Evaluation of Cloud Computing Modelling Tools: Simulators and Predictive Models

Dhahi Alshammari

Submitted in fulfilment of the requirements for the

Degree of Doctor of Philosophy

School of Computing Science

College of Science and Engineering

University of Glasgow

December 2018

Abstract

Experimenting with novel algorithms and configurations for the automatic management of Cloud Computing infrastructures is expensive and time consuming on real systems. Cloud computing delivers the benefits of using virtualisation techniques to data centers instead of physical servers for customers. However, it is still complex for researchers to test and run their experiments on data center due to the cost for repeating the experiments. To address this, various tools are available to enable simulators, emulators, mathematical models, statistical models and benchmarking.

Despite this, there are different methods used by researchers to avoid the difficulty of conducting Cloud Computing research on actual large data centre infrastructure. However, it is still difficult to chose the best tool to evaluate the proposed research. This research focuses on investigating the level of accuracy of existing known simulators in the field of cloud computing. Simulation tools are generally developed for particular experiments, so there is little assurance that using them with different workloads will be reliable. Moreover, a predictive model based on a data set from a realistic data center is delivered as an alternative model of simulators as there is a lack of their sufficient accuracy. So, this work addresses the problem of investigating the accuracy of different modelling tools by developing and validating a procedure based on the performance of a target micro data centre.

Key insights and contributions are: Involving three alternative models for Cloud Computing real infrastructure showing the level of accuracy of selected simulation tools. Developing and validating a predictive model based on a Raspberry Pi small scale data centre. The use of predictive model based on Linear Regression and Artificial Neural Networks models based on training data set drawn from a Raspberry Pi Cloud infrastructure provides better accuracy.

Acknowledgements

I thank my God 'Allah' for his support and providing me with the patience to successfully complete the work. I express my sincerest gratitude and thanks to my supervisor, Dr. Jeremy Singer, for his valuable support, guidance and assistance in conducting and shaping this research with his recommendations. I also extend my gratitude to Dr Timothy Storer for his comments and aid in shaping the research and producing work with high-quality research. I really appreciate the support and the funds for my Ph.D. from the University of Hail and I am grateful to the school of Computing Science in University of Glasgow for its support. Furthermore, I would like to thank the academic community for sharing their knowledge and providing me with the necessary equipment for my research. Finally, I wish to extend special thanks to my family for their encouragement and support throughout my Ph.D. I am thankful to my mother, my wife and my daughter for their patience.

Contents

List of Tables

List of Figures

ACRONYMS

- **AB** Apache Bench
- **ANN** Artificial Neural Networks
- **AR** Auto Regressive
- **ARIMA** Auto Regressive Integrated Moving Average.
- **ARMA** Auto Regressive Moving Average
- **AWS** Amazon Web Services
- **CDOs** Cloud Deployment Options
- **CPU** Central Processing Unit
- **DT** Data Traffic
- **EC2** Elastic Compute Cloud
- **ES** Exponential Smoothing
- **GUI** Graphical User Interface
- **JVM** java virtual machine
- **K-Fold** CV K-Fold cross-validation
- **KCCA** Kernel Canonical Correlation Analysis
- **LM** Linear Regression
- **MCDM** Multi-Criteria Decision making
- **MIPS** Million Instructions Per Second
- **ML** Machine Learning
- **MPI** Message Passing Interface

- **NET** Networking
- **NoSQL** Not Only Structured Query Language
- **PWR** Power Consumption
- **QoS** Quality of Service
- **RAM** Random Access Memory
- **RMSE** Root Mean Square Error
- **RPI2** Raspberry Pi version 2
- **RPI3** Raspberry Pi version 3
- **SD** Secure Digital
- **SDN** Software Defined Network
- **SSH** Secure Shell
- **SVMs** Support Vector Machines
- **VM** Virtual Machine

Chapter 1

Introduction

1.1 Introduction

Cloud computing infrastructure is composed of large-scale distributed systems with complex architectural stacks (Armbrust et al. [32]; Mell and Grance [119]). Researchers are thus often unable to deploy cloud computing experiments on real cloud infrastructures, due to the prohibitive nature of the overheads in terms of time and cost, and therefore they often use simulators to model pertinent aspects of a cloud computing infrastructure to perform such experiments. This means that most cloud computing research is performed using simulators. There are many such simulators available, and different simulators are specialised for various cloud workloads and experiment types ("Makaratzis et al. [115]; Gustedt et al. [74]; Senyo et al. [132]). Developers of cloud computing systems may thus choose from a wide variety of simulators for use in their empirical research projects (Barker et al. [36]; Li and Deng [106]).

All software simulation tools provide opportunities to repeat experiments and to control variables with respect to the architectural model of a target data centre environment; however, simulators may have specific features and variants based on the typical use case for simulation. There are thus many limitations and trade-offs to consider when investigating data centre systems issues by means of simulation, which is one explanation for the vast number of cloud simulators available. As an alternative, researchers have therefore proposed employing micro data centres, based on micro servers (Hosman and

Baikie [78]; Kyaw et al. [103]) used instead of normal servers to reduce costs. According to Greenberg et al. [72], micro data centres have also been proposed for use alongside normal servers to improve performance and reduce costs. Micro data centres can therefore be used as platforms for researchers to conduct experiments Toosi et al. [143]. However, questions have been raised about the efficiency of micro data centres as a replacement of commercial servers by Zhao et al. [162], and different techniques that can be used to predict the performance of cloud computing data centres such as predictive models and emulation modeling are also popular. In this research, a predictive model is therefore applied to examine the performance of Raspberry Pi infrastructure. The most important objective of this research into a micro data centre built on Raspberry Pi is to use the predictions of performance to validate existing cloud computing simulators and to generate a publicly accessible data set for predicting the performance of large data centres with differing architectures.

