It is vital for the system that all the install locations are identical as the programs will only be executed from a single location. As this is research software, all the dependencies are expected to be in as they would be for an Intel architecture system rather than the arm architecture in the Raspberry Pi modules. This means the compiler must be told where all the dependencies are.

Table 3, Table of dependencies

Reason	Dependency
displaying the GUI and generating the output graphs	X11 and Xpm Tcl/Tk ImageMagick
compiling the application	Gcc bison Fortran Compiler
to execute mathematical operations	Fftw Fftw3
For managing load across the cluster	MPI for parallel runs HDF5 for parallel dump files,

From the extracted xgrafix folder the following command configures xgrafix for installation

```
sudo TCL_LIBDIR_PATH=/usr/lib/arm-linux-gnueabihf/
TK_LIBDIR_PATH=/usr/lib/arm-linux-gnueabihf/ \
sh run_conf.sh
--prefix=/home/pi/Network/xgrafix \
--with-tclsh=/usr/bin/tclsh8.6 \
--with-X11_LIBDIR=/usr/lib/arm-linux-gnueabihf \
--with-tclconfig=/usr/lib/tcl8.6 \
```

*--with-tkconfig=/usr/lib/tk8.6 *
*--with-tclhdir=/usr/include/tcl8.6 *
*--with-tkhdir=/usr/include/tcl8.6 *
*--with-sfftw-incdir=/usr/include *
--with-xpm=/usr/lib/arm-linux-gnueabihf
Then compile and install the program with the command:

sudo make install -j 4

From the extracted xoopic folder the following command configures xoopic for installation

sudo TCL_LIBDIR_PATH=/usr/lib/arm-linux-gnueabihf/
*TK_LIBDIR_PATH=/usr/lib/arm-linux-gnueabihf/ *
sh run_conf.sh
--prefix=/home/pi/Network/xoopic/
--with-xgrafix-libdir= /home/pi/Network/xgrafix/lib
--with-xgrafix-incdir= /home/pi/Network/xgrafix/include
*--with-tclsh=/usr/bin/tclsh8.6 *
*--with-X11_LIBDIR=/usr/lib/arm-linux-gnueabihf *
*--with-tclconfig=/usr/lib/tcl8.6 *
*--with-tkconfig=/usr/lib/tk8.6 *
*--with-tclhdir=/usr/include/tcl8.6 *
*--with-tkhdir=/usr/include/tcl8.6 *
*--with-sfftw-incdir=/usr/include *
--with-xpm=/usr/lib/arm-linux-gnueabihf
Then compile and install the program with the command:

sudo make install -j 4

An NFS sharing software must be installed on the master node and configured to share the Network folder. Each of the nodes should be configured to load mount the share on boot.

Execution can then take place by running the following command.

mpirun -n 4 --hostfile hosts xoopic -nox -i default.inp -dp 500-s 1000
Where -n indicates the number of cores to execute on -i the input file, -dp the dump period (in steps) and -s the number of steps. The command above executes *"default.inp"* on four cores to 1000 steps dumping the output every 500 steps.

6.2 Testing and Benchmarking

Cluster Computers will only perform faster when they are given an appropriate load. Processes which require sequential data operations gain limited speed increases whereas processes, where all operations can be done concurrently, can have a linear speed increase when the core count is increased. Unfortunately, in the real world, it is unlikely that an algorithm does not have elements of both. Benchmarks are designed to test both internode communication and the scalability of systems through different numbers of nodes. To properly test the system, Benchmarks testing both the extreme scenarios and a real-world scientific scenario have been

chosen. These should be able to identify weaknesses and strengths for cluster-based systems. Each of the benchmarks will be run with differing numbers of nodes and compared against the same benchmarks ran on a pc with the same software installed. This will indicate the effectiveness of the cluster and how it can be used with minimal waste. It is important to note that these benchmarks are written for efficiency on x86 or x64 based systems. This means that the benchmarking will only be testing the CPU so several high-performance components such as the GPU could be removed or utilised if the code was rewritten to include them.

6.2.1 Plasma Codes

2-Dimensional Plasma Codes are a good benchmark for Cluster computing. The model area is split into regions as shown in Figure 6-1 each of the regions can then be assigned to a separate core on the system. The benchmarks will be based on xoopic, an object orientated particle in cell simulator[25]. The simulation observes the change in energy of the particles in the field. To calculate the next frame for each particle the energy of those surrounding it must be known from the previous frame. When this is all contained within a region it is a trivial issue[26]. However, when close to the edge of a region there is a need to communicate between the nodes handling on the adjacent regions. This is a good example of a problem with a highly parallelisable solution with a bottleneck dictating the maximum speed of operation. The effect of this slowdown can be minimalised by assigning adjacent regions to the same nodes and minimalizing the communication drawn between the different nodes this is where a low core count node lacks performance compared to a high core count node. The results to all calculations are dumped into files labelled as appropriate

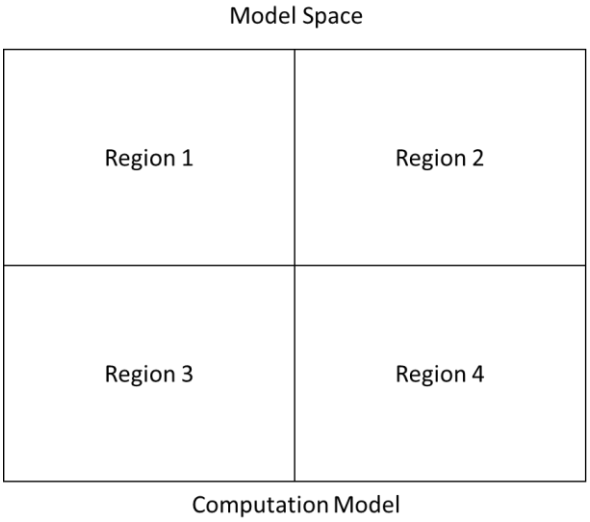


Figure 6-1, 2-Dimensional plasma simulation regions

To Appropriately test the system with different values multiple plasma-based benchmarks have been carried out on a variety of information. This should give us an overall idea of the speed increase introduced from both increasing the number of nodes and the minimum sample size for it to be effective to use a cluster versus not using a cluster. The Plasma benchmarking

is focused around results calculated from 2 profiles: The Default profile and the Afterburner4 profile.

Verboncoeur et al.'s Default.inp [27] states the input profile defines a *"Beam in right circular cylinder. This is useful for obtaining beam spreading curves, for example, by adjusting the injection current and the magnetic field."*

Verboncoeur et al.'s Afterburner4.inp [27] states the input profile defines a *"High-energy electron bunch enters a quiet plasma in cylindrical geometry -- Modeling the SLAC "afterburner" concept of Tom Katsouleas. … This simulation models a beam-plasma wake-field accelerator"*

The default profile is limited to 10 executing cores whereas the Afterburner can execute with all 24 Computational cores. The default benchmark will be used for ensuring the system fully works and can appropriately distribute the workload whereas the Afterburner profile will be used for producing a worked performance result.

6.2.2 Benchmark

A range of performance tests will be executed to allow us to analyse how the cluster performs with plasma-based applications. These will be split into preliminary testing and full testing.

In preliminary testing, the aim is to show how the cluster can perform compared to the workstation when a simple task is applied. To test this, it is important to compare single core performance for both the Raspberry Pi and the workstation for a range of results. To determine if the cluster is working a basic program with a reduced number of cores should be used. As the default profile is used for this testing, the system is limited to 10 cores on the cluster. To effectively provide a realistic result two cores from the first five nodes will be used and the full four cores of the workstation. Each setup will be tested with seven variations starting with 10^0 and finishing with 10^6 increasing by a power of 10 each time. This should show the saturation point where it becomes faster to use more nodes and will allow us to demonstrate the characteristic of the cluster. By observing the point at which it is equivalent speed for one core and ten cores to execute it is possible to understand the critical relationship between the number of cores assigned to a problem and the execution time.

The available cores can be controlled by limiting slots defined in the host file given to mpirun. Contents of the host file is shown below:

```
# Hostfile
ClusterNode0 slots=2
ClusterNode1 slots=2
ClusterNode2 slots=2
ClusterNode3 slots=2
ClusterNode4 slots=2
ClusterNode5 slots=2
```

Each line of this declares a host address, and the number of slots (Processing cores) available on each of the machines this test gives a good idea of how the performance scales while executing smaller faster calculations.

When testing it is important that the cluster has reached a point where it is faster to engage all the nodes. Therefore, a number of steps substantially greater than the point identified in preliminary testing must be selected. Running a benchmark with an ascending number of cores each time will provide us with an idea of both how parallelisable the code is and if Amdahl's Law provides an appropriate approximation for the relationship.

For more complete testing, the afterburner profile will be used. Once a large enough step value (over an order of magnitude larger) has been chosen, the time taken to execute the code with different numbers of cores can be recorded. A simple comparison of how the speed of communication between cores in a single node and cores in multiple nodes will affect the computation time will be tested by executing on 4 four cores.

Another key factor in analysing the success of the project is how power consumption compares to the workstation machine. For this monitoring the consumption of different aspects of the cluster and the apparent cost of the simulation is necessary. It is possible to make assumptions based on the maximum power consumption of both the nodes and the workstation. Using complex power management ideas suggested by Alves Filho et al. will show the potential system efficiency[28].

7 Results

7.1 Preliminary testing

To justify the testing methodology for the project a brief set of testing was done with reduced testing size, for this a variety of steps of execution was tested with different numbers of cores being used each time.

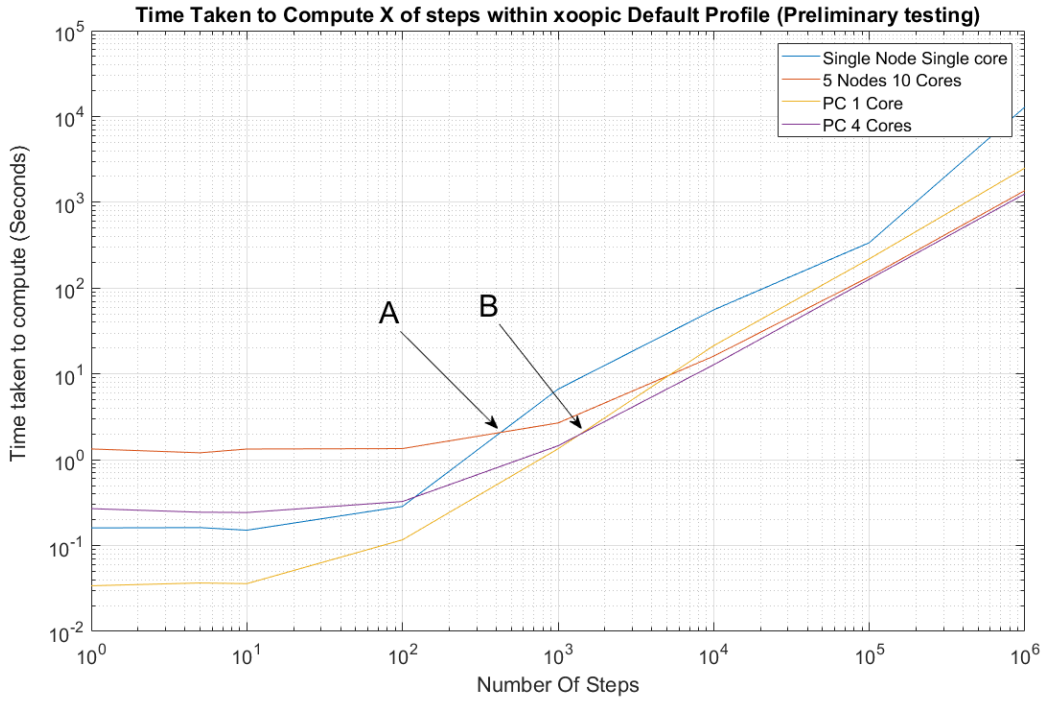


Figure 7-1, Graph Showing Results of Preliminary tests with xoopic Default Profile

Figure 7-1 is marked with 2 points A and B. A corresponds to the point in which the problem begins to be more effective to execute across ten cores rather than 1. This shows when the problem gets larger, it begins to scale more effectively. With this problem, at the point where the system is executing more than 428 steps, it becomes more efficient to use the cluster. B corresponds to the point where it is more effective to use four cores on the workstation computer. As each of the cores on the workstation is more powerful it becomes more effective after a greater number of steps than with the single core of the Raspberry Pi. The workstation becomes more efficient at around 1370 steps.

