

FRAMEWORK FOR EVALUATION OF PARALLEL ALGORITHMS ON CLUSTERS

BY MEHUL WARADE

SUPERVISED BY
PROF. JEAN-GUY SCHNEIDER
AND
DR. KEVIN LEE



Submitted as partial fulfilment of the requirements for the degree of

Bachelor of Software Engineering (Honours)

School of Information Technology

Deakin University

October 2020



DEAKIN UNIVERSITY CANDIDATE DECLARATION

I certify the following about the thesis entitled

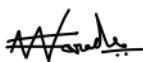
Framework for Evaluation of Parallel Algorithms on Clusters

submitted for the degree of **Bachelor of Software Engineering (Honours), School of Information Technology, Deakin University**

- (a) I am the creator of all or part of the whole work(s) (including content and layout) and that where reference is made to the work of others, due acknowledgment is given.
- (b) The work(s) are not in any way a violation or infringement of any copyright, trademark, patent, or other rights whatsoever of any person.
- (c) That if the work(s) have been commissioned, sponsored or supported by any organisation, I have fulfilled all of the obligations required by such contract or agreement.
- (d) That any material in the thesis which has been accepted for a degree or diploma by any university or institution is identified in the text.
- (e) All research integrity requirements have been complied with.

I certify that I am the student named below and that the information provided in the form is correct

Full Name: Mehul Vikas Warade

Signed: 

Date: October 30, 2020

Abstract

For several decades, clusters have been used in high performance computing to achieve exceptional performance. Single Board Computer (SBC) clusters were developed as an alternate to the traditional clusters as they require no active cooling, less estate and consume a fraction of the energy consumed by traditional cluster while providing competing computational performance. Traditional evaluation protocols for cluster computing mainly focus on their run-time, accuracy and cross compatibility. The downside to most current cluster evaluation frameworks is that they do not exploit one of the strong points of single board computers — their energy consumption. Motivated by this, a novel idea is proposed to develop a framework for evaluating parallel algorithms on different cluster architectures. A detailed illustration of the proposed framework, named FEPAC, is outlined. FEPAC focuses on the energy consumption of the cluster and provides the analysis of a computation in the form of performance per Watt (GFLPOS/W) and value for money (GFLOPS/\$) for different cluster architectures. This is particularly useful in developing more energy efficient algorithms and to provide an analysis of the effects of different cluster architectures on the performance and cost of computation. A qualitative and quantitative evaluation of the framework is performed to showcase its working, confirm the results and align them to the requirements of the thesis. To show how the approach can be applied, parallel algorithms from different domains are used to demonstrate the ability of FEPAC. A detailed case study of implementation and evaluation of FEPAC for several parallel algorithms being executed on a 7-node RPi3B+ cluster is used to evaluate the approach. FEPAC enables the energy and computation characteristics of diverse algorithm to be measured and aids researchers in choosing a suitable cluster configuration for their computation needs.

Acknowledgments

Firstly, I would like to thank my supervisors - **Jean-Guy Schneider** and **Kevin Lee**, for their guidance and support throughout the year, as well as years prior.

I would also like to thank the Deakin University for providing the necessary hardware, which made the evaluation of this work possible.

Finally, I would like to thank my family for their support, and friends for tolerating me throughout this journey.

Contents

Abstract	iii
Acknowledgments	iv
Contents	v
List of Figures	viii
List of Tables	x
Definitions of terms	xi
List of Abbreviations	xii
1 Introduction	1
1.1 Research Questions	2
1.2 Contributions	2
1.3 Thesis Structure	3
2 Background	4
2.1 Methodology for literature review	6
2.2 Hardware	8
2.2.1 Processors	8
2.2.2 Memory	10
2.2.3 Cluster computing	11
2.2.4 Single Board computers	12
2.2.5 Summary	13
2.3 Software	13
2.3.1 Operating System	13
2.3.2 Inter-node communication interface (MPI)	13
2.3.3 Storage Framework	14
2.3.4 Job Schedulers	14
2.3.5 Computing Libraries	15

2.3.6	Parallel Algorithms	15
2.3.7	Summary	18
2.4	Related Work	18
2.4.1	Experimental Cluster Computing	18
2.4.2	Energy Aware Cluster Computing	19
2.4.3	Related work in benchmarking and evaluating clusters	20
2.5	Open Issues/ Gaps	21
2.6	Summary	22
3	Methodology	24
3.1	Research questions	25
3.2	Paradigm	25
3.3	Data Collection	26
3.4	Validity and Reliability of Data	26
3.5	Data Analysis Strategy	26
3.6	Requirements	27
3.7	Summary	29
4	Design	31
4.1	Cluster	32
4.2	Framework	33
4.2.1	User Responsibilities	34
4.2.2	Framework Functionalities	34
4.3	Summary	38
5	Implementation	39
5.1	Implementation of Computing Cluster	41
5.1.1	Node Computer	41
5.1.2	Retrieving Energy consumption values	42
5.1.3	Cluster Setup	44
5.1.4	Final Setup	46
5.2	Implementation of Framework	48
5.2.1	Framework language	48
5.2.2	Configuration file	49
5.2.3	Development of Individual Functionality	50
5.2.4	Final Setup	53
5.3	Summary	55
6	Evaluation	56
6.1	Qualitative Evaluation	56

6.2	Quantitative Evaluation	60
6.2.1	Multiple run of single experiment on different cluster configuration	60
6.2.2	Multiple run of single experiment with different data-set	62
6.2.3	Expansion on the functionality of the designed system	64
6.2.4	Algorithm Evaluation: Matrix Multiplication	65
6.2.5	Algorithm Evaluation: Kmeans	67
6.2.6	Algorithm Evaluation: OpenCV filtering	68
6.3	Summary	69
7	Discussion	70
7.1	Helping researchers in deciding best optimal node for computing	70
7.2	Finding relationship between data set and energy	70
7.3	Helping researchers in predicting the computation time for an algorithm	71
7.4	Helping researchers in predicting the energy consumption of the cluster	73
7.5	Limitations	74
8	Conclusion	76
8.1	Overview of Thesis	76
8.2	Contributions	77
8.3	Future Work	78
8.3.1	Expansion to real-world workloads	78
8.3.2	Cluster Improvements	79
8.3.3	Framework Improvement	80
8.4	Concluding Remarks	81
	Bibliography	82
A	Configuration File	86
B	Matrix Multiplication: Raw Data and Calculations	88
C	OpenCV Algorithm: Raw Data	91
D	Kmeans Algorithm: Raw Data	93
E	Addition of new functionality through code (Using Plotly)	94
F	Plotly Dashboard with the data from the framework	96

List of Figures

2.1	Flynn’s Taxonomy for parallel computing architecture (Flynn, 1966)	5
2.2	Classification of microprocessor.	9
4.1	Proposed Design for the Cluster hardware	32
4.2	Proposed Design of the Framework	33
4.3	Proposed set up of nodes in the cluster	34
4.4	Splitting of data for cluster computing	35
4.5	Running of an algorithm in a cluster	36
4.6	Collecting energy consumption values	36
4.7	Collecting and storing algorithm output data	37
4.8	Exporting collected data to a file	37
5.1	Implementation of the proposed cluster hardware	40
5.2	Implementation of the proposed framework	41
5.3	Raspberry Pi3B+ (Source: www.raspberrypi.org)	42
5.4	Netgear GS110TP Fully Managed Switch (Source: www.netgear.org)	43
5.5	PoE Splitter (Source: https://core-electronics.com.au)	44
5.6	Web Interface for management of the switch	44
5.7	Network Booting Steps	46
5.8	Final Set Up of the Implemented Cluster	47
5.9	Final Setup - Netboot and Master node	47
5.10	Final Setup - Slave nodes	47
5.11	Collection of energy consumption values through code	50
5.12	Energy Values stored in local MySQL Database	51
5.13	Using MPI to execute algorithm across multiple nodes	52
5.14	Exported data in JSON format	53
5.15	Final Implemented Framework and its Menu	54
6.1	Configuration File extract for target cluster configurations	58
6.2	Configuration file extract for instructions on execution of Algorithms	59

6.3	Evaluation of Algorithm: Matrix Multiplication with different cluster configurations . . .	61
6.4	Evaluation of Algorithm: Matrix Multiplication (Data-set of 600)	62
6.5	Evaluation of Algorithm: Matrix Multiplication (Data-set of 720)	62
6.6	Evaluation of Algorithm: Matrix Multiplication (Data-set of 840)	63
6.7	Evaluation of Algorithm: Matrix Multiplication (Data-set of 960)	63
6.8	Exported data in a JSON format	65
6.9	Algorithm Evaluation: Matrix Multiplication and Energy consumption	66
6.10	Algorithm Evaluation: Matrix Multiplication and Computation Time	66
6.11	Algorithm Evaluation: Kmeans and Energy Consumption	67
6.12	Algorithm Evaluation: Kmeans and Computation Time	68
6.13	Algorithm Evaluation: OpenCV and Energy Consumption	69
6.14	Algorithm Evaluation: OpenCV and Computation Time	69
7.1	Effects of different data-sets on Energy Consumption of an Algorithm	71
7.2	Ability to predict Algorithm Computation Time based on previous data	72
7.3	Ability to predict Energy Consumption based on previous data	73

List of Tables

2.1	Comparison of widely used single board computers. Derived from (Cloutier et al., 2016)	12
2.2	HPL benchmark results on different computers. Derived from (Cloutier et al., 2016)	21
5.1	RPi Operating System Distributions	45
6.1	List of Algorithms Evaluated and their Parameters	60
B.1	Raw Data Output of Matrix Multiplication Algorithm	90
C.1	Raw Data Output of OpenCV Filtering Algorithm	92
D.1	Raw Data Output of Kmeans Algorithm	93

Definitions of terms

Access	The reading or writing of data
Algorithm	A set of rules for solving a problem in a given number of steps.
Bandwidth	The maximum data transfer rate of a network or Internet connection.
Code	A language for expressing operations to be performed by a computer.
Computing	Any goal-related activity comprising of computers.
Configuration	The particular hardware elements and their interaction in a computer system for a particular period of operation.
Database	Accessible collection of information
Dataset	A file or group of files associated with one part of a study.
Energy consumption	The amount of power used in a particular time
Framework	Computer programs that perform various tasks
Instance	A particular occurrence of an object defined by a class
Master-slave	A relationship in which slave software obtains services from a master on behalf of a person
Memory	The fastest storage device of a computer.
Microprocessor/ CPU	The main computer chip providing the speed and capabilities of the computer.
Node	A member of a network
Operating System	Software that controls the basic, low-level hardware operations, and file management.
Parallel computing	Computations performed parallelly.
Parameter	A value supplied to an algorithm.
Port	The portion of a computer through which a peripheral device may communicate
Portable	Able to be used by a variety of software on a variety of hardware platforms.
Program	A set of actions or instructions that a machine is capable of interpreting and executing.
<i>etc.</i>	<i>etc.</i>

List of Abbreviations

AI	Artificial Intelligence
ALU	Arithmetic Logic Unit
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FEPAC	Framework for Evaluation of Parallel Algorithms on Clusters
FLOPS	FLoating point Operations Per Second.
FORTRAN	FORMula TRANslation
GHz	GigaHertz
GUI	Graphical User Interface
I/O	Input/Output
IC	Integrated Circuits
IP	Internet Protocol
Js	JavaScript
MB	MegaByte
MIMD	Multiple Instruction Multiple Data
MISD	Multiple Instruction Single Data
OpenCV	Open Computer Vision
RAM	Random Access Memory
ROM	Read Only Memory
SBC	Single Board Computer
SIMD	Single Instruction Multiple Data
SISD	Single Instruction Single Data
<i>etc.</i>	<i>etc.</i>

CHAPTER 1

INTRODUCTION

Energy consumption is one of the top challenges faced while achieving the next generation of super-computing (Jin et al., 2017). Many power-saving techniques have been developed, and most hardware components are approaching their physical limits. Parallel computing provides us with the technology and means to achieve the ever-increasing demands of computing. Traditionally, clusters were developed with performance in mind and hence they were infrastructures requiring expensive cooling mechanism installed and high energy consumption. This has promulgated the creation of huge data centres that have heightened the energy demand (Kaur and Chana, 2015).

Single board computers (SBCs) are complete computers developed on a miniaturised single circuit board. They were introduced for educational purposes and to teach the workings of a computer, but development in processors have led to introduction of more advanced and powerful Single Board Computers. Architecturally, SBCs are multiprocessor-based hardware units that provides all the features of a traditional computers. They have the ability to perform reasonably well when compared to contemporary devices in terms of cost and power consumption (Cloutier et al., 2016). Due to their multi-core nature, they can be optimised to achieve computations faster by using their inherent parallel capabilities.

The performance or execution time of a parallel computation depends on many factors such as architecture, the compiler, Operating System used, the environment used for parallel programming and the model used by the environment (Rauber and Runger, 2013). While developing or choosing for a platform to execute an algorithm all these factors need to be considered. However, there are many complex interactions between these factors, and it is therefore difficult to consider them all (Rauber and Runger, 2013). This thesis investigates the current state of the art in parallel computing and investigates the evaluation methods for different computations. This thesis also investigates how to take advantage of SBC computing platforms and how they have been used in clusters to achieve improved processing/computing times.

The focus of this thesis is on how to aid researchers in choosing energy efficient software or algorithms for their computation needs. Specifically, the research presented in this thesis aims to investigate the effects of computing (using different parallel algorithms) on different cluster configuration as seen

from an energy point of view. The proposed solution aims to be platform independent with minimal installation required and number of configurable parameters according to the users needs. Using our system, parallel computation done by a researcher can be evaluated and analysed to provide additional analysis from performance and energy aspects.

The remainder of this chapter is structured as follows: in Section 1.1, an overview of this thesis with the base research questions the work aims to answer is given. In Section 1.2 main contributions of the work to the field of research are presented. Finally, in Section 1.3, the structure of the thesis is summarised.

1.1 Research Questions

The research presented in this thesis aims to investigate the following research questions. These questions are derived from the open issues and gaps identified in Chapter 2. The work presented aims to answer these research question through formal research methods.

- How do different parallel algorithms executing on different cluster architectures affect the overall performance?
- What is the best architecture design and specs of clusters for a researcher that want to test their algorithm effectively?
- How does different cluster configuration affect the energy consumption during computation?
- How do researchers choose a particular cluster for their computation needs?
- How to achieve a platform independent framework with consistent results?

1.2 Contributions

The main contributions of this work are as follows:

- **Literature review in the field of parallel computing**

The state-of-the-art in the field of parallel computing and the research gaps found during the survey of related work in the field are documented in Chapter 2. The literature review is important as it provides the required background knowledge for understanding the work done in this thesis and provides information on the current research being conducted in the field.

- **Design of a Framework for evaluating parallel algorithms**

The design of a system which evaluates different parallel algorithms on different cluster configuration is the major contributing factor as different implementations can be done for an abstract design which can meet the research questions. Evaluation of one such implementation, called FEPAC, is provided and show how the research questions are addressed.

- **Evaluation of the Framework using qualitative and quantitative means**

Extensive evaluation on FEPAC by-using both qualitative and quantitative analysis is performed. The evaluation helps in understanding the aims of the thesis and how they co-relate to the framework's design and implementation.

1.3 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2 introduces the field of parallel and cluster computing. A comprehensive analysis of the state-of-the-art in the field of parallel cluster computing is presented with focus on energy consumption. The chapter concludes with a brief discussion on the research gaps identified during the review of the relevant literature.

Chapter 3 presents the general methodology adopted while developing the framework. It includes a discussion of the approach undertaken and principals followed while designing and evaluation of the framework. The Chapter also discusses the requirements for the proposed design of the system. The requirements were devised after analysis of the research questions and the final goal of the thesis.

Chapter 4 presents a detailed design of the proposed framework. The chapter includes high level abstract working of the project and the how it can achieve the requirements set out in Chapter 3. This consists of a comprehensive design based on requirements set out in Chapter 3 and goals of this thesis. This chapter, along with Chapter 3, enables a researcher to understand the underlying workings of the framework so that they can modify it to meet their particular needs.

Chapter 5 presents the central contribution of the thesis – FEPAC, a framework developed to evaluate parallel algorithms on different cluster configurations. The framework is implemented following the abstract design in Chapter 4. The implementation is divided into two parts - the hardware and software implementation. The chapter explains in detail the specific steps undertaken to achieve the final proposed system.

Chapter 6 presents the qualitative and quantitative evaluation of the framework. Different functionalities of the framework are evaluated and quality checked based on their relevance with the thesis's goals and showcases how the framework achieves to answer the research questions and close the research gaps. A quantitative evaluation is performed to show the workings of the framework and to guide the user on different conclusions that can be devised from the results. Chapter 3, 4 and 5 together can allow a researcher to replicate the whole work and compare its workings with the original results.

Chapter 7 discusses the findings in the thesis and connects them to the research question. The results from Chapter 6 are used to showcase other scenarios in which the designed system can help researchers understand and analyse their computation. The quantitative data in Chapter 6 is used to discuss and showcase a number of scenarios where the framework can help the user to achieve the problem statement of our thesis. Finally, Chapter 8 highlights the major results of the work presented in this thesis, connecting them to the research questions, and discussing potential future expansions of the work.

CHAPTER 2

BACKGROUND

The modern world is full of data driven technologies which demand more and more analysis and management to produce effective results. As predicted by Moore's law (Moore et al., 1965), advances in processors and technology has led to increased performance of modern hardware. Algorithms are used to reduce the manual work and effectively utilise the ever-growing computer technologies to their full extent. For the purposes of this study, an algorithm is considered as a well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values as output (Cormen et al., 2009). Newer technologies and advancements lead to exponential growth in the amount of data that needed processing (Berman and Paul, 1996). Hence, computers with powerful computation capabilities are used to process the amount of data generated.

Parallel computing is defined as simultaneous use of multiple computer resources to solve a computational problem (JéJé, 1992). Parallel computing is achieved through implementation of parallel algorithms on a computing platform. Following limitations of serial computers paved a way for research in parallel computing (Barney et al., 2010):

1. Transmission speed: The speed of a computer directly relates to the speed of data in the hardware. Limits of hardware speed (Copper wire – 9cm/nanosecond) have already been reached. Hence, there was a need to find other ways to improve performance.
2. Limits to miniaturisation: There is a limit up-to which the number of transistors on the chip can be increased.
3. Economic limitations: Multi-cores processors were used when single core processors were not enough to fulfil the computing needs. It was expensive to make a single processor faster than that whereas using many moderately fast processors to achieve the same (or better) performance is less expensive.

Michael Flynn, in 1966, classified parallel computer architectures based on the instructions, data and processing sequences (or streams).

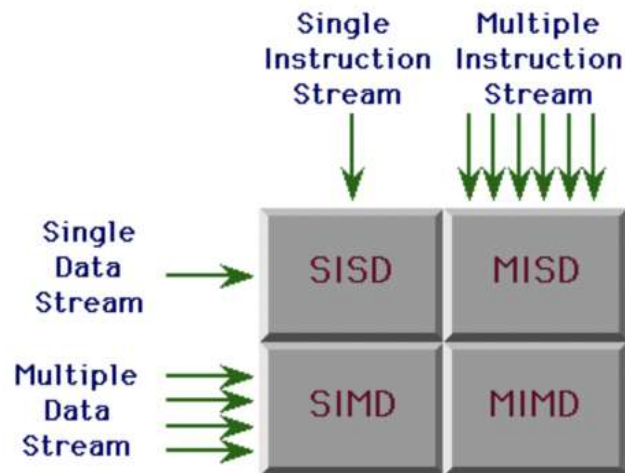


Figure 2.1: Flynn's Taxonomy for parallel computing architecture (Flynn, 1966)

1. Single Instruction and Single Data Stream (SISD):

SISD is the conventional sequential computer in which the CPU fetches a single instruction (I) from the memory and processes a single stream of data (D). There is no parallelism either in instructions or data streams. Examples include Minicomputers, Uni-processors and old mainframes.

2. Single Instruction and Multiple Data Stream (SIMD):

In SIMD, a single instruction is set to be executed for multiple data. SIMD based computers are the most natural form of parallel computers. Examples include graphics processor units (GPUs), Intel and AMD multi-core processors.

3. Multiple Instruction and Single Data Stream (MISD):

In this architecture, multiple instructions operate on the same data independently during a clock cycle. This architecture is rarely used. Examples include pipe-lined computers and real time systems such as space shuttle control system.

4. Multiple Instruction and Multiple Data Stream (MIMD):

MIMD architecture is used in parallel systems. Multiple instruction works on multiple data stream independently of each other. They execute asynchronously and each processor works on its own instruction. Examples include supercomputers and all multi-core PCs.

An MIMD architecture system that shares a common memory is known as multiprocessors, while those that uses an interconnection network is known as multi-computer (Alghamdi and Alaghband, 2020). Distributed systems are a type of multi-computer system where the processors are physically

far away from each other and communicate with each other through network. The distance between the nodes can range from being in the same room to being across the globe.

During initial stages, computers used to process the data sequentially (step-by-step) to produce results. Introduction of multiprocessors led to development in the field of parallel processing and algorithms. The main purpose of using parallel processing is to perform computations faster by utilising a number of processors concurrently (JéJé, 1992). In parallel computing, a complex problem is split into different parts that can be executed in parallel (simultaneous), synchronised and then recombined to form the final outcome. It improves the response time along with making use of all the available computing resources (Berman and Paul, 1996; Xu and Wunsch, 2005). The concept of a cluster was introduced when people first tried to spread different jobs over more computers and then gather back the data those jobs produced (Weiss 1992). A cluster is a collection of number of processors which are distributed on different computer systems and are connected to each other through a communication medium (e.g. a network). All the processors work in unison to execute a parallel algorithm and provide the necessary speedups that modern computation needs.

The remainder of this section is structured as follows. Section 2.1 presents a brief discussion of the methodology used while identifying related work. Section 2.2 and Section 2.3 focuses on the hardware and software requirements for cluster computing respectively. This will help the reader understand the basics of the work presented in this thesis. Section 2.4 provides the state of the art in SBC cluster computing domain with focus on experimental and energy aware computing. The section also lists popular clusters developed in the past, and their unique features. Different techniques and methods used to evaluate, and benchmark clusters have been documented in Section 2.4 as well. Section 2.5 provides a list of open issues and gaps found in the literature. Finally, Section 2.6 summarises and analyses the whole literature review with a list of open issues and gaps found in the field. These gaps form the base for our thesis and the aim is to solve these gaps through the proposed solution in the thesis.

2.1 Methodology for literature review

The literature review is the first and the most important step in research process for qualitative, quantitative and mixed research studies (Onwuegbuzie et al., 2012). A good literature review can help researchers or readers with a wide variety of benefits which can help summarise the related work in whole and provide some strong conclusions on the literature (Onwuegbuzie et al., 2010). For the purposes of this study, following six steps for good literature review as presented by Fraenkel et al. (1993) were followed:

1. Define the research problem as precisely as possible.

The research problem is the high level aim of the thesis. The research problem defined for this thesis was - “How does different cluster architectures affect different algorithms and how all of this affect the overall performance, energy and quality?”. It is important to define the research

problem precisely as this needs to be answered at the end of a research project. The research problem was identified by a number of steps - choosing a domain of interest, doing preliminary research and asking high-level and open-ended questions regarding the topic.

2. Look at relevant secondary sources.

Once the central aim and research problem is defined and understood, the next step is to search for the sources which can help in better understanding the research problem. Secondary sources of information include anything that analyses or interprets events in the past. For this literature review, secondary sources of information included review papers, journal article, conference proceedings, etc. These sources reflected the current theories and understanding of the past.

3. Select and peruse one or two appropriate general reference works.

The second step resulted in a collection of vast variety of available data and literature relevant to the research problem. The next step was to filter and select the literature which is more relevant to the aims of the research. This filtering process was done using a wide number of parameters including but not limited to the date of publications, credibility of authors, reliability and relevance.

4. Formulate search terms (key words or phrases) pertinent to the problem or question of interest.

Based on the general reference chosen, the next step included an in-depth research of the research problem. The first step in doing so was to find the most relevant search terms from the references found in previous steps. Search terms are words or phrases that you enter into search engines. They represent the main concepts of your research topic and without the right keywords, its very hard to find relevant articles.

The search terms used to find relevant material for this research included - Distributed systems, Algorithms, Parallel Computing, Partitioning, Scheduling, Distribution, Cluster, Server, Master, Slave, Cloud Clusters, Scaling, Supercomputer, Computer architecture, Low cost, Efficient, Energy, Power Measurement.

5. Search the general references for relevant primary sources.

Primary sources provide first-hand information regarding a topic. This can include the basics of a theory or concepts which have been introduced long time ago or the latest experiments conducted to validate a theory. This literature might not be directly related to the work but they provide the base which is required to understand the work presented. For this thesis, primary sources of information included books on basic concepts relevant to the research problem, experiments conducted and results published by researchers, etc.

6. Obtain and read relevant primary sources, and note and summarise key points in the sources.

The last step include documenting the most relevant bits and parts from all the sources and presenting them in a concise and easy to understand manner.

2.2 Hardware

As with any computation, the first step is to acquire the hardware on which the computation will be performed. The resources used to implement parallel processing can be diverse and can include single computer with multiple processors (multiprocessor) or multiple networked computers (multi-computer or clusters) (Zargham, 1996). Multi-computer or cluster refers to many computers connected to each other and performing like a single entity. Hardware is one of the most important and basic part of any computation. Any form of computing is not possible without the availability of required hardware resources. Also, slight changes in the hardware can result in huge improvement or decline in performance. Hence, it is very important to discuss and understand the basics working hardware in computing and also its impact.

As discussed above, clusters work by combing the computation power of individual nodes to solve a problem. All the individual nodes and their hardware affect the overall performance of the cluster to a huge extent. Choosing the correct hardware for nodes is one of the most fundamental decision in starting to build a cluster. Processing unit (PU) is the common component of any computer. Apart from the main processing unit, development of a cluster requires a number of other hardware such as inter-connection cables, storage devices and up-to a certain limit cooling hardware.

2.2.1 Processors

Processor or Central processing unit (CPU) is the most basic component of a computer that includes a processing unit and control unit (CU). It performs basic arithmetic, logic, controlling and input/ output operations of the computer. Most modern CPU's are microprocessors where the CPU is contained on a single integrated circuit (IC) chip. The microprocessors in the CPU's have been developed and improved over the time and there have been number of different microprocessors based on different needs developed. High-level classification of the available microprocessors with some widely known examples are presented in Figure 2.2.

The main difference between CISC and RISC processors is that CISC approach attempts to minimise the number of instructions by having a greater number of cycles per instruction whereas RISC reduces the number of cycles per instruction at the cost of the number of instructions per program (Akram, 2017). Most of the modern computers contain RISC processors and the progression from 8 to 16 to 32 bit architecture essentially forced the need for RISC architectures to be implemented (Blem et al., 2013). Based on the RISC architecture and the need to miniaturise the CPU for portability, ARM processors were developed. ARM stands for Advanced RISC (Reduced instruction Set Computer) machine. ARM processors are extensively used in consumer electronics such as smartphones and other wearable. They are small in size and lower power consumption while delivering high performance make them ideal processor for miniaturised devices.