1.2 Problem and Motivation

When developing an efficient cloud computing simulator, the capability of the simulator to model all aspects of cloud computing environments with appropriate levels of accuracy is the main influential factor. The level of accuracy required for simulation models can be checked by conducting a quantitative study via empirical comparison against actual measurements. The simulator capability can also be investigated by qualitative study that provides in-depth analysis of the existing simulator features. Despite the growing popularity of Cloud Computing simulators, their general level of accuracy has not yet been well investigated, however, and it is therefore difficult to choose an optimal simulator based on reliability. The precision of results produced by various cloud computing simulators relative to a particular real-world infrastructure can vary due to a number of factors, including infrastructure topology, benchmark task characteristics, and the fidelity of the simulator. Different network topologies clearly affect the performance of data centres in term of total execution time, but it is also known that different workloads behave differently based on the workflow of the workload and the way that it runs on the machines, whether

in parallel or sequentially. As each simulator is generally built for a particular type of experiment, the network topology and the workload behaviour must be considered during the modelling.

Other modeling tools, such as predictive models, can be used to predict the performance of data centre infrastructure. However, choosing accurate tools is difficult due to the lack of investigation of levels of accuracy in the literature. The difficulty in investigating level of accuracy lies in the complexity and cost of conducting the required cross validation experiments against large data centres; this cross validation work needs to be repeated, which makes it even more difficult to conduct verification work.

1.3 Thesis Statement

The purpose of the research statement is to articulate the way in which a methodology for evaluating a range of cloud computing simulators is demonstrated. Despite the fact that numerous cloud simulators have been designed to simulate cloud infrastructures, the current work focuses only on three: CloudSim, Mininet, and GreenCloud. Thus, the statement defines the scope of the research, as well as presenting a set of specific research questions to guide the study.

One of the chief benefits associated with the utilisation of cloud simulators in the context of business is dramatic reductions in capital expenditure and, critically, the freedom not to have to provision or maintain data centres. As aforementioned, there is a wide range of cloud infrastructure simulators available alongside the three that are analysed in the current research, such as CloudAnalyst Wickremasinghe et al. [152], NetworkCloudSim Garg and Buyya [70], and MDCSim Lim et al. [109]; in light of this, generating an understanding of various systems' respective cores, novel features, and disadvantageous features is the central motivating factor for developing an evaluation methodology.

This work is vital for cloud computing researchers using simulated models in their research, as well as for developers of cloud computing simulators. Researchers can benefit from understanding the level of accuracy, features, and advantages of several existing simulators, while this research is also important for developers of cloud computing simulators

wishing to prove the validity and reliability of their simulators, based on doubts about the validation of CloudSim in the work by Velho et al. [148].

Research Hypothesis: Different classes of modelling tools for Cloud Computing experiments provide characteristically different levels of accuracy. The accuracy of cloud computing simulators can, however, be systematically characterised using empirical data drawn from micro data centre configurations. Further, the run time of a micro data centre can be accurately predicted using appropriate Machine Learning algorithms trained on a data set of representative workloads.

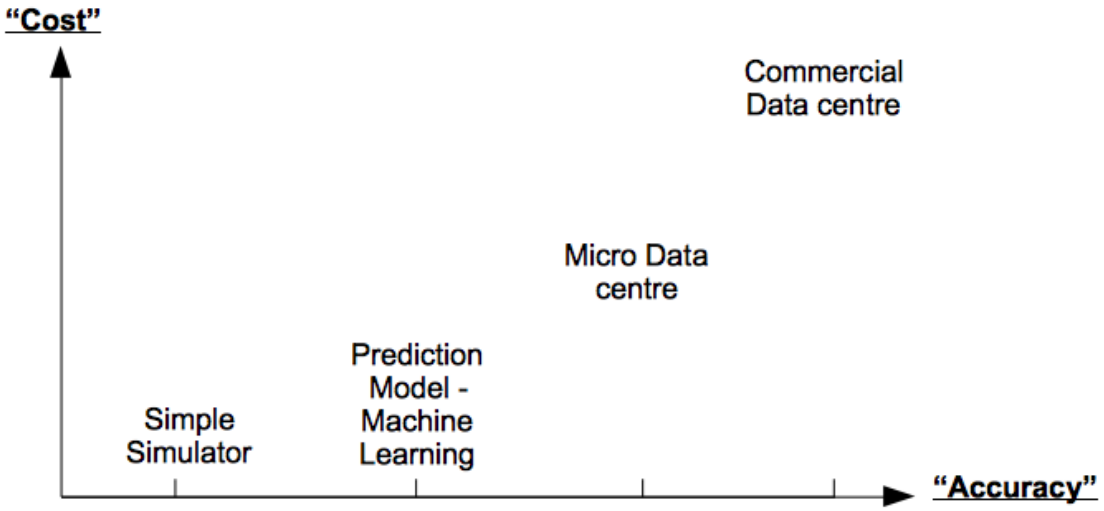

Figure 1.1: *The Hypothesis Overview Diagram, Showing Four Classes of Modelling Tools with their Cost and Accuracy*

Figure 1.1 shows the research hypothesis in terms of the differing levels of accuracy and costs for the different tools researchers may use for cloud computing experiments. For instance, simple simulators are generally cheaper than running experiments on commercial data centres, and the level of accuracy of simulation models' results is sufficient for many different types of workloads. Moreover, there are other tools that can be used to conduct experiments, such as prediction models based on machine learning models or the use of micro data centres to predict the performance of other infrastructure features. Therefore,

this research focuses on ascertaining and comparing the level of accuracy obtained by various different tools that may be used for cloud computing experiments by applying a comparative cross validation method on different modeling tools for Cloud Computing and assessing the gaps between simulated results and actual performance. The computed levels of accuracy will be presented for use in further research and the limitations and drawbacks of the simulation tools thus identified will also be highlighted for investigation. In this research, the following research questions are posed in order to address the overall research statement:

RQ1 How are existing simulators categorised in terms of delivering the required modelling and features for experiments on Cloud Computing? ***Chapters 3 and 5***

RQ2 What methods and benchmarks should be used to evaluate any Cloud Computing simulator? ***Chapter 6 and 7***

RQ3 How do simulators accurately model Cloud Computing experiments? ***Chapter 7***

RQ4 Which types of modeling tools, such as simulation models or statistical prediction modeling, can most accurately model a small scale data centre based on Raspberry Pi devices? ***Chapters 7 and 8***

This research is conducted with the aim of developing an accurate way to model cloud computing infrastructure with fewer absolute and relative errors for future researchers to use in their cloud computing research. This aim should be completed by fulfilling the following objectives:

- To study existing techniques for cloud computing modeling, such as simulation tools and predictive models, in terms of their drawbacks, advantages, and functional and non-functional characteristics.
- To generate a data set for the performance of micro data centres running against various benchmarks that can be used for evaluating existing models and as a training data set for developing a predictive model.

- To analyse the level of accuracy for existing simulation tools by applying a cross validation method against micro data centre infrastructure built on Raspberry Pi devices.

- To develop a predictive model based on the performance of Raspberry Pi devices that acts to predict the performance when running applications on different devices or different sizes of data centre.

- To validate the results from this predictive model by statistical analysis and cross validation.

This thesis responds to the conundrum of choosing an appropriate simulator by providing a systematic procedure for selecting a suitable simulator based on in-depth analysis of a set of existing simulators. This systematic procedure can be applied by researchers to any set of simulators. Moreover, the study of these simulators should allow researchers to characterise the different features of simulations relative to real cloud computing infrastructures. New features can then be developed for each individual simulator based on the results of investigating the limitations of existing simulators. Furthermore, formally analysing selected extensions should lead to a better understanding of how to extend a new simulator from any existing simulator.

To address the other questions, two types of investigations are conducted. A qualitative study into the advantages and limitations of a set of existing simulators in the cloud computing field is undertaken, followed by a quantitative study on a set of three simulators that applies cross validation experiments between the chosen simulators and a small scale data centre based in the researcher's university. This also offers the main source of data collection for the evaluation component of this thesis.

In order to develop a methodology for evaluating the target cloud computing simulators and for evaluating any cloud computing simulator, there are a range of techniques that can be used to verify computer simulation models delivered by (Sargent [131]and Beckert et al. [37]). In this research, we refer to empirical verification technique which has been applied in the current research by fulfilling the following steps:

- Carrying out a cross validation work between each simulator and the micro data centre.

- Existing benchmarks with parameterised configurations are run on the micro data centre.

- Modelling the benchmarks and infrastracture configurations on each simulator from the set CloudSim, GreenCloud, and Mininet.

- The results compared across each simulator

This first requires the collection of data sets for both the evaluation results and micro-benchmarks in order to sanity check the new simulators. This process is a critical part of the evaluation as it facilitates the collection of specific data to a greater degree than examining the general characteristics of each cloud simulator, which include flexibility, scalability, functionality, and effective performance in various aspects.

The data set parameters examined can be categorized into three areas: simulators; benchmarks and parameters; and other elements such as the topology and nodes forming the target data centre architecture. Along with this, two non-functional characteristics are also measured, as they are important factors when evaluating cloud computing simulators for actual use: execution time and power consumption.

The level of precision for different simulators is compared by modelling different workloads based on various features such as size, communication type, and computation models in all tested simulators. Selecting different workloads leads to modelling different types of applications and experiments on the simulators, and allows the research to provide a real infrastructure framework based on micro servers on Raspberry Pi v2 and v3 with profiling tools that automatically profile the performance of real workloads on actual infrastructure. This provides a standard framework for testing real infrastructure performance running various types of workloads. The infrastructure is based on Raspberry Pi devices as these are already used as data centres in several works in the literature on cloud computing experiments such as Abrahamsson et al. [19], Varghese and Buyya [147] and Toosi et al. [143].

Several types of micro data centres are used for various research and commercial purpose in field of cloud computing, as micro data centres offer excellent opportunities for conducting research into cloud computing in developing countries because of their reduced costs as compared to the containerised data centres often used in developed countries (Hosman and Baikie [78]).

Another way of modeling the cloud computing data centre infrastructure is prediction modelling based on machine learning algorithms; this uses the data set generated from an evaluation of a micro data centre to develop a predictive model for data centre architecture. This is used in this research to confirm the feasibility of using micro data centre infrastructure to predict the performance of other types of data centre.

Overall, future researchers will be able to use this empirical study of the features of existing simulators alongside other surveys in the literature to help select the most suitable simulators for their experiments. It will also allow researchers to invent new features for individual simulators or to develop new simulators with specific features. Evaluating existing simulators which possess the same features for a particular type of workload also helps provide a level of confidence for researchers conducting experiments on more than one simulator. Furthermore, quantifying the simulated performance of a cloud infrastructure relative to a matching real cloud infrastructure informs the developers of the level of accuracy of the simulators to either prove the validity of the simulator or to allow them to add new features to improve reliability and usability. Insofar as there are already numerous forms of cloud architecture, the value of this development of an effective evaluation methodology primarily lies in the fact that it addresses the current and growing demand for effective cloud computing simulators in both industrial and academic settings. As the evaluation methodology illuminates those areas most pertinent to cloud simulators, an effective system that allows users to identify the ideal cloud simulator for their work in the most efficient way is generated, helping to save energy, costs, and time. This work thus constructs and characterises a general framework that offers different types of benchmark performance to validate any new simulators arising.