At this point, it is possible to calculate the speed increase provided by each additional core the system can use to execute. Figure 7-2 shows us the time taken to calculate 428 steps with different numbers of cores and allows us to observe how each simulation can have an optimal number of cores to execute on. For the size of the problem being executed it is most efficient on three cores. This demonstrates that if the number of nodes is too large to effectively split the problem into then it becomes more efficient to the simulation on fewer cores. This is expected . The most appropriate solution is to ensure that the cluster is predominantly utilised for more extensive simulations.

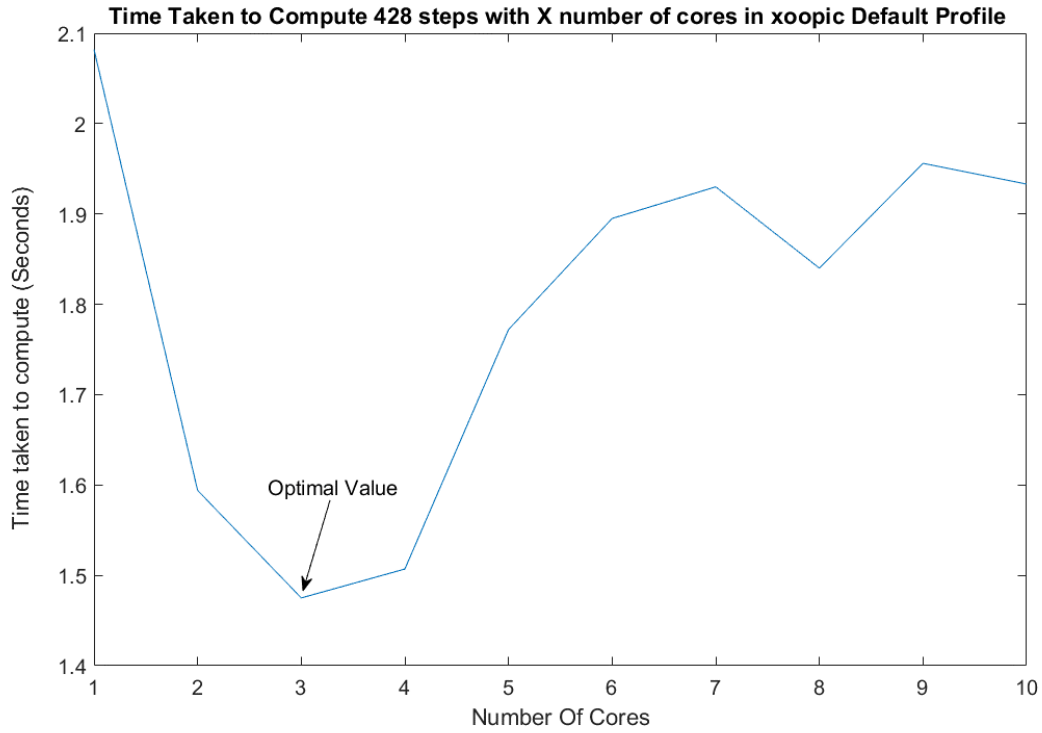


Figure 7-2, Graph showing the time taken to compute 428 steps with varying numbers of cores.

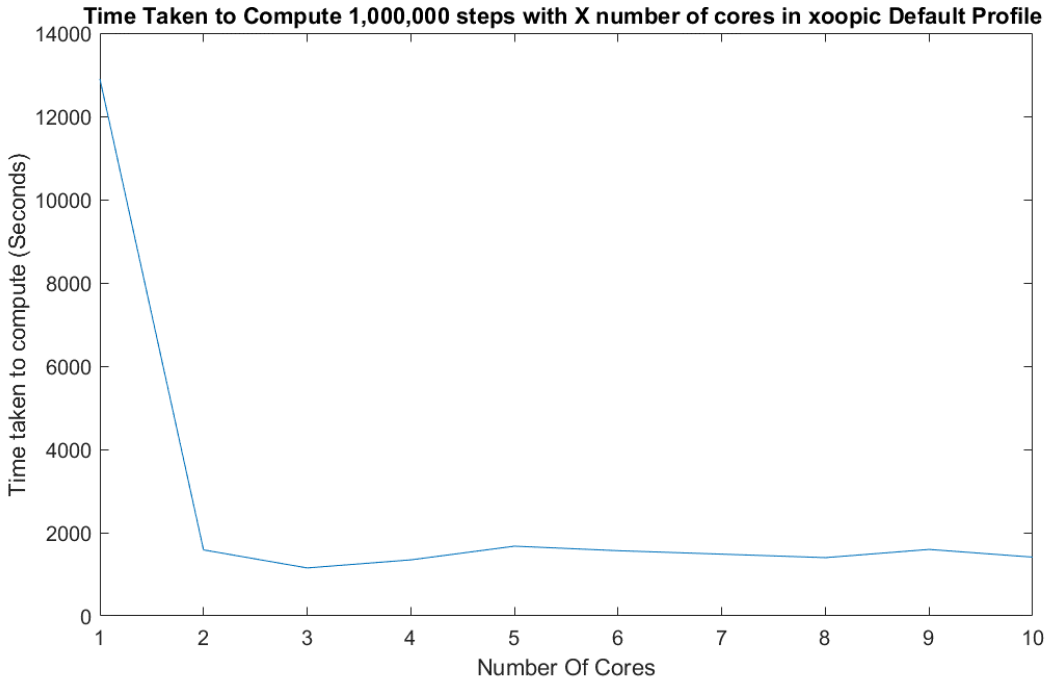


Figure 7-3, Graph showing time taken to compute 1,000,000 steps with varying numbers of cores

Shown above in Figure 7-3, when the problem approaches a larger size, there is a more defined decrease in execution time from increasing the number of cores. It is clear with this many execution steps that the cluster begins to fully utilise the 10 cores assigned to this task.

7.2 Benchmarking Results

Figure 7-4 shows the ratio of time taken to simulate a beam plasma Wakefield generator to 1000 steps (Afterburner4 input file), in an ideal situation, with a large enough load. The more cores used, the more effective the cluster becomes. Figure 7-4 also features a fitting line with the equation

$$y = \frac{2379\, e^{-\frac{2191\, x}{5000}}}{5000} + \frac{1403\, e^{-\frac{1837\, x}{100000}}}{2000},\ x > 1,\ x \in R$$

Equation 3, Fitting Line for Figure 7-4

Using this equation, it is possible to extrapolate the point at which it is no longer beneficial to use the cluster as shown in Figure 7-5. The marked point is where the execution time is 10% of the single node execution time. This shows that 106 cores (or 27 Raspberry Pis) are needed to achieve this goal. Each node added to the system will speed the system up by a smaller degree than the last. This limits the viability of using the cluster for this application.

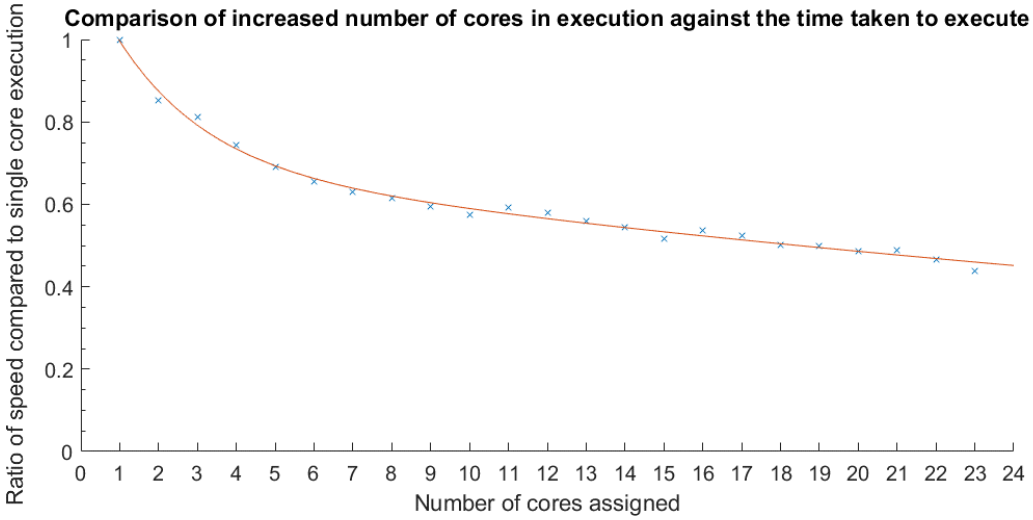


Figure 7-4, Graph showing speedups when cores added to the afterburner profile with 1000 steps.

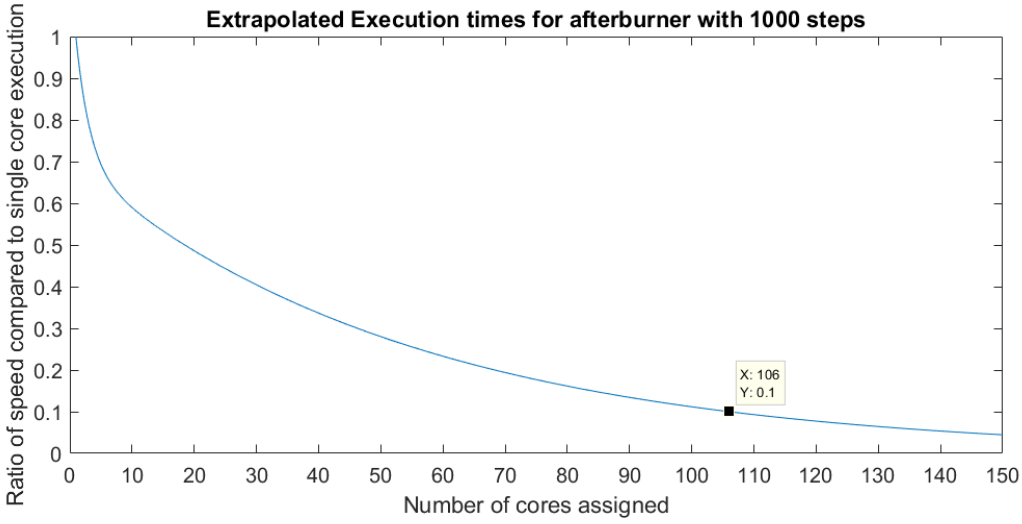


Figure 7-5, Graph showing extrapolated speedup to 150 execution cores.