CPU comprises of a basic processing unit called a core. Cores work independently of each other and they execute program instructions, as if the computer had several processors. The microprocessors currently used in almost all personal computers are multi-core. A multi-core processor is an CPU

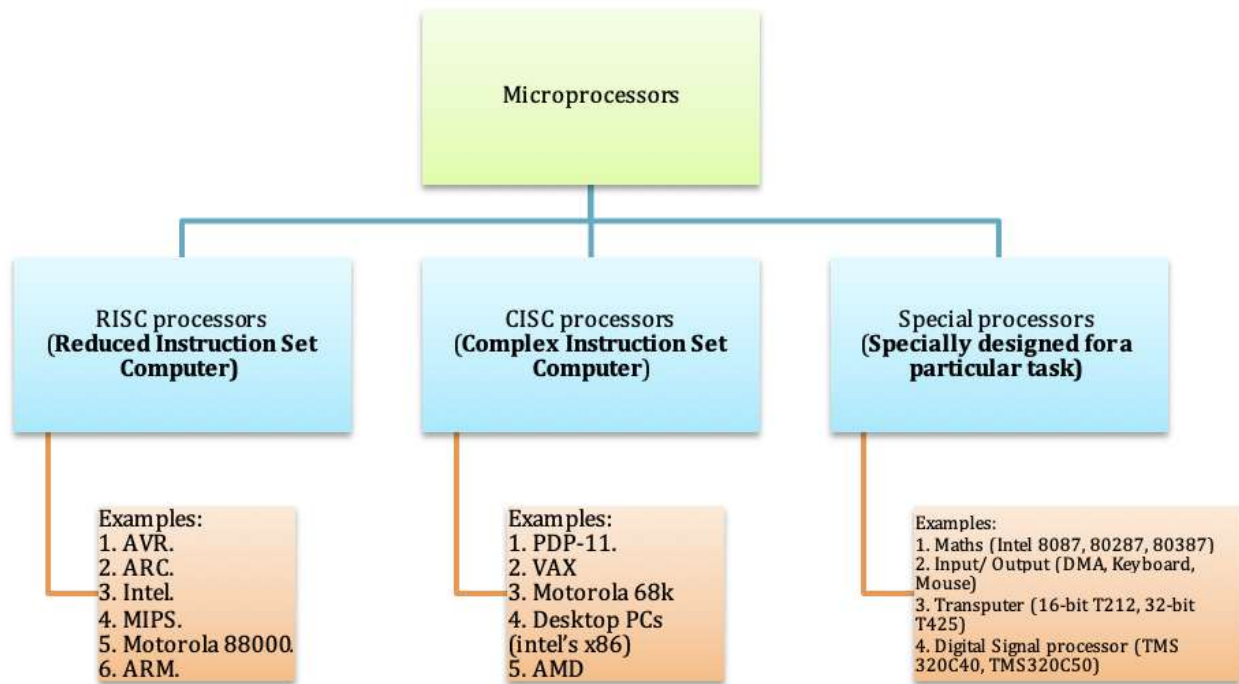


Figure 2.2: Classification of microprocessor.

with two or more separate cores working together to achieve improved speeds and functionalities. Due to their independent nature, cores need to communicate with each other through some medium to produce final computation results. Based on the communication medium, different cores can be coupled tightly or loosely. Cores are said to be tightly couple when they share a common cache. In loosely coupled cores, the inter-core communication is achieved through implementation of message passing or physically share-memory interface. A multi-core processors supports multiprocessing without any additional requirements as multiple cores can work individually on different computation and produce results faster.

In computing, clock rate typically refers to the frequency at which a processor can execute instructions (Davidson, 1965). It is also the frequency at which processors generate pulses, which are used to synchronise the operations of its components (Davidson, 1965). It is generally an indicator of how fast a processor is. Its measured in Hertz (Hz). Higher clock rate means faster processors and better computing times (Srinivasan et al., 2018). Clock rate can be increased or decreased, in certain limits, by varying the energy provided to the processor. Overvolting is increasing the clock rate and hence providing more energy for optimal performance. Undervolting is the opposite where the clock rate is reduced, and energy provided is decreased. The need to increase or decrease clock rate is totally the user's decision and most processor manufacturers provide a safe limit in which their clock rates can be tweaked without loosing performance. Most modern devices are tolerant of over-clocking or under-clocking. Most of the processors will have a maximum or minimum "stable" speed where they still operate correctly. If operating a processor outside these limits, the processor might completely fail or produce incorrect results.

2.2.2 Memory

Computer memory is defined as any physical device capable of storing information. Every computer has two kinds of memory – RAM, which stores information temporarily and ROM, which stores information permanently. In computation, RAM is often referred to as memory. For the purposes of this study, memory terminology is used to refer to RAM memory of a computer. RAM is a super-fast and temporary data storage space that computer needs to access right now. It plays a very important role during computation as RAM is the memory where most of the synchronisation, output, inputs, etc. resides.

There is a physical memory on each computer which is accessible by CPU to process. With the introduction of multi-core computers, the memory was needed to be shared across different cores. Each core can have its own space in memory of share a collective big space with other cores. The latter is called a shared memory model of computing where many individual computing processes communicate with each other using a shared memory. In this situation, the memory is physically shared across different cores on a single computer.

In a distributed system, multiple computers with their own individual memory might implement a Virtual Shared Memory (VSM) model for synchronisation between processors. Each node receives an abstract shared memory which provides the connected processors with impression of being in a shared memory environment. This is mainly achieved through a message passing interface which provides the synchronisation across different processors. Memory allocation in such systems can be simulated through number of ways. Shared memory can be simulated on distributed systems by implementing message passing interface for synchronisation, where a distributed memory can be simulated on a computer by restricting access of individual cores to a particular memory space.

Shared memory and Message passing models have been widely used for parallel programming (Zargham, 1996). Shared memory model refers to programming on a multiprocessor where communication between processes is achieved through shared or global memory and message passing model refers to programming in a multi-computer environment where communication is achieved through some kind of message switching mechanism (Zargham, 1996).

Cache memory is a hardware or software component that stores commonly requested data. In a computing terms, it might be a certain block of data which needs to be accessed multiple times during computing. Cache memory is faster than RAM and is usually relatively smaller than RAM as larger resource implies greater physical distances. In computing, data is constantly being read and written and this can take a long time if the data store is slow. Cache is used in such situations to write results of computation or reading a recently requested data by CPU faster. Cache request by CPU are faster than any other data store and hence, the more requests it can serve from cache, the faster the system performs.

2.2.3 Cluster computing

Cluster computing refers to many computers connected to each other and performing like a single entity. Clusters can also be defined as two or more computers that are networked together to provide solutions to given problems. Clusters were introduced when the traditional methods of computing were not enough to solve the modern-day technologies and advancements. Researching and developing powerful multiprocessors is a very expensive task and as discussed in the previous sections the speed of the processors cannot be increased any more due to inherent limitations of them. Clusters were introduced as an alternative to powerful multiprocessors in which many moderately powerful computers were combined to provide exceptional results. A cluster of computer joins computational powers of individual computers to provide a combined computation power. Cluster computing offers solutions to solve complicated problems by providing faster computational speed, and enhanced data integrity. Cluster computing exploits the parallelism into the computing requirements and provides exceptional results. Clusters can be developed with any type of architecture and the performance differs according to the underlying processor chosen.

Clusters have been classified into a number of ways based on their features. Based on the proximity of the cluster nodes, they are classified as multi-computer (computer nodes close to each other – in the same room) and distributed systems (computer nodes far away from each other – different cities). Inter-node communication in a cluster mainly depends on the bandwidth of the connecting medium. Bandwidth is defined as the maximum data transfer rate of a network. For a given computation needs, distributed systems with high bandwidth can be a better option than a multi-computer cluster with slower bandwidth. To gain faster computing, physical distance can be considered a secondary characteristic and more priority should be given to bandwidth of inter-node communication. The bandwidth between nodes can be evaluated using ping pong benchmark (See Section 2.4.3).

An alternate high-level classification based on the unique functionality of the clusters have also been documented (Yeo et al. 2006; Buytaert 2000):

- **Fail-over Cluster:** They consist of 2 or more network connected computers with a separate heartbeat connection between the 2 hosts. They are monitored and as soon as a service on one machine breaks down the other machines try to take over (Buytaert, 2000).
- **Load-balancing clusters:** When needed, the cluster checks which machine is the least busy and then sends the request to that machine thus balancing the load among all the machines and not just one machine (Buytaert, 2000).
- **High Performance Computing Cluster:** The machines are configured specially to produce extreme performance that the user requires. In this, the processes are being parallelised and routines that can be execute separately are distributed on different machines instead of having to wait till they get done one after another (Buytaert, 2000).

Many clusters might have more than one feature of the other, but the expected basic functionality is what leads to the development of the clusters. The computers are configured according to the need and

the ideal situation would be creation of a cluster with perfect balance between balance, performance and effectiveness.

2.2.4 Single Board computers

Single Board Computers (SBC) is a complete computer built on a single circuit board, providing all the features of a traditional computer. Due to their miniaturised nature (Section 2.2.1), most of the single board computers are developed on ARM processors. They perform reasonably well when compared with the cost and power consumption (Kecskemeti et al., 2017; Schot, 2015). Also, development in ARM processors have led to more and more advanced and more powerful single board computers. SBCs were first introduced as an object of curiosity, but they are gaining more technological importance in the present generation (Basford et al., 2020). In initial stages their features did not compare anywhere near to the physical computers and hence were only used for educational purposes but these past few years have been significant changes in SBC hardware capabilities (Basford et al., 2020). This has led to SBC being seen as a potentially useful technology rather than just an object of curiosity.

The advancements in SBC technology has led to around 20 times the performance as compared to its predecessors in just 5 years (Basford et al., 2020). This directly translates to a massive improvement in SBC clusters. Comparatively, an SBC will always have limitations as compared to the newer computers and technologies (Basford et al., 2020) but use of many SBCs in clusters provide almost the same or better performance in some cases while providing extra benefits in the same process. Table 2.1 lists some of widely used single board computers.

Single Board Computer	Family	CPU	Memory
Raspberry Pi Zero	ARM1176	Broadcom 2835	512 MB
Raspberry Pi Model A+	ARM1176	Broadcom 2835	256 MB
Raspberry Pi Model B	ARM1176	Broadcom 2835	512 MB
Raspberry Pi Model B+	ARM1176	Broadcom 2835	512 MB
Beagleboard-xm	Cortex A8	TI DM3730 512	512 MB
Beaglebone Black	Cortex A8	TI AM3358/9	512 MB
Pandaboard ES	Cortex A9	TI OMAP4460	1 GB
Raspberry Pi Model 2-B	Cortex A7	Broadcom 2836	1 GB
ODROID-xU	Cortex A7	Exynos 5 Octa	2 GB
Raspberry Pi Model 3-B	Cortex A53	Broadcom 2837	1 GB
Dragonboard	Cortex A53	Snapdragon 410	1 GB
Jetson TX-1	Cortex A57	Tegra X1	4 GB

Table 2.1: Comparison of widely used single board computers.
Derived from (Cloutier et al., 2016)

During the writing of this thesis two of the most researched and advanced single board computers include the Raspberry Pi and the Odroid series. The latest board in the RPi series include the RPi 4B (ARM Cortex-A72 quad core CPU, 8GB RAM, gigabit Ethernet, USB 3.0 and USB C for power) while Odroid series include Odroid XU (ARM Cortex-A7 octa core CPU, 2GB RAM, USB 3.0). These

SBCs provide benefits that of laptops and computers while being small in size and energy efficient. Many researches have been conducted including these two SBCs and when used in clusters they provide competing performance to those of high-end computers (Section 2.4.3).

2.2.5 Summary

The section provided background related to the hardware aspect of computing. Basic understanding of a computer architecture is briefly discussed. The processors and their evolution through times provide a brief background in the development of computers. Different memory types and their effects on computing are also discussed. Widely used models for parallel computing in clusters have been documented and the section ends with a list and features of single board computers. The development of single board computers and their significance in the field of computing have also been discussed.

2.3 Software

Effective cluster computing depends not only on hardware but also on the software. The software needs to be interacting with the hardware in the most efficient way. If the software is incompatible with the hardware, then there can be performance issues and crashes which can affect the overall computation being carried out. A little change in the software on a cluster can provide huge difference in the performance of the whole cluster and hence, a lot of improvements have been done to provide the best and compatible software for a particular cluster architecture. Cluster computing requires installation of supporting software such as the computing libraries for particular problem, message passing interface for communication, etc. installed on each individual node to function properly and effectively (Alghamdi and Alaghband, 2020).

2.3.1 Operating System

The operating system is the most important software on a computer that can affect the performance and reliability of all the dependent software. Linux is the most commonly used OS on clusters as its best known and most-used open source operating system. Linux is the software that sits underneath all other software on a computer and facilitates the communication between a software and computer's hardware. Being an open source OS, Linux has been highly modified to suit particular needs. Ubuntu, Fedora, Debian, Arch Linux, etc. are the most commonly used distributions of Linux.

2.3.2 Inter-node communication interface (MPI)

Cluster comprises of lots of nodes working together. These are individual computers which are connected to each other through physical hardware. They have their own individual CPU's and memory blocks which do not interfere with each other and hence work independently. The individual nodes/ processes communicate to each other using message passing interfaces (MPI). MPI is a

standardised and portable message passing standard designed to function on a wide variety of parallel computing architectures. There are many efficient and well tested implementation of MPI. Some of them include MPICH and OpenMP supports programming in C, C++ and FORTRAN. Many libraries extend the MPI implementation by binding them to the existing implementation such as MPICH and OpenMP. MPI4Py is a binding for MPI in python while Rmpi is binding for MPI in R language. As these are essentially just the bindings for existing implementation there is no performance hit to the computation while using them

2.3.3 Storage Framework

Applications on a cluster must have a quick, dependable, concurrent and always available access to storage framework. The storage demands can be met by having the exact same storage on each individual node or bringing together a storage node (most of the time master node) that will be in charge of providing storage framework to other nodes. As all the nodes are connected to each other, Network File system (NFS) is the most commonly used protocol to access and make changes to the storage file system. The master server hosts the NFS daemon process to make data available to clients and each individual node requests access to data by mounting it at a particular location. NFS is not a requirement and clusters can work effectively even without it.

2.3.4 Job Schedulers

As discussed in previous sections, Job schedulers and workload managers are used to easily and effectively manage the task of execution of tasks and monitoring them. Job schedulers such as Torque/maui, torque/moab and SLURM have been heavily used in many HPC clusters (Yoo et al., 2003). Torque/maui has been neglected for few years and torque/moab has paid features to effectively use it. SLURM has open model and is easy to use as all the configuration is done only in single file. SLURM has modern design in its design and implementations and hence is preferred among all other job schedulers and workload managers (Yoo et al., 2003). Other Job schedulers include Portable Batch System (PBS) and Sun Grid Engine (SGE).

Job Scheduler is a resource management system which performs important functions like scheduling user jobs, monitoring machine and job status, launching user applications, and managing machine configuration (Yoo et al., 2003). Scheduling can be achieved using the help of a workload manager such as SLURM. Slurm is an open source, fault tolerant and highly scalable cluster management and job scheduling system for Linux clusters (Yoo et al., 2003). Slurm works on the existing kernel and requires no modifications for its operation and is self-contained (Iserte et al., 2014). As a job scheduler, Slurm performs three main functions: Allocating access to resources (compute nodes) to users for performing work, providing framework for starting, executing and monitoring a parallel work and managing a queue of pending work (Yoo et al., 2003). Slurm has been successfully used in HPC (high performance computing) for job scheduling and process queuing (Cloutier et al., 2016).

2.3.5 Computing Libraries

All of the previous software's are used to setup the cluster and the computing processes (Alghamdi and Alaghband, 2020). Apart from these, individual libraries are required which are dependent on the type of computing the cluster is doing. Computing libraries in the domain of vector and matrix operations include the Basic Linear Algebra Sub-Programs (*BLAS*), Automatically tuned Linear Algebra Software (*ATLAS*), *OpenBLAS* and Linear Algebra PACKage (*LA-PACK*) is heavily used by High performance Lin-pack (*HPL*) benchmark to calculate the efficiency of clusters (Cloutier et al., 2016; Papakyriakou et al., 2018; Balakrishnan, 2012). Intel developed their own library called Math kernel Library (*MKL*) which is heavily optimised to work alongside Intel processors to provide the best possible performance and efficiency. In domain of image processing and graphic processing Open-CL framework is used to execute programs on GPUs and other digital signal processors (Cloutier et al., 2016). Other commonly used libraries for computing are SciPy, NumPy and Sklearn.

2.3.6 Parallel Algorithms

For a long time, sequential algorithms are used for computation purposes, in which operations must be executed step by step (Zargham 1996). Introduction of multiprocessors and multi-computer meant that computation can be done relatively faster due to the inherited parallelism. Better and improved parallel algorithms were introduced over time by researchers in order to gain maximum performance from computing platforms (Chai and Bose, 1993). A parallel algorithm for a parallel computer can be broadly defined as a set of processes that may be executed simultaneously and may communicate with each other in order to solve a given problem (Kung 1980). A process is defined as a part of program that is execute on a processor.

Designing a parallel algorithm comprises of lots of steps and considerations. An effective parallel algorithm needs to fulfil few characteristics and conditions effectively to be able to be used in parallel computing. Chai and Bose (1993) presents few of the important conditions and characteristics:

- The algorithm needs to be able to use the full CPU resources.
- The algorithm should not waste any CPU cycle during processing.
- There needs to be effective communication between individual processes that results in overall better performance of the algorithm.
- The algorithm needs to be scalable. It should work on any number of processors.
- The algorithm needs to balance the load evenly on individual processes.

This list is not exhaustive and there are many other conditions that need to be fulfilled by the algorithm to be considered effective. There are many parallel algorithms designed for the same functionality and each of them focus on an individual characteristic. Some algorithms focus on improving the efficiency while some focus on reducing the communication overheads. Algorithms are constantly under improvement as there is always a chance to make it perform faster and better.

Inhibitors of parallelism

Achieving true parallelism in any computing situation is hard to achieve as it depends on many factors such as the architecture of computer, cost of communication between the processes and even the way in which the algorithm is written. The architecture of the computer on which the algorithm is executed affects the performance of it as a particular algorithm may be efficient on one architecture while being inefficient on other and poorly written algorithms may lead to communication time being greater than actual computation time. The designer needs to consider all these inhibitors in order to design an efficient parallel algorithm.

- **Data Dependence:** All the dependencies of an algorithm need to be identified and properly managed so that they don't interfere with each other. A data dependency occurs when multiple instructions use the same location in storage for different tasks and flow dependency occurs when an instruction depends on the output of the previous instruction. These dependencies need to be carefully removed and if not possible, the algorithm is considered to be non-parallelizable.
- **Overheads:** Communication overhead between the parallel processes is the most important consideration while designing a parallel algorithm. The cost for communication in a parallel algorithm affects the performance and the efficiency of the algorithm. For some algorithms, communication time may be greater than actual computation time. This may lead to degraded overall performance. Communication includes dividing the tasks, sending the tasks to nodes, retrieving the output of nodes and other inter communication of data between the processors or nodes.
- **Load Balancing:** The main objective of load balancing is to engage all the cores so that the tasks are completed in minimum elapsed time. This often takes care of two key overheads: inter-process communication and idle CPU processes. Load balancing refers to distribution of equal amount of work to all the processors or nodes so that the processing units are kept busy with the least interference from anything else.
- **Hardware Dependence:** The hardware on which the algorithm needs to be executed should also be considered while developing an algorithm. For example, an algorithm which requires a GPU to execute cannot be executed on system that has no GPU or does not support that execution. This is an important consideration, since the same algorithm may be very efficient during the designing process but very inefficient on another architecture later on.

Evaluating parallel algorithms

Once an algorithm has been designed, the next step is to evaluate the algorithm. An algorithm can be evaluated in a number of ways and each of the evaluations explains more about the algorithm and proves to be a basis for further improvement. Some of the evaluation methods for an algorithm include its run-time, accuracy, scalability, ease of use and cross compatibility.

execute time is one of the most common measurement to evaluate the performance or efficiency of an algorithm. It is also referred to as elapsed time or completion time. It is the time taken by the algorithm to solve a problem. More specifically, it is the elapsed time between the start of the first process and the termination of the last process.

The accuracy of the result can change due to many constraints and the solution to a given problem should be same by any algorithm. If the result is not as accurate or correct as the one obtained from some other algorithm, then there might be errors in further computing or analysis.

A parallel algorithm is designed assuming that it will be executed parallelly on a number of computers or processors. The algorithm needs to be scalable and it should be able to work at the same efficiency on 2 computers as well as thousands of computers. The number of processors that the algorithm will be executed on is not known and hence this is an important measurement of robustness of an algorithm. It might also happen that an algorithm executes efficiently on few processors but slows down on large number of processors. Evaluating and considering all this will lead the algorithm to be more scalable and robust.

This is not a preferred evaluation method, but it is good practise for the algorithm to be easily understandable by anyone who did not design it. Its implementation needs to be easy and the instruction clear on how to use it. Harder to understand algorithm can lead to its wrong implementation and incorrect results.

An algorithm needs to be able to execute on any operating system with very limited number of pre-requirements. An algorithm which supports only a single system or needs a very specific requirements cannot be collaborated by others easily.

Popular algorithms in parallel computing

There are vast number of algorithms and with rapidly growing technologies, it's nearly impossible to be exhaustive while listing them. Book on parallel algorithms by Alan Gibbons and Wojciech Rytter try to explain and list the algorithms in details (Gibbons and Rytter, 1989). Most of the low-level parallel algorithms can be highly classified into following seven categories. Newer algorithms can have more than one features (Hillis and Steele Jr, 1986; Gibbons and Rytter, 1989). A non-exhaustive high-level classification of parallel algorithms are provided:

- Recursive algorithms (Strassen algorithm, tree traversals, quicksort).
- Dynamic programming algorithm (Insertion Sort).
- Backtracking algorithm (Sudoku solving).
- Divide and conquer algorithm (Merge sort and Quicksort).
- Greedy algorithm (knapsack problem, Huffman compression trees, task scheduling).
- Brute Force algorithm (Travelling Salesman problem).
- Randomized algorithm (Randomised quicksort, Monte Carlo algorithm).

2.3.7 Summary

The section provided background related to the software aspect of computing. The section starts with the software requirements for parallel computing. Each of the requirements are explained while providing a brief discussion of their effect on the computation performance. Just like hardware, software plays a very important role in computing. Computing libraries, Operating systems and inter-node communication libraries are few of the basics requirements for parallel computing. Parallel algorithms, their inhibitors and the methods of evaluating them are also described in this section. The section concludes with a list of the most popular algorithms which are used in most parallel computing situations.

2.4 Related Work

2.4.1 Experimental Cluster Computing

Clusters were mainly built to gain high performance computation. When the focus shifted to find the balance between cost and performance, many ideas were put forward by researchers theoretically to achieve perfect balance. Cluster performance depends on a lot of factors, and there is no ideal way to develop clusters. Lot of researchers developed their own novel clusters with specific configurations to tackle the problems that they want to solve. A lot of experiments were conducted on clusters to find the optimal solution for the problems.

Apache Hadoop infrastructure is very expensive and requires a lot of energy, real estate, cooling, etc. Big data issues like storing, retrieving and handling have been addressed and solved by a number of experiments involving SBC clusters. SBC clusters have been concluded as an effective alternative for mobile Hadoop clusters and robust computing performances (Qureshi and Koubaa, 2017; Schot, 2015; Srinivasan et al., 2018). Comparative study of performances of clusters with different numbers of nodes showed that 10 nodes SBC cluster was more superior in performance to a single computer and 5 nodes SBC cluster by 20% and 80% respectively.

Novel cluster architectures were introduced to tackle the problems of Edge computing and Big Data (Basford et al., 2020; Schot, 2015). PiStack focuses on power efficiency and thermal output while providing an optimal performance in edge computing conditions. It implements some of novel ideas to reduce cluttering by powering nodes through the cluster case and saving power by introducing heartbeat functionalities for each node (Basford et al., 2020). Similar to PiStack, novel alternatives for 1U rack in big data centres involving stacking RPi's to achieve maximum accessibility and easier replacement (Schot, 2015). They also prove that SBC clusters can be a feasible approach to solve Big Data issues of storing and retrieving data.

Using SBC clusters to perform cloud simulations and provide virtualisation of resources have been studied (Kecskemeti et al., 2017; Tso et al., 2013). Cloud infrastructure has been simulated by iCanCloud (Nunez et al., 2011) and CloudSim (Calheiros et al., 2011) in the past. Same functionality has been achieved using SBC clusters in Glasgow Raspberry PiCloud (Tso et al., 2013). PiCloud

simulates every layer of the cloud infrastructure, ranging from resource virtualisation to network behaviour (Tso et al., 2013).

Image processing is one of the applications which makes use of in-built parallelism during processing. Computation time is a very important factor during image processing and clusters have been proved to reduce the computation time significantly. Algorithms in OpenCV libraries exploit parallelism to process each image faster and in real time. SBC clusters executing algorithm which use OpenCV library significantly outperformed single computers in frame processing of a live stream video (Pomaska, 2019). Accuracy of SBC cluster in image learning was studied by using the Scikit Image library and by executing two parallel algorithms – Watershed and Edge detection on number of images (Markovic et al., 2018). OpenMP is an application programming interface that supports shared-memory multiprocessing. It has been used in clusters to perform image processing faster and to provide comparative study of different image processing libraries (Patel et al., 2015; Rahmat et al., 2019).

2.4.2 Energy Aware Cluster Computing

Clusters have been used in a lot of disciplines which requires high computation power. Along with maximum performance, there has been a need to design a cluster which stays inside the power budget (Cloutier et al., 2016). Supercomputers have been consuming vast amounts of electrical power and produce so much heat that large cooling facilities needed to be constructed to ensure proper performance. In order to motivate this approach, a Green500 list was generated which consists of energy efficient supercomputers.

A lot of studies proved that single board computers (SBC) are more energy efficient than daily use computers and this paved a pathway towards studies about energy consumption in different SBC clusters (Saffran et al., 2016). Terms like power/ energy consumption (GFLOPS/W) and value for money (GFLOPS/\$) are used frequently to depict the effectiveness of the cluster in terms of performance and cost.

Apache Hadoop framework is commonly used for analysis of data intensive operations such as Big Data analysis where large volumes of data need to be analysed effectively (Qureshi and Koubaa, 2017). Hadoop's Map/Reduce model has been used as a benchmarking tool for comparing performance and energy consumption of various architectures (Feller et al., 2015). Comparative study of energy consumption in big Hadoop clusters show that SBC clusters can be an effective alternative to big data centres (Conejero et al., 2016; Krish et al., 2014; Qureshi and Koubaa, 2017; Tiwari et al., 2016).

Data mining algorithms are essential tools to extract information from the increasing number of large data-sets, also called Big Data (Saffran et al., 2016). They are high power and performance demanding algorithms. Executing two of these algorithms (Apriori and K-means) on SBC clusters and daily-use computer showed that SBC clusters out-performed while proving to be the most cost effective (Aroca and Gonçalves, 2012; Cloutier et al., 2016; d'Amore et al., 2015; Saffran et al., 2016). They compare the results from SBC clusters with that of a high-performance computing (HPC) platform and

concluded that even though SBC cluster performs lower than HPC platform, they are very effective at energy consumption and value for money.