1.4 Contributions and Publications

This research makes the following contributions to the field of cloud computing modeling and performance evaluation:

- A systematic literature review and a qualitative content analysis of existing simulators for the selection of the most suitable tools at the earliest stages of conducting cloud computing experiments. *(Chapter 5)*

- An evaluation of the performance of a micro data centre built on Raspberry Pi devices running different types of workloads; these are a Data Traffic (DT) running a message passing interface for high performance computing jobs; Apache Bench (AB) for web server applications, a Cassandra-Stress for NoSQL database applications, and word count on Spark for big data applications. *(Chapter 6)*

- Generating a method for accuracy and sanity checking any cloud computing simulators based on micro data centre infrastructure, and more specifically, presenting the level of accuracy of three extant cloud computing simulators, CloudSim, GreenCloud, and Mininet. *(Chapter 7)*

- Developing a predictive model for data centre performance based on micro data centre performance by applying Artificial Neural Networks (ANN) and Linear Modelling (LM). *(Chapter 8)*

Figure 1.2 shows an overview of the contributions delivered in the research. During the course of this research, two peer reviewed conference papers were also produced in collaboration with the supervisory team Alshammari et al. [28, 29]. Material from these papers is incorporated into chapters 5 and 6.

- **D. Alshammari, J. Singer, and T. Storer, Does cloudsim accurately model micro datacenters? in Proceedings of the 10th IEEE International Conference on Cloud Computing (IEEE CLOUD). June 2017,**
  DOI: 10.1109/CLOUD.2017.97

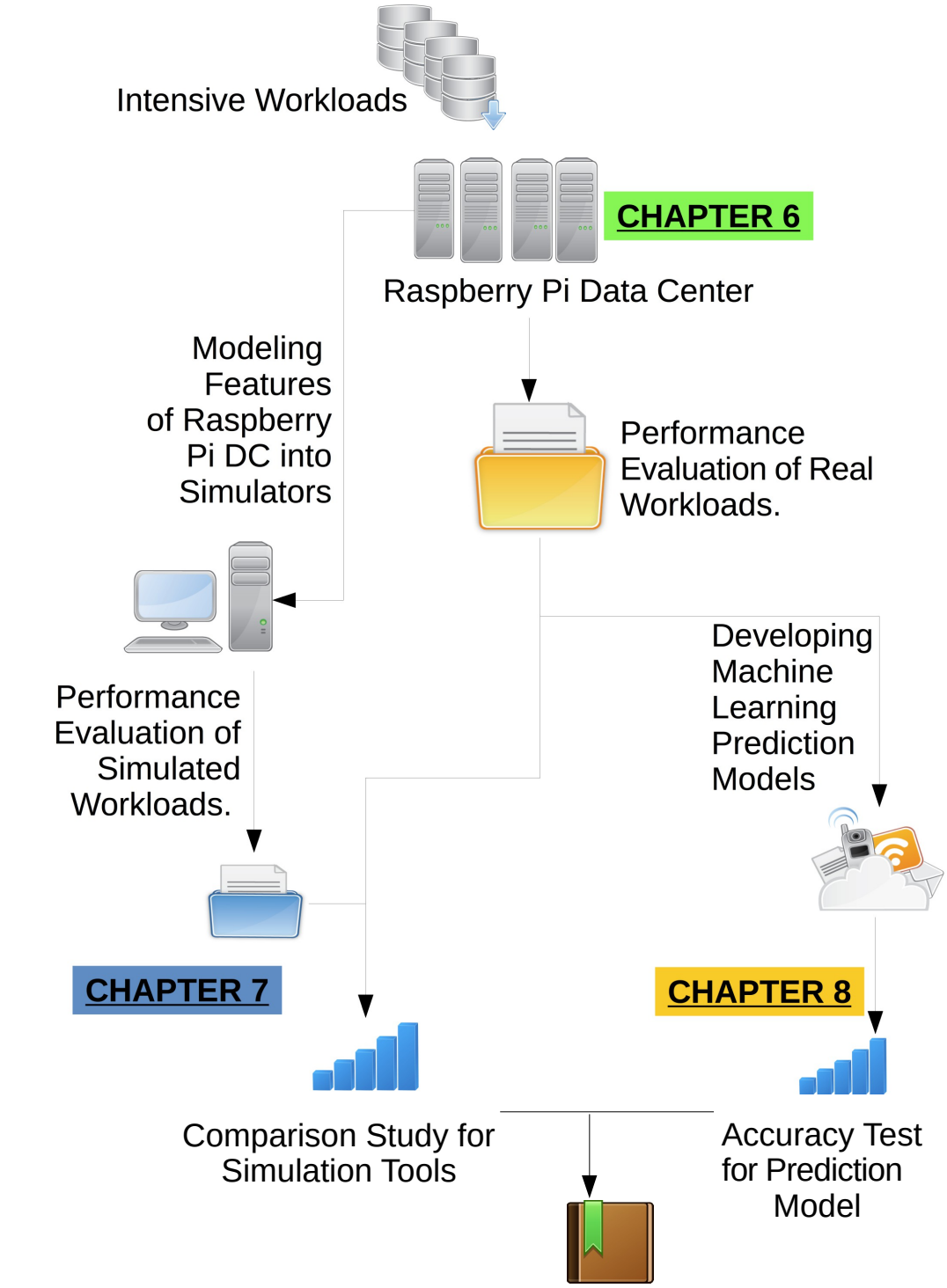

Figure 1.2: *The Outcomes of the Dissertation*

- **D. Alshammari, J. Singer, and T. Storer, Performance Evaluation of Cloud Computing Simulation Tools, in Proceedings of the 3rd IEEE International Conference on Cloud Computing and Big Data Analysis. (IEEE ICCCBDA) April 2018, DOI: 10.1109/ICCCBDA.2018.8386571**

1.5 Dissertation Outline

This section provides overview and briefly describes the proposed dissertation in my research as follow:

1. Chapter 1: **Introduction**

   This chapter presents the research hypotheses, statements, and questions. The contents of the thesis are briefly outlined to give the reader an overview of the dissertation's flow and content.