This data can be used to evaluate the worth of each node in the system. Therefore, each node is worth less than the last as it provides less performance increase.

$$\lim_{x\to\infty}\left(\frac{2379\,e^{-\frac{2191\,x}{5000}}}{5000}+\frac{1403\,e^{-\frac{1837\,x}{100000}}}{2000}\right)=0$$

Equation 4, Limit to infinity of Equation 3

As shown above in Equation 4, theoretically that every node added to the system causes a reduction in execution time. Figure 7-2 shows that this is only the case when the number of cores is not higher than the optimal number of cores consequently this relationship is only true when the number of cores is less than or equal to the optimal number of cores. As cores are added the processing time reduces provided that the number of iterations executed also tends to infinity. The most appropriate way to ensure this is the case is to increase the size of the simulation to the maximum simulation size possibly needed or to maintain a modest core count.

If Amdahl's law [16] is applied to this result it shows that the plasma simulation code is around 62% parallelisable this assumes that there is no performance loss due to slow networking or other bottlenecks. As said previously [2] this is not the case, therefore, it is more appropriate to conclude that this software instead contains greater than 62% parallelisable code.

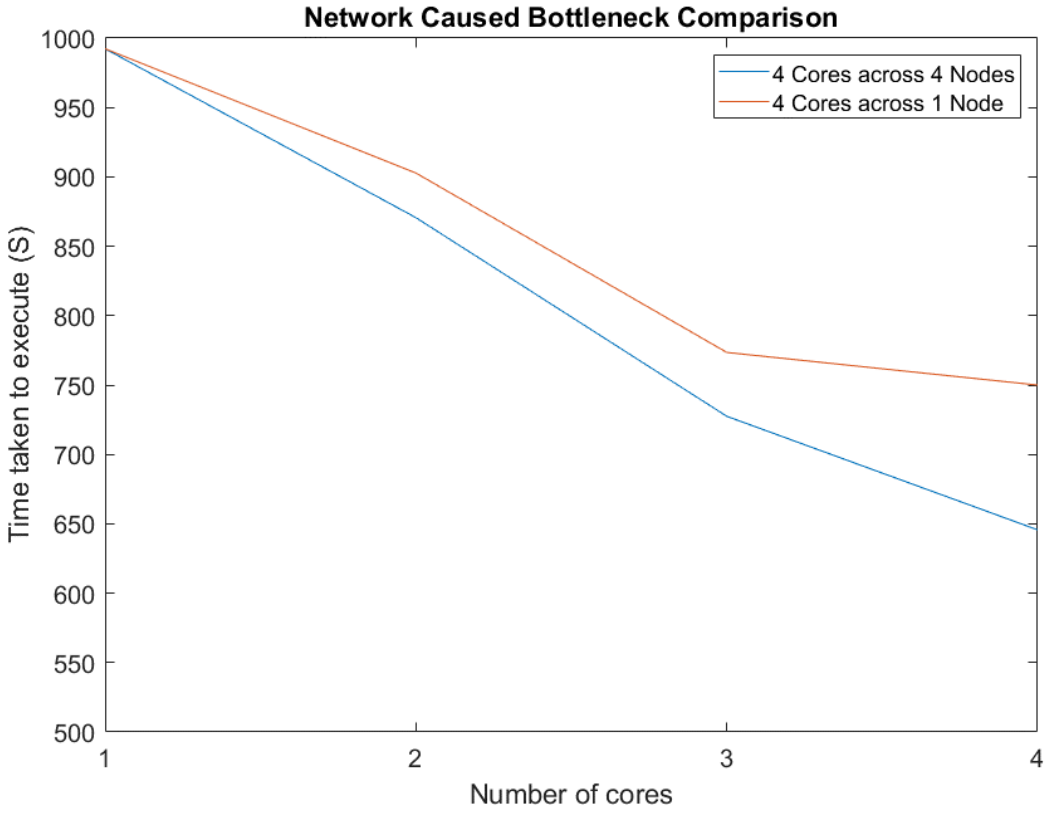


Figure 7-6, Graph showing network-based Slowdown

Figure 7-6 shows the effect of separating the tasks over a local network. This shows that execution time using a single core per node is much quicker than when a single node is used to execute the code. This was an unexpected result however overall it shows that the task executing has a wider resource requirement then just CPU utilisation. This means that during this benchmark it is possible to say that there is an additional bottleneck in the system.

7.3 Efficiency

Figure 7-7 illustrates the maximum power consumption of different elements within the system compared to the full power draw of a traditional workstation. This shows that the Power draw of the cluster is only just greater than the power draw of just the CPU in the workstation.

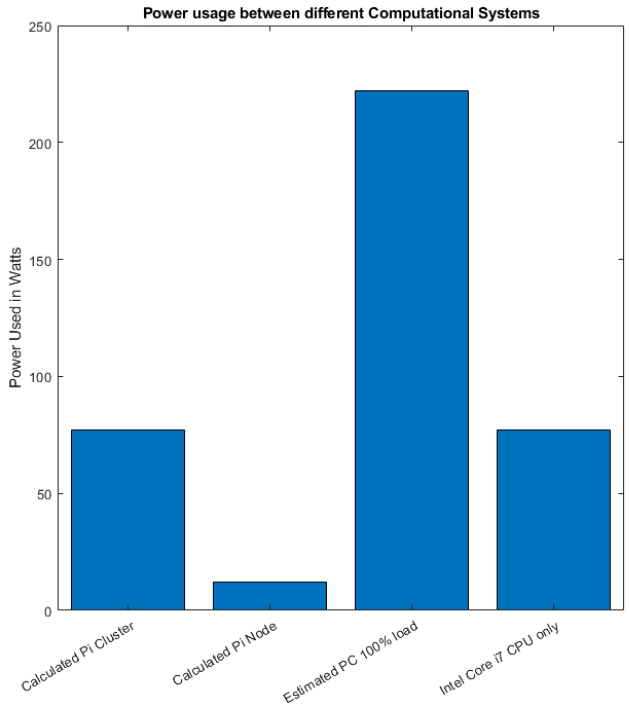


Figure 7-7, Power Consumption Comparison

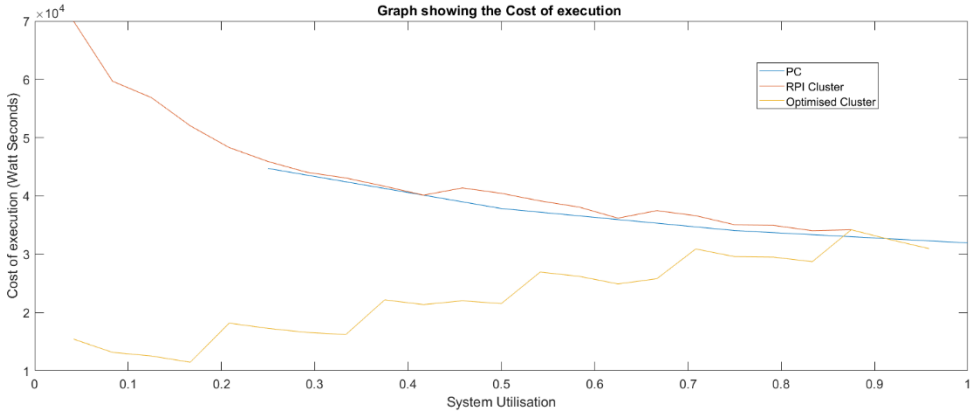


Figure 7-8, Graph showing the cost of execution for 1000 steps

Therefore, from Figure 7-8 the cost of execution on the pi cluster and a workstation pc remains similar. This assumes that there is a negligible decrease in power consumption from less than 100% utilisation. If the cluster were optimised as per ideas discussed by Alves Filho et al. [28], the cost of execution would be further reduced. This would increase the efficiency of the system.

8 Discussion

The timing data was gathered using the time command. The delay caused by the time command has been assumed to be negligible as the execution of the time command takes approximately 0.01 seconds. While this assumption has been made, it is worth noting that this is a considerable portion of the execution time in the data shown for single core pc execution in Figure 7-1 due to this it would be inappropriate to use data in this region for calculative purposes. After these preliminary tests, a key factor of execution was ensuring that the testing parameters were large enough to allow us to ignore this unpredictable error. An additional source of error is present in Figure 7-7 and Figure 7-8. Due to the unavailability of measuring equipment to calculate power draw, the power draw has been calculated based on maximum values given by the minimum power supply values. This shows the minimum efficiency it is possible for each system to meet. Clouter et al.[29] discovered that under full their 25 node cluster drew a maximum of 92W using similar devices. The final control measure introduced was to ensure that applications were only ran benchmarked after a full system reboot to ensure that no key files are still available in RAM.

For the execution of Particle in Cell simulation codes using a small cluster can be beneficial in terms of the cost and time of computation when compared to a workstation however there is a considerably more involved setup. This could counter the benefit of using a cluster for computational models where budget is not and will not be an obstacle. Although the range of testing has been limited beyond this scope, further investigation into a more extensive variety of applications would allow us to analyse how effective the cluster is compared to other existing solutions. Furthermore, in the current form, the cluster constructed here is limited in scalability due to the insufficient speed of localised components being unable to maintain the simultaneous access. The data supports that for smaller problems it is important to not underutilise the cluster as this results in a slower execution with a higher number of cores and therefore a higher power draw. The data shown in Figure 7-3 and Figure 7-4 follows a clear trend with some relation to Ahmdal's law as defined in section 4. The deviation from this trend is dictated by bottlenecks in the system such as memory bandwidth and I/O bandwidth. These bottlenecks are characterised by the difference between Equation 3 and Equation 1 where the efficiency of the code is assumed to be >62%. An additional limitation which must be accounted for is that due to the way the problem is broken up there is no advantage of having nodes with any greater performance than the least powerful node. This is due to each frame of data needed to be completed before the next can begin being calculated.

If the cluster maintains 100% load for a year the overall cost will be a maximum of £95 this is considerably cheaper than an average workstation which would cost around £270 this 35% cost reduction. While adding nodes to the cluster will also increase the running cost linearly, each node consumes a fraction of the power consumption of a traditional workstation. This shows the excellent scalability of low powered nodes in cluster computing. Furthermore, if advanced system management principals are applied to the cluster it would be possible to

tailor the setup based on a speed cost curve which balances the cost of calculating solution with the time taken.

A different possible alternative for further investigation is the innovation of custom silicon such as dedicated integrated circuits which excel at specific tasks. During the last few years high profile mathematically based problems, such as bitcoin mining have generated the profits necessary to fund custom development into application specific integrated circuits (ASICs) which for their dedicated task vastly outperform existing computing solutions [30].