Cloud computing is an example of edge computing in which a large number of computations are performed remotely on the data centres through some form of message passing (Qureshi and Koubaa, 2017). Edge computing means having compute resources near to the data-sources (Basford et al., 2020). In the modern world there is less and less space available for installation of large computing hardware. Hence, more and more data centres and installations are done in remote locations. There are very limited power resources in remote locations and hence power efficiency in terms of GFLOPS/W is an important consideration (Basford et al., 2020). SBC clusters for cloud and edge computing have been compared to find the best architecture to provide maximum performance in terms of low network latency, communication overhead, low power and energy consumption (Basford et al., 2020; Qureshi and Koubaa, 2017).

2.4.3 Related work in benchmarking and evaluating clusters

Evaluating and benchmarking a cluster is very important as it lets the user know how much performance the cluster can provide and how to make it better. There are many benchmarking libraries and frameworks developed to test different aspects of a cluster. Following lists few of the benchmarking frameworks used to evaluate different functionality of a cluster:

- Performance and speed:

CoreMark: Benchmarks the number of iterations per second. It calculates computation on matrixes and lists such as sorting, searching, reversing, etc.

HPL: High performance Linpack benchmarks number of aspects in a cluster. It performs Algebra and matrix computation and calculates the maximum speed and performance of the cluster.

- Memory bandwidth:

STREAM: Measures sustainable memory bandwidth in MB/s and vector kernels such as Copy, Scale, Add and Triad.

- Accuracy of results:

Linpack: Calculates average rolled and unrolled performance of a cluster by solving NxN system of linear equations and using double precision floating points.

HPL: HPL determines double precision floating point performance for a given computation.

- Network Speed:

PingPong: Tests the latency and bandwidth of network communication between two nodes. It transmits and receives a message of certain size and measure the time taken by it to come back.

Many researchers have bench-marked and documented the performance of different computers. Figure 2.2 provides an extract of results obtained by executing different benchmarks on computers.

The energy consumption of each is also documented in the similar table. All of the benchmarks were done using OpenBLAS libraries.

#	Name	GFLOPS/W	Performance (GFLOPS)	Average Power (Watt)	Processor Type	RAM (GB)
1	RPi 4B 64-bit	2.02	13.5	6.6	Cortex A72	4
2	Haswell desktop	1.56	145	92.	hsw i7-4770	4
3	RPi 3B+	0.73	5.3	7.3	Cortex A53	1
4	Dragonboard	0.450	2.10	4.7	Cortex A53	1
5	RPi 2	0.432	1.47	3.4	Cortex A7	1
6	fam16h-a8-jaguar	0.354	14.1	39.7	A8-6410	4
7	RPi zero	0.236	0.319	1.3	BCM2835	0.5
8	RPi B+	0.118	0.213	1.8	BCM2835	0.5
9	Beaglebone-black	0.026	0.068	2.6	Cortex A8	0.5
10	Sparc	0.003	0.456	140.7	Ultra	0.5

Table 2.2: HPL benchmark results on different computers.
Derived from (Cloutier et al., 2016)

2.5 Open Issues/ Gaps

Parallel and Cluster computing has been extensively studied and implemented in a number of different domains to solve particular problems. They have been constantly improved on and updated keeping in mind the present computing needs. Algorithms on clusters have always been a hot topic in research as there are innumerable factors which can improve the process of computing and modern-day problems demands more and more powerful algorithms capable of high-performance computing.

During the survey of literature in the field of parallel computing it was noted that most of the algorithms designed or evaluated follows a constant pattern – to be the most efficient or the fastest. Also, that cluster designing, or cluster computing have been traditionally evaluated through its performance and very few frameworks are present which provide overall evaluation and analysis of clusters.

Speed and performance of an algorithm or a cluster is undoubtedly very important in computing but in modern times, they are not the only characteristics that needs focusing on. Major issue affecting the modern computing systems include – high energy consumption, major cooling and maintenance infrastructure resulting in higher costs, high real estate cost. Related research has already been done in the field of reducing the overall energy consumed during big data analysis (Srinivasan 2018) and gaining maximum performance from limited energy resources in edge computing. (Philip J. Basford 2018).

Non-exhaustive list of all the issues and concerns raised during the survey of literature in the field of parallel and cluster computing is given below:

- All the algorithms are used, optimised and evaluated considering its performance and speeds.

- Very few algorithms developed till now focus on the energy aspect of computation. They don't factor in the value or the cost of performing the computation.
- There is no definite framework to evaluate the algorithms based on its energy consumption. The existing framework evaluates their performance and efficiency which leads to improvement in the same. Development of a framework to evaluate the energy consumption along with performance will lead to development of more energy efficient algorithm with balance between performance and cost.
- Sometimes the algorithms are designed just to access the inherent parallelism in the computation. They don't focus on the load balancing and hence their algorithm is not as efficient as it's supposed to be.
- No framework to evaluate, analyse and benchmark clusters with different configuration. All the benchmarks test clusters with a single configuration.
- No framework to evaluate the cost/ performance analysis of cluster. Calculating performance per dollar spent can help in understanding the energy aspect and consumption for a computation.
- Energy consumption of an SBC cluster has not been researched in depth and there are gaps in literature in relations between performance and power or resources used and power consumption.

2.6 Summary

This chapter has introduced the field of parallel and cluster computing. Section 2.1 discussed the methodology of identifying and reviewing relevant literature. This is a general section and has been done with sole purpose of explaining the steps undertaken to reach this point and writing the literature review. As the focus of this thesis is that of a generic framework for evaluating parallel algorithms on different hardware architectures, it is important to know the factors that can affect the performance of a computation (hardware and software aspects). To do this Section 2.2 and 2.3 provided a brief background in the field of cluster computing from hardware and software point of view respectively. They both argue that there are innumerable factors that can affect the final performance of a cluster. This can range from a line of code in algorithm to the configuration of the cluster.

In Section 2.4, related work undertaken by other researchers have been included. The introduction of parallel computing and energy efficient computers allowed a wide range of research to be conducted in this field. The section documented the previous work done in experimental and energy aware cluster computing. Existing frameworks used to evaluate, and benchmark clusters have also been listed. Comparative evaluation of HPL benchmark test on a number of computers have been included.

The reviews and analysis in this chapter lay important ground work for forthcoming chapters. Firstly, by having reviewed the effects of hardware and software components on the performance, a framework that will cover all these parameters and evaluate accordingly can be developed. Secondly, reviewing and analysing the work of past researchers allows a definite set of requirements to be

outlined for addressing the research question and open gaps. The literature also allows a comprehensive evaluation of our proposed system. Finally, the literature provides us with the basic understanding of designing, implementing and evaluating the proposed system for a particular architecture system.

CHAPTER 3

METHODOLOGY

As discussed in Chapter 1, the approach chosen to evaluate different cluster architectures is by creating a standardised framework which can help users to evaluate the performance of an algorithm on different cluster specification while providing a cost-performance and energy-performance analysis at the same time. As discussed in the previous chapter, there is little related work in the field of energy aware cluster computing. However, none of them have the aim of developing a framework with the intentions of evaluating performance of a cluster in relation to its energy consumption. In addition, none of them aim to support multiple cluster configurations. They evaluate the cluster setup by executing a single algorithm which do not produce a comparative analysis to the user on the cluster performance.

The purpose of this chapter is to provide enough information regarding the approach undertaken while developing the framework, in order to be able to replicate it in future. The chapter explains the underlying concepts used while conducting the experiments without going too much in detail about the technology used. The specific technologies and approach used for this thesis will be discussed in Chapter 5. This chapter investigates the principals while designing and conducting an experiment. The chapter starts with a broader approach taken towards the experiment and narrows it down to the specifics in the experiment.

Firstly, Section 3.1 discusses the research questions identified from the open gaps in Chapter 2. Section 3.2 provides an introduction to the overall approach towards the experiments conducted. It provides an argument why the approach was used and how it relates to other approaches in similar experiments. Section 3.3 discusses the data collection methods and provides a detailed analysis of why the method was better than other methods used in previous researches. Then Section 3.4 elaborates on how the data collected was validates and deemed reliable whereas Section 3.5 describes the strategy used while analysing the data which is collected and validated. This section also lists other ways in which researchers have analysed data in past in similar experimental setups. Section 3.6 discusses a list of requirements for a generic framework for evaluating parallel algorithms. These requirements provide a base for the design and implementation of the framework and later focus for the evaluation. Finally, Section 3.7 summarises the chapter while emphasising on the relation between the methodology and the aims of this thesis.

3.1 Research questions

This thesis aims to propose a method of evaluating parallel algorithms on different cluster configurations and to provide the researchers with ability to assess their computation needs with respect to their energy needs. The proposed idea is the introduction of a framework, which can be used by researchers to help choose a cluster configuration for their particular needs. Extensive literature review led to the following set of research questions to be developed. The goal of the work presented in this thesis is to explore these research questions, with possibility of future research in the similar field.

- How do different parallel algorithms executing on different SBC cluster architectures affect the overall performance of the system?
- What is the best architecture design and specs of SBC clusters for a researcher that want to test their algorithm effectively?
- How does different cluster configuration affect the energy consumption during computation?
- How do researchers choose a particular cluster for their computation needs?
- How to achieve a platform independent framework with consistent results?

3.2 Paradigm

For a long time, quantitative and qualitative paradigms of research have been the two different ways of solving a problem (Patton, 2014). A quantitative study to solve a problem is based on rigorous and controlled techniques whereas qualitative studies are based on social realities. Both studies have different way to look at and solve a research problem. The methodology adopted by quantitative research is mostly experimental with focus on hypothesis testing (Patton, 2014). It quantifies variables and solves problems using numeric assessment. In qualitative study, it is believed that the constructed reality is based on experiences, circumstances and situations. There is no single correct answer in qualitative study whereas a quantitative study produces a single definite result.

For the purposes of this study, both quantitative and qualitative approaches were chosen as the research questions partly relate to the functionalities of our framework and partly on the experimental data collected and analysed during the experiments conducted. The answers to our research questions will be provided in two ways: one through evaluating the qualitative aspects of our framework and second through quantitative data of experiments being executed on our cluster. Related work in similar fields have chosen similar approaches have been found to be the most practical ones (Cloutier et al., 2016; Basford et al., 2020; Qureshi and Koubaa, 2017; Saffran et al., 2016). This approach was chosen as experiments involving data related to energy consumption, performance, cost, etc. cannot be analysed with a qualitative approach whereas verifying that the framework follows the requirements and answers the research questions cannot be analysed using quantitative approach.

3.3 Data Collection

Data collection methods in a qualitative approach are mainly through observations and action research. Quantitative data can be collected in a number of ways – through experiments, controlled observations, surveys, etc. The method by which you collect the data can impact the way you analyse the data. Quantitative data requires a large population of data-sets to be considered useful in drawing definite conclusions whereas qualitative data depends on a finite set of criteria's to be considered (Patton, 2014). Data collection can be primary (collect data generated by yourself) or secondary (use data from different sources).

For the purpose of this study, the data is primarily collected through experimental results. The data is stored and used in analysis which will be used to produce conclusions and provide a base for future work. The sampling size depend on the type of data collected. Sampling size can also be affected by the physical limitations of the hardware or software. Some hardware might not support data generation for a particular sample size whereas some software may crash if the sampling rate is too high. The errors in the collected data have to be analysed according to the specifications of the instrument used. The data is collected and stored in database for future use. Similar methods have been followed by past studies (Cloutier et al., 2016; Basford et al., 2020; Qureshi and Koubaa, 2017; Saffran et al., 2016).

3.4 Validity and Reliability of Data

The quality of research is evaluated by the reliability and validity of the results. Reliability is about the consistency of experiment whereas validity is about the accuracy of the experiment. Especially in quantitative research, it is important to consider the reliability and validity of the experiments while creating the research design, planning methods, writing of results and concluding the experiments. It is important to explain the validity and reliability of data in an experiment so that it provides a base of analysis for future experiments repeated under the same conditions by other researchers.

For the purpose of this study, the quantitative data collected was tested against multiple iterations to validate the results. The data was collected from industry graded instruments and computers, which meant few to no human errors. Multiple iterations of the same experiment were conducted to check if the data collection methods were valid and reliable. The instruments were tested as well under different conditions and they proved to be a reliable source to generate data for different conditions such as different sampling rates, duration and different physical setups.

3.5 Data Analysis Strategy

Systematically applying statistical and/or logical techniques to describe, recap or evaluate data is called data analysis. Data analysis is required where some definite conclusions are to be derived from a large set of data. While analysing data, many factors such as integrity, reliability, validity, data recording methods, issues, etc. need to be considered.

For qualitative approaches, data is analysed towards a set criterion and the final aim is to achieve all the initial requirements for the project. For experimental setups, data is usually analysed at the end of data collection. The data is collected as a whole during the experiments and then analysed by using the statistical or logical methods. For our study, the data was stored into a database and analysed only after the end of experiments. The analysis of data, both quantitative and qualitative, proved to a base for creating the framework. It is important to wait for the experiment to end as many factors affecting the data can be inferred only after analysing the whole data-set (Diwedi and Sharma, 2018; Rahmat et al., 2019)

3.6 Requirements

Chapter 1 showcases the motivation behind the research while arguing that there is lack for a generic framework for evaluation of algorithms on different cluster configurations. The following requirements for the proposed system have been derived from the analysis of Chapter 2.

R1 The proposed system shall help researchers in deciding the best optimal performance for their energy constraints.

The system helps the user in executing their algorithms on different cluster configuration and provide data in easily accessible format to be able to help researchers in analyse the data and reach conclusions.

R2 The proposed system shall support the identification of bottlenecks for a given computation for a cluster configuration.

The system should be able to identify the limitations or the issues in its computation. The system shall be able to identify them for the computation it is performing, or it shall help users in identifying them. This requirement is hard to implement as there might be innumerable things that might affect the performance of an algorithm.

R3 The proposed system will allow the specification of a range of repeated and repeatable experiments which varies algorithm parameters and cluster configurations.

There are many combinations of algorithms, their parameters and cluster configuration. Providing comparative analysis of all the variable aspects of a computing is beyond the scope of this research. Some of the variables affecting the computation process are discussed in this thesis. The proposed system should allow re-run of experiments for these algorithms with different parameters on different cluster configurations. This will allow the user to check the data and compare it against different data sets. This is very helpful in predicting results when the relation between data sets is found by analysis.

R4 The proposed system shall automatically configure the target cluster for the computation needs as defined in the experiment configuration.

The system needs to be able to reduce the manual work of configuring the cluster for computation. Most of the modern cluster management systems like Kubernetes and workload managers like SLURM uses configuration file to configure the clusters and get them ready for computing. Most of these configuration files are complex with a number of variables which are, most of the time, unnecessary for a given computation and overwhelming (Suresh et al., 2019). Kubernetes have implemented Dynamic Kubelet configuration which achieves this goal to a certain extent, but it restarts the nodes and uses the new configuration which is not a preferred way to do it (Pahl et al., 2016). SLURM also does not have any known way to make the changes to the nodes once the slurmd daemon starts on the cluster (Yoo et al., 2003). The proposed system should be executing the required algorithms on the cluster configuration as requested by the user in a single configuration file. This feature is important as the using a single and simple configuration allows users to provide a concise instruction to the system and allows easier debugging in case of errors.

- R5 The proposed system shall automatically execute different algorithms on different cluster configurations.

The system allows evaluation of parallel algorithms on different cluster configurations while reducing the manual work done in execution of the algorithms, configuring the cluster, collecting data from experiments, and visualising them. The proposed system needs to be able to do all this automatically as per the user needs as specified in the configuration file. The system should be able to configure the cluster and algorithms dynamically without restarting the node or services on the nodes. This is very useful in high performance computing where each node plays an important role and can affect the performance drastically.

- R6 The proposed system shall support multiple algorithms from different languages.

The proposed system will form a framework for future developments and improvements. A framework is defined as a platform for developing software and it provides a foundation on which software developers can build their own software depending on their specific needs (Scacchi, 2002). Hence, it is very important for a software framework to be as generalised as possible and support a wide range of modifications and support (Scacchi, 2002). The proposed system will be evaluated against different algorithms and hence, it should be able to support multiple programming languages or provide an easy way to support them for future use.

- R7 The proposed system shall provide a comparative output (within suitable time) of energy consumption for different cluster configurations and algorithms.

The requirement can be broken down into three different aspects. First – being able to provide a comparative output. The system shall provide a wide range of computing output and different aspects like energy consumed by each node or by cluster during computation, the run-time of algorithm, run-time of the computing part of algorithm etc. This will help the user in better understanding their algorithm and how it is related to the energy consumed and performance

of the cluster. Second – within suitable time. Large computation may take longer time and users might not have that long for them to be able to understand the cost-performance of their cluster. The system should be able to provide consistent results within short amount of time which can be used to predict results for larger computations. Third – Energy consumption. The system mainly focuses on energy consumption of a particular computation and hence this is an important aspect. The system should be able to monitor and log energy consumption of the cluster and use that data to provide results to user.

R8 The proposed system shall provide a performance-energy (FLOPS/W) analysis of a different cluster configurations.

The research conducted is aiming to bring about awareness on the energy consumption of cluster computing. The system developed as a result should be able to do the same. The system should be working towards helping a user understand the energy aspects of their computing and make better decisions regarding their computing needs based on energy. To achieve this, cost-performance (FLOPS/\$) and performance-energy (FLOPS/W) analysis of their computation is very important. The analysis should also be presented in a easily accessible format to allow easy discussion and portability of data.

R9 The proposed system shall be able to export the experiment data for further analysis.

Every framework should be able to be compatible with other software's for analysis or easy portability of data. To achieve this, the system shall be able to export the data in a universally compatible format so that the user can use the data to easily analyse the data or work on it through some analysis software. The exporting of data can also be facilitated by implementing a graph functionality which will further help analyse the data easier.

R10 The proposed system shall be compatible across different processor architectures.

Compatibility is the main requirement of any framework. It needs to be cross compatible with most of the available platforms. It is not a compulsory requirement but it is better to develop a software framework in language which has a wide range of supported platforms (Scacchi, 2002). This assures the wide usability of the software and multiple improvements can be done to make the software better. Also, it is hard to replicate or confirm results if the framework is restricted to a particular platform.

3.7 Summary

This chapter has presented the methodological approach undertaken during the progress of this thesis. It provides arguments as to why the particular approaches were decided and how they relate to previous studies in the similar field. The chapter starts from a broader approach towards the study to more specific methods for individual steps in the experiments. Related work in similar fields and their approaches have been studied and concluded that the methodological approach undertaken for the

work presented in this thesis needs to be both - qualitative and quantitative. Both approaches and how they can be achieved are explained in detail with references to similar work done by other researchers.

CHAPTER 4

DESIGN

Chapter 2 presented the current state of the art in parallel and cluster computing and argued that the lack of a generic framework for evaluation of parallel algorithms on different cluster configuration was harming the field of parallel computing both in terms of energy efficiency and performance. It also explored the field of benchmarking and evaluation of cluster computing and showed that the current evaluation frameworks were inadequate for parallel computing in terms of usability, energy consumption analysis, run-time reconfiguration and ability to tweak independent variables or parameters in computing. The conclusion of the chapter was that the development of a framework for evaluation of parallel algorithms which addresses these shortcomings was desirable.

Chapter 3 then discussed the methodological approach adopted while undertaking this research. It outlines the methods used during each step of the research and argues why these methods were decided and links the chosen methods to research in the similar field by other researchers.

This chapter presents the generic high level design of “Framework for Evaluation of Parallel Algorithms on Clusters” - FEPAC, a lightweight and flexible framework for evaluation of parallel algorithms on different cluster configuration. The design follows and aims to meet all the user requirements outlined in Section 3.6. The design presented in this chapter allows FEPAC to be a platform independent framework with minimal installation required for its setup and working.

In addition to these core features, the design also supports collection of the experiment data including the energy consumption (both the cluster and individual nodes), run time and algorithm output. It also provides an option to export data so that it can be used as a backup or in further analysis of the experiments. The main aim of this chapter is to make the reader understand how the system achieves the aims and goals of this thesis and enable them to develop their own implementations of system with the help of a generic design.

The remainder of this chapter is as follows. Section 4.1 provides in detailed information on the design process of the computing cluster. The generic design generated for the setup and assembling of the hardware is described here. Section 4.2 explains the design process of the framework. This section provides the general and generic idea for the system that needs to be implemented to be order to full-fill the aims of this thesis. Section 4.3 provides the summary of the chapter and links it to

Chapter 5.

4.1 Cluster

The first step in designing a computing system is the setup of the hardware. For the purposes of this study, we setup our own cluster using individual hardware. As discussed in Chapter 2, there are few basic installations needed to get the computing system up and running. These include the individual computing nodes (Processors - Section 2.2.1), operating system on them (OS - Section 2.3.1), inter node communication medium (Hardware - Section 2.2) and the software which facilitates the inter-node communication (MPI - Section 2.3.2). This section provides information on the design process used while setting up of the cluster for computing.

Based on the requirements and past designs of other researchers, cluster setup as shown in Figure 4.1 was chosen to be implemented for cluster computing.

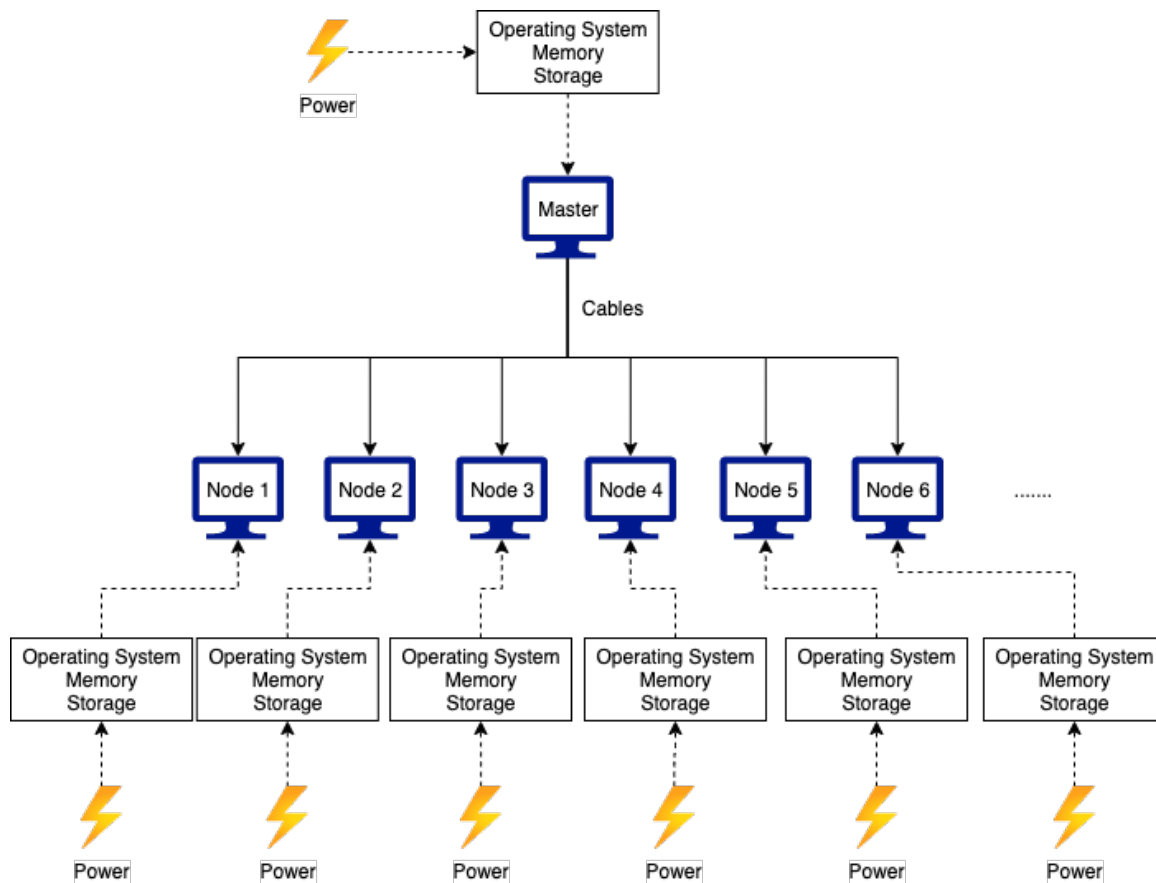


Figure 4.1: Proposed Design for the Cluster hardware

Individual nodes

Cluster computing system comprises of a number of inter-connected computing platforms. Section 2.2.3 describes the many models used for cluster computing. The master-slave model used in our design includes asymmetric communication where one device (master) controls one or more other

devices (slaves) and acts as a communication hub. Each node has its own operating system, memory to compute, storage systems and individual power sources. The master harnesses the computing power of individual nodes to perform computation faster.

Inter-node communication

Inter-node communication can be achieved wired or wireless depending on the type of cluster to be developed (See 2.2.3). Note that, in the design, inter-node communication is achieved using cables. This can also be achieved using wireless connectivity but using wired connection provides many benefits such as speed, security, reliability and efficiency. Traditionally, most of the inter computer connections are done using Ethernet cables. They are used because of their high speed (up to 10Gbps), less prone to hacker attacks, no environmental interruptions and very low amount of power.

4.2 Framework

In the previous section we described the design of developing and setting up the hardware components for computing purposes. In this section we will describe the design process used to develop the framework which will be used to answer the research questions described in 2.5. Based on the requirements outlined in Section 3.6, the functionalities of the framework can be identified and implemented accordingly to full-fill the requirements completely. The functionalities that are important and are compulsorily needed to be implemented are described in this section and explained on how they are achieved. The high level working of the framework can be described as given in Figure 4.2

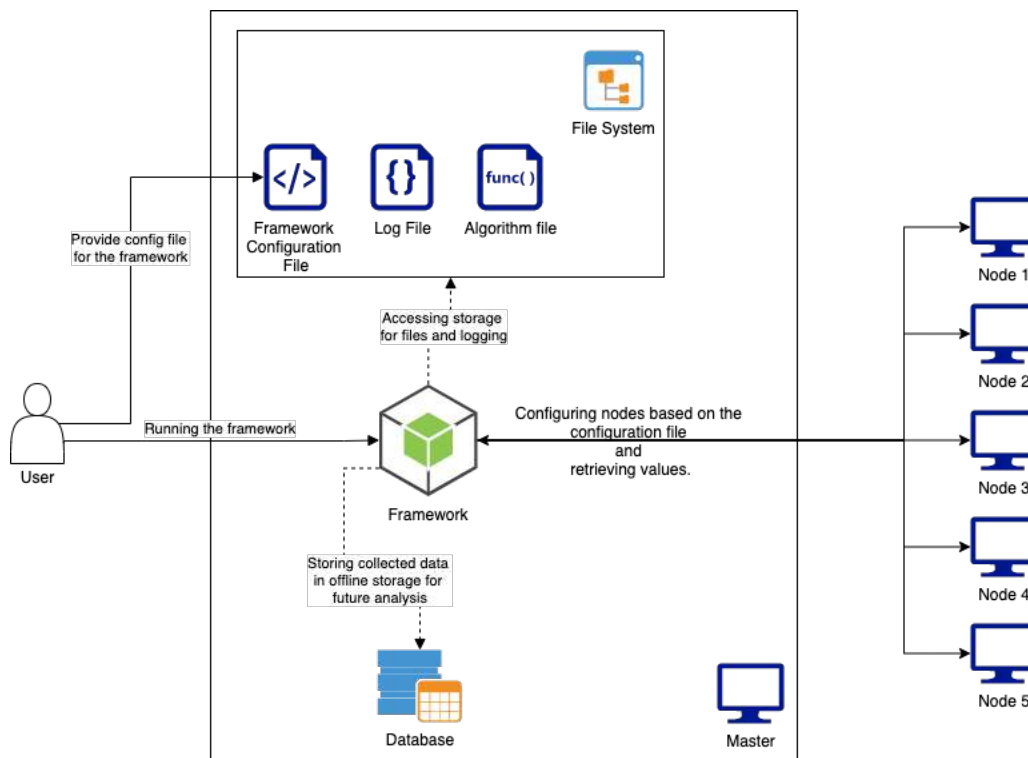


Figure 4.2: Proposed Design of the Framework

4.2.1 User Responsibilities

The user is responsible for running the framework and providing the framework with correct files and permissions required for it to perform the given tasks.