2. Chapter 2: **Background**

   This chapter explains the concepts and architecture of the selected simulators, which are CloudSim, GreenCloud, and Mininet. The concept of the micro data centre, built on Raspberry Pi. is also outlined. These descriptions represent the core of each platform without focusing on performance or validation. Existing verification and validation techniques for computing simulation models are also presented in this chapter.

3. Chapter 3: **Literature Review**

   This chapter examines existing studies in the area of cloud computing, particularly in the field of simulation modelling and verification. Key findings in the literature are linked to the current research to highlight similarities and differences. In addition, the chapter examines several different parts of the evaluation work, including evaluating the performance of cloud computing systems that built on micro servers, evaluating simulators in cloud computing systems, and examining existing machine learning models for predictive modelling of cloud computing experiments.

4. Chapter 4: **Research Methodology**

   This chapter presents the methods that used in the research. The first method was to select the most suitable simulators based on a systematic review of existing simulators, while the second method was to evaluate any cloud computing simulators based on data on micro data centre performance.

5. Chapter 5: **Qualitative Study of Existing Cloud Computing Simulators.**

   This chapter includes a description of several cloud computing simulators based on their features, proposed goals, architecture, and other concepts. Simulators that have not been explained and used in the verification analysis are reviewed and a historical analysis offered. Finally, the relational links between simulators designed for the same purposes are elucidated.

6. Chapter 6: **Performance of Benchmarking a Micro Data Centre**

   This chapter presents the performance results of the micro data centre running different workloads. More details about the tools and techniques are presented, including data centre structure, workloads, profiling tools, and the analytic test. The chapter reports on the performance of the micro data centre running intensive workloads such as Cassandra and Spark for big data application, thereby passing performance information into the public domain for further research. A comparison of the micro data centre against the real cloud data centre will also constitute a key component of the performance measurements.

7. Chapter 7: **Performance Evaluation of CloudSim, GreenCloud, and Mininet**

   This chapter focuses on explaining the methods used to evaluate the selected cloud simulators and presenting the results of simulated performance in existing simulators. The respective levels of accuracy are also compared with the actual performance of the micro data centre.

8. Chapter 8: **Predictive Model for Cloud Computing Experiments**

   This chapter delivers the predictive model that the study contributes to predicting

the performance of any data centre infrastructure; this is based on the performance of the micro data centre as evaluated in chapter 6.

9. Chapter 9: **Conclusion**

   This chapter concludes the work by presenting the key contributions, offering a dissertation summary, and providing recommendations for further work.

Chapter 2

Technical Background

2.1 Introduction

This chapter offers a general overview of the concept of cloud computing, the selected simulators and the realistic data centre based on Raspberry Pi that form the background of this research. This chapter thus defines and discusses cloud computing simulation, with a focus on the attributes and structures of CloudSim, Mininet, and GreenCloud. The Glasgow Raspberry Pi Data Centre is a small-scale data centre that is also examined in this chapter, with details of its architecture and software design being provided. This chapter also offers an overview of existing verification and validation techniques for computer systems. The subsequent chapter then moves into a comprehensive literature review of related research into cloud computing systems and tools.

2.2 A View of Cloud Computing

The concept of cloud computing emerged alongside the development of mainframe computing in the 1950s and was supported by the virtualization technologies that appeared in the 1970s Ahuja and Muthiah [22]. It is now widely accepted that cloud computing brings great advantages to modern businesses and supports the use of information technology in general (Marinescu [116]; Wang et al. [150]; and Gupta et al. [73]. According to Mell and Grance [119], "Cloud computing is a model for enabling ubiquitous, conve-

nient, on-demand network access to a shared pool of configurable computing resources e.g., networks, servers, storage, applications, and services that can be rapidly provisioned and released with minimal management effort or service provider interaction".
The main characteristics of cloud computing are as follows (Mell and Grance [119]):

- **On-demand self-service:** Allows consumers to access the required services, such as server time and network storage, without any human communication from the service provider side.
- **Broad network access:** Cloud computing facilities are available and accessed via network for several types of platforms from the consumer side.
- **Resource pooling:** Resources and facilities from the providers can be shared by different consumers with separate specifications for each resource based on consumer demand.
- **Rapid elasticity:** Efficiency of resource provision can be provided in automatic way of scaling outward and inward resources for end users based on demand.
- **Measured service:** The cloud system can control the leverage and provision of services through monitoring techniques for the utilized services for both providers and consumers.

According to Dillon et al. [60]; Marinescu [116]; Wang et al. [150]; and Mell and Grance [119] there are four classifications of cloud computing based on system accessibility:

- **Public Cloud**: Accessible at any time, from any place, by anyone via the internet. Examples of a public cloud are the Amazon Web Services Amazon Web Services (AWS) [10], Microsoft Azure [11], and Google AppEngine [12].
- **Private Cloud**: Specific to an organization, it overcomes the security issues regarding sensitive information that can be accessed in the public cloud.
- **Community cloud**: Differs from the private cloud as the providing organization has shared concerns e.g. security requirements, policy, and compliance considerations.

- **Hybrid Cloud**: Delivers a combination of public and private systems, providing services to both the providers themselves and the consumers.

There are three primary services that cloud computing provides to consumers:

- **Software as a Service SaaS**: This allows the consumer to use the providers' applications via a client interface such as a web browser. Limited access to configure the application may also be provided.

- **Platforms as a Service PaaS**: This allows end users to deploy and configure their applications to run on top of the provider's infrastructure, using the provider's libraries, programming languages, and other support tools.