Reflecting upon the specification first dictated in October 2018 [1] the success of the system can be evaluated. The Hardware has been successful as shown by the cluster shown in Figure 5-5. The software element of the cluster can be deemed effective and working. For the benchmarking applications have been executed simultaneously across all nodes on the cluster. This shows that the software section has been achieved. The software was extended to include a basic GUI for display purposes. Due to unexpected complexity with the plasma-based codes, the benchmarking section was shortened and not as extensive as initially planned. Due to this obstacle benchmarking of the workstation machine was limited to showing the competitivity of the system. Also, the industry leading benchmarks for cluster computing (LINPACK [31]) have been omitted. This would be the next stage in cluster benchmarking.

Further research into the field would allow full testing of larger scale clusters across a range of low power architectures. These could be combined with high power nodes to execute serial operations while lower power nodes conduct largely parallel operations.

9 Conclusion

To conclude, while effective in cost of execution a newer infrastructure required to implement the system into modern HPC solutions would be expensive with little increase in performance. A solution to this would be to head towards a bespoke solution with a focus on networking speed and bandwidth between components. While this could expensive to design, a bespoke solution would allow the removal of unnecessary features and the integration of specialised coprocessor units. Unfortunately, this would probably prove non-viable for most companies. There is much research to be done in this area with the integration of higher speed components and bespoke systems. There are many advancements to be made in this field however substantial investment would be needed.

10 References

[1] B. Hague, "Final Year Project Specification Report," 2018.

[2] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. O'Brien, "Iridis-pi: a low-cost, compact demonstration cluster," *Cluster Comput.*, vol. 17, no. 2, pp. 349–358, Jun. 2014.

[3] P. Abrahamsson *et al.*, "Affordable and Energy-Efficient Cloud Computing Clusters: The Bolzano Raspberry Pi Cloud Cluster Experiment," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, 2013, pp. 170–175.

[4] K. Candelario, C. Booth, A. St. Leger, and S. J. Matthews, "Investigating a Raspberry Pi cluster for detecting anomalies in the smart grid," in *2017 IEEE MIT Undergraduate Research Technology Conference (URTC)*, 2017, pp. 1–4.

[5] John Russell, "Supercomputer Sales Drove 2016 HPC Market Up to Record $11.2 Billion," *HPC Wire*, 06-Apr-2017. [Online]. Available: https://www.hpcwire.com/2017/04/06/supercomputer-sales-drove-2016-hpc-market-record-11-2-billion/. [Accessed: 10-Apr-2019].

[6] S. Tichenor and A. Reuther, "Making the business case for high performance computing: a benefit-cost analysis methodology," *CTWatch Q.*, vol. 2, no. 4a, 2006.

[7] T. Ludwig, "The costs of HPC-based science in the exascale era," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, 2012, pp. 2120–2188.

[8] K. Asanovic, B. C. Catanzaro, D. A. Patterson, and K. A. Yelick, "The Landscape of Parallel Computing Research : A View from Berkeley," *Electr. Eng. Comput. Sci. Univ. Calif. Berkeley Tech. Rep. No UCBEECS2006183 December*, 2006.

[9] P. Kogge *et al.*, "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems," 2008.

[10] M. A. Laurenzano *et al.*, "Characterizing the Performance-Energy Tradeoff of Small ARM Cores in HPC Computation," 2014, pp. 124–137.

[11] A. Olofsson, T. Nordström, and Z. Ul-Abdin, "Kickstarting High-performance Energy-efficient Manycore Architectures with Epiphany," Dec. 2014.

[12] M. A. Kuhail, S. Cook, J. W. Neustrom, and P. Rao, "Teaching Parallel Programming with Active Learning," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2018, pp. 369–376.

[13] R. Brown, J. Adams, S. Matthews, and E. Shoop, *Teaching Parallel and Distributed Computing with MPI on Raspberry Pi Clusters*. 2018.

[14] J. Kiepert, "Creating a Raspberry Pi-Based Beowulf Cluster," 2013.

[15] J. C. Kiepert, "WSNFS: A Distributed Data Sharing System for In-Network Processing," Boise State University, 2014.

[16] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *American Federation of Information Processing Societies: Proceedings of the {AFIPS} '67 Spring Joint Computer Conference, April 18-20, 1967, Atlantic City, New Jersey, {USA}*, 1967, vol. 30, pp. 483–485.

[17] J. L. Gustafson and J. L., "Reevaluating Amdahl's law," *Commun. ACM*, vol. 31, no. 5, pp. 532–533, May 1988.

[18] J. Parker Mitchell *et al.*, "PERFORMANCE, MANAGEMENT, AND MONITORING OF 68 NODE RASPBERRY PI 3 EDUCATION CLUSTER: BIG ORANGE BRAMBLE (BOB) for Modeling & Simulation International (SCS)," 2017.

[19] Raspberry Pi Foundation, "Rasbian." 2013.

[20] R. Buyya, "High Performance Cluster Computing: Architectures and Systems, Volume 1."

[21] A. Barak and O. La'adan, "The MOSIX multicomputer operating system for high performance cluster computing," *Futur. Gener. Comput. Syst.*, vol. 13, no. 4–5, pp. 361–372, Mar. 1998.

[22] J. J. Dongarra, S. W. Otto, M. Snir, and D. Walker, "An Introduction to the MPI Standard," 1995.

[23] "MPI: A Message-Passing Interface Standard," 2015.

[24] Python Software Foundation, "Python Language Reference." 2001.

[25] J. P. Verboncoeur, A. B. Langdon, and N. T. Gladd, "An object-oriented electromagnetic PIC code," 1995.

[26] P. J. Mardahl and J. P. Verboncoeur, "Progress in Parallelizing XOOPIC."

[27] J. P. Verboncoeur, T. Gladd, P. J. Mardahl, K. Cartwright, and B. Peter, "XOOPIC." PTSG, 2010.

[28] S. E. Alves Filho, A. M. F. Burlamaqui, R. V. Aroca, and L. M. G. Goncalves, "NPi-Cluster: A Low Power Energy-Proportional Computing Cluster Architecture," *IEEE Access*, vol. 5, pp. 16297–16313, 2017.

[29] M. Cloutier, C. Paradis, V. Weaver, M. F. Cloutier, C. Paradis, and V. M. Weaver, "A Raspberry Pi Cluster Instrumented for Fine-Grained Power Measurement," *Electronics*, vol. 5, no. 4, p. 61, Sep. 2016.

[30] M. Bedford Taylor, "Bitcoin and the age of Bespoke Silicon," in *2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, 2013, pp. 1–10.

[31] "The Linpack Benchmark | TOP500 Supercomputer Sites." [Online]. Available: https://www.top500.org/project/linpack/. [Accessed: 28-Apr-2019].

11 Appendix

Appendix A Changed files from Raspberry Pi

All taken from the master node ClusterNode0. Small changes are apparent on different nodes within the cluster

/etc/dhcpcd.conf

```
# A sample configuration for dhcpcd. See dhcpcd.conf(5) for details.

# Allow users of this group to interact with dhcpcd via the control socket.
#controlgroup wheel

# Inform the DHCP server of our hostname for DDNS.
hostname

# Use the hardware address of the interface for the Client ID.
clientid
# or
# Use the same DUID + IAID as set in DHCPv6 for DHCPv4 ClientID as per RFC4361.
# Some non-RFC compliant DHCP servers do not reply with this set.
# In this case, comment out duid and enable clientid above.
#duid

# Persist interface configuration when dhcpcd exits.
persistent

# Rapid commit support.
# Safe to enable by default because it requires the equivalent option set
# on the server to actually work.
option rapid_commit

# A list of options to request from the DHCP server.
option domain_name_servers, domain_name, domain_search, host_name
option classless_static_routes
# Most distributions have NTP support.
option ntp_servers
# Respect the network MTU. This is applied to DHCP routes.
option interface_mtu

# A ServerID is required by RFC2131.
require dhcp_server_identifier
```

```
# Generate Stable Private IPv6 Addresses instead of hardware based ones
slaac private

# Example static IP configuration:
interface eth0
static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0
```

/etc/hostname

```
ClusterNode0
```

/etc/hosts

```
127.0.0.1       localhost
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

127.0.1.1       ClusterNode0
192.168.0.10    ClusterNode0
192.168.0.2     ClusterNode1
192.168.0.3     ClusterNode2
192.168.0.4     ClusterNode3
192.168.0.5     ClusterNode4
192.168.0.6     ClusterNode5
```

/etc/exports

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
```

*/home/pi/Network *(rw,sync,no_root_squash,no_subtree_check)*

Appendix B GUI Code

Code for local execution

```
##########################################
#       Ben Hague, MPI Xoopic GUI      #
# Written 2019 for Python 3 with TKinter #
# Compatible cross platform written for  #
# rasbian linux on Raspberry pi Cluster  #
##########################################
# Import TKinter for GUI interface
from tkinter import *
# Import system for running external programs
from os import system

# Program wide variables (For portability between machines)
# Location of input file (for ease of demonstration)
Benchmark = "/home/pi/Network/xoopic/xoopic/inp/"
# Location of Xoopic install without MPI
XoopicNOMPI = "xoopic"
# Machine Specific variables (4 cores 1 node for pc, 4 cores 67 nodes for cluster)
MaxNodes = 4
MaxCores = 4

# Function for launching xoopic to execute with the specified file in the GUI
def launchXoopic():
   system("mpirun -n " + str(Nodes.get()*Cores.get()) + " xoopic -nox -i " + Benchmark
+INPFILE.get()+".inp  -dp "+ str(Iterations.get()) +" -s " + str(Iterations.get()))
# Function for launching xoopic to view result with the specified file in the GUI
def launchResult():
   system(XoopicNOMPI + " -i " + Benchmark +INPFILE.get()+".inp  -dp "+
str(Iterations.get()) +" -s " + str(Iterations.get()))

# define window object and call title
window = Tk()
window.title('Xoopic MPI Demonstrator')

# Write label for file name to window
INPLab = Label(window,text = "Input File Name")
INPLab.pack()
# Write text entry box to window with default text being "default"
INPFILE = Entry(window)
INPFILE.insert(10,"default")
INPFILE.pack()

# Write label for nodes slider to window
NLabel = Label(window,text = "Number of Nodes")
NLabel.pack()
# Write slider to window with default value of 1
```

```
Nodes = Scale(window, from_=1, to=MaxNodes, orient=HORIZONTAL)
Nodes.set(1)
Nodes.pack()

# Write label for Cores slider to window
CLabel = Label(window,text = "Number of Cores per Node")
CLabel.pack()
# Write slider to window with default value of 1
Cores = Scale(window, from_=1, to=MaxCores, orient=HORIZONTAL)
Cores.set(1)
Cores.pack()

# Write label for execution steps slider to window
ILabel = Label(window,text = "Number of Steps to execute")
ILabel.pack()
# Write slider to window with default value of 1000
Iterations = Scale(window, from_=1, to=100000, orient=HORIZONTAL)
Iterations.set(1000)
Iterations.pack()