- Configuration File

The configuration file is responsible for providing the framework with most of the information needed for it to run successfully and complete its operations. It includes all the information such as the dependencies, logging, algorithm file details, cluster configurations to run the algorithm on and other important connection details such as IP addresses or login credentials.

- Running the framework

As shown in the Figure 4.2, the framework needs to be run on the master node where it will use the configuration file to complete initialization. It will then configure the whole cluster and get it ready for computing. After, the cluster is configured, the framework can then use the configuration file to run the experiments on the cluster and process the results. The user needs to start the cluster and debug any issues that the cluster might show in its initial setup. After the issues are resolved the cluster will begin the computation and will showcase the results whenever they are available.

4.2.2 Framework Functionalities

Setting up the cluster

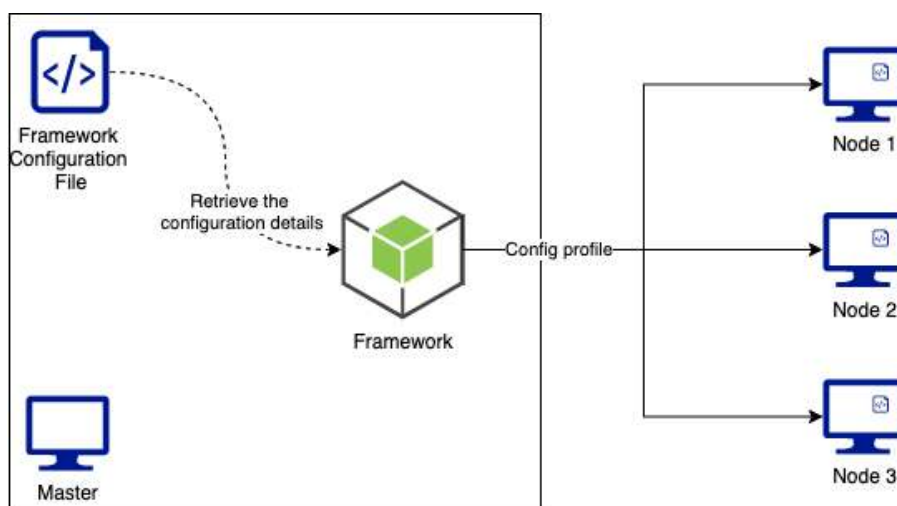


Figure 4.3: Proposed set up of nodes in the cluster

The framework needs to be able to automatically set up the cluster and its nodes for computation (Requirement R4). This can be achieved by using the MPI protocols during the run time (Refer Figure 4.3). The user needs to provide the framework with the preferred configuration using the configuration file. This is an optional requirement for the framework and by default, the framework

will run the computation on all the configurations possible to provide a wide variety of data for better analysis. Once the nodes receive their particular configurations, algorithms and collection of data can start.

Splitting the problem

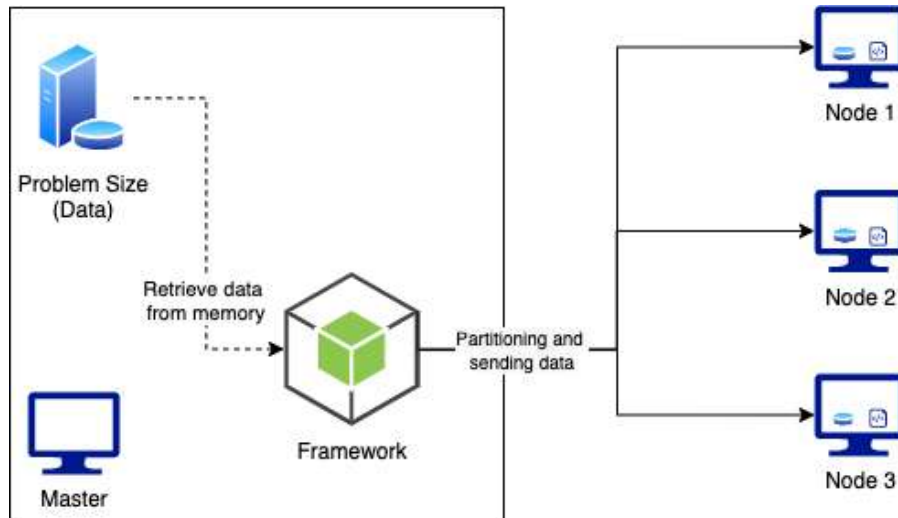


Figure 4.4: Splitting of data for cluster computing

After the nodes are configured for computation, the next step is the actual computation (R5 and R6). This includes providing each node with their individual slice of data to compute on. This can also be achieved through MPI protocols. The big data on the master node is sliced into smaller chunks so that each nodes has a small part to compute and return the results to master. As discussed in Chapter 2, this is the main essence of parallel computing - to reduce the computation time and increase performance by reducing the work load on individual nodes by distributing the overall computing work across number of nodes.

Running the algorithm

After the cluster is setup and each node provided with their part of problem to compute, we need to provide the nodes with the actual set of instructions to perform the computation (algorithms)(R5 and R6). The user needs to provide the actual algorithm file and declare its usage in the configuration file. The configuration can include specifications such as location of the file, parameters needed for algorithm, the dependencies, the output format, and other needed factors which are important in running the experiments. The framework should send the algorithm file to each node and the nodes are supposed to run the algorithm as per the specifications provided by the framework (Refer Figure 4.5).

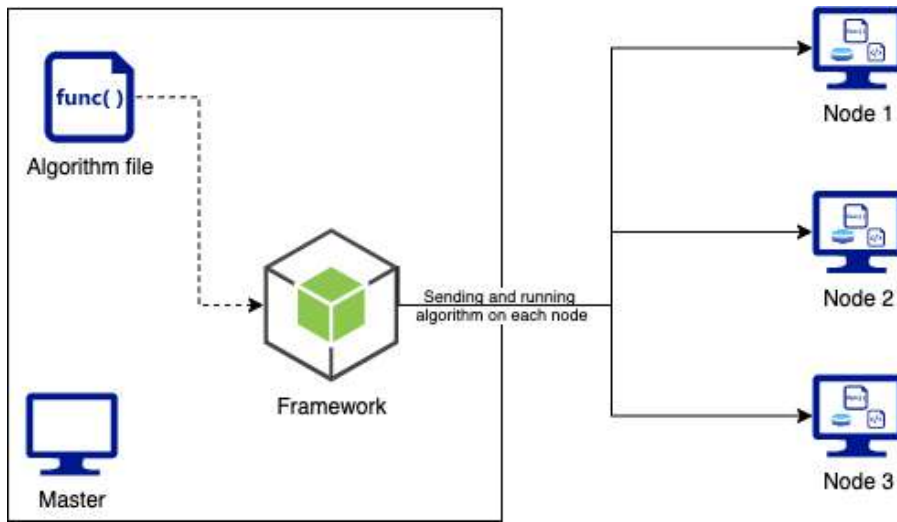


Figure 4.5: Running of an algorithm in a cluster

Collecting energy data

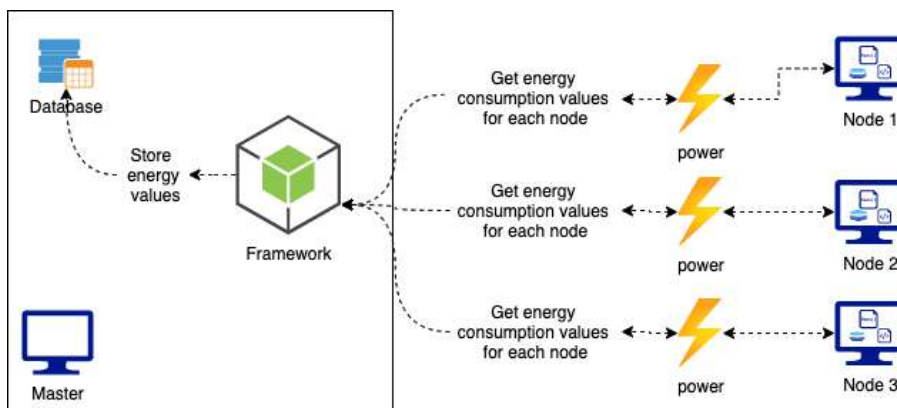


Figure 4.6: Collecting energy consumption values

The requirements R7 and R8 can be achieved through implementing a functionality to monitor and log the energy consumption of each node. Measuring the energy consumption of a particular node is hard to achieve using purely software methods. For accurate measurements there needs to be a hardware interface installed at the power source which can monitor and send the energy consumption data to the framework. This can be achieved through a number of steps. A digital power source with sensors can help in collection of the energy data (Refer Figure 4.6).

Storing the algorithm output in the database

Collection of data from algorithm is an important aspect of any framework. Algorithms can have many different outputs which can range from just a log entry to an important result of a computation. The framework cannot and should not restrict users on the output their algorithm provides and hence, a framework should be able to accommodate whatever output the algorithms generates. This functionality will help in achieving requirements R3, R9 and R1. The data collected through this medium is the

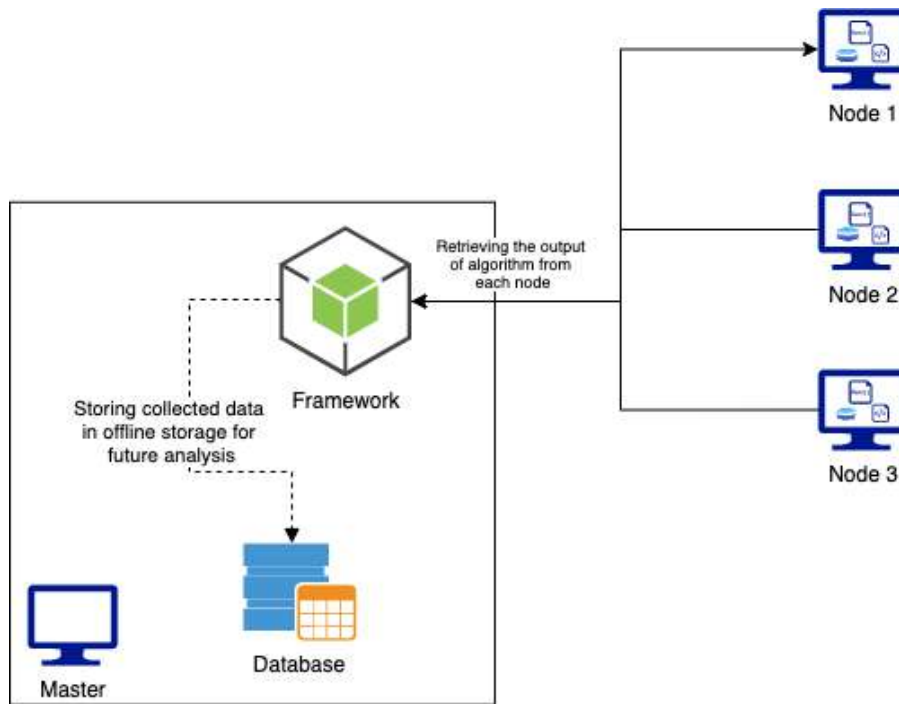


Figure 4.7: Collecting and storing algorithm output data

raw data generated by a particular algorithm and can help the researcher in monitoring, evaluating or debugging the algorithm when needed.

Exporting the data

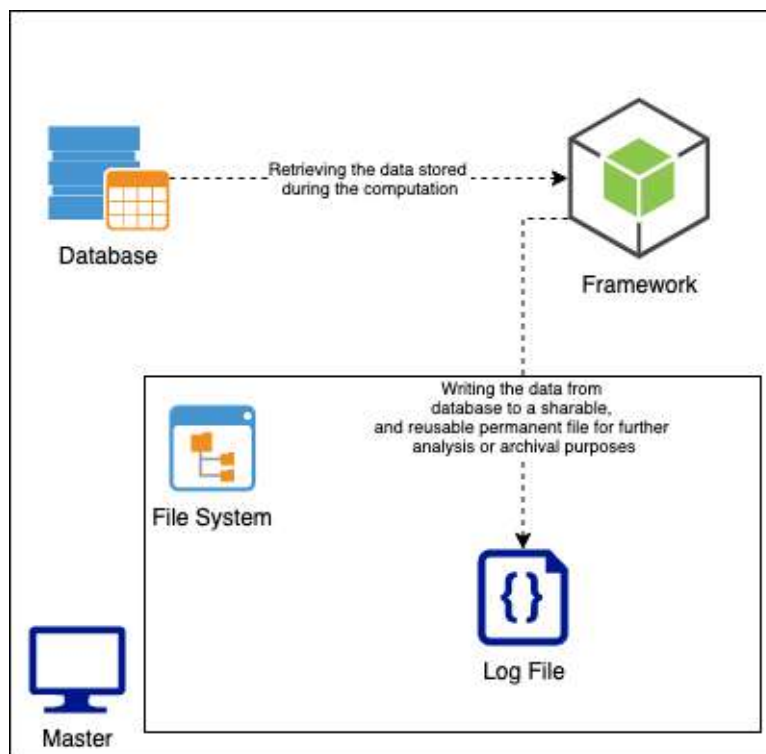


Figure 4.8: Exporting collected data to a file

Data exporting is a great way to permanently store and archive the data. It can also lead to future improvements by implementing new functionality to use this data. The framework stores the data from the computation into a local database. The user can then export that data into a compatible file format which can be used to analyse the data further or add new functionalities to the data. This functionality is important to the goals of this thesis as this allows the researcher more control over the data and freedom to analyse the data in number of ways. Requirement R9 can be full-filled through this design and this can help in achieving Requirement R2 as well.

4.3 Summary

This chapter provides an abstract idea developed to meet all the requirements in Section 3.6. As discussed, two-fold implementation was required to be able to provide answers to the research questions. Section 4.1 provided a generic design for the cluster setup. It shows the process in which individual nodes can be connected to each other and used for the purposes of this study. Section 4.2 provided the design for the framework. Functionalities of the framework and the way to achieve them were also shown. The chapter concluded with a general idea which will be used to develop an implementation in the further chapters.

CHAPTER 5

IMPLEMENTATION

A two-fold implementation was needed to achieve the goal of the thesis. As discussed in Chapter 4, high level conceptual idea was designed which would help in answering the research questions set out for this work. Based on Section 4.1 and Section 4.2, both hardware and software implementations were performed to achieve the necessary design. Both implementations had their own sets of requirements, characteristics and set-backs. This chapter provides the steps undertaken to implement the design, justification for those steps and the issues encountered while doing the same. The aim of this chapter is to provide the reader with the understanding and means to exactly replicate the work performed in this thesis.

The first implementation was that of the design and development of the required computing hardware system. For the purposes of this thesis, a cluster of 8 RPi3B+ was designed and implemented based on the design in Section 4.1. Steps undertaken while developing the cluster is explained in-depth in Section 5.1. The thought-process behind choosing the node computers and other important decisions required while developing the framework have also been described in Section 5.1. Figure 5.1 shows the final cluster setup achieved after the implementation of the design in Section 4.1.

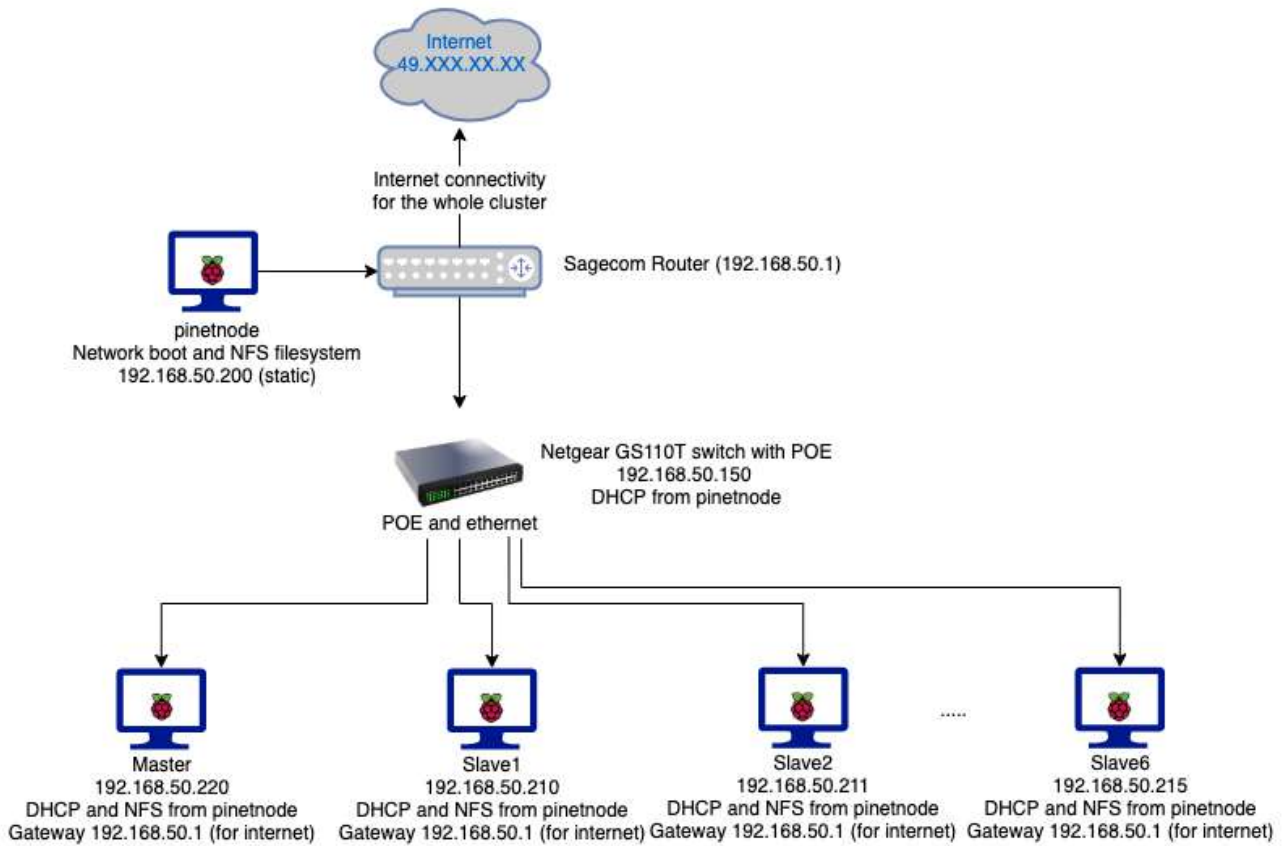


Figure 5.1: Implementation of the proposed cluster hardware

The second implementation was that of a software framework based on the design set out in Section 4.2, full-filing the requirements in Section 3.6 and achieving the research questions in Section 1.1. This implementation is the main goal of this thesis and was designed and implemented with all the aims of this thesis in mind. More information on the design and the process of development is described further in Section 5.2. Figure 5.2 shows the final high-level workings of the framework after the implementation of the design in Section 4.2.

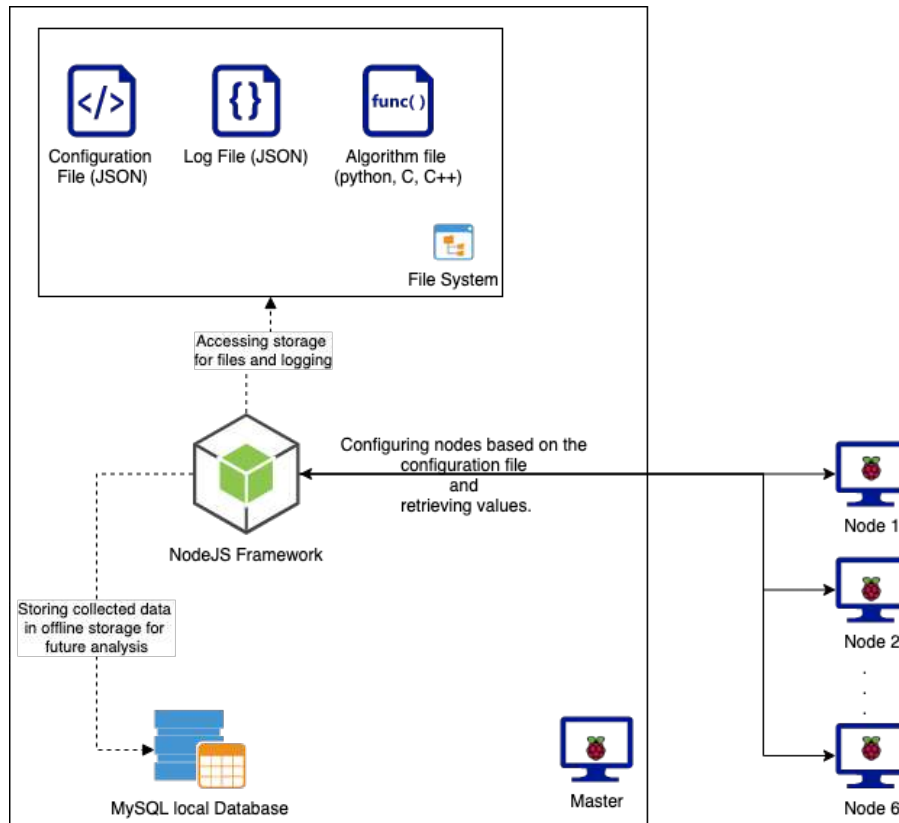


Figure 5.2: Implementation of the proposed framework

5.1 Implementation of Computing Cluster

The aim of the thesis is to develop a framework that facilitates the evaluation of different cluster configuration focusing on their energy consumption. The first step in achieving this is to obtain a computing platform or, in this sense, a cluster. This section describes the process of development of the cluster and the decisions undertaken while its development.

5.1.1 Node Computer

As described in Chapter 2, a cluster comprises of a number of nodes inter-connected to obtain an increase in performance. These individual nodes are computers with their own memory, storage and other characteristics. As the main aim of this thesis is to focus on the energy consumption of a cluster, individual nodes which performed comparatively well for low energy consumption were ideal for nodes. After extensive research in the field of cluster computing (Section 2.2.4) and benchmarking (Section 2.4.3), it was decided to use a Single Board Computers for the development of cluster. SBC performed reasonably well for their low energy consumption (Cloutier et al., 2016). For the purposes of our study, RPi was chosen as the SBC for cluster as it has been a center of attraction due to its advancements and characteristics from last few years (Basford et al., 2020). Many researchers in the similar field have also used RPi over other SBC boards for their computing needs (Saffran et al., 2016; Cloutier et al., 2016; Basford et al., 2020; Qureshi and Koubaa, 2017; Srinivasan et al., 2018; Schot,

2015; Markovic et al., 2018; Rahmat et al., 2019; Tso et al., 2013).

Raspberry Pi foundation is a UK based charity responsible for the manufacturing and development of Raspberry Pi boards. During the writing of this thesis, the latest RPi released by Raspberry Pi foundation was the RPi4 Model B. The specifications of the board includes Broadcom BCM2711, Quad core Cortex-A72 processor. The user can choose from different variants of RPi4B model based on the memory (2GB, 4GB or 8GB). Due to availability and constraints of buying ability, RPi4B could not be used for the cluster development process. Also, RPi4B has been known to exhibit a lot of bugs along with corruption of system after extended use. Support or compatible libraries are also not available for RPi4B due to it being announced recently. The next powerful board developed by Raspberry Pi foundation - RPi3B+ (Figure 5.3) was chosen. It features 1GB RAM with Broadcom BCM2837B0 quad-core A53 processor. RPi3B+ is a lot stable as compared to RPi4B and has support for maximum libraries used for computing. It contains Gigabit Ethernet connection (Section 4.1) for faster data transmission.



Figure 5.3: Raspberry Pi3B+ (Source: www.raspberrypi.org)

5.1.2 Retrieving Energy consumption values

The ability to be able to monitor the energy consumption of a particular node or the cluster as whole is an important aspect of our system. All the past researchers installed extra hardware such as USB energy monitor (Saffran et al., 2016), monitoring hardware between power source (Cloutier et al., 2016; Qureshi and Koubaa, 2017) and the nodes and theoretical calculations (Rahmat et al., 2019) to obtain the energy consumption values for a computing node. For the purposes of this study the power monitoring sensors installed in the industry grade fully managed switch - Netgear GS110TP were used to monitor the energy consumption. Figure 5.4 shows the chosen switch for our cluster design.



Figure 5.4: Netgear GS110TP Fully Managed Switch (Source: www.netgear.org)

Following three important features were used to determine which switch to choose for our cluster:

1. Software accessible energy consumption values

This was the most important factor in deciding the switch to be chosen. The framework is a software based and it needs to be able to access the energy consumption values easily through executable scripts. After extensive research, it was found out that GS110TP switch had a telnet server residing on it which can be exploited to retrieve values from the switches sensors. Using the telnet server, all the configuration and generated data can be controlled and read through from any local computer connected to it. This was the main deciding factor while choosing the switch as this features meant that the framework can be truly automated and can be advanced to include newer features in future.

2. Power Over Ethernet support (PoE)

Managed switches contain Ethernet ports which support power supply through them. This also enables the switches to log the energy consumption values locally as seen in the previous section. GS110TP contains 8 PoE enabled ports which can provide a stable of 7.5 Watt of energy to the connected computers. Ideal energy consumption of a RPi is only 2 Watt and hence, the switch provided more than enough energy to power all the nodes. PoE also reduced the cluttering required to power the nodes. PoE splitter is required to dissipate the power to the nodes and the data transmission to the ethernet port on nodes. Figure 5.5 shows the PoE splitter used for our cluster.

3. Web Interface for easy management of nodes

GS110TP managed switches contain a web server on it which provides the user with a easy and configurable web interface to be able to interact with the switch. Figure 5.6 shows the web interface option which shows the energy consumed by each port. Apart from this, the web interface also provides many other options like turning power on and off remotely, monitoring the traffic, security, authentication, etc. This feature is very useful when the user does not have physical access to the switch and wants to remotely config it or debug an issue.