- **Infrastructure as a Service IaaS**: This provides processing, storage, networks, and other fundamental computing resources to the end consumers to allow them to run their own software, operating systems, and applications. Users can deploy their applications without any consideration for the low level infrastructure such as network devices, storage, VMs, and CPU capacity; however, there are only limited possibilities in terms of selecting networking components.

Cloud computing is widely used in many areas of information technology as it effectively reduces costs and maximises benefits for contemporary business and companies (Armbrust et al. [32] Mell and Grance [119] Marston et al. [117]). However, cloud computing features inherent challenges, such as security, cost modeling, energy management, and virtual machine migration. Researchers thus conduct experiments to address cloud computing issues by using physical infrastructures, simulators, or mathematical models (Ahmed and Sabyasachi [21]; Barker et al. [36]).

The cloud computing field offers many pitfalls and opportunities for researchers. Work by Barker et al. [36] has generally argued that the academic researcher should concentrate their efforts more on software as a service and platform as a service aspects than on infrastructure as a service. Non-reproducible results are also a general pitfall in cloud computing research, as there are huge difficulties using real cloud infrastructure without a

simulator application such as CloudSim (Calheiros et al. [46]) or GreenCloud (Kliazovich et al. [100]).

Many academic researchers have been accused of simply rebranding grid activities and research to focus on cloud computing, while industrial users of the cloud are increasing production levels simply by using real cloud infrastructures. However, there appear to be several opportunities concerning future cloud research, including user-driven research and programming models, that could lead to more effective results in the academic field. Focusing on the elastic applications of the cloud rather than working on its infrastructure is thus advised.

2.3 Overview of Existing Cloud Computing Simulators

In order to evaluate simulation models for cloud computing experiment, the current work focused on choosing popular simulators for various common types of experiments. This led to the selection of CloudSim (Calheiros et al. [46]), GreenCloud (Kliazovich et al. [100]), and Mininet (Min [8]), which offer a diverse range of simulation types in terms of platform and purpose. All three simulators are able to simulate both computation and communication; however, these simulators were developed with different focuses, these being computation for CloudSim and communication in mininet, and power consumption in GreenCloud. Moreover, these simulators are built on different platforms: Java for CloudSim, C++ for GreenCloud, and a virtual environment to be run on top the physical machine for Mininet.

The simulation tools used in this research are based on mathematical equations that cause them to behave like the actual nodes. They have been programmed by developers based on the characteristics of actual data centres in computer programming languages such as Java, C, and Python. For example, in CloudSim, the class VM has attributes representing the features of virtual machines, while the class Cloudlet has the features of the tasks to be run on the machine. When executing a task, the VM is capable of executing 1,000 Million Instructions Per Second (MIPS), so the time for a 1,000,000,000 instruction task will be one second in CloudSim, resulting in the mathematical expression:

Execution time = Cloudlet. Number of Instruction / VM. MIPS

The predictive models in this research are based on the algebraic equations used to train existing data sets based on actual measurements. The prediction models give the predicted results based on the previous performance of the data centre and other data centre characteristics. The prediction model can predict the performance of data centres based on several types of data centre running similar workloads or the same data centre running different types of workloads.

It is thus clear that the simulation tools are based on only mathematical equations that are expected to behave like the actual data centre, while the predictive models are based on algebraic methods trained using actual data sets to predict unknown data centre performance.

In the next section, the base CloudSim, Mininet, and GreenCloud tools are described in detail. It is necessary to understand the architecture and design principles of these tools in order to appreciate the demand for extensions to the base CloudSim system made in Chapter 5. Moreover, it is important to deeply investigate the design and architecture of all tools involved in the research to justify the modeling results arising from the cross validation experiments on the micro data centre.

2.3.1 CloudSim

Cloudsim (Calheiros et al. [46]; Buyya et al. [43]) is a configurable generic tool that simulates and models cloud computing environments. Cloud concepts and entities are modelled as a set of Java classes that simulate the entire operation of the cloud computing systems and their workloads. It has several features that make it particularly flexible, such as the possibility of simulating small- and large-scale cloud computing deployments, including federated cloud and inter-cloud typologies.

Work by Calheiros et al. [46] generated a self-contained framework that can be used to simulate the workload of the cloud using an end-to-end network architecture. Figure 2.2 shows the architecture of the CloudSim framework insofar as it consists of data centre, hosts, virtual machines, and tasks. The CloudSim framework (Calheiros et al. [46]; Buyya et al. [43]) is thus logically divided into various modules:

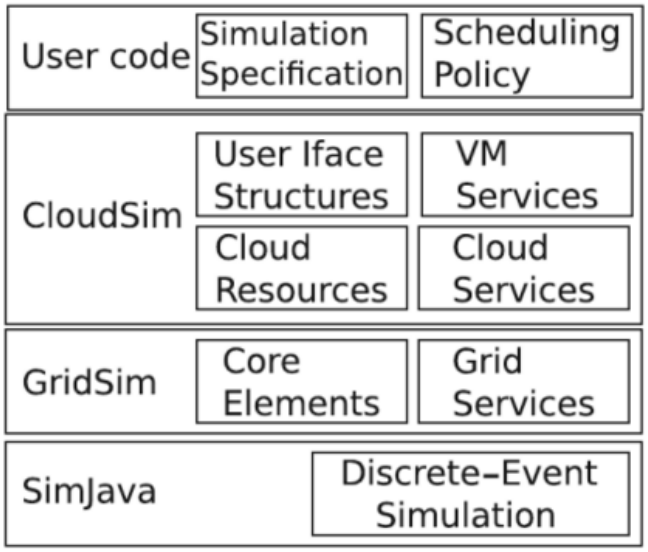

Figure 2.1: *The Architecture layers of CloudSim Framework Dubitzky et al. [62]*

- User Code: code that implements the user requirements and the application configuration.
- Broker: algorithms that link user interfaces and the data centre and schedule policies.
- User Interface: acts as a bridge between the user and the simulator, which lies between the user code and the underlying cloud resources.
- VM services: performs execution of the application and management of the VM Cloud Services; this covers several activities such as provisioning VMs, and allocating underlying physical resources (CPU, memory, etc).
- Cloud resources: CloudSim must coordinate low-level events in the data centre.
- Network: management of network features, such as message delay calculation and network topology.