# Write view results and run benchmark buttons to window
# Command links them to their appropriate function
Launch = Button(window,text = 'Launch Benchmark', command=launchXoopic)
Results = Button(window,text = 'View Results', command=launchResult)
Launch.pack(padx=150, pady =10)
Results.pack(padx=150, pady=20)

# show the window
window.mainloop()
```

Code for remote execution

```
##########################################
#       Ben Hague, MPI Xoopic GUI      #
# Written 2019 for Python 3 with TKinter #
# Compatible cross platform written for  #
# PC control on local network to cluster #
##########################################
# Import TKinter for GUI interface
from tkinter import *
# Import system for running external programs
from os import system

# Program wide variables (For portability between machines)
# Location of input file (for ease of demonstration)
Benchmark = "/home/pi/Network/xoopic/xoopic/inp/"
# Location of Xoopic install without MPI
XoopicNOMPI = "xoopic"
# Machine Specific variables (4 cores 1 node for pc, 4 cores 67 nodes for cluster)
```

```
MaxNodes = 4
MaxCores = 4

# Function for launching xoopic to execute with the specified file  on the cluster
def launchXoopic():
   Hostgen()
   system("ssh ClusterNode0 mpirun -n " + str(Nodes.get()*Cores.get()) + " --hostfile hosts
xoopic -nox -i " + Benchmark +INPFILE.get()+"  -dp "+ str(10**Dumpfreq.get()) +" -s " +
str(10**Iterations.get()))
# Function for launching xoopic to view result with the specified file in the local GUI
def launchResult():
   system(XoopicNOMPI + " -i " + Benchmark +INPFILE.get()+".inp  -dp "+
str(Dumpfreq.get()) +" -s " + str(Iterations.get()))
# Generate the hostfile
def Hostgen():

   for i in range(0,Nodes.get()):
      system("echo ClusterNode" +str(i)+" slots="+str(Cores.get()) + " >> hosts")
   system("scp hosts ClusterNode0:hosts")
   system("rm hosts")


# define window object and call title
window = Tk()
window.title('Xoopic MPI Demonstrator for PC')

# Write label for file name to window
INPLab = Label(window,text = "Input File Name")
INPLab.pack()
# Write text entry box to window with default text being "default"
INPFILE = Entry(window)
INPFILE.insert(10,"default")
INPFILE.pack()

# Write label for nodes slider to window
NLabel = Label(window,text = "Number of Nodes")
NLabel.pack()
# Write slider to window with default value of 1
Nodes = Scale(window, from_=1, to=MaxNodes, orient=HORIZONTAL)
Nodes.set(1)
Nodes.pack()

# Write label for Cores slider to window
CLabel = Label(window,text = "Number of Cores per Node")
CLabel.pack()
# Write slider to window with default value of 1
Cores = Scale(window, from_=1, to=MaxCores, orient=HORIZONTAL)
Cores.set(1)
Cores.pack()
```

```
# Write label for execution steps slider to window
ILabel = Label(window,text = "Number of Steps to execute (10^n)")
ILabel.pack()
# Write slider to window with default value of 1000 (10^3)
Iterations = Scale(window, from_=1, to=6,resolution=0.1, orient=HORIZONTAL)
Iterations.set(3)
Iterations.pack()

# Write label for Dump Freqency slider to window
DPLabel = Label(window,text = "Number of steps between Data Dumps (10^n)")
DPLabel.pack()
# Write slider to window with default value of 1000 (10^3)
Dumpfreq = Scale(window, from_=1, to=6,resolution=0.1, orient=HORIZONTAL)
Dumpfreq.set(3)
Dumpfreq.pack()

# Write view results and run benchmark buttons to window
# Command links them to their appropriate function
Launch = Button(window,text = 'Launch Benchmark', command=launchXoopic)
Results = Button(window,text = 'View Results', command=launchResult)
Launch.pack(padx=150, pady =10)
Results.pack(padx=150, pady=20)