Figure 5.5: PoE Splitter (Source: <https://core-electronics.com.au>)

PoE Port Configuration

PoE Port Configuration													
PORTS													
Port	Admin Mode	Max Power	Priority Level	Detection Mode	Class	Timer Schedule	Output Voltage (Volt)	Output Current (mA)	Output Power (Watt)	Power Limit Type	Power Limit (mWatt)	Status	
<input type="checkbox"/> g1	Enable	16.2	Low	802.3af 4point Only	3	None	47	64	3.0	Class	15400	Delivering Power	
<input type="checkbox"/> g2	Enable	16.2	Low	802.3af 4point Only	3	None	47	77	3.600	Class	15400	Delivering Power	
<input type="checkbox"/> g3	Enable	16.2	Low	802.3af 4point Only	3	None	47	64	3.0	Class	15400	Delivering Power	
<input type="checkbox"/> g4	Enable	16.2	Low	802.3af 4point Only	3	None	47	66	3.100	Class	15400	Delivering Power	
<input type="checkbox"/> g5	Enable	16.2	Low	802.3af 4point Only	3	None	47	65	3.0	Class	15400	Delivering Power	
<input type="checkbox"/> g6	Enable	16.2	Low	802.3af 4point Only	3	None	47	66	3.100	Class	15400	Delivering Power	
<input type="checkbox"/> g7	Enable	16.2	Low	802.3af 4point Only	3	None	47	64	3.0	Class	15400	Delivering Power	
<input type="checkbox"/> g8	Enable	16.2	Low	802.3af 4point Only	0	None	0	0	0.000	Class	15400	Searching	

Figure 5.6: Web Interface for management of the switch

5.1.3 Cluster Setup

As shown in Figure 5.1, the final cluster implementation involved of eight individual RPi3B+ connected to each other using Ethernet cables, switch and router. Exact process of installation and setup of cluster for computing is documented in this section.

Seven of the eight RPi3B+ are computing nodes while one RPi3B+ acts as a networking configuration node. It provides other seven with the required network configurations necessary to facilitate the cluster computing.

Operating System

Operating System is the most basic component of any computer. RPi3B+ only provides a bare hardware. Operating system installed on it will facilitate the use of processors and other components

for computation purposes. As the aim of this thesis and any computing cluster is to maximize the performance from the cluster, a lightweight and highly optimised Operating System needs to be chosen which provides the most resources for computing. Official and unofficial operating system distributions for RPi were compared based on their size, support and reliability(Refer to Table 5.1).

#	Name	Size (GB)	Source
1	Raspberry Pi OS	7.37	https://www.raspberrypi.org/downloads/
2	Raspberry Pi OS Lite	1.85	https://www.raspberrypi.org/downloads/
3	Dietpi	1.07	https://dietpi.com
4	TinyCore Linux	0.18	http://tinycorelinux.net/downloads.html

Table 5.1: RPi Operating System Distributions

DietPi was chosen to be used as an Operating System for our system. DietPi is a Debian-based Linux distribution, primarily developed for single board computers and hence it is highly optimized to gain maximum performance. Also, DietPi comes with an Software Manager that can be used to install software easily. Unlike the official Operating System which includes a lot of unneeded software, DietPi only comes with the necessary packages installed. Also, DietPi works on Command Line Interface, due to which resources used by a GUI based OS are used for computing purposes.

Networking

Traditionally, RPi's are booted by flashing the image of Operating System onto a SD card and then inserting the SD card into RPi. Due to the lightweight nature of DietPi and other benefits listed below, it was decided to use network booting for each nodes in a cluster. Network booting basically means that booting a computer from a network rather than a local drive. There is a server and a client in network booting. Client requests server for the boot instructions. Server then assigns a IP address to the client and provide boot instructions over network. Network boot provides three significant benefits over normal boot:

1. Client machine does not need any hardware storage or Operating System installed.
2. Client machine can be restarted remotely in case of system failure.
3. Client configuration can be easily changed. Just need to change once on the server and after restart all the clients have the same configuration.

As a part of network booting, clients are provided an IP address and boot instructions. Dynamic Host Configuration Protocol (DHCP) server are installed on the server for allocation of IP addresses and to provide information to each node in a cluster for efficient communication between endpoints. Generally, in network booting, the boot files and the root file-system are all stored on the server and

the clients use Network File System Protocol (NFS) to access the files and folder on the server. This is useful where the changes to the files on server are required to be updated on clients as well.

Figure 5.7 shows the very high level idea of network booting in our system.

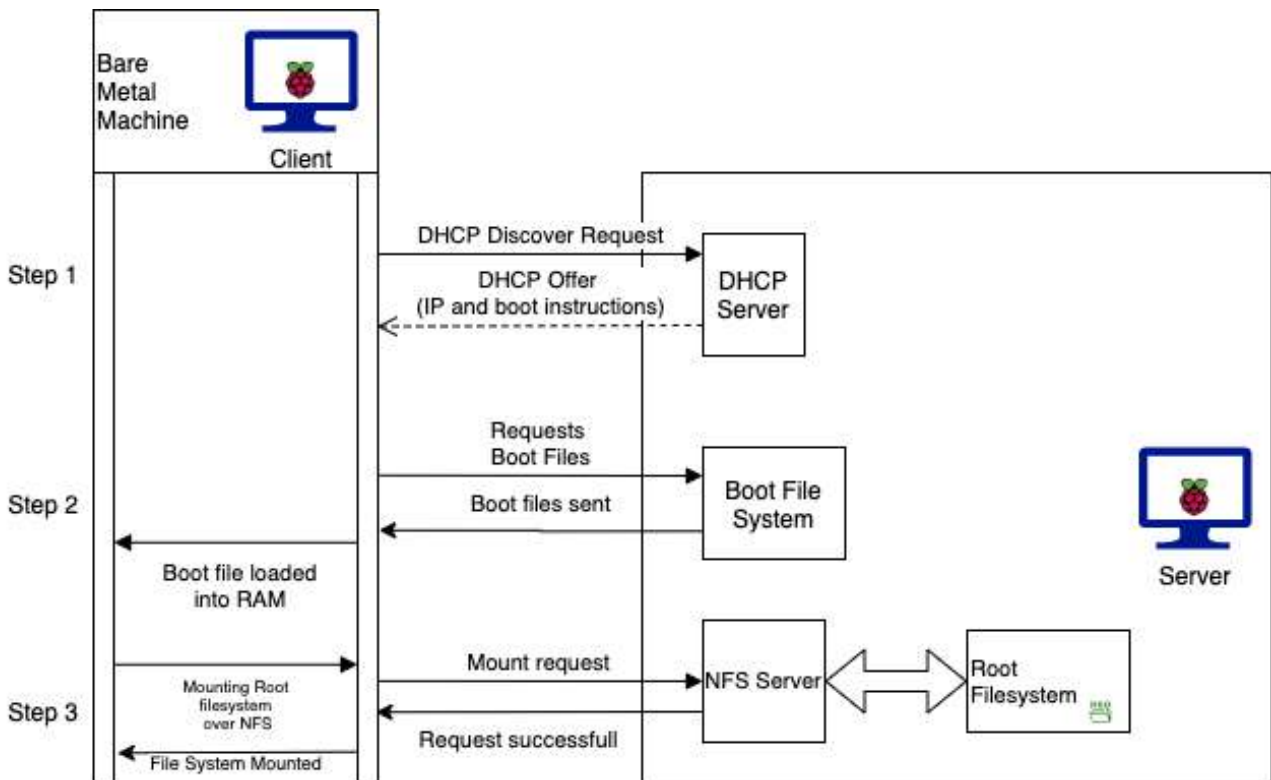


Figure 5.7: Network Booting Steps

Software Setup

Once the file-system is mounted and boot completed, next step included installing a Message Passing Interface (MPI) library which is necessary for parallel computing. Open MPI (<https://www.open-mpi.org>) was used for message passing and process handling in our system. Open MPI supports the latest MPI-3.1 standards along with dynamic process spawning. This is very important functionality for our system as it will allow the system to remotely spawn processes (execute algorithms) on computing nodes. It is network fault tolerant and supports all types of networks. A simple “helloworld” program was executed using MPI to check its correct installation on all the nodes.

5.1.4 Final Setup

The cluster setup actually implemented can be seen in Figures 5.8, 5.9 and 5.10. Due to lack of hardware acquisition, no sturdy mounting hardware was available. Hence, six RPi3B+ were mounted on top of each other using screws. Two RPi3B+ were mounted together to distinguish them from the cluster. The top RPi is the one providing the network boot and file-system to other nodes (Denoted

by B) whereas the bottom one is the master node (Denoted by M). The six RPi's were purely used for computing purposes (Labelled as S1, S2, S3, S4, S5, S6).

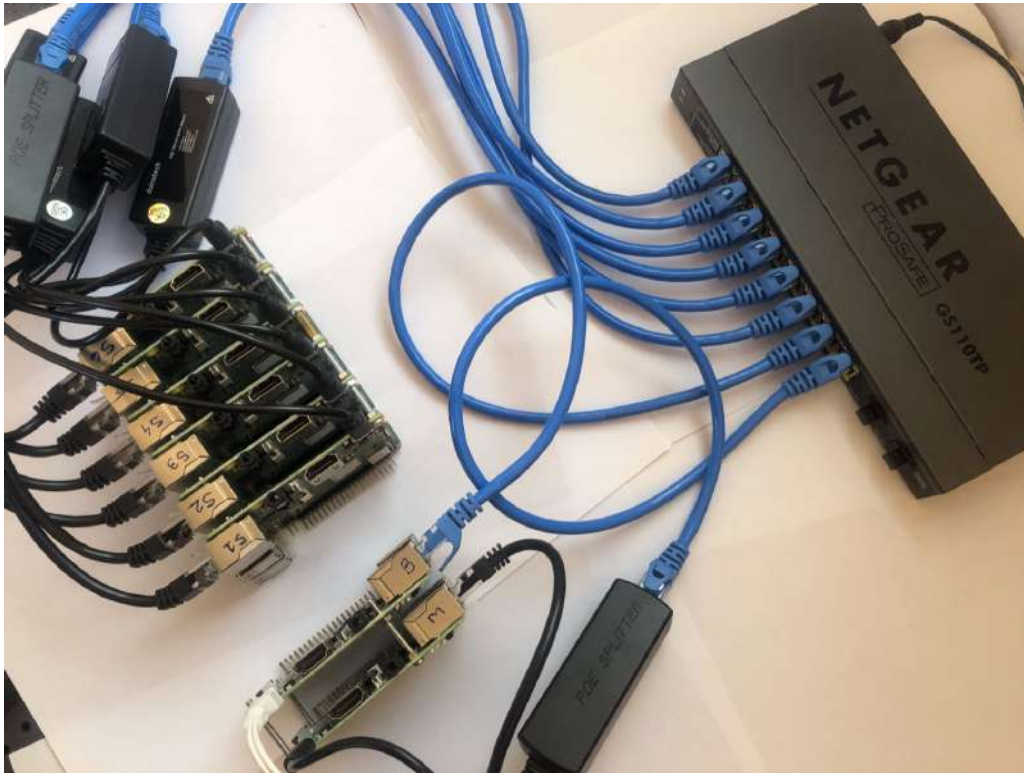


Figure 5.8: Final Set Up of the Implemented Cluster



Figure 5.9: Final Setup - Netboot and Master node



Figure 5.10: Final Setup - Slave nodes

5.2 Implementation of Framework

A fully set up cluster is developed in which individual nodes are booted, have IP addresses and can effectively communicate with each other. Figure 5.2 showcases the high level workings of the framework. Once the cluster was set up, implementation of the framework started. Each independent and individual functionality of the framework was developed separately and then combined in the end. This section describes the process of development of the framework and its different components.

5.2.1 Framework language

The development of any software framework starts by choosing the programming language to develop it in. For our study, JavaScript was chosen as the preferred language due to its cross-compatibility and wide use. JavaScript is mainly used to generate interactive web-pages. NodeJS is a cross-platform run-time environment that executes JavaScript outside of web browsers. For this study, the framework was decided to be developed so that it can be accessed via the Command Line Interface (CLI). As NodeJS also supports development of web based GUI applications, it won't be hard to develop a GUI for our CLI based framework. The framework was developed and evaluated using NodeJS environment because of the following reasons:

1. Synchronisation

NodeJS is a asynchronous platform which does not wait of operations to complete before starting new operations. Instead it uses callbacks. The callback function is called when the block of code finishes execution. NodeJS also supports synchronous methods which block the execution of code until the result from a function is not received. This is particularly important feature as this allows the framework to execute a function in background while blocking another function in foreground.

2. Integration with Computer System Utilities

NodeJS have a wide base of modules which are developed to implement a particular functionality or to improve on another one. NodeJS has modules which can interact with the Command Line Interface (Module: `child_process`), read and write JSON files (Module: `edit-json-file`), connecting to servers (Module: `telnet-client`, `mysql`), remote management of computers (Module: `node-ssh`), etc. All these modules were used to develop each individual functionality of the framework.

3. Widely supported and used

NodeJS have been widely used as a server-side environment to develop JavaScript applications for web-pages. It supports almost every architecture (MAC, Linux, Windows) and is heavily backed by the community to find and solve issues.

5.2.2 Configuration file

The configuration file is most important part of the framework. The framework is dependent on the configuration file for each part of its functioning. An extract from the configuration file is given below (Refer Appendix A for full configuration file). This section explains the different fields of the configuration file and how each of them affect the workings of the framework.

```

"__comment":
    "Configuration file for FEPAC",
"app": {
    "hostfile_folder":
        "algorithms/hostfile",
    "max_nodes": 7,
    "max_threads": 4
    "data_file": null,
    "start_node": null,
    "start_thread": null
}
"switch_telnet": {
    "IP": "192.168.50.150",
    "port": 60000,
    "login": "admin",
    "password": "password",
    "interval_to_call": 1000
},
"db": {
    "user": "root",
    "password": "qazwsxedc"
},
"algorithm": {
    "matrix_multiplication": {
        "language": "python3",
        "dependency": "mpi4py numpy",
        "parameters": "3000 3000",
        "command": "python3 multiply.py"
    },
    "kmeans_c": {
        "language": "C",
        "dependency": "GCC",
        "parameters": "288000 48",
        "command": "./kmeans_c.exe"
    }
}

```

The “app” field describes the parameters that the framework uses to configure the target cluster for computing. “hostfile_folder” is the location of host file, containing the IP addresses for each node in a cluster. “max_nodes” and “max_threads” define the maximum number of nodes and threads to be used on each node respectively. “data_file” is the location of the file where the temporary log of the framework is to be stored. “start_node” and “start_thread” defines the exact configuration of the cluster to execute the experiment on.

The “switch_telnet” field defines and declares the information needed for the framework to connect to the telnet server located on the switch. As discussed in Section 5.1, the energy consumption values for the cluster are obtained from the telnet server located on the switch. “interval” is the time delay between two consecutive retrievals of data.

The “db” field defines the credential required by framework to connect to the local MySQL database. The database is used to store log information and results from computations.

The “algorithm” field defines the parameters and information required by the framework to effectively execute the algorithm based on the cluster configuration declared in “app” field. “Language” defines the programming language in which the algorithm is written. “Dependency” helps the framework in understanding the required libraries for that particular algorithm. The framework attempts to install these libraries if they do not exist. “Parameters” field define the actual values to be passed to the algorithm while execution. The framework uses the “command” field to execute the algorithm on different nodes

5.2.3 Development of Individual Functionality

This section explains in detail few of the important functionalities of the framework. Three main functionalities were identified for the framework during its design.

Retrieving energy values

As explained in the Section 5.1, there was no external hardware installed to measure the energy consumption of the cluster. It was done through retrieval of values from the internal sensors in the switch via a telnet server. The telnet server on the switch contains a lot of data regarding the working of the switch, devices connected to it, monitoring of those devices, etc. The main issue was that the telnet server provided information as a long buffer (a very long string) with pairs of fields and values. As shown in Figure 5.11, the location of the values in the buffer was found out using trial and error methods and the values were then sliced from it and stored into an array. MySQL queries were then used to store the values into local database. Figure 5.12 shows the format in which the values are stored in the local database.

```
var pwr = [];

pwr[1] = parseFloat(res.slice(1163, 1223).slice(15, 18))
pwr[2] = parseFloat(res.slice(1224, 1284).slice(15, 18))
pwr[3] = parseFloat(res.slice(1285, 1345).slice(15, 18))
pwr[4] = parseFloat(res.slice(1346, 1406).slice(15, 18))
pwr[5] = parseFloat(res.slice(1407, 1467).slice(15, 18))
pwr[6] = parseFloat(res.slice(1468, 1528).slice(15, 18))
pwr[7] = parseFloat(res.slice(1529, 1589).slice(15, 18))

var qinsertpwr = 'INSERT INTO ${tb_name}
(timestamp, description, p1, p2, p3, p4, p5, p6, p7) VALUES
(${Date.now()}, ' ', ${pwr[1]} , ${pwr[2]} , ${pwr[3]} ,
${pwr[4]} , ${pwr[5]} , ${pwr[6]} , ${pwr[7]})';
```

Figure 5.11: Collection of energy consumption values through code

```
MariaDB [mytestdb]> select * from test;
```

timestamp	description	p1	p2	p3	p4	p5	p6	p7
1599371624774	Start	NULL	NULL	NULL	NULL	NULL	NULL	NULL
1599371627814		3	3.1	3	3	3	3.5	2.6
1599371628812		3	2.8	3	3	3	3.5	2.6
1599371629813		3	2.8	3	3.5	3.1	3	3
1599371630814		3	2.8	3	3.5	3.1	3	3
1599371631814		3.5	3.1	3	3.5	3.1	3	3
1599371632816		3.5	3.1	3.2	3	3.1	2.9	3
1599371633817		3.5	3.1	3.2	3	3.1	2.9	3
1599371634817		3	3.1	3.2	3	3.1	2.9	3
1599371635819		3	3.1	3	2.7	3	3.7	2.7
1599371673858		3	2.8	3	2.7	3.1	3	3.7
1599383262098	Start	NULL	NULL	NULL	NULL	NULL	NULL	NULL
1599383265141		2.6	3.6	3.5	3	3.2	3	2.9
1599383266139		2.6	3.6	3.5	3	3.2	3	2.9
1599383267140		3.3	3.1	3	3.1	3	3.1	3
1599383268141		3.3	3.1	3	3.1	3	3.1	3
1599383269142		3.3	3.1	3	3.1	3	3.1	3
1599383270142		3	2.8	2.7	3.8	2.7	3.1	3
1599383271142		3	2.8	2.7	3.8	2.7	3.1	3
1599383370610	Start	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 5.12: Energy Values stored in local MySQL Database

Running experiments

The experiments are executed by the framework in accordance to the configuration parameters by the researcher in the configuration file. The framework retrieves the values from the configuration files shown in Section 5.2.2 and runs the experiments accordingly. The framework uses the MPI to configure the cluster as shown in Figure 5.13. The framework then spawns a pseudo-terminal with help of remote management module on each node and executes the command provided in the “command” field. A simple example of “helloworld” program being executed on 1 Node 4 Threads is shown in Figure 5.13. The hostfile parameter is a file that contains all the IP addresses or host-name of other nodes and also declares the number of threads available on each in the cluster.

```
pi@boot:~/mpi $ cat hostfile
localhost slots=4
master slots=4
slave1 slots=10
slave2 slots=10
slave3 slots=10

pi@boot:~/mpi $ mpirun -np 4 --hostfile hostfile python3 helloworld.py
Boot: Hello, World! I am process 2 of 4 on boot.
Boot: Hello, World! I am process 3 of 4 on boot.
Boot: Hello, World! I am process 1 of 4 on boot.
Boot: Hello, World! I am process 0 of 4 on boot.
```

Figure 5.13: Using MPI to execute algorithm across multiple nodes

Exporting data

As shown in Figure 5.12, the data stored in the database consists of instantaneous values of energy consumption. Analysis of such data is tedious and it was decided that for a particular algorithm, average value of energy consumption would be calculated based on the start and end time of the algorithm. The export functionality of the framework automatically does the averaging of data. The researcher can export the raw data as well if required, but by default the framework exports the data in format as shown in Figure 5.15. The data is exported in JSON format so that it can be used in other analysis software easily. The first field in the exported file is the algorithm name and then the configuration of the cluster. Second and third field denotes the number of nodes and threads being used respectively. On fourth level is the timestamp - duration for which the algorithm was running and the average power values of the cluster for that duration. p1-p7 denotes the individual nodes where p1 is the master node and other six are computing nodes.


```

"opencv": {
  "1": {
    "2": {
      "timestamp": {
        "start": "1598965940040",
        "end": "1598966009730"
      },
      "pwr_avg": {
        "p1": 4.239130403684533,
        "p2": 3.1246376175811323,
        "p3": 2.947826091794,
        "p4": 3.0826086894325586,
        "p5": 2.9565217391304346,
        "p6": 3.0956521241561226,
        "p7": 3.056521747423255
      }
    },
  },
  "5": {
    "1": {
      "timestamp": {
        "start": "1599003659274",
        "end": "1599003754051"
      },
      "pwr_avg": {
        "p1": 3.3585106464142496,
        "p2": 3.4968085035364678,
        "p3": 3.258510635254231,
        "p4": 3.407446808003365,
        "p5": 3.467021287755763,
        "p6": 3.015957403690257,
        "p7": 3.0212765982810486
      }
    },
  },
}

```

Figure 5.14: Exported data in JSON format

5.2.4 Final Setup

In this section, few snippets of the final framework are included.

```
FEPAC
```

Welcome to Framework for Evaluation of Parallel Algorithms on Clusters (FEPAC).

Please select your option:

- 1) Test MySQL connection
- 2) Test Telnet connection
- 3) Run FEPAC
- 4) Exit

Answer: 3|

```
FEPAC
```

Welcome to Framework for Evaluation of Parallel Algorithms on Clusters (FEPAC).

Please select your option: Run FEPAC

FEPACK menu:

- 1) MySQL <-> Telnet connection (Power)
- 2) Stop mysql insert
- 3) Get avg
- 4) Run algorithm
- 5) Generate graph
- 6) Go Back

(Move up and down to reveal more choices)

Answer: 4

```
FEPAC
```

Welcome to Framework for Evaluation of Parallel Algorithms on Clusters (FEPAC).

Please select your option: Run FEPAC

FEPACK menu: Run algorithm

Algorithm menu:

- 1) matrix_multiplication
- 2) kmeans_c
- 3) OpenCV
- 4) opencv_filter
- 5) opencv_hash
- 6) kmeans_python

(Move up and down to reveal more choices)

Answer: |

Figure 5.15: Final Implemented Framework and its Menu

5.3 Summary

The implementation of framework which can help in evaluation of different parallel algorithms on different cluster configurations is presented in this Chapter. A two-fold implementation undertaken for this thesis is showcased. Section 5.1 describes in detail the steps in implemented to achieve a final cluster of 8 RPi3B+ nodes based on design in Section 4.1. Section 5.2 includes the implementation of framework outlined in Section 4.2. This chapter aims to aid researchers in replicating our implemented work and achieve similar output.

CHAPTER 6

EVALUATION

Chapter 5 provides the implementation of the design outlined in Chapter 4. Both the hardware and the software implementation of the work in this thesis has been provided. This chapter evaluates the designed system based on the methodology in Chapter 3. The system is evaluated using two different methods - Qualitative and Quantitative.

Qualitative evaluation (Section 6.1) focuses on the contribution of our proposed system in terms of its quality. The section evaluated the designed system based on a comparative study with the requirements in Section 3.6. Qualitative evaluation is achieved by reviewing the requirements in Section 3.6 with relation to the achieved functionality of the system. This is then used to discuss and co-relate the findings back to the research questions of this thesis. For the purposes of this study, the system is considered to be successful in qualitative evaluation when it achieves the requirements set-out in Section 3.6.

Quantitative evaluation is done by running a series of experiments with the help of the designed system to achieve quantitative data. The main focus of this method of evaluation is to relate the results of these experiments with the requirements and the research questions of this thesis. The results of those experiments are not the main focus of this chapter but rather what those results depict. The experiments are based on the real life computing scenarios including 6 different types of experiments. This will also help the readers in understanding the workings of the system and how to interpret the results in future.

6.1 Qualitative Evaluation

The system is evaluated qualitatively in this section by showing that the system achieves the requirements given in Section 3.6. The impact of each requirement shown and up to what extent the requirements are being met or not met has been discussed.

- R1 The proposed system shall help researchers in deciding the best optimal performance for their energy constraints.

This is a domain level requirement which is closely related to the research questions for this thesis. The system designed meets this requirement by providing a method to support researchers in evaluating different algorithms on different cluster configurations. The system allows a researcher to specify a range of cluster and algorithm configurations to evaluate. After experiments are performed, data can be exported which can then be tabulated and graphed for further analysis. The results from the system can help the researchers in their decision making and identification of best possible scenarios for their given computation.

- R2 The proposed system shall support the identification of bottlenecks for a given computation for a cluster configuration.

The system designed meets this requirement by providing them with the results from the conclusions which can then help them in analysing the data. This analysis can lead to identification of discrepancies in the results, inconsistent data or irregular patterns which all suggests involvement of bottlenecks. When the data is consistent with similar experiments in literature or follows a certain pattern, the researchers can conclude that the bottlenecks are not present or that they are not significant. This analysis of the data will help motivate further research in the field and can help researchers in finding ways to identify and overcome or minimize these bottlenecks for their particular computation.

- R3 The proposed system will allow the specification of a range of repeated and repeatable experiments which varies algorithm parameters and cluster configurations.

The system designed meets this requirement by allowing a wide range of parameters and configurations to be passed while conducting the experiments. A researcher can specify, in the configuration file, exactly how the cluster needs to be configured for computing, how many times to iterate the experiment and also specify parameters for the computation. This will help researchers gain larger control over their experimental setup and the whole experiment can be repeated or reconfigured by editing the configuration file and allowing the system to do the manual tasks.

- R4 The proposed system shall automatically configure the target cluster for the computation needs as defined in the experiment configuration.

The designed system meets this requirement by providing researcher with a method to specify the maximum nodes to use and also the number of threads to be used on each (Refer Figure 6.1). By default the system performs the experiments from one node one thread upto the “max_nodes” and “max_threads” as specified in the configuration file. The system also provides a method to over-ride this functionality. “current_node” and “current_thread” fields specify the system which configuration to configure the target cluster. Using this, a researcher can perform an experiment with specific configuration to get the computing results faster and more specific to their goals.

- R5 The proposed system shall automatically execute different algorithms on different cluster configurations.

```

"__comment": "Configuration file for FEPAC",
"app": {
  "__comment1": "NodeJS code",
  "port": 8000,
  "master_node_ip": "192.168.50.220",
  "folder_location_on_nodes": "~/Honours_Framework",
  "max_nodes": 7,
  "max_threads": 4,
  "__comment2": "Fill this to overwrite the default values used.
Default: Name of algorithm.",
  "data_file": null,
  "current_node": null,
  "current_thread": null
},

```

Figure 6.1: Configuration File extract for target cluster configurations

The designed system meets this requirement by providing researchers a method specify all the algorithms available for the system to run the experiments on. The system chooses one algorithm at a time and executes the algorithm for all the specified cluster configurations. The researcher can then specify if they want to execute any other algorithm. If the researcher want to makes changes to the configuration specification then they have to stop the current execution of algorithm and re-run it after editing the configuration file.