CloudSim core simulation is based on top of GridSim (Buyya and Murshed [44]), which in turn is built on top simjava (Howell and McNab [79]). However, GridSim Buyya and Murshed [44] manages the simulation inside the framework based on simjava's event simulation model, which is a tool-kit for building working models of complex systems. However, CloudSim 2.0 and later releases manages event simulation internally, rather

than using simjava (see Figure 2.2). This overcomes limitations in simjava, such as an inability to reset events during the run time, overheads creation and complexity in the multi threading nature of simjava. Instead, CloudSim stores future event in a queue until they are due for execution, and then moves completed events to a DeferredQueue which provides flexibility management of simulation. The CloudSim design enables four capabilities for each event in the simulation: deactivation, switching of entities, creating new entire and restarting the simulation at run time Calheiros et al. [46]. Figure 2.3 shows the core event management framework for CloudSim. Several key CloudSim (Calheiros et al. [46]) classes are outlined below as the fundamental classes that represent the simulation of cloud computing environment in CloudSim:

- `Cloudlet`: to model application-level services.
- `CloudletScheduler`: to class is extended by the implementation of different policies that determine the share of processing power among cloudlets in a VM.
- `Datacenter`: to model infrastructure-level compute services.
- `Datacenter Broker`: to simulate the broker in the cloud with responsible on linking the layers SaaS with cloud providers.
- `BwProvisioner`: to model the policies of provisioning of bandwidth to the VM.
- `CloudCoordinator`: to extend the data centre of the cloud to the federation.
- `Host`: to models a physical nodes as resources in data centres .
- `NetworkTopology`: to indicate network behavior
- `RamProvisioner`: to allocate primary memory (RAM) to the VMs.
- `SanStorage`: to model a storage area network.
- `Sensor`: to be interface that must be implemented for Cloud Coordinator in monitoring specific performance parameters.

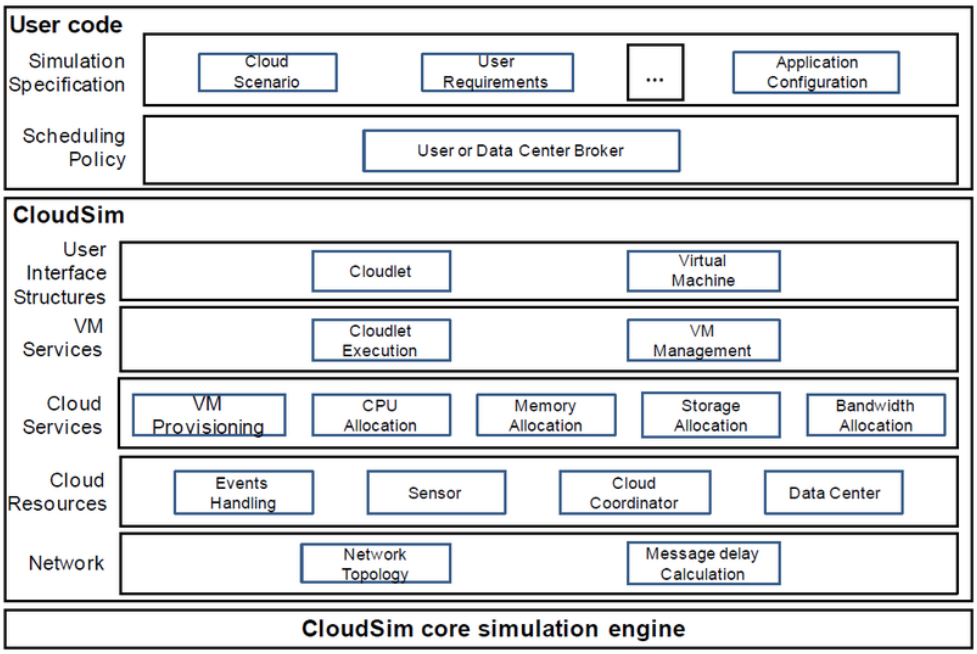

Figure 2.2: *The Design of the CloudSim, reproduced from Buyya et al. [43]*

- `Vm`: to model a virtual machine that runs on a particular host with its specifications such as RAM

- `VmmAllocationPolicy`: to set provisioning policy that a VM Monitor utilizes for allocating VMs to hosts.

- `VmScheduler`: to apply allocating processor cores to VMs by a Host component that models the policies (space-shared, time-shared).

Figure 2.4 shows the class design diagram of CloudSim, demonstrating that CloudSim has provision classes, allocation policy classes, and scheduler classes. Provision classes are responsible for providing the cloud data centre with all its major components, such as virtual machines, memory, and bandwidth; allocation policy classes manage time- and space-sharing allocation policies; and scheduler classes are responsible for implementing the time- and space-sharing allocation policies in terms of allocating cores for VMs (Calheiros et al. [46]; Atiewi and Yussof [33]; Buyya et al. [43]).