# show the window
window.mainloop()
```

Appendix C Specification Report

Specification report for project 'Raspberry Pi Computing Cluster'

Author: Ben Hague (201146260)

Project Supervisor: Dr James Walsh

Project Assessor: Dr Mohammad Hasan

Department of Electrical Engineering and Electronics

12 October 2018

Abstract

Cluster Computing is used widely in industry for computing large quantities of data in what would be an unrealistic timescale for an individual normal computer. During the last 5 years a massive decrease in price has led to a massive increase in availability of small low power single board computers. Combining the technologies used to process large data with clusters and cheaper single board computers there is a potential to substantially lower the cost of processing large data. This report outlies the specification for a project investigating how a cluster could be constructed and investigate the benefits and drawbacks of using lower power hardware to conduct calculations on large quantities of data.

Declaration

I confirm that I have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with any other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).

Contents

Specification Report Ben Hague Raspberry Pi Computing Cluster

Introduction

Cluster computing is used on a day to day bases to compute large amounts of data in far less time than possible by normal methods. It is used to split tasks into sets of smaller tasks that can be executed concurrently. This report outlines a rough timescale for each part of the project to produce a small computing cluster, the overall goals of the project and how the project will be validated.

Project Description

The aim of the project is to produce a cluster of Raspberry Pi computers capable of executing plasma code in a suitable period. A key objective of the project are to investigate the economic effectiveness of using arrays of cheaper single board computers above using a more expensive workstations or even traditional clusters. Over the project it will be important to ensure that software and hardware are correctly configured and working effectively.

Project Specification

The project has been divided into 4 separate categories, Hardware, Software, Benchmarking

and Paperwork. Each of these categories is a distinct deliverable which must be done to achieve success in the project. Within each of the Deliverables there is a small number of work packets which relate to the goals which must be completed for the section to succeed.

The outcome of this Project is to create a 6 node 24 Core Cluster out of Raspberry Pi 3 units, the cluster should also be able to run Plasma Simulation code from the Plasma Theory and Simulation Group (PTSG). The cluster should also be able to demonstrate the speed difference between a single node and a full cluster and show how this compares to using a traditional workstation.

Specification and aims for each deliverable:

- Hardware - The outcome of this deliverable is to create a 6 node 24 Core Cluster out of Raspberry Pi 3 units.
    - Construct Hardware Network
    - Design Casing
    - Construct Casing for Cluster

- Software – The outcome of this deliverable is to use an MPI to allow the cluster to work together and execute code, the cluster should also be able to run Plasma Simulation code from the Plasma Theory and Simulation Group (PTSG) [1]
    - Build Linux Software for network
    - Set up and install MPI software
    - Compile Benchmarking code
- Benchmarking – The outcome of this deliverable is to Test, Benchmark and improve the cluster built. The cluster should also be able to demonstrate the speed difference between a single node and a full cluster and show how this compares to using a traditional workstation.
    - Construct Benchmarking Tests
    - Conduct Cluster Benchmarking
    - Conduct Single Node Benchmarking
    - Conduct Workstation Benchmark
- Paperwork The outcome of this deliverable is to ensure that all the correct permissions and orders have been put forward at the correct time to allow the other parts of the project to be conducted in good time.
    - Project Forms (Appendix 3,4,5)
    - Specification Report
    - Order Form
    - Order Casing
    - Poster
    - Report

Methodology

The cluster is going to be based on the Beowulf style cluster computer. This means that one node is dedicated as the master node, the others are all designated as slave nodes. The master node uses a Message Passing Interface (MPI) to transfer data and commands to the slave nodes [2]. This makes one of the key tasks to install and set up a working MPI instance.

Following this, the next important set up will be to compile the code written by PTSG [1]. This has been done before for raspberry pi [3]. It is important to ensure that the codes are compiled compatible with the MPI and able to effectively distribute the load needed and provide a higher level of processing power.

Project Plan

The Project has been broken down into discrete work packets which appear on the Gantt chart shown in Appendix 1. Each packet of work has been scheduled with an appropriate deadline making it easy to identify the appropriate workload to be doing at any given time.

Work Packets and Deadlines

- Identify software dependencies and requirements by 31st October 2018
- Constructed hardware network by the 30th November 2018
- Build Linux software for network by the 30th November 2018
- Set up a working MPI with 6 nodes by 17th December 2018
- Compile Plasma Software for benchmarking by 31st January 2019
- Begin benchmarking the cluster in February 2019
- Calculate baseline benchmark from Single Raspberry Pi by 28th February 2019
- Calculate baseline benchmark from PC by 15th March 2019
- Construct casing for Raspberry Pi cluster by 1st April 2019

During this period as work is done a log will be kept of how work is going, any key discoveries, obstacles and solutions. Documentation for each section of the project will be made to produce a user manual as to how to use the cluster at the end of operation.

The project has been planned and split up in a work breakdown structure (WBS), a Gantt chart and network diagram is shown in Appendix 1 and 2 respectively.

Project Rationale and Industrial Relevance

This project aims to show how we could reduce the power consumption and therefore relative running costs associated with using cluster computers. Regardless of power consumption the project also shows that a lower cost cluster computing set up, would allow learning of how-to code for large numbers of nodes without using expensive and highly demanded supercomputer time. Problems which scale well for parallel are known as embarrassingly parallel examples of these are processes such as Monte Carlo Simulations or Simulations where the next calculation does not depend on the result of the previous one. Using this cluster to explore the suitability of small computing clusters when calculating plasma simulations will be useful those people who are conducting research in the plasma field.

Literature Review