R6 The proposed system shall support multiple algorithms from different languages.

The designed system meets this requirement by adopting an universal and generic approach towards the execution of algorithms. The system utilises Command line system to execute the algorithm. The researchers provides the command used to execute the algorithm and also specifies the supporting libraries needed in the configuration file (Refer to Figure 6.2). The system will attempt to install the missing libraries and execute the algorithm using the command given. As command line system is available universally, our system supports any and all algorithms which can be executed using this method.

R7 The proposed system shall provide a performance-energy (FLOPS/W) analysis of a different cluster configurations.

The designed system partially meets this requirement by providing the researcher with results from the computation in terms of Energy Consumption (Watt-hr) and the specifications of the experiment conducted. Different cluster configurations will result into variety of Floating Point Operations Per Second (FLOPS) to be conducted. The researcher can use the data provided by the system to calculate the FLOPS/W and analyse based on the calculations. Note that this value needs to be adjusted to reflect and accommodate for other factors like energy used by non-computing resources.

```

"matrix_multiplication": {
  "__comment": "All paths should be relative. Create a data file by
algorithm name at /logs/*.json",
  "language": "python3",
  "dependency": "mpi4py numpy openMPI",
  "parameters": "3000 3000",
  "command": "python3
algorithms/matrix_multiplication/matrixmultiplication.py",
},
"kmeans_c": {
  "__comment": "All paths should be relative. Create a data file by
algorithm name at /logs/*.json",
  "language": "C",
  "dependency": "GCC openMPI",
  "parameter": "288000 48",
  "command": "./algorithms/kmeans_c/kmeans_c.exe",
  "file_path": "algorithms/kmeans_c/kmeans_c.exe",
}

```

Figure 6.2: Configuration file extract for instructions on execution of Algorithms

R8 The proposed system shall be able to export the experiment data for further analysis.

The designed system meets this requirement by providing user an option to export data into a JSON format file. The data for the specified experiment is retrieved from the local database and written to a file which can then be used to perform further analysis. The system allows researchers to specify parameters while exporting of data. These parameters also allow filtering of the data so that researchers can export only the relevant data. The parameters include the cluster configuration, algorithm type, algorithm specification such as the parameters or output, energy thresholds, etc. This functionality will help researchers in implementing new specific features easily to make use of the exported data and analyse or conclude results further.

R9 The proposed system shall be compatible across different processor architectures.

The designed system itself is developed on JavaScript, a programming language which supports wide range of computer architectures and Operating systems. The system is developed to minimize the installation process and every functionality in the system is implemented using generic approaches. The system as whole is not restricted by any processor architecture and as long as the processor has support for JavaScript, the system will be compatible. This requirement helps ensure that the system is available for use by researchers with different resources available for computing.

6.2 Quantitative Evaluation

This section showcases the results from different experiments conducted using the designed system. Quantitative data of all these requirements are presented and the requirements are met by the designed system in real life based computing scenarios. Note that, the results by themselves are not related to the research questions but they help in understanding how the research questions are achieved in real life scenarios. The in-depth discussion is later provided in Chapter 7

The aims of the experiments are to showcase the workings of the system designed with the help of a particular use case of algorithm and parameters. The experiments relates to Requirements R1 to R8. As the main goal of the system is to be able to evaluate different algorithms on different cluster configurations, each of these experiments focus on a single algorithm and the results obtained from the experiments conducted on those algorithms. The results are then co-related to the requirements and their relevance to the research questions is justified. Table 6.1 briefly summarises the algorithms evaluated and their parameters.

Name	Application	Computation details
Matrix Multiplicaiton	Graph Theory	Dot product of matrices of 3000x3000 elements
Kmeans	Data Mining	Clustering of 288000 data points into 48 clusters
OpenCV Filtering	Image Processing	5 Images (1920x1080 pixels) Applied filters on each image: blur, sepia, emboss, warm, cold and increased brightness.

Table 6.1: List of Algorithms Evaluated and their Parameters

6.2.1 Multiple run of single experiment on different cluster configuration

The aim of this experiment is to test Requirements R1, R2, R3 and R4. In order to check that the system helps in identifying the bottlenecks, the system needs to be pushed to its limits. Theoretically, there is no limit to the system's capacity to carry out the computation, but for this experiment a 3,000 elements by 3,000 elements array were used for the matrix multiplication algorithm. The algorithm is executed for upto 12 threads on each nodes.

The experiment is setup with installation of basic libraries required for this computation. The algorithm uses numpy, a python library which supports computation large arrays. To support parallelism the cluster also needed installation of mpi4py package, a Python binding for the Message Passing Interface (MPI) standard. The basic python3 package is being used. The algorithm is given the max parameters – 6 nodes and 12 threads and was allowed to execute until it finishes the computation. Ideal nodes are powered off to save energy. The final data is exported in a JSON file (Requirement R9 and

Section 6.2.3) but is shown as tabular and graphical format for ease of use and understanding (Refer to Figure 6.3).

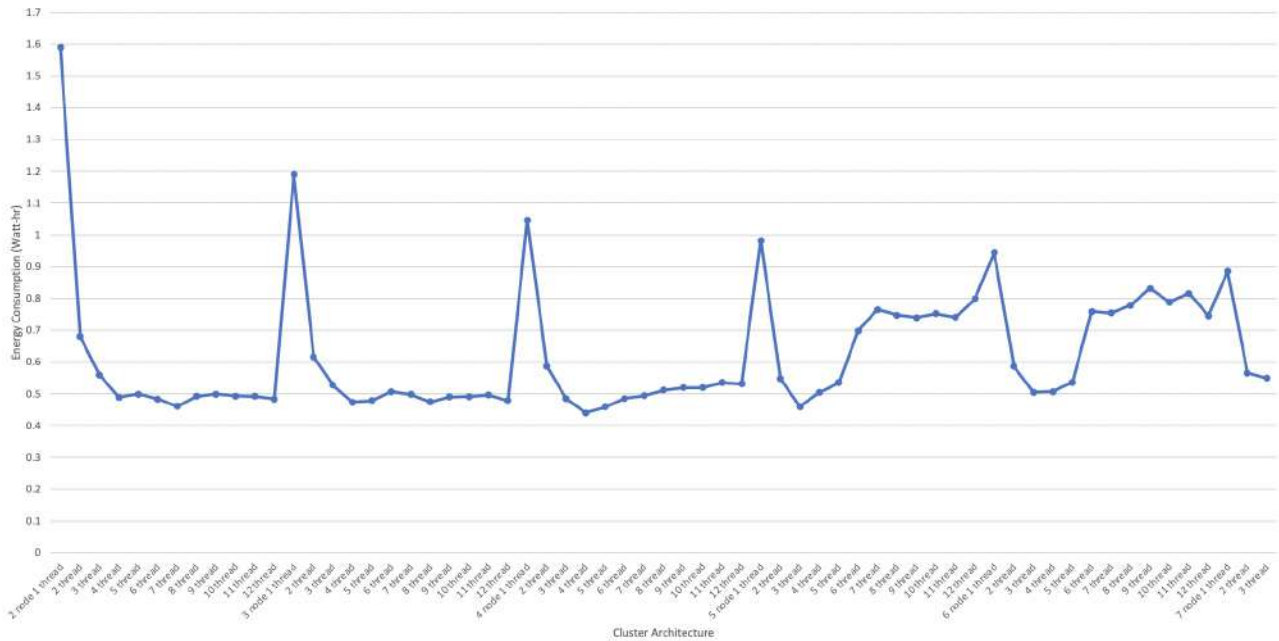


Figure 6.3: Evaluation of Algorithm: Matrix Multiplication with different cluster configurations

The Y-axis shows the Watt-hr values and the X-axis shows the node configuration. As seen from the graph and the raw data, a researcher can clearly conclude saying that increase in nodes leads to faster computation and less energy consumption for the duration of computation. The energy consumed is definitely increasing when a new node is added, but that is compensated by the speedup in the performance. The graph also shows that for small amount of nodes that the energy consumption stays constant or gets worse after 4 threads on each node.

Analysing the result with relation to the aims of the experiment (Achieving Requirements R1, R2, R3 and R4), it can be concluded that all of the requirements are successfully met.

- Requirement R1 involves aiding the user in choosing the best possible cluster configuration. This is successfully met as a researcher can use the data given above to stop the experiments from running on more than 4 threads on each node to save time and energy in future.
- Requirement R2 includes identification of bottlenecks in the computing, which has been fulfilled (No improvement after 4 threads on each node).
- Requirement R3 includes the specification of a range of repeated experiments. This requirement and Requirement R4 are successfully met, shown by the number of different experiments conducted in one single run of the system. The system configured the cluster and re-ran the whole computation on the new configured cluster automatically without the need to be restarted.

6.2.2 Multiple run of single experiment with different data-set

The aim of this experiment is to test requirements R1, R3, R5, and R7. The experiment is conducted to compare the energy consumption of a single node when different data sets are provided for it to compute. This experiment will help us understand the effects of variable data-sets on the performance and energy of a node.

The experimental setup is same as that of Section 6.2.1, but instead of using multiple nodes, the experiment is performed on a single node, single thread, with different data sets. The experiment is iterated 4 times, each time with a different data-set (starting from 600 to 960 elements each time). Two arrays of each size are randomly generated and their product is computed using the function `numpy.dot`. The results of the experiment were exported into a JSON file (Requirement R9 and Section 6.2.3) but is shown as tabular and graphical format for ease of use and understanding (Refer to Figures 6.4, 6.5, 6.6, 6.7). The energy measured is the instantaneous energy for any given particular time.

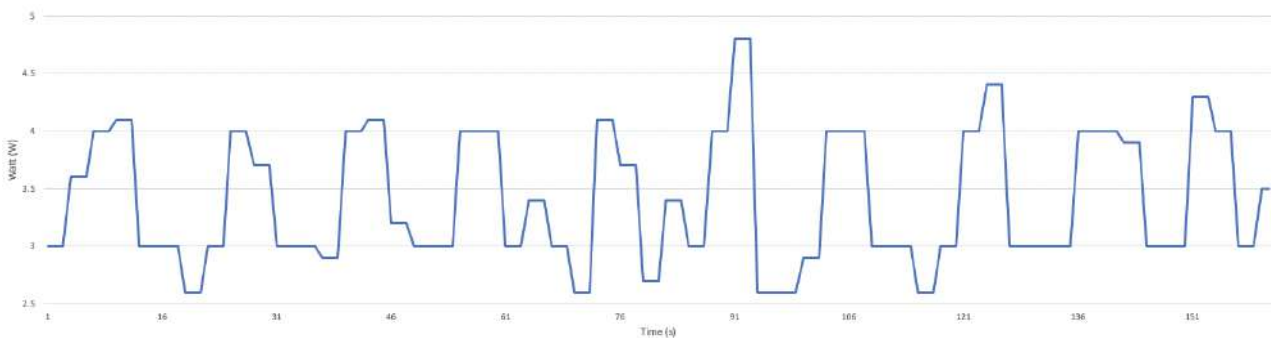


Figure 6.4: Evaluation of Algorithm: Matrix Multiplication (Data-set of 600)

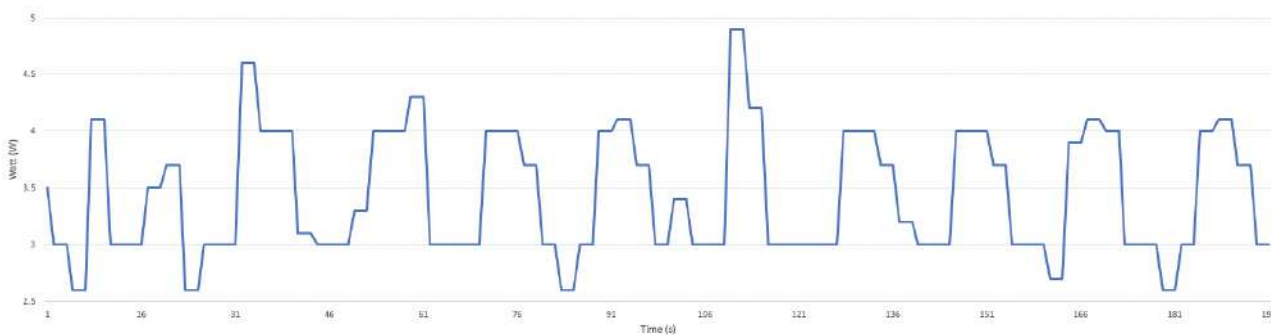


Figure 6.5: Evaluation of Algorithm: Matrix Multiplication (Data-set of 720)

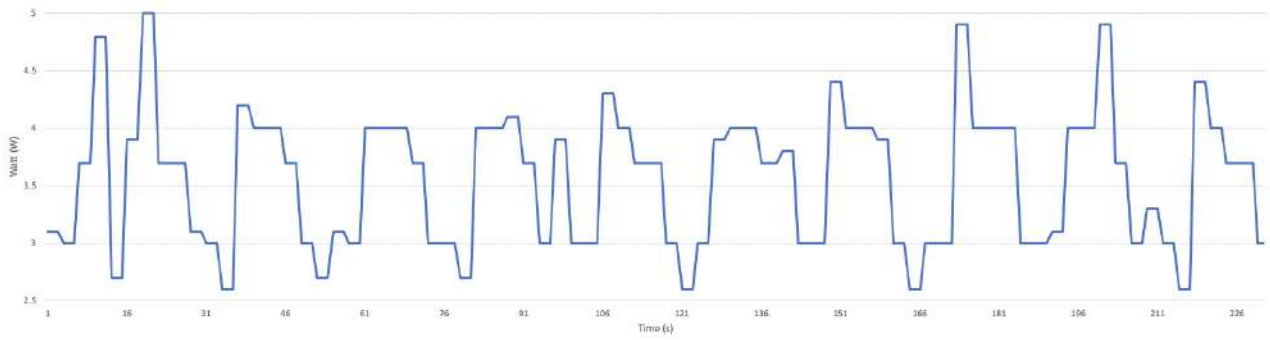


Figure 6.6: Evaluation of Algorithm: Matrix Multiplication (Data-set of 840)

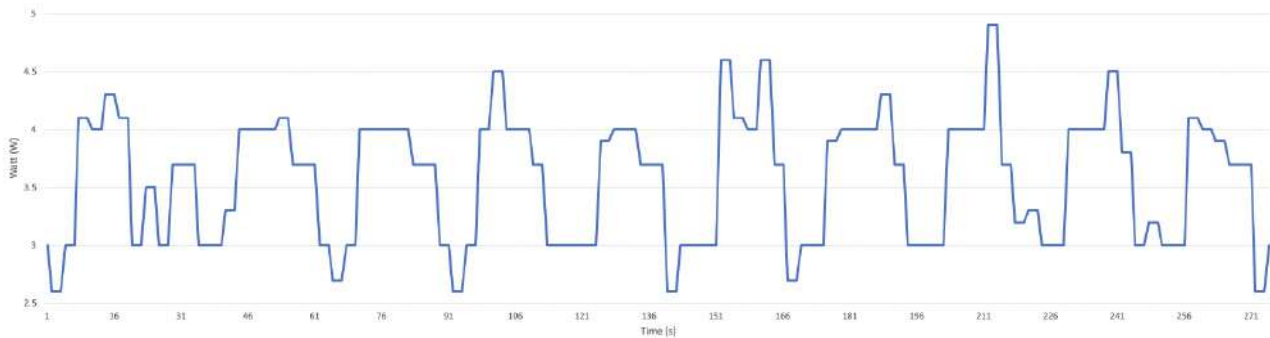


Figure 6.7: Evaluation of Algorithm: Matrix Multiplication (Data-set of 960)

The Y-axis shows the instantaneous energy consumed in Watts and the X-axis shows the data set size. As seen from the graphs and the raw data, a clear maximum and minimum limit of energy consumption can be identified. For different data-sets, for any given time the maximum energy consumed is found to be 5 Watts and minimum 2.6 Watts. The factor that changes is the time taken to compute. As the energy consumed is capped up to a certain limit, the processor is being pushed to its maximum and has to compute for a long time to finish the big data-sets.

Analysing the result with relation to the aims of the experiment (Achieving Requirements R1, R3, R5 and R7), it can be concluded that all of the requirements are successfully met.

- Requirement R1 involves aiding the user in choosing the best possible cluster configuration. This is successfully met as a researcher can use the data given above to calculate the time taken by a node to compute a certain data. Using the maximum energy consumption and the FLOPS operations that a core can perform, a estimated completion time of an algorithm can be calculated. This can then be used to choose the optimal cluster configuration for that particular computing.
- Requirement R3 includes specification of a range of repeated experiments. The system in itself repeats the experiment multiple times with different data parameters to the algorithm. This is all achieved through the configuration file.

- Requirement R5 includes the execution of different algorithms automatically with support for its changing parameters. This requirement has been partially met in this experiment as it supports the change of parameters for the algorithm. Support for multiple algorithms is shown in further Sections.
- Requirement R7 includes comparison of energy consumption for different computations. This requirement is met as seen from the graphs. Energy consumption can be compared to that of other computations in the same domain and conclusions can be devised.

6.2.3 Expansion on the functionality of the designed system

The aim of this experiment is to test Requirement R9. This is a goal level requirement and adds value to the system. The functionality is implemented with the goal of easing further analysis. This will help in the usability and portability of experiment data. The section also describes how this functionality be used to implement new features and help achieve our research questions better.

There is no special experiment setup required for this experiment. This experiment builds upon previously ran experiments whose data is available in local database.

Figure 6.8 shows an extract of the output file generated by the system. It is exported in JSON format for easier use and cross-compatibility. The first object identifies the name of the algorithm. The second and third level objects represent the number of nodes and threads the algorithm is being executed on respectively. The fourth level object is a tuple data structure comprising of the timestamps of that particular experiment and the average power consumption during the experiment.

In order to be able to conclude that the framework meets Requirement R9, an implementation of this functionality is shown. The raw exported data can be used to improve on the functionality of the system. A new function in the system is implemented which reads this exported data and generates an online graph and provides the link to access it. Appendix E shows an extract of the functionality being included. Plotly is an online tool which can receive data through API requests and generates graph based on the parameters (Refer Appendix F).

```

"opencv": {
"1": {
  "2": {
    "timestamp": {
      "start": "1598965940040",
      "end": "1598966009730"
    },
    "pwr_avg": {
      "p1": 4.239130403684533,
      "p2": 3.1246376175811323,
      "p3": 2.947826091794,
      "p4": 3.0826086894325586,
      "p5": 2.9565217391304346,
      "p6": 3.0956521241561226,
      "p7": 3.056521747423255
    }
  },
  "3": {
    "timestamp": {
      "start": "1598966016296",
      "end": "1598966149690"
    },
    "pwr_avg": {
      "p1": 4.513533785827178,
      "p2": 3.0887217467888854,
      "p3": 3.0127819534531213,
      "p4": 3.0721804253140785,
      "p5": 3.0338345864661656,
      "p6": 3.0563909344207074,
      "p7": 3.0992481116961716
    }
  }
}
}
}

```

Figure 6.8: Exported data in a JSON format

6.2.4 Algorithm Evaluation: Matrix Multiplication

Matrix multiplication algorithm is being used in almost all sort of research, ranging from quantum mechanics in physics, graph theory in mathematics to gene expression in biology. This algorithm was chosen as it has been heavily optimised to work with parallel computers. The experimental setup is not different from the ones provided in previous Sections. The system is provided with the algorithm file, configuration parameters and the system produces the results. The experiment is running the function `np.dot` for matrix multiplication on matrices with 3,000x3,000 elements.

The results of the experiment were exported in a JSON file (Requirement R9 and Section 6.2.3). Refer to the Appendix B for the raw data. Different parameters of the algorithm are shown in graphical format for clarification (Refer to figures 6.9 and 6.10).

As shown in Section 6.2.1, it was decided to showcase results for experiments up to 4 threads

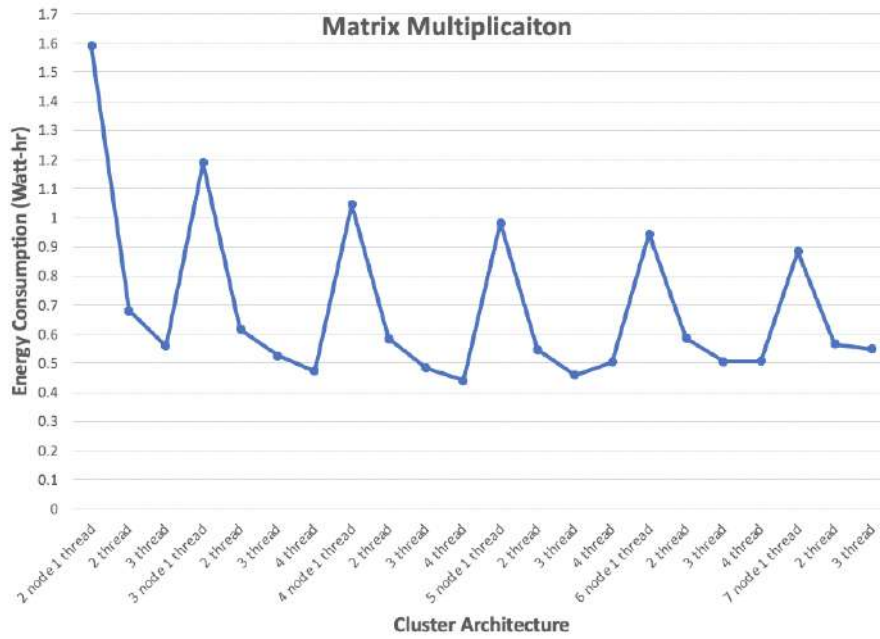


Figure 6.9: Algorithm Evaluation: Matrix Multiplication and Energy consumption

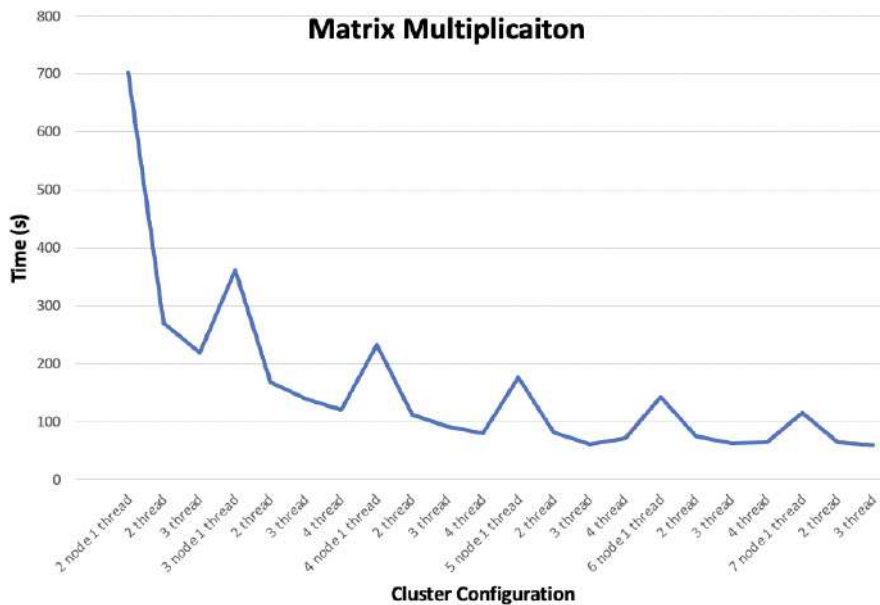


Figure 6.10: Algorithm Evaluation: Matrix Multiplication and Computation Time

on each node to clearly convey the results and to reduce the garbage data. As seen from Figures 6.9 and 6.10, matrix multiplication produces the expected results from a parallel computing algorithm. Increase in the number of computing nodes leads to decrease in the computing time and increase in performance. It was seen that the time to compute slowly starts flattens out at the end of 6 nodes. The power consumption, as expected, is supposed to go up as more nodes are added or increase the computation per node.

A researcher can calculate theoretically or experimentally the lowest computation time achievable for the give data set and algorithm using the results. This is very useful in predictive analysis where

the algorithm output can be predicted based on its past performance. Researchers can also identify the bottlenecks from the graph saying that addition of each node degrades the performance and then using the nodes to their maximum capacity leads to increase in performance. The bottleneck here might be the cache memory which has to load the array first time any node is added to the cluster.

6.2.5 Algorithm Evaluation: Kmeans

The Kmeans algorithm is a method that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster (Saffran et al., 2016). This algorithm was chosen as the algorithm heavily relies on the communication between master and slave nodes for computing. Kmeans have less independent work units which leads to more synchronization when the parallel work finishes. A C implementation of Kmeans is used to show the portability of our system as well as its compatibility with FORTRAN libraries.

The experimental hardware setup is the same. For this experiment, the system was provided with an algorithm file which in turn was provided a set of 288,000 data points to sort into 48 clusters. The algorithm ran iteratively until the cluster centers did not change their position. Refer to the Appendix D for the raw data and Figures 6.11 and 6.12 for the results in graphical format.

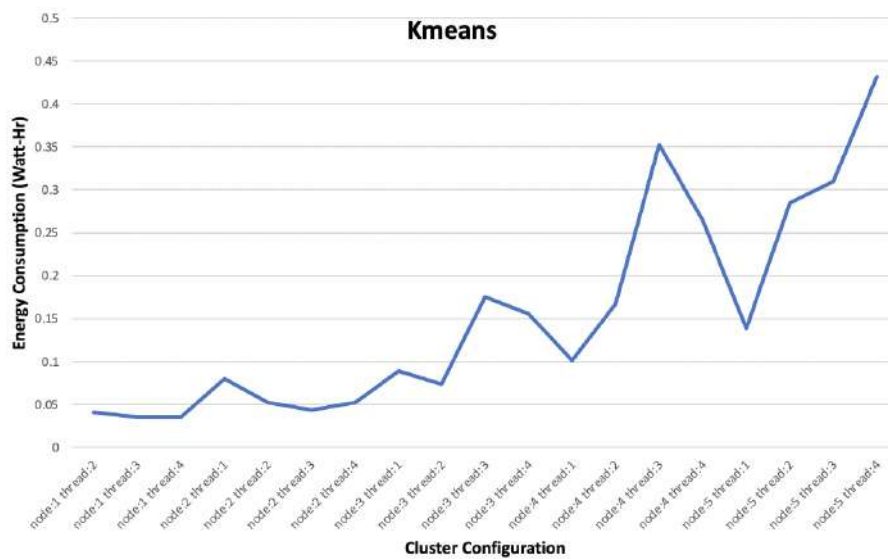


Figure 6.11: Algorithm Evaluation: Kmeans and Energy Consumption

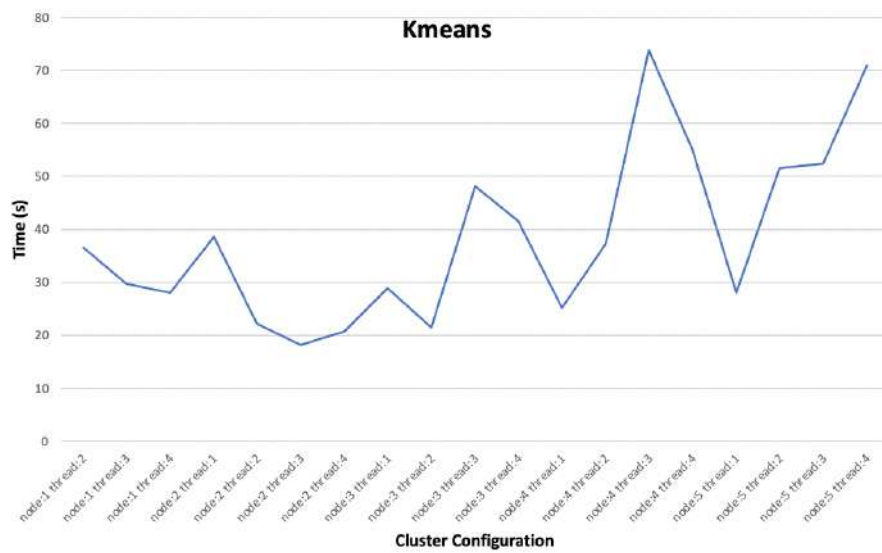


Figure 6.12: Algorithm Evaluation: Kmeans and Computation Time

This algorithm was chosen as researchers have tried to parallelize the algorithm but could not gain any significant improvements in it (Saffran et al., 2016). Looking at it from an energy perspective led to the same conclusion. The algorithm does not improve or in some cases worsens the performance as well as energy consumption when adding new nodes to compute.