During the research period of this project I have investigated basic principles and different types of cluster setups how they are used and what the advantages are [4]. I also specifically researched instances where Raspberry Pi clusters have been made for different uses. I specifically noted a conference that happening in Cambridge, USA where they discussed the use of a cluster for detecting anomalies in power grids. [5] The algorithms used for this application happened to be extremely scalable and led them to the conclusion that using an 8 node cluster matched the performance of a workstation for a significantly lower cost whilst using under half the power.

Conclusion

This project aims to construct a cluster of single board low power computers. The idea behind doing this is to show how cluster computers work and understand how to use them effectively. The end goal is to run plasma codes with higher efficiency then when a PC or traditional cluster is used. Each of the key deliverables has defined requirement for success and each of the work packets has been analysed and given a suitable time period and deadline. This is shown in the Network diagram in Appendix 2.

References

[1] Plasma Theory and Simulation Group, Electrical Computer Engineering Department, Michigan State University, "Plasma Theory and Simulation Group," [Online]. Available: https://ptsg.egr.msu.edu/.

[2] D. Nath, "Running an MPI Cluster within a LAN," MPI Tutorial, 2015. [Online]. Available: http://mpitutorial.com/tutorials/. [Accessed 08 10 2018].

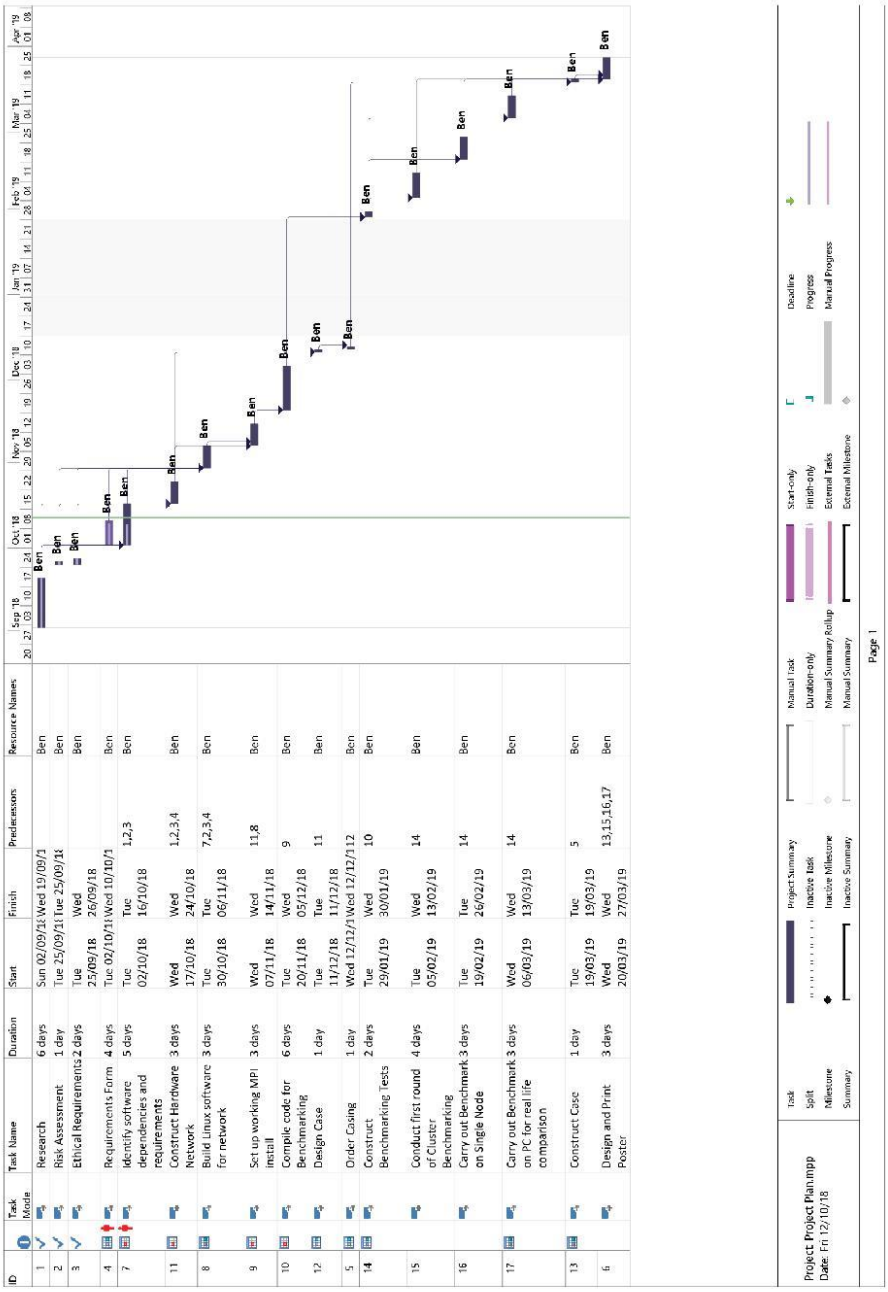
[3] Particle In Cell Consulting LLC, "Fun with Raspberry Pi – plasma simulation code performance," Particle In Cell Consulting LLC, 16 05 2016. [Online]. Available: https://www.particleincell.com/2016/raspberry-pi-performance/. [Accessed 08 10 2018].

[4] The Camber Group, "Clustering: A basic 101 tutorial," IBM Developer Works, 03 04 2002. [Online]. Available: https://www.ibm.com/developerworks/aix/tutorials/clustering/clustering.html. [Accessed 08 10 2018].

[5] K. Candelario, C. Booth, A. S. Leger and S. J. Matthews, "Investigating a Raspberry Pi cluster for detecting anomalies in the smart grid," in *Investigating a Raspberry Pi cluster for detecting anomalies in the smart grid*, Cambridge, MA, USA, 2017.

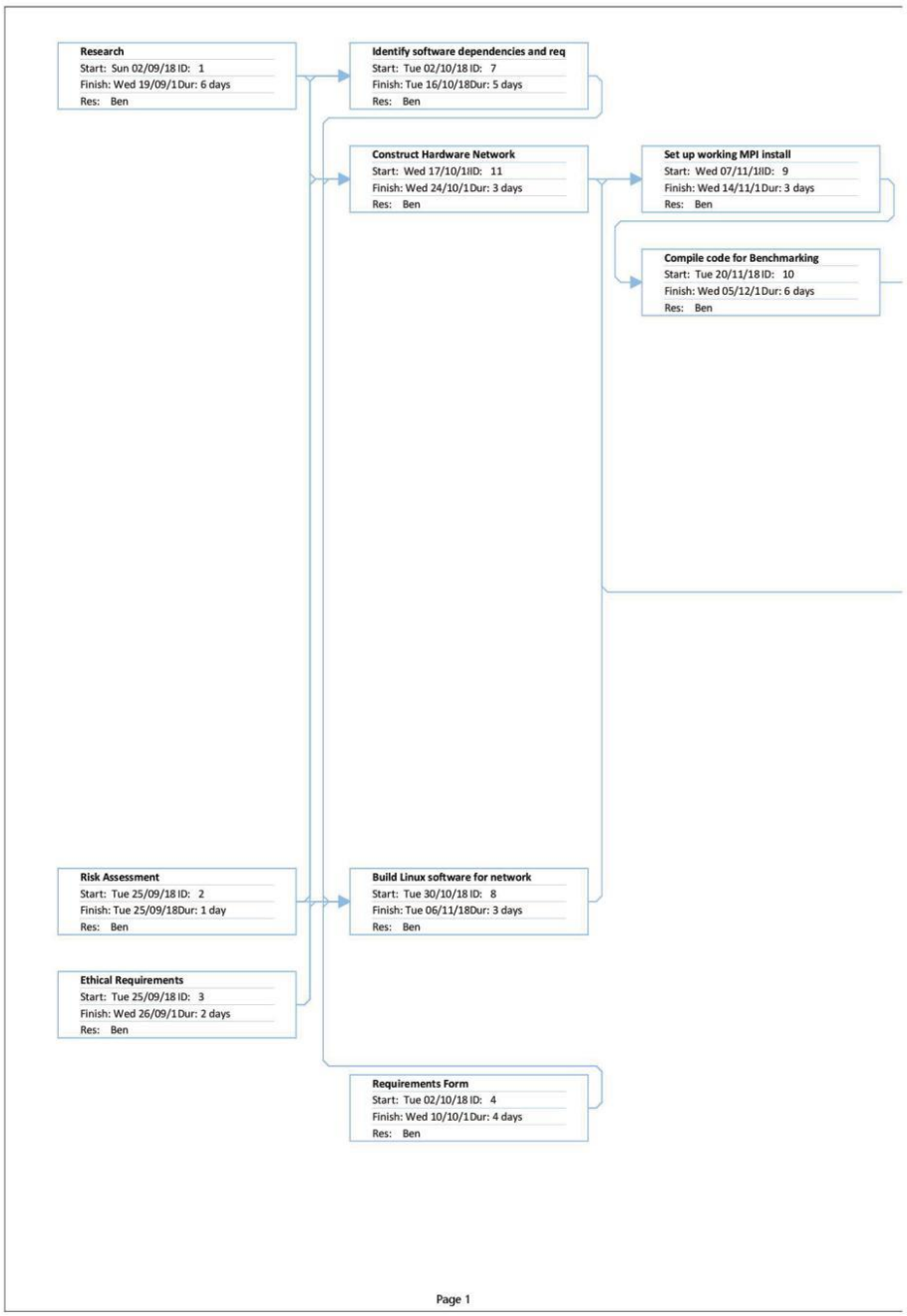
Appendix

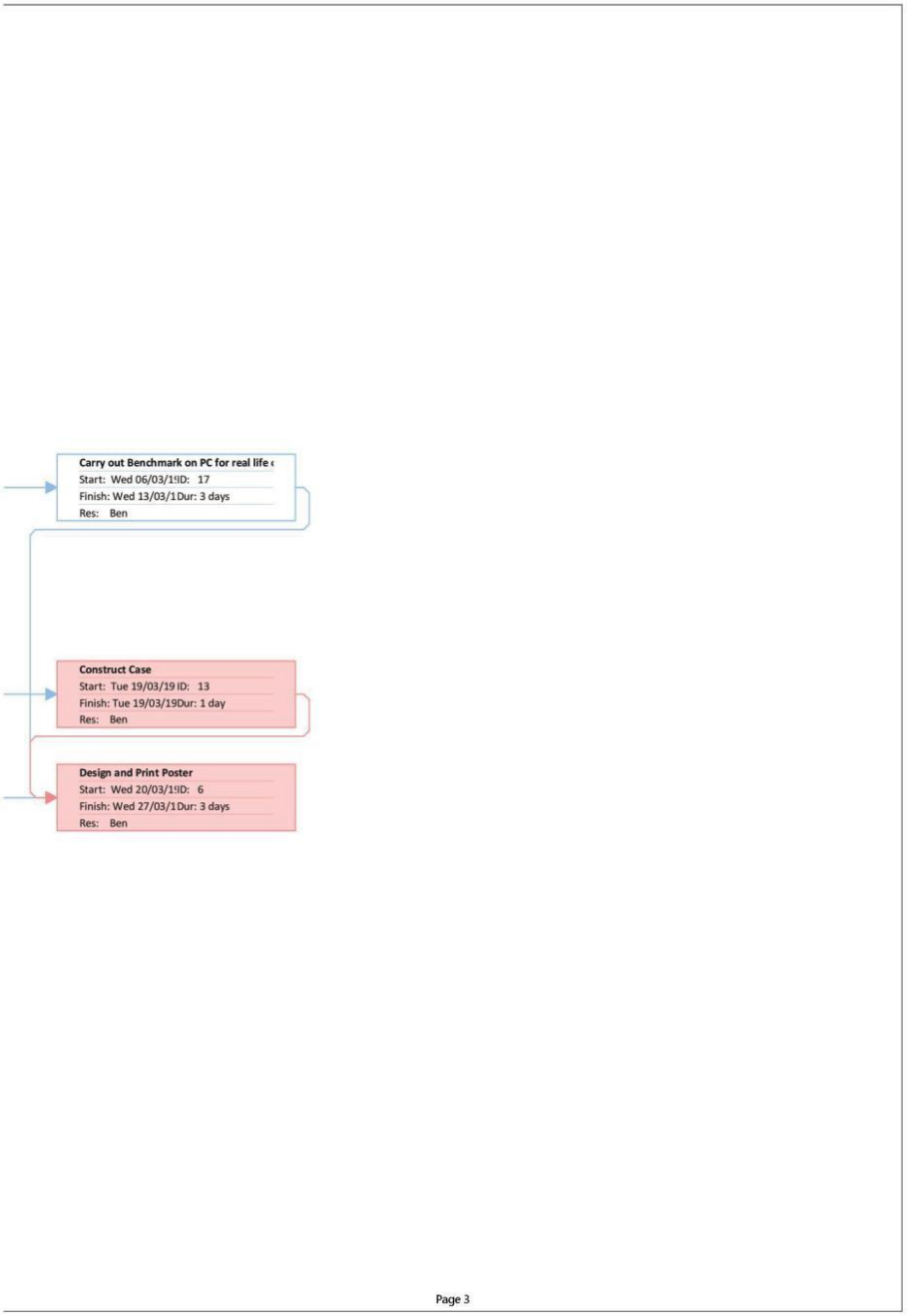
Appendix 1

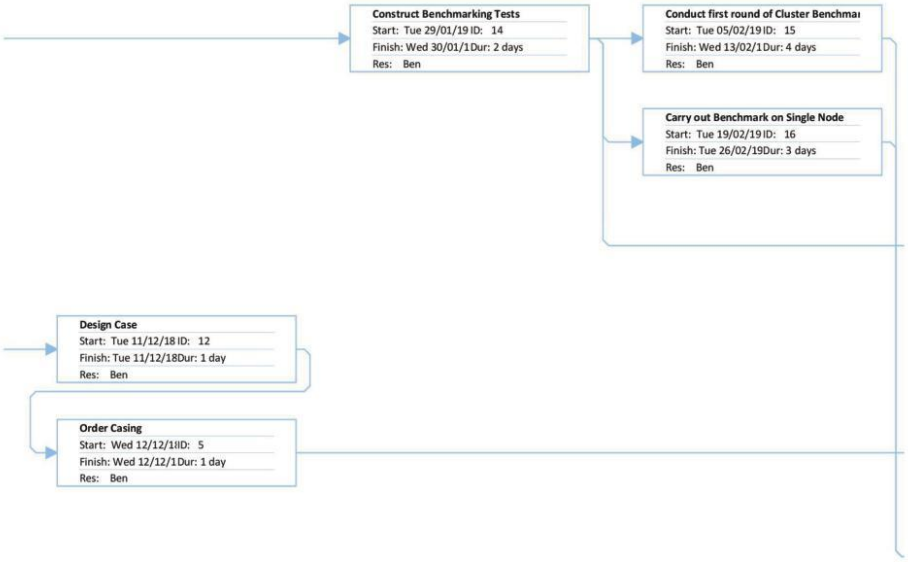


ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		Research	6 days	Sun 02/09/18	Wed 19/09/18		Ben
2		Risk Assessment	1 day	Tue 25/09/18	Tue 25/09/18		Ben
3		Ethical Requirements	2 days	Tue 25/09/18	Wed 26/09/18		Ben
4		Requirements Form	4 days	Tue 02/10/18	Wed 10/10/18		Ben
7		Identify software dependencies and requirements	5 days	Tue 02/10/18	Tue 16/10/18	1,2,3	Ben
11		Construct Hardware Network	3 days	Wed 17/10/18	Wed 24/10/18	1,2,3,4	Ben
8		Build Linux software for network	3 days	Tue 30/10/18	Tue 06/11/18	7,2,3,4	Ben
9		Set up working MPI install	3 days	Wed 07/11/18	Wed 14/11/18	11,8	Ben
10		Compile code for Benchmarking	6 days	Tue 20/11/18	Wed 05/12/18	9	Ben
12		Design Case	1 day	Tue 11/12/18	Tue 11/12/18	11	Ben
5		Order Casing	1 day	Wed 12/12/18	Wed 12/12/18		Ben
14		Construct Benchmarking Tests	2 days	Tue 29/01/19	Wed 30/01/19	10	Ben
15		Conduct first round of Cluster Benchmarking	4 days	Tue 05/02/19	Wed 13/02/19	14	Ben
16		Carry out Benchmark on Single Node	3 days	Tue 19/02/19	Tue 26/02/19	14	Ben
17		Carry out Benchmark on PC for real life comparison	3 days	Wed 06/03/19	Wed 13/03/19	14	Ben
13		Construct Case	1 day	Tue 19/03/19	Tue 19/03/19	5	Ben
6		Design and Print Poster	3 days	Wed 20/03/19	Wed 27/03/19	13,15,16,17	Ben

Appendix 2


Research
Start: Sun 02/09/18 ID: 1
Finish: Wed 19/09/1 Dur: 6 days
Res: Ben
Identify software dependencies and req
Start: Tue 02/10/18 ID: 7
Finish: Tue 16/10/18 Dur: 5 days
Res: Ben
Construct Hardware Network
Start: Wed 17/10/1 ID: 11
Finish: Wed 24/10/1 Dur: 3 days
Res: Ben
Set up working MPI install
Start: Wed 07/11/1 ID: 9
Finish: Wed 14/11/1 Dur: 3 days
Res: Ben
Compile code for Benchmarking
Start: Tue 20/11/18 ID: 10
Finish: Wed 05/12/1 Dur: 6 days
Res: Ben
Risk Assessment
Start: Tue 25/09/18 ID: 2
Finish: Tue 25/09/18 Dur: 1 day
Res: Ben
Build Linux software for network
Start: Tue 30/10/18 ID: 8
Finish: Tue 06/11/18 Dur: 3 days
Res: Ben
Ethical Requirements
Start: Tue 25/09/18 ID: 3
Finish: Wed 26/09/1 Dur: 2 days
Res: Ben
Requirements Form
Start: Tue 02/10/18 ID: 4
Finish: Wed 10/10/1 Dur: 4 days
Res: Ben
Page 1


Carry out Benchmark on PC for real life
Start: Wed 06/03/1ID: 17
Finish: Wed 13/03/1Dur: 3 days
Res: Ben
Construct Case
Start: Tue 19/03/19 ID: 13
Finish: Tue 19/03/19Dur: 1 day
Res: Ben
Design and Print Poster
Start: Wed 20/03/1ID: 6
Finish: Wed 27/03/1Dur: 3 days
Res: Ben
Page 3


Construct Benchmarking Tests
Start: Tue 29/01/19 ID: 14
Finish: Wed 30/01/1 Dur: 2 days
Res: Ben
Conduct first round of Cluster Benchmar
Start: Tue 05/02/19 ID: 15
Finish: Wed 13/02/1 Dur: 4 days
Res: Ben
Carry out Benchmark on Single Node
Start: Tue 19/02/19 ID: 16
Finish: Tue 26/02/19 Dur: 3 days
Res: Ben
Design Case
Start: Tue 11/12/18 ID: 12
Finish: Tue 11/12/18 Dur: 1 day
Res: Ben
Order Casing
Start: Wed 12/12/1 ID: 5
Finish: Wed 12/12/1 Dur: 1 day
Res: Ben

Specification Report Ben Hague Raspberry Pi Computing Cluster

Appendix 3

UNIVERSITY OF LIVERPOOL

DEPARTMENT OF ELECTRICAL ENGINEERING AND ELECTRONICS

Ethical Approval Questionnaire 2017-2018

Final Year BEng (ELEC340) and Year 3 MEng (ELEC440)

Student Name: Benjamin Hague _____ Module: ~~ELEC340 /~~ ELEC440 (delete one)

Supervisor: Dr James Walsh _____ Student ID No: _____ 201146260

Project Title: Raspberry Pie computing Cluster _____

Formal ethical approval must be obtained for all research projects '*involving research on human subjects or human tissues or databases of personal information to be carried out by University staff or students on University premises, or at any location, where there is no other acceptable provision for ethical consideration*'. Final year projects (ELEC340) and year 3 MEng projects (ELEC440) involving human participation must be undertaken in a way that safeguards the dignity, rights, health, safety, and privacy of those involved.

It should be noted that this policy covers **all** research methodologies including such activities as informal interviews, accessing personal files in an archive, or on-line data gathering. The requirement to obtain ethical review applies with equal force to projects undertaken by undergraduate students. For these projects, it is the responsibility of the supervisor to ensure that the ethical issues of the research are fully assessed and that formal ethical approval is obtained before the project commences.

Does your project involve any human participants (including situations where you are a participant as well as the investigator)?	~~YES~~	NO
Does your project involve any human tissues (including your own)?	~~YES~~	NO
Does your project involve any databases of personal information (including your own personal information)?	~~YES~~	NO
Does your project involve experiments using animals?	~~YES~~	NO

Delete either YES or NO on each line above

If any of the answers given above are YES then you, along with your project supervisor, must investigate the requirement to apply for ethical approval. Details of how to apply for ethical approval can be found at www.liv.ac.uk/intranet/research-support-office/research-ethics/ (for human participation) and at https://www.liv.ac.uk/research-integrity/biomedical-research/ (for use of animals).

Student Signature: _____ Date: 28-9-2018

Supervisor's Signature: _____ Date: _28-9_____

Appendix 4

UNIVERSITY OF LIVERPOOL

RISK ASSESSMENT FORM

School/Department: EEE	Building: EEE
Task: Building Raspberry Pi Computer Cluster	
Persons who can be adversely affected by the activity: Only individual doing the work	

Section 1: Is there potential for one or more of the issues below to lead to injury/ill health (tick relevant boxes)

People and animals/Behaviour hazards

Allergies		Too few people		Horseplay		Repetitive action	☑	Farm animals	
Disabilities		Too many people		Violence/aggression		Standing for long periods		Small animals	
Poor training		Non-employees		Stress		Fatigue		Physical size, strength, shape	
Poor supervision		Illness/disease		Pregnancy/expectant mothers		Awkward body postures		Potential for human error	
Lack of experience		Lack of insurance		Static body postures		Lack of or poor communication		Taking short cuts	
Children		Rushing		Lack of mental ability		Language difficulties		Vulnerable adult group	

What controls measures are in place or need to be introduced to address the issues identified?

Identified hazards	CURRENT CONTROLS	RISK SCORE	ADDITIONAL CONTROLS REQUIRED (To include responsibilities and timescales)	RESIDUAL RISK SCORE
Repetitive action caused strains	Take Regular Breaks Ensure Good Posture	2	NA	2

Eye Strain	Take Regular breaks	1	NA	1

Section 2: Common Workplace hazards. Is there potential for one or more of the issues below to lead to injury/ill health (tick relevant boxes)

Fall from height	Poor lighting	Portable tools	Fire hazards	Chemicals	Asbestos		
Falling objects	Poor heating or ventilation	Powered/moving machinery	Vehicles	Biological agents	Explosives		
Slips, trips, falls	Poor space design	Lifting equipment	Radiation sources	Waste materials	Genetic modification work		
Manual handling	Poor welfare facilities	Pressure vessels	Lasers	Nanotechnology	Magnetic devices		
Display screen equipment	Electrical equipment	☑ Noise or vibration	Confined spaces	Gases	Extraction systems		
Temperature extremes	Sharps	Drones	Cryogenics	Legionella	Robotics		
Home working	Poor signage	Overseas work	Overnight experiments	Unusual events	Community visits		
Late/lone working	Lack of/poor selection of PPE	Night work	Long hours	Weather extremes	Diving		

Section 3: Additional hazards: are there further hazards NOT IDENTIFIED ABOVE that need to be considered and what controls are in place or needed? (list below)

Additional hazards	CURRENT CONTROLS	RISK SCORE	ADDITIONAL CONTROLS REQUIRED (To include responsibilities and timescales)	RESIDUAL RISK SCORE

What controls measures are in place or need to be introduced to address the issues identified?

Identified hazards	CURRENT CONTROLS	RISK SCORE	ADDITIONAL CONTROLS REQUIRED (To include responsibilities and timescales)	RESIDUAL RISK SCORE	
Electric Shock	Only low voltages are to be used Avoid coming into direct contact with powered circuits	1	NA	1	

Section 4: Emergency arrangements (List any additional controls that are required to deal with the potential emergency situation)

Emergency situation	Additional control required

Risk assessor (signature)..... .....Date..... 28/09/18..... Authorised by (signature)..... .....Date.....

Appendix 5

Initial Start-up form 2018-2019

Student Name	Ben Hague	Student ID Number	201146260	Contact Email (In case of problems with orders etc)	sgbhague@liv.ac.uk
Project Title	Raspberry Pie computing cluster				
Supervisor	Dr J Walsh				
Order ID No.	(Assigned by 4th floor Technicians on receipt of form)				

This form is to be completed by the student in co-operation with their supervisor. On completion the form **must** be signed by the supervisor in the space provided on the rear on this form, and then returned to the Technicians. Forms must be submitted even for those students with nil requirements.

Type of project
Is the project completely software based? ~~YES~~/NO Delete as appropriate
Is the project completely Hardware based? ~~YES~~/NO Delete as appropriate
Is the project mixed software/Hardware based? YES/~~NO~~ Delete as appropriate

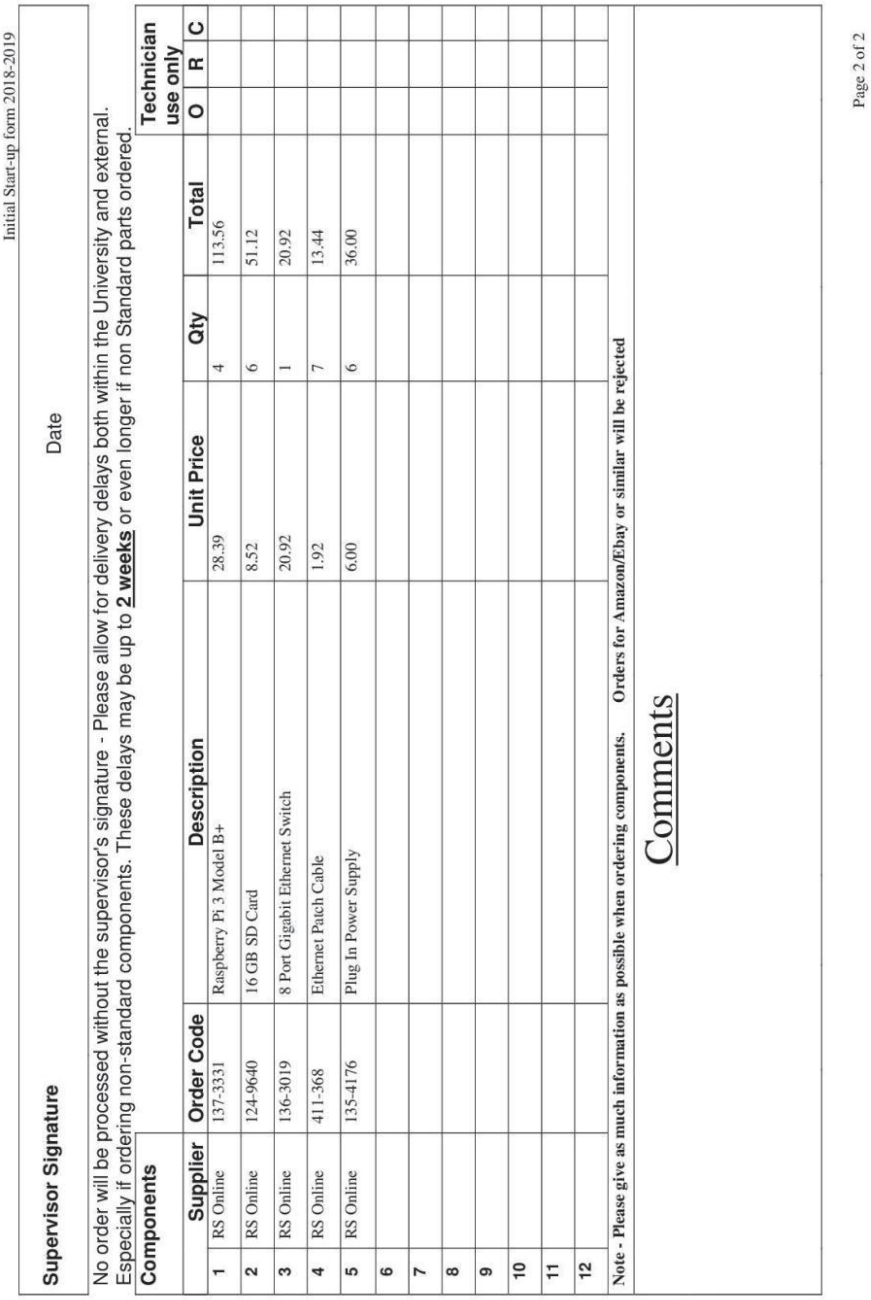
Small parts/PCB manufacture etc.
You must NOT approach any of the departmental workshops directly. All such requests should be made through the 4th floor Technicians from whom further information can be obtained. Note – Considerable delays may be experienced regarding the manufacture of small parts &/or PCB's for student projects dependent upon the workshop work loading at the time of request submission.

Location
Please state any special requirement(s) that will affect the location of the project within the laboratory area. Software projects will NOT be allocated a specific location until shortly before the bench inspections. Details of the arrangements for the bench inspections will be issued during session 2017/18 semester 2

Additional equipment, other than standard laboratory equipment to be supplied by supervisor

Equipment / Components	Available from

Page 1 of 2



Initial Start-up form 2018-2019

Supervisor Signature Date

No order will be processed without the supervisor's signature - Please allow for delivery delays both within the University and external.
Especially if ordering non-standard components. These delays may be up to **2 weeks** or even longer if non Standard parts ordered.

Components

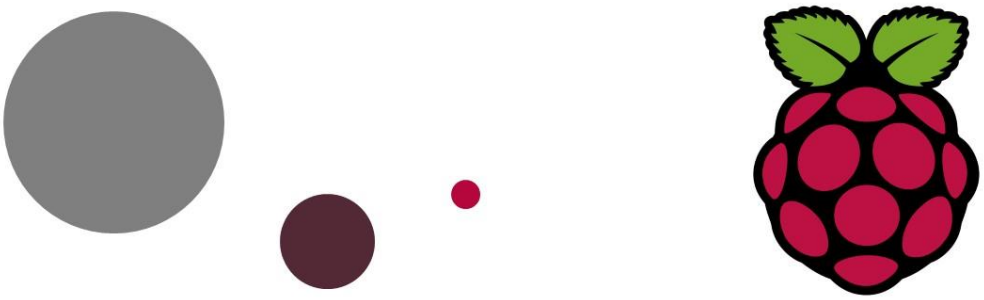
	Supplier	Order Code	Description	Unit Price	Qty	Total	Technician use only O	R	C
1	RS Online	137-3331	Raspberry Pi 3 Model B+	28.39	4	113.56			
2	RS Online	124-9640	16 GB SD Card	8.52	6	51.12			
3	RS Online	136-3019	8 Port Gigabit Ethernet Switch	20.92	1	20.92			
4	RS Online	411-368	Ethernet Patch Cable	1.92	7	13.44			
5	RS Online	135-4176	Plug In Power Supply	6.00	6	36.00			
6									
7									
8									
9									
10									
11									
12									

Note - Please give as much information as possible when ordering components. Orders for Amazon/Ebay or similar will be rejected

Comments

Page 2 of 2

Appendix D Oral Presentation Slides



Raspberry Pi Computing Cluster

Ben Hague
(201146260)

Content

- Project Aim
- What is a Cluster?
- Why Cluster?
- Why Raspberry Pi?
- Parallelisation of Plasma Codes
- Obstacles
- Progress
- Roadmap
- Questions

Project Aim

"Produce a Cluster of Raspberry Pi Computers capable of executing Plasma simulation codes from the Plasma Theory and Simulation group at Michigan State University. "

What is a Cluster?

A computer cluster is a single "unit" consisting of multiple computers that are linked through a network.

The networked computers can then be used as a single, more powerful machine.

Clusters excel at Performing operations on large data sets or batch style operations.

Why Cluster?

Performance

Redundancy

Why Raspberry Pi?

Raspberry Pi	4 cores running at 1.4Ghz	1GB of Ram	With each node drawing 5.1W Max	16GB of storage	£36.61
Raspberry Pi Cluster	24 cores running at 1.4Ghz each	With 6GB of ram	Drawing just over 30.6 W	And ~100GB of storage	£289.90
PC Similarly Priced (Dell OptiPlex 3050)	2 Cores Running at 3.4Ghz	4GB of Ram	Drawing 65W	500GB of storage	£349.00

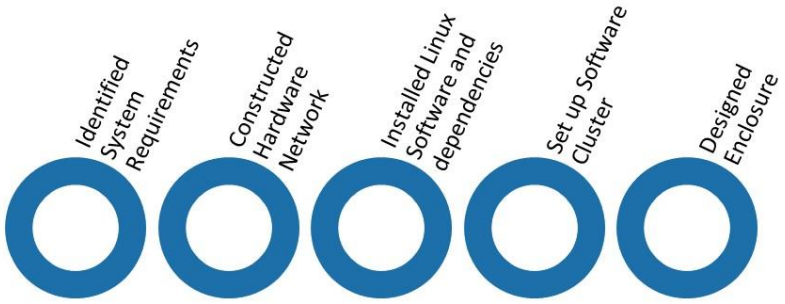
Parallelisation of Plasma Codes

Obstacles

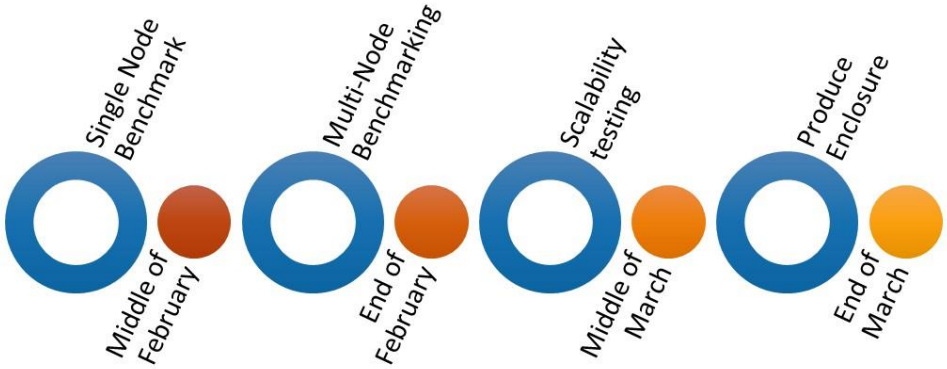
Dependencies

Networking

Progress



Roadmap



Questions