The results can be used to conclude a number of things. It clearly showcases the bottleneck in Kmeans algorithm - its communication and synchronisation. Also, it shows that adding new nodes to a cluster degrades the performance of the algorithm. This result can help researchers in choosing another algorithm or other ways to improve on algorithm for their computation needs.

6.2.6 Algorithm Evaluation: OpenCV filtering

OpenCV is a library of programming functions mainly aimed at real-time computer vision (Markovic et al., 2018). It is mainly used in image and video processing where a large amount of data needs to be processed in real time. The main aim of the experiment is to find the effects of image processing libraries and its different methods on the energy consumption of a cluster and to identify its bottlenecks.

As shown in Table 6.1 six different types of filtering methods were used on 5 high-res images recursively. Python language was used to write the algorithm. Experimental setup was similar to that of matrix multiplication with different supporting libraries installed (opencv (3.2.0+dfsg-6)). Refer to Appendix C for raw data and Figures 6.13 and 6.14 for the results in graphical format.

Figures 6.13 and 6.14 shows the Energy Consumption of the cluster and the Time taken by the algorithm to finish computing respectively. As expected, the energy consumption increases with an increase in nodes and resources being used. As for Figure 6.14, there is no significant trend in the time required by the algorithm, but a clear bottom limit of the speed can be seen, after which the algorithm starts to degrade performance.

The system ran the algorithm with the given configuration and provided the results shown in the

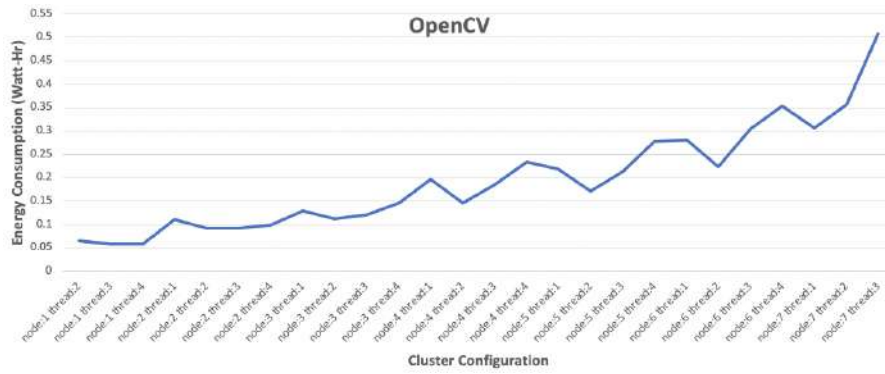


Figure 6.13: Algorithm Evaluation: OpenCV and Energy Consumption

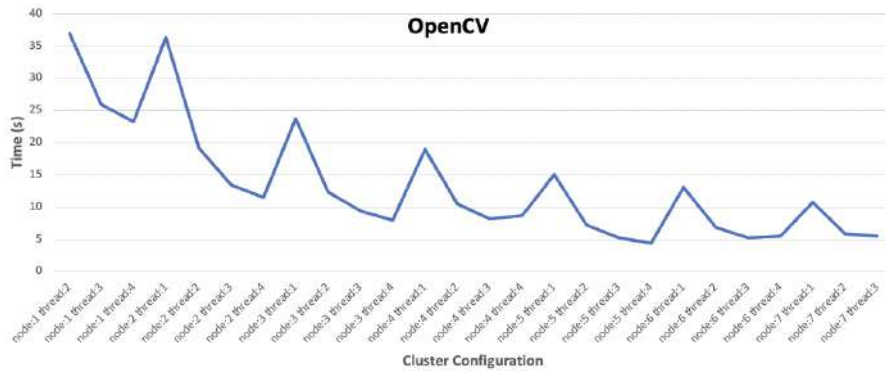


Figure 6.14: Algorithm Evaluation: OpenCV and Computation Time

graphs. From the graphs bottleneck for the algorithm can be hypothesized - data set being too small for computation. No clear trend in the data also can suggest researchers to find better ways to optimise the algorithm or motivate them to find the reason behind it.

6.3 Summary

This chapter has presented an evaluation of the designed system, FEPAC, for a wide range of parallel algorithms on different cluster configurations. It consists of two main parts, a detailed qualitative evaluation of how FEPAC meets the requirements and quantitative evaluation of FEPAC when ran for three different algorithms - Matrix Multiplication, OpenCV and Kmeans.

Section 6.1 shows that FEPAC successfully met all the required criteria defined in previous chapters. The requirements are then evaluated with the help of evidence or demonstration the functionality corresponding to the requirement.

The evaluation of FEPAC on three algorithms provided a detailed case study which can be used to understand the workings of the system in a real case scenario. Aims of each experiments were discussed before the experiment results and an in-depth analysis of each experiment's data is also presented. Section 6.2 showcased the results of each experiment and argues that the designed system successfully answers the research questions through quantitative results.

CHAPTER 7

DISCUSSION

This Chapter provides discussion of the experiments performed in Chapter 6. The results in the evaluation are co-related to already known facts and different interpretation of the results are provided. Apart from results of individual experiments, all the data as whole can help researchers conclude important features, shortcomings or results. The chapter aims to document many such interpretations and conclusions that researchers can draw out from the data generated by FEPAC.

7.1 Helping researchers in deciding best optimal node for computing

The system helps in answering this concern through quantitative evaluation of different algorithms. Figure 6.3 shows the results of using FEPAC to execute Matrix Multiplication algorithm on different node configurations. It can be clearly seen that on an average four threads on each node produced better performance than any other case. This can be explained by the fact that RPi3B+ have a quad core processor and the four threads access each of the four cores independently and hence present better performance. The reason for degraded performance for configurations under 4 threads could be due to the ideal nature of the non-computing cores. For other situations the degraded performance can be explained due to context switching where one process has to wait for the core to finish computing before starting work on another.

7.2 Finding relationship between data set and energy

Figure 7.1 shows the instantaneous energy consumption for Matrix Multiplication algorithm for one single node with one single thread. The variable parameter in this experiment is the data size that the algorithm has to compute. The algorithm was provided with four different data-sets and their energy consumption for the duration of experiments was stored for analysis.

As seen from the figure, there is a clear upper bound and lower bound for the duration of experiment. Also, it can be seen that the bigger data-set took more time to compute than others. This is an expected

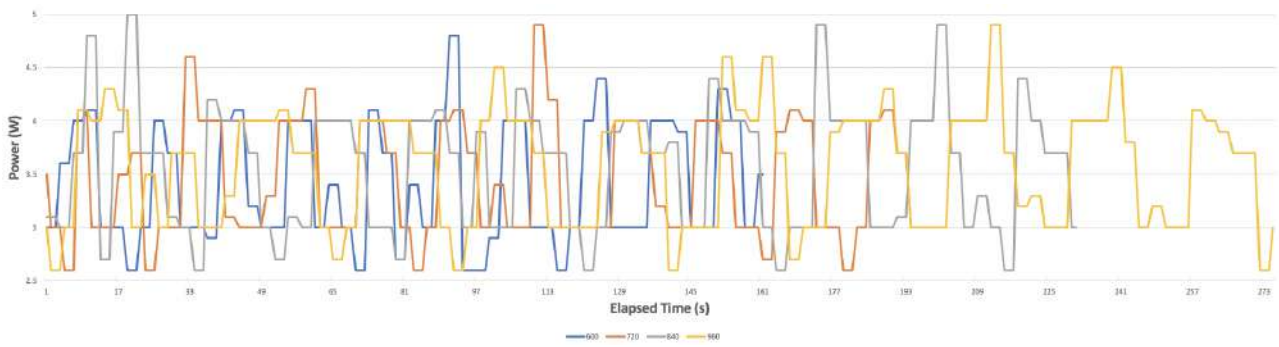


Figure 7.1: Effects of different data-sets on Energy Consumption of an Algorithm

behaviour because the node can not consume as much energy as needed to perform computations. The energy being consumed by the node at any time is regulated by the on-board resistors to avoid damage to the nodes. As the energy stays constant, the processor have to wrong longer to computer bigger data sets.

Researchers can use this data to predict the time that the algorithm will take in future. A relationship between the increase in time and the increase in data-sets can be calculated and used to base future predictions on. Also, seeing that there is a upper bound limit of a computation, the theoretically maximum energy that a cluster can consume for the duration of experiments can be predicted as well.

7.3 Helping researchers in predicting the computation time for an algorithm

The data given above comprises of the average time taken by the algorithm to complete the computation. The three experiments conducted are different in the algorithms and the parameters passed to each of them. The relation between the time taken by an algorithm and the cluster configuration can help in a number of conclusions such as prediction of data, identification of bottlenecks, motivation for improvement, etc.

As seen from the comparative graphs of computation time of all three algorithms, inconsistency of data can be seen across different algorithms. As the algorithm is being provided with more computation power on each iteration, the expected output is a reduction of computation time. Matrix multiplication and OpenCV produce results as expected but Kmeans performance degrades with an increase in computation power. This can mainly happen due to the inherent non-parallelism of kmeans algorithm. There are very less independent work units available in kmeans algorithm and much time is spent in synchronisation than actual computation. This results can help researchers in choosing correct cluster configuration for the computation or motivate them to develop a more energy efficient algorithm.

Matrix multiplication and OpenCV follow expected patterns in respect to their computation times. This data can be used and analysed to find the point of lowest computation time. It is fundamentally

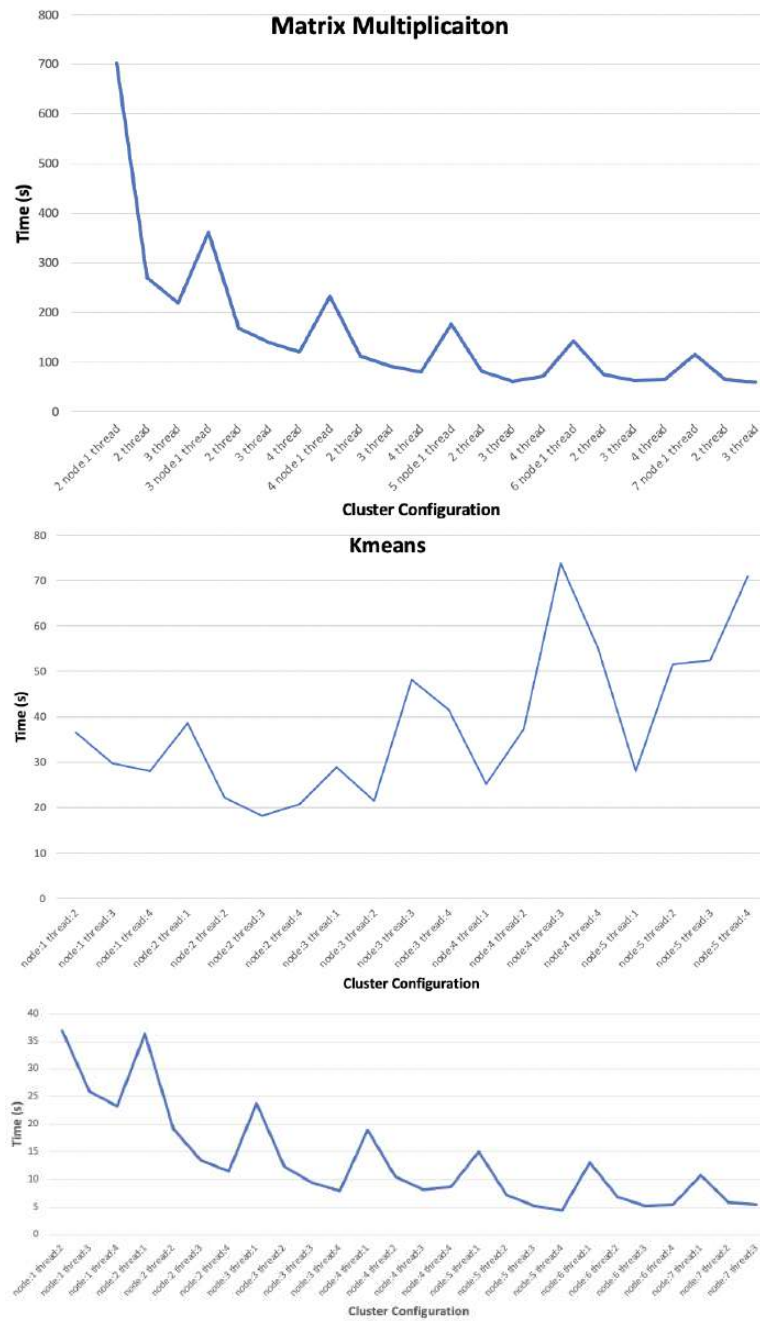


Figure 7.2: Ability to predict Algorithm Computation Time based on previous data

known that computation time can never reach zero and hence, the graphs are expected to flatten out after a certain amount of time. The computation time is also expected to increase after this time as more time will be spent on communication overheads than actual computation.

7.4 Helping researchers in predicting the energy consumption of the cluster

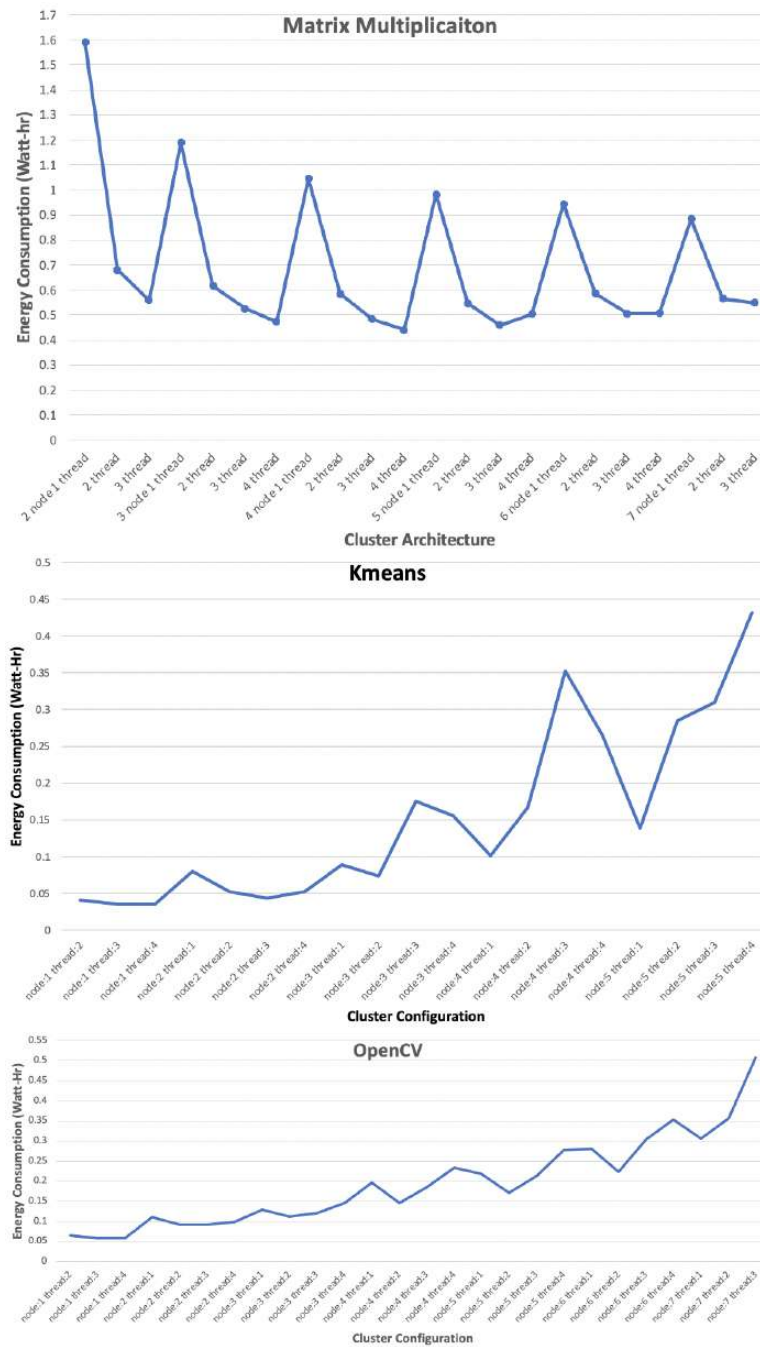


Figure 7.3: Ability to predict Energy Consumption based on previous data

The data given above comprises of the average energy consumption values for a given algorithm for a particular cluster configuration. All three experiments showcases values from different algorithms

with different parameters. Note that the energy consumption values are dependent on two factors- the total energy consumed during a computation and computation time. Comparative results from all three experiments can help in identification of a trend or to show other results.

As seen from the comparative graphs of energy consumption of all three algorithms, it can be seen that the energy consumption increase for Kmeans and OpenCV algorithms when the number of nodes increase. This is expected as a new node in the cluster will definitely need more energy. Researchers can use the data to calculate and predict on increase in the energy for a certain number of nodes. This can help researchers, who are trying to conduct experiments in same domain, to choose and analyse whether the particular cluster is suitable for their computation needs or not (R1).

Matrix multiplication does not follow the expected results. This can be mainly because of the speedup obtained during the computation. As seen in Figure 7.2 the computation time of the algorithm reduces to a great extent when more computation power is provided. Due to this reason, even though energy consumption of the cluster increases due to increase in a computing node, the resulting speedup can help compensate for it. Researchers can use this conclusion to predict the cluster configuration where the energy consumption is no longer compensated by the speedup. This can result in many interesting conclusions.

7.5 Limitations

This section lists some limitations of FEPAC

1. Support for Workloads

Workload is the ability of the system to handle and process work. The work presented and evaluated in this thesis supports execution of algorithms and automatic configuration of clusters. These are the lowest forms of workloads and the system currently have not been evaluated to support complex workloads used by other researchers.

2. Support for scientific workflows

Workflows is a group of instructions that can be executed to achieve a certain objective. FEPAC can be called a very simplistic pseudo-workflow engine which contains instruction on evaluation of clusters and executions of algorithm. As the configuration files is needed to be provided by the researcher, the system will never be truly a workflow engine. Instead, the system can be developed to support other workflows used in other fields.

3. Container based Execution

Even though the system is designed with the requirement of minimal installation steps, there can arise situations where the system can work in some environment but might give error in another. This is the limitation of the system and can be overcome by running it through a container. Containers are complete packages developed with their own set of dependencies. Therefore there will not be any compatibility issues when developing the system in such manner.

4. Support for Graphical Processing Units

Graphical Processing Units or GPUs work in similar manner as to that of a computer. They have their own computing cores and share a common memory. GPUs carry out symmetric multiprocessing in which all the processes work together to perform certain computation. No process is reserved for any special purpose and hence it would be interesting to apply our system on such architectures. The data provided by experiments conducted on GPUs can help in developing more energy efficient GPUs or perform other analysis on its workings. This is a limitation in current scenario as the computing nodes used to evaluate the system do not possess the GPU capabilities.

5. Predictive model for identification of trends and analysis of data

The system developed in this thesis provides a way for researcher to automatically execute algorithms and configure clusters. The results are then provided to the researchers. The system does not provide any of its own suggestions and therefore the interpretation of data can differ from one person to another. This is a limitation of the system as the data is then open to bias and can be wrongfully interpreted.

6. Inter-node communication

The work presented in this thesis presents a system in which the communication and synchronisation is only done between the master and the slave nodes. This increases the time spent in communication. Facilitating inter-node communication so that the nodes synchronise themselves with each other and only communicate with master node for the results can lead to increase performance and reduced computation times. This is a purely academic idea and the lack of knowledge for developing a system that can handle such processing is a limitation for this thesis.

CHAPTER 8

CONCLUSION

This chapter presents the conclusions of the thesis. Section 8.1 contains an overview of the thesis, reviewing the key-points of each of the chapters. The main contributions of the work presented in this thesis are discussed in Section 8.2. Section 8.3 discusses potential future work and expansion on the work presented in this thesis. Finally, Section 8.4 includes some concluding remarks.

8.1 Overview of Thesis

This thesis has described and evaluated a generic framework for evaluation of parallel algorithms on different cluster configurations. Chapter 1 provides an introduction to the whole thesis. The chapter introduced the field of parallel computing and shows that energy consumption is one of the challenges faced while achieving high performance computing. The chapter argues that the use of single board computers in computing can solve this challenge due to their low power consumption. It was also shown that the traditional methods of evaluation did not focus on energy consumption of a computation. In conclusion, it was argued that there was a requirement for a new evaluation method for parallel computing which focused more on energy consumption and factors affecting it.

Chapter 2 presented the background in the field of parallel computing. The hardware and software aspects of parallel computing have been discussed. The chapter also presents a discussion of the state of the art in the field of experimental computing and energy aware cluster computing. State of the art in the field showed that although researchers have accepted energy as a concern in high performance computing, the research in this field is stunted because of the lack of evaluation method which focuses on the energy consumption. The chapter also shows that the current methods of evaluation of computing focus heavily on the performance and ways on how to improve performance. The chapter concluded by commenting on the lack of a framework to evaluate energy consumption in parallel and that this would motivate further research in the field.

Chapter 3 presented a discussion of research questions which were devised from the open issues found in the literature. The chapter presented the methodological approach adopted for finding the solutions to the research questions. It compared the strategies adopted by researchers in similar fields

and provided a justification for the approach undertaken to achieve the objectives of this research. The chapter concluded with a list of requirements for the system that will help achieve the research questions.

Chapter 4 presented an abstract idea which would help in finding the solution to the research questions. The idea was developed to meet the requirements and provided a generic outline of the features and functionalities that needs to be achieved by the system in order to be able to find solution to the research problems.

Chapter 5 presented FEPAC as a generic framework for evaluating parallel computation with focus on energy consumption. The chapter documented the two-fold implementation required to achieve the final goal. One being the implementation of a hardware cluster and second being that of a software framework. The implementation is built using the abstract idea developed in previous chapters.

To illustrate that FEPAC achieves the requirements and helps in answering the research questions, Chapter 6 applied qualitative and quantitative evaluation approaches on the implemented design. Qualitative evaluation concluded that the system designed met the requirements set out in previous chapters whereas quantitative evaluation helped in understanding the workings and functionalities of the system. To achieve the quantitative evaluation FEPAC was used to execute three different algorithms, Matrix Multiplication, OpenCV and Kmeans. The main focus of using FEPAC on these algorithms was to draw conclusion based on results and to showcase the relevance of the designed system to the research questions of this thesis. The conclusions of the chapter were that FEPAC had effectively meet the system requirements and can quantitatively justify that it can provide answer to the research questions.

Chapter 7 presented a discussion of the results found during the evaluation of FEPAC. This chapter included an in-depth analysis of the framework and its relevance to the research questions. High level functionalities of the designed system are discussed and justified using quantitative data. The chapter concludes with a list of limitations identified during the evaluation and discussion of the proposed system.

8.2 Contributions

High quality scientific contributions of this thesis are presented in this section.

Literature review in the field of parallel computing

A major contribution of this thesis is the derivation of research gaps found in the field of high performance computing based on a comprehensive analysis of state of the art in parallel and cluster computing. The research gaps found were the need for energy aware computing, the lack of research focusing on the energy consumption of a computation, the lack of motivation for research in the similar field and the need for a generic programming model for evaluation of computing which focuses on energy. The thesis aims to address these research questions and provide solution for them.

Design of a Framework for evaluating parallel algorithms

The thesis aims to tackle the research questions in Chapter 2 by developing a framework, namely FEPAC, that evaluates different parallel algorithms on different cluster configuration and provide the researchers with a comparative cost-performance analysis of the algorithm. The framework focuses on energy consumption of the cluster and aims to motivate researchers to use or develop a more energy efficient algorithm or method of computation respectively. This thesis has presented the design of a flexible software framework which supports wide range of configurations and parameters. An in-depth evaluation of FEPAC on an implementation of a eight node single board computer cluster was then used to illustrate the potential of the framework.

Evaluation of the Framework using qualitative and quantitative means

Extensive evaluation on FEPAC by-using both qualitative and quantitative analysis is performed. The framework is evaluated by discussing its qualitative aspects and how they align with the thesis's goals. It is also evaluated by comparison of the data collected from executing 3 real-world algorithms on different cluster architectures. The evaluation results show the relevance of the proposed model with respect to the research gaps found in Chapter 2 and confirms the significance of this work in Chapter 7. The evaluation contributes majorly in terms of the functionality provided by the framework and how they can help in answering the research questions.

High quality ready-to-submit research paper is prepared

A high quality research paper related to the work presented in this thesis has been prepared and will be submitted to the AusPDC 2021 conference. AusPDC 2021 is a conference as part of the 2021 Australasian Computer Science Week (ACSW).

8.3 Future Work

In this thesis, the need for a framework which evaluates different cluster configurations is investigated. The workings of the framework is shown by running it on a 8 node RPi3B+ cluster. The work presented in this thesis can be expanded in a number of different ways and some of the future plans to enhance the research in the field is documented in this section.

8.3.1 Expansion to real-world workloads

Workloads are defined as a computer system's ability to handle and process work. The work presented in this thesis addresses smaller workloads in terms of algorithms and their execution. This can further be expanded by providing an functionality to be able to handle actual real-world workloads. Scientific workflows and container-based computing executions are some of the workloads that can be focused on in the future.

Scientific Workflows

A scientific Workflow is the description of a process for accomplishing a scientific objective. It includes a list of tasks or dependencies which are mainly used for scientific simulations or data analysis. Scientific Workflows are being used in computation field to easily express multi-step computational tasks. The work presented in this thesis can be expanded to include support for workflow algorithm. Workflow algorithm make use of different algorithms to perform higher level tasks. The expanded work can then help in analysis of the energy consumption of different workflows and parameters affecting it.

Container-based Computing Execution

Container based computation are being used in almost all parts of computations. Container is defined as an entire run-time environment on its own. It includes all the execution code and its dependencies in a single package. It allows the execution to occur quickly and reliably. As all the dependencies are already included in the software package there is no compatibility issues while working with container-based executions. The work presented in this thesis can be expanded to include evaluation of Container-based Executions. This will facilitate a wide range of experiments to be conducted on the containers and their results being used to suggest improvements in the containers.

8.3.2 Cluster Improvements

Upgrading the cluster to use RPi 4B

RPi4B is the newer generation of SBC developed by Raspberry Pi foundation which features significant improvement in hardware as well as computation. Some benchmarks have already been done on RPi4 which suggest improvements in its computation power as well as being energy efficient 2.2. RPi4B has 4 different model with different memory – 1GB, 2GB, 4GB and 8GB. RPi4B has the same footprint as the 3B+, so this would mean minimal changes to the way the clusters are setup. The framework was designed to be compatible with different architectures and hence there won't be any changes needed to be made to the framework. During the writing of this thesis, only stable OS for RPi4 has been Raspbian buster and LibreELEC. Both of them are heavy weight OS and won't provide the best performance from the SBC. The main worry while using the RPi4 is the heat generation, the costs for cooling and corruption of the the boot memory (which is highly unstable). Many dependent computing libraries also do not support RPi4 yet. If these issues are resolved then RPi4B could be the next step in gaining more performance while being energy efficient.

Implementing inter-node communication for faster computation

The current system is a master-slave configuration where all the communication and final computation is handled by the master node. The slave nodes only perform the computation that the master node requests. A possible improvement to this topology can be to implement inter-node communication to

reduce the overheads of relaying data back and forth from the master. This implementation can lead to improvement or deterioration of performance based on the way the algorithm handles communication. With comparisons to the algorithms evaluated in this thesis, it can be predicted that Kmeans algorithm will have a highly degraded performance when ran using this cluster setup.

8.3.3 Framework Improvement

Powering nodes as per requirements

The ideal nodes in the current system are running in the background. The power consumption for these nodes have not been included in the thesis as they were powered off manually when not needed. The framework can be improved by providing it with the ability to power on and off the nodes based on the demand for computing. This may improve the energy efficiency of the cluster as whole and might provide different performance based on the latency of the nodes being powered on and off.

Expanding the domain of algorithms to be tested

A framework that can perform detailed computing using the parameters given and analysis of the data is provided in this thesis. The work presented in this thesis can be expanded by including a case study comparing results of algorithms from similar domains or algorithms from different domains. For example - similar domain (arithmetic) could be matrix multiplication, sorting algorithms, finding accuracy of a calculation, etc. Different domain could include image processing, cryptography, compression, Artificial Intelligence, Machine Learning, video processing, etc.

Implementing a predictive model and better analysis based on the data

The current proposed system provides the computation data to the user and the analysis is totally up-to the user. They can interpret the results however they want. This manual work could be automated by providing a set input to the framework and using machine learning modules to learn from previous results in similar domain or the similar algorithms to provide a predictive response based on the need of the user. This can also be improved to include in-depth analysis of the bottlenecks of the framework and explaining to the user how to overcome them or to show how these bottlenecks affect the performance.

Another improvement can be to include the analysis of relation between energy and performance. How does the energy relate to the performance for a particular algorithm or a cluster configuration. For example, an analysis model can be implemented which focuses on over-volting of the nodes in the cluster. As described in 2.2.1 over-volting is one situation where a processor can provide improved performance for increased power provided to it. It will be interesting to check the relation between this increased speed and energy and how do they relate to each other in different scenarios. Such analysis provide in-depth understanding of the trade-offs between energy and performance.

What are the trade-offs between energy and performance? Does the performance increase or decrease for different energy supply to the cluster? How does the performance compare with the initial cost of developing, implementing and running the cluster? Is the return of investment worth it?

Other analysis from the financial point of view can be provided. This mainly could comprise of the initial cost of developing, implementing and running the cluster and what is the return of investment for the same. The results can be compared with the costs of running the similar experiments on already established infrastructure like Amazon Web Services. The later could save a researcher the initial cost of setup but could lead to increased cost of hiring the infrastructure and reduced performance.

8.4 Concluding Remarks

With the increasing demands for high performance computing, more and more infrastructures are developed with the focus on providing the computing resources needed. This is currently being performed by a mixture of highly energy inefficient and often very expensive high performance infrastructures. Single board computers offer potential to revolutionise the field of high performance computing by offering a low-power, low cost and highly-programmable alternative.

The evaluation of the current state of the art in the field of parallel computing presented in this thesis has argued that the lack of a generic programming support for evaluation of clusters based on their energy consumption is harming the motivation for research and development in the field of Energy aware computing. This problem is both hardware and software related. Low cost and low energy consumption are hardware related issues whereas software that facilitates evaluation of computations based on their energy is a software related issue. Research focusing on energy consumption of a computation have been performed by other researchers by using single board computers. It has been shown that computation achieved through embedded processors do not provide the raw numerical power comparable to that of any established infrastructures. It has also been argues that they such computations are amazingly low cost and when compared using their cost and energy consumption, these embedded processors can prove to be an effective alternative to that of high performance computing infrastructures.

This thesis has demonstrated that it is possible to design and implement a framework supporting the evaluation of parallel computation based on a number of factors including its energy consumption and performance. It also stipulates that single board computers are the future of low-energy high performance computing. It is the author's hope that research such as the one presented in this thesis will contribute to the evolution of the Energy-aware Computing field.

Bibliography

- Akram, A. (2017). *A Study on the Impact of Instruction Set Architectures on Processor's Performance*. PhD thesis, Western Michigan University.
- Alghamdi, T. and Alaghband, G. (2020). High performance parallel sort for shared and distributed memory mimd. *arXiv preprint arXiv:2003.01216*.
- Aroca, R. V. and Gonçalves, L. M. G. (2012). Towards green data centers: A comparison of x86 and arm architectures power efficiency. *Journal of Parallel and Distributed Computing*, 72(12):1770–1780.
- Balakrishnan, N. (2012). Building and benchmarking a low power arm cluster. *Master's thesis, University of Edinburgh*.
- Barney, B. et al. (2010). Introduction to parallel computing. *Lawrence Livermore National Laboratory*, 6(13):10.
- Basford, P. J., Johnston, S. J., Perkins, C. S., Garnock-Jones, T., Tso, F. P., Pezaros, D., Mullins, R. D., Yoneki, E., Singer, J., and Cox, S. J. (2020). Performance analysis of single board computer clusters. *Future Generation Computer Systems*, 102:278–291.
- Berman, K. A. and Paul, J. (1996). *Fundamentals of sequential and parallel algorithms*. PWS Publishing Co.
- Blem, E., Menon, J., and Sankaralingam, K. (2013). Power struggles: Revisiting the risc vs. cisc debate on contemporary arm and x86 architectures. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–12. IEEE.
- Buytaert, K. (2000). The openmosix howto.
- Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50.
- Chai, J. S. and Bose, A. (1993). Bottlenecks in parallel algorithms for power system stability analysis. *IEEE Transactions on Power Systems*, 8(1):9–15.

- Cloutier, M. F., Paradis, C., and Weaver, V. M. (2016). A raspberry pi cluster instrumented for fine-grained power measurement. *Electronics*, 5(4):61.
- Conejero, J., Rana, O., Burnap, P., Morgan, J., Caminero, B., and Carrión, C. (2016). Analyzing hadoop power consumption and impact on application qos. *Future Generation Computer Systems*, 55:213–223.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press.
- Davidson, J. C. (1965). Clock system for electronic computers. US Patent 3,226,648.
- Diwedi, D. V. and Sharma, S. J. (2018). Development of a low cost cluster computer using raspberry pi. In *2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN)*, pages 11–15. IEEE.
- d'Amore, M., Baggio, R., and Valdani, E. (2015). A practical approach to big data in tourism: a low cost raspberry pi cluster. In *Information and Communication Technologies in Tourism 2015*, pages 169–181. Springer.
- Feller, E., Ramakrishnan, L., and Morin, C. (2015). Performance and energy efficiency of big data applications in cloud environments: A hadoop case study. *Journal of Parallel and Distributed Computing*, 79:80–89.
- Flynn, M. J. (1966). Very high-speed computing systems. *Proceedings of the IEEE*, 54(12):1901–1909.
- Fraenkel, J. R., Wallen, N. E., and Hyun, H. H. (1993). *How to design and evaluate research in education*, volume 7. McGraw-Hill New York.
- Gibbons, A. and Rytter, W. (1989). *Efficient parallel algorithms*. Cambridge University Press.
- Hillis, W. D. and Steele Jr, G. L. (1986). Data parallel algorithms. *Communications of the ACM*, 29(12):1170–1183.
- Iserte, S., Castelló, A., Mayo, R., Quintana-Ortí, E. S., Silla, F., Duato, J., Reaño, C., and Prades, J. (2014). Slurm support for remote gpu virtualization: Implementation and performance study. In *2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing*, pages 318–325. IEEE.
- JéJé, J. (1992). *An introduction to parallel algorithms*. Reading, MA: Addison-Wesley.
- Jin, C., de Supinski, B. R., Abramson, D., Poxon, H., DeRose, L., Dinh, M. N., Endrei, M., and Jessup, E. R. (2017). A survey on software methods to improve the energy efficiency of parallel computing. *The International Journal of High Performance Computing Applications*, 31(6):517–549.
- Kaur, T. and Chana, I. (2015). Energy efficiency techniques in cloud computing: A survey and taxonomy. *ACM computing surveys (CSUR)*, 48(2):1–46.

- Kecskemeti, G., Hajji, W., and Tso, F. P. (2017). Modelling low power compute clusters for cloud simulation. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 39–45. IEEE.
- Krish, K., Iqbal, M. S., Rafique, M. M., and Butt, A. R. (2014). Towards energy awareness in hadoop. In *2014 Fourth International Workshop on Network-Aware Data Management*, pages 16–22. IEEE.
- Markovic, D., Vujcic, D., Mitrovic, D., and Randic, S. (2018). Image processing on raspberry pi cluster. *International Journal of Electrical Engineering and Computing*, 2(2):83–90.
- Moore, G. E. et al. (1965). Cramming more components onto integrated circuits.
- Nunez, A., Vazquez-Poletti, J. L., Caminero, A. C., Carretero, J., and Llorente, I. M. (2011). Design of a new cloud computing simulation platform. In *International Conference on Computational Science and Its Applications*, pages 582–593. Springer.
- Onwuegbuzie, A. J., Leech, N. L., and Collins, K. M. (2010). Innovative data collection strategies in qualitative research. *Qualitative Report*, 15(3):696–726.
- Onwuegbuzie, A. J., Leech, N. L., and Collins, K. M. (2012). Qualitative analysis techniques for the review of the literature. *Qualitative Report*, 17:56.
- Pahl, C., Helmer, S., Miori, L., Sanin, J., and Lee, B. (2016). A container-based edge cloud paas architecture based on raspberry pi clusters. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 117–124. IEEE.
- Papakyriakou, D., Kottou, D., and Kostouros, I. (2018). Benchmarking raspberry pi 2 beowulf cluster. *International Journal of Computer Applications*, 975:8887.
- Patel, S., Potdar, M., and Gohil, B. (2015). A survey on image processing techniques with openmp. *International Journal of Engineering Development and Research*, 3(4):837–839.
- Patton, M. Q. (2014). *Qualitative research & evaluation methods: Integrating theory and practice*. Sage publications.
- Pomaska, G. (2019). Stereo vision applying opencv and raspberry pi. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.
- Qureshi, B. and Koubaa, A. (2017). Power efficiency of a sbc based hadoop cluster. In *International Conference on Smart Cities, Infrastructure, Technologies and Applications*, pages 52–60. Springer.
- Rahmat, R. F., Saputra, T., Hizriadi, A., Lini, T. Z., and Nasution, M. K. (2019). Performance test of parallel image processing using open mpi on raspberry pi cluster board. In *2019 3rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM)*, pages 32–35. IEEE.

- Rauber, T. and Rünger, G. (2013). *Parallel programming*. Springer.
- Saffran, J., Garcia, G., Souza, M. A., Penna, P. H., Castro, M., Góes, L. F., and Freitas, H. C. (2016). A low-cost energy-efficient raspberry pi cluster for data mining algorithms. In *European Conference on Parallel Processing*, pages 788–799. Springer.
- Scacchi, W. (2002). Understanding the requirements for developing open source software systems. *IEE Proceedings-Software*, 149(1):24–39.
- Schot, N. (2015). Feasibility of raspberry pi 2 based micro data centers in big data applications. In *Proceedings of the 23th University of Twente Student Conference on IT, Enschede, The Netherlands*, volume 22.
- Srinivasan, K., Chang, C.-Y., Huang, C.-H., Chang, M.-H., Sharma, A., and Ankur, A. (2018). An efficient implementation of mobile raspberry pi hadoop clusters for robust and augmented computing performance. *Journal of Information Processing Systems*, 14(4).
- Suresh, L., Loff, J., Kalim, F., Narodytska, N., Ryzhyk, L., Gamage, S., Oki, B., Lokhandwala, Z., Hira, M., and Sagiv, M. (2019). Automating cluster management with weave.
- Tiwari, N., Bellur, U., Sarkar, S., and Indrawan, M. (2016). Identification of critical parameters for mapreduce energy efficiency using statistical design of experiments. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1170–1179. IEEE.
- Tso, F. P., White, D. R., Jouet, S., Singer, J., and Pezaros, D. P. (2013). The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures. In *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, pages 108–112. IEEE.
- Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678.
- Yoo, A. B., Jette, M. A., and Grondona, M. (2003). Slurm: Simple linux utility for resource management. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer.
- Zargham, M. R. (1996). *Computer architecture: single and parallel systems*. Prentice-Hall, Inc.

APPENDIX A

CONFIGURATION FILE

```
"__comment": "Configuration file for FEPAC",
"app": {
  "__comment1": "NodeJS code",
  "port": 8000,
  "master_node_ip": "192.168.50.220",
  "folder_location_on_nodes": "~/Honours_Framework",
  "max_nodes": 7,
  "max_threads": 4,
  "__comment2": "Following is used for testing purposes.
  Fill this to overwrite the data file usage.
  Default: Name of algorithm.",
  "data_file": null,
  "start_node": null,
  "start_thread": null
},
"plotly": {
  "uname": "mwarade",
  "api": "OKshuW2zDgDRh6Nm3mlp"
},
"switch_telnet": {
  "__comment": "Telnet credentials for connecting to Netgear Switch GS110TP",
  "IP": "192.168.50.150",
  "port": 60000,
  "login": "admin",
  "password": "password",
  "interval_to_call": 1000
},
```

```
"db": {
  "__comment": "Local mysql database credentials",
  "host": "localhost",
  "user": "root",
  "password": "qazwsxedc",
  "db_name": "mytestdb"
},
"algorithm": {
  "hostfile_folder": "algorithms/hostfile",
  "matrix_multiplication": {
    "__comment": "All paths should be relative.
Create a data file by algorithm name
at /logs/*.json",
    "language": "python3",
    "dependency": "mpi4py numpy",
    "command": "python3 algorithms/matrix_multiplication/matrixmultiplication.py",
    "__comment1": "Put is_output true if you have any SINGLE line output from the code",
    "is_output": true
  },
  "kmeans_c": {
    "__comment": "All paths should be relative.
Create a data file by algorithm name
at /logs/*.json",
    "language": "C",
    "dependency": "GCC",
    "command": "./algorithms/kmeans_c/kmeans_c.exe",
    "file_path": "algorithms/kmeans_c/kmeans_c.exe",
    "__comment1": "Put is_output true if you have any SINGLE line output from the code",
    "is_output": true
  }
}
}
```

APPENDIX B

MATRIX MULTIPLICATION: RAW DATA AND CALCULATIONS

Watt-hr calculations are done as follows:

$$\text{Watt-hr} = [(\text{Total Watt})/3600] * (\text{Total Time})$$

Cluster Architecture		Time to compute (s)	Power Consumption (Watt)							Watt-hr
node	threads		master	slave 1	slave 2	slave 3	slave 4	slave 5	total	
2	1	702.49	3.98	4.17	0	0	0	0	8.14	1.59
2	2	269.74	4.21	4.87	0	0	0	0	9.07	0.68
2	3	219.51	4.48	4.70	0	0	0	0	9.18	0.56
2	4	182.58	4.63	5.01	0	0	0	0	9.64	0.49
2	5	190.42	4.60	4.84	0	0	0	0	9.44	0.50
2	6	160.91	5.14	5.67	0	0	0	0	10.81	0.48
2	7	166.20	4.82	5.16	0	0	0	0	9.98	0.46
2	8	180.32	4.71	5.12	0	0	0	0	9.82	0.49
2	9	184.52	4.63	5.11	0	0	0	0	9.74	0.50
2	10	182.10	4.63	5.11	0	0	0	0	9.75	0.49
2	11	181.42	4.62	5.15	0	0	0	0	9.76	0.49
2	12	178.01	4.60	5.18	0	0	0	0	9.77	0.48
3	1	361.03	3.74	4.09	4.03	0	0	0	11.86	1.19
3	2	167.55	4.16	4.70	4.39	0	0	0	13.25	0.62
3	3	139.26	4.46	4.69	4.48	0	0	0	13.62	0.53
3	4	119.55	4.63	4.91	4.72	0	0	0	14.27	0.47
3	5	120.72	4.64	4.94	4.68	0	0	0	14.26	0.48
3	6	118.17	4.91	5.39	5.13	0	0	0	15.43	0.51

3	7	123.42	4.59	5.08	4.84	0	0	0	14.51	0.50
3	8	118.25	4.58	5.13	4.75	0	0	0	14.46	0.47
3	9	123.45	4.55	5.04	4.70	0	0	0	14.29	0.49
3	10	123.18	4.58	5.09	4.69	0	0	0	14.36	0.49
3	11	124.31	4.58	5.09	4.69	0	0	0	14.36	0.50
3	12	119.28	4.60	5.08	4.75	0	0	0	14.43	0.48
4	1	231.75	3.98	4.19	3.99	4.07	0	0	16.23	1.05
4	2	112.48	4.46	4.88	4.61	4.79	0	0	18.74	0.59
4	3	90.84	4.72	4.96	4.65	4.86	0	0	19.19	0.48
4	4	79.93	4.87	5.09	4.87	5.01	0	0	19.84	0.44
4	5	84.40	4.81	5.10	4.79	4.86	0	0	19.56	0.46
4	6	90.99	4.51	4.94	4.60	5.13	0	0	19.19	0.49
4	7	92.72	4.60	4.94	4.68	4.96	0	0	19.18	0.49
4	8	96.02	4.57	4.99	4.68	4.94	0	0	19.18	0.51
4	9	97.43	4.53	5.04	4.63	4.97	0	0	19.18	0.52
4	10	96.82	4.57	5.07	4.71	4.96	0	0	19.31	0.52
4	11	90.41	5.05	5.98	5.07	5.19	0	0	21.29	0.53
4	12	99.44	4.57	5.08	4.67	4.92	0	0	19.24	0.53
5	1	176.05	3.97	4.19	4.00	4.01	3.90	0	20.07	0.98
5	2	80.82	4.67	5.05	4.86	4.88	4.88	0	24.34	0.55
5	3	61.10	5.27	5.53	4.99	5.69	5.62	0	27.11	0.46
5	4	71.28	5.00	5.21	4.81	5.16	5.28	0	25.46	0.50
5	5	78.23	4.90	5.01	4.71	5.05	4.98	0	24.64	0.54
5	6	107.52	4.41	4.97	4.49	4.81	4.71	0	23.40	0.70
5	7	109.10	4.71	5.58	4.90	5.02	5.02	0	25.23	0.76
5	8	108.05	4.61	5.49	4.84	4.98	4.98	0	24.89	0.75
5	9	108.80	4.55	5.33	4.76	4.90	4.93	0	24.48	0.74
5	10	110.65	4.60	5.32	4.81	4.86	4.89	0	24.48	0.75
5	11	108.90	4.65	5.27	4.79	4.89	4.88	0	24.48	0.74
5	12	117.19	4.61	5.29	4.79	4.94	4.93	0	24.56	0.80
6	1	142.88	3.88	4.02	3.95	3.97	3.94	4.02	23.78	0.94
6	2	75.07	4.45	4.99	4.41	4.69	4.67	4.88	28.09	0.59
6	3	62.81	4.58	4.83	4.44	4.96	5.03	5.10	28.94	0.50
6	4	63.74	4.57	4.88	4.54	4.88	4.86	4.89	28.63	0.51
6	5	66.96	4.82	4.87	4.58	4.87	4.84	4.83	28.80	0.54
6	6	95.46	4.44	5.01	4.59	4.77	4.84	4.96	28.62	0.76
6	7	95.46	4.47	5.10	4.64	4.72	4.78	4.72	28.43	0.75

6	8	97.17	4.49	5.22	4.71	4.80	4.83	4.78	28.83	0.78
6	9	103.12	4.51	5.29	4.77	4.88	4.82	4.77	29.03	0.83
6	10	96.57	4.58	5.35	4.80	4.91	4.89	4.88	29.39	0.79
6	11	100.32	4.56	5.35	4.82	4.92	4.84	4.78	29.28	0.82
6	12	91.90	4.57	5.29	4.84	4.84	4.84	4.81	29.19	0.75

Table B.1: Raw Data Output of Matrix Multiplication Algorithm

APPENDIX C

OPENCV ALGORITHM: RAW DATA

#	Cluster Architecture	Energy Consumption (watt-hr)	Total Time (s)
0	node:1 thread:2	0.067	49.7
1	node:1 thread:3	0.058	38.1
2	node:1 thread:4	0.059	38.6
3	node:2 thread:1	0.110	50.3
4	node:2 thread:2	0.092	34.8
5	node:2 thread:3	0.092	30.6
6	node:2 thread:4	0.098	31.7
7	node:3 thread:1	0.128	38.2
8	node:3 thread:2	0.113	29.1
9	node:3 thread:3	0.120	28.2
10	node:3 thread:4	0.145	34.0
11	node:4 thread:1	0.196	42.5
12	node:4 thread:2	0.146	28.0
13	node:4 thread:3	0.184	32.2
14	node:4 thread:4	0.233	40.8
15	node:5 thread:1	0.218	39.7
16	node:5 thread:2	0.172	27.0
17	node:5 thread:3	0.213	31.7
18	node:5 thread:4	0.277	40.3
19	node:6 thread:1	0.280	41.2
20	node:6 thread:2	0.222	28.9
21	node:6 thread:3	0.304	38.1
22	node:6 thread:4	0.353	43.2
23	node:7 thread:1	0.306	38.5

Table C.1 continued from previous page

24	node:7 thread:2	0.356	36.9
25	node:7 thread:3	0.507	51.5

Table C.1: Raw Data Output of OpenCV Filtering Algorithm

APPENDIX D

KMEANS ALGORITHM: RAW DATA

#	Architecture	Energy consumption (watt-hr)	total_time (s)
0	node:1 thread:2	0.040	36.6
1	node:1 thread:3	0.035	29.7
2	node:1 thread:4	0.035	28.0
3	node:2 thread:1	0.080	38.6
4	node:2 thread:2	0.051	22.1
5	node:2 thread:3	0.042	18.1
6	node:2 thread:4	0.051	20.7
7	node:3 thread:1	0.088	28.9
8	node:3 thread:2	0.073	21.5
9	node:3 thread:3	0.175	48.0
10	node:3 thread:4	0.155	41.5
11	node:4 thread:1	0.101	25.2
12	node:4 thread:2	0.167	37.2
13	node:4 thread:3	0.352	73.7
14	node:4 thread:4	0.264	54.9
15	node:5 thread:1	0.138	27.9
16	node:5 thread:2	0.285	51.6
17	node:5 thread:3	0.309	52.3
18	node:5 thread:4	0.431	70.9

Table D.1: Raw Data Output of Kmeans Algorithm

APPENDIX E

ADDITION OF NEW FUNCTIONALITY THROUGH CODE (USING PLOTLY)

```

var watthr = {
  x: x_axis,
  y: watt_hr,
  name: "watt-hr",
  type: "scatter"
};
var timea = {
  x: x_axis,
  y: time,
  name: "total_time",
  yaxis: "y2",
  type: "scatter"
};
var com_timea = {
  x: x_axis,
  y: output,
  name: "algorithm_output",
  yaxis: "y2",
  type: "scatter"
};
var data = [watthr, timea, com_timea];
var layout = {
  title: algo_name,
  yaxis: { title: "Energy consumption (Wh)" },
  yaxis2: {
    title: "Time (sec)",
    // title: "milliseconds",
    overlaying: "y",
    side: "right"
  }
};

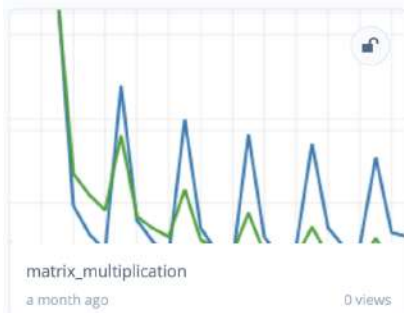
var graphOptions = { layout: layout, filename: algo_name, fileopt: "overwrite" };

plotly.plot(data, graphOptions, async function (err, msg) {
  if (err) return console.log(err);
  await console.log(chalk.magenta('\n
The graph is available at:', chalk.underline.yellow(msg.url)));
  await functions.spinnerStop();
});

```

APPENDIX F

PLOTLY DASHBOARD WITH THE DATA FROM THE FRAMEWORK

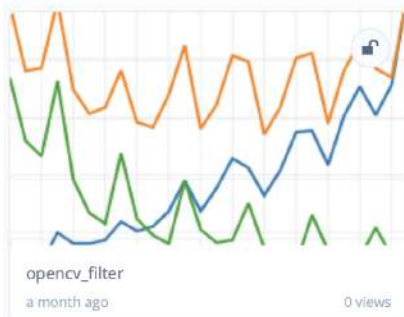


watt-hr, x; to	watt-hr, y	total_time, y
node:1 thread:	None	None
node:1 thread:	None	None
node:1 thread:	None	None
node:2 thread:	1.589048277543	None
node:2 thread:	0.679928553856	None
node:2 thread:	0.559680666552	None
node:2 thread:	0.488659064135	None
node:3 thread:	1.189786147815	None
node:3 thread:	0.616738577906	None

Grid 154
a month ago 0 views

watt-hr, x; to	algorithm_outp
node:2 thread:	782.49
node:2 thread:	269.74
node:2 thread:	219.51
node:2 thread:	182.58
node:3 thread:	361.83
node:3 thread:	167.55
node:3 thread:	139.26
node:3 thread:	119.55
node:4 thread:	231.75

Grid 153
a month ago 0 views



watt-hr, x; to	watt-hr, y	total_time, y
node:1 thread:	0.066050817694	49.73
node:1 thread:	0.057881191268	38.123
node:1 thread:	0.059273354858	38.635
node:2 thread:	0.110616955822	50.344
node:2 thread:	0.092352499894	34.85
node:2 thread:	0.092000999458	30.667
node:2 thread:	0.098214465884	31.755
node:3 thread:	0.128754856656	38.249
node:3 thread:	0.113265655594	29.188

opencv_filter Grid
a month ago 0 views