Initial releases of CloudSim used SimJava (Howell and McNab [79]) as the discrete event simulation engine, but CloudSim has been developed through six official public releases, and the latest version of CloudSim is v4.0, which provides support for container

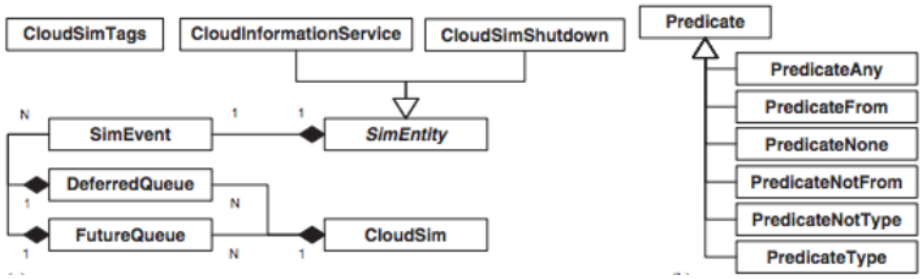

Figure 2.3: *CloudSim core simulation framework class diagram, reproduced from Calheiros et al. [46]*

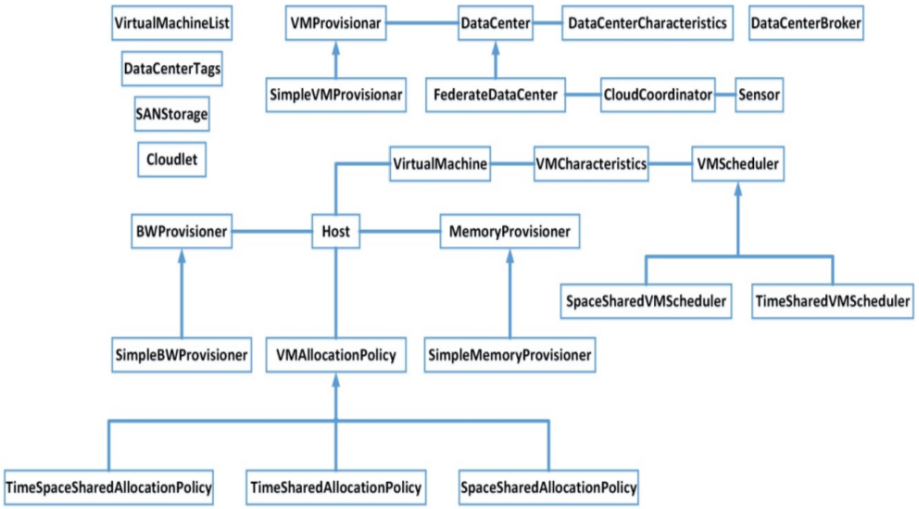

Figure 2.4: *The CloudSim design diagram, reproduced from Calheiros et al. [46] Buyya et al. [43]*

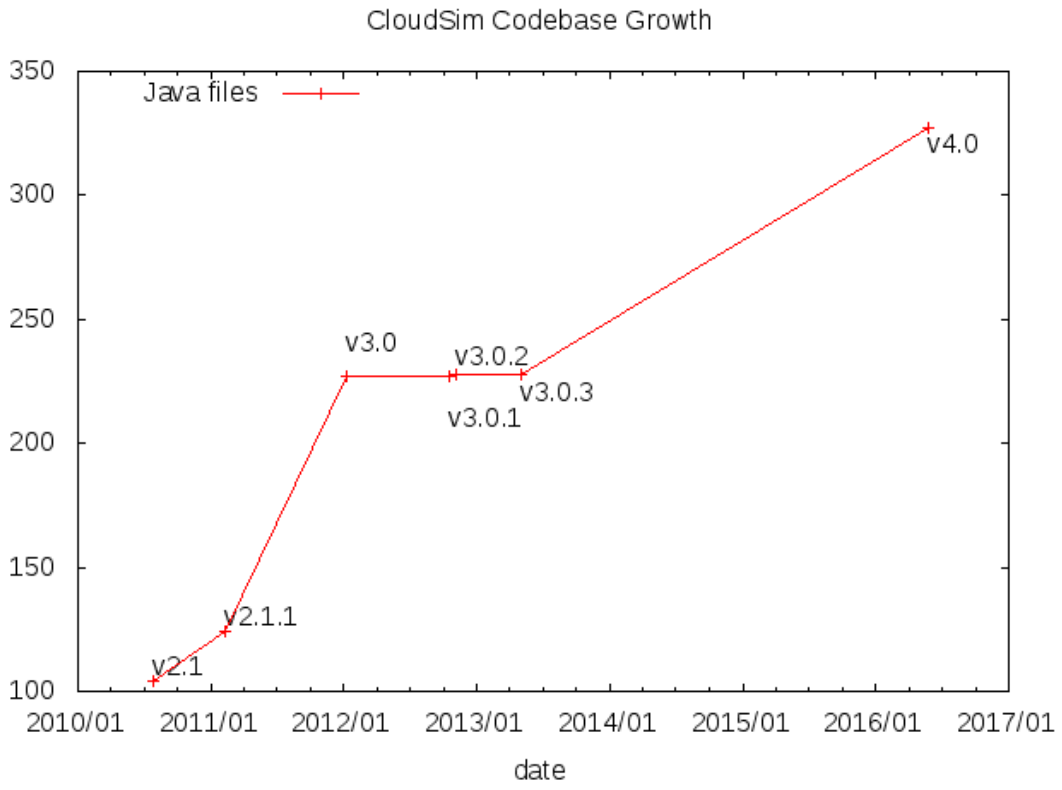

Figure 2.5: *Growth of CloudSim in Number of JAVA Files*

virtualization, required by many mainstream cloud providers (Clo [3]). The previous release (v3.0.3) improved the mathematical accuracy and allowed the minimum time between events to be user-configurable. It also added a new constructor for defining the data size to increase simulation granularity. Generally, new public releases of CloudSim aim to improve simulations by modelling new entities (e.g. internal networks) or supplying new policies (e.g. for VM selection) Clo [3]. Figures 2.5 and 2.6 show the growth of the framework, with an increase in number of java files and code lines, since it was first publicly released in 2010 to 2016.

CloudSim has been used as a simulator in different areas of cloud computing such as resource allocation management (Abar et al. [18]) and energy consumption (Kord and Haghighi [101]; Shaoling et al. [133]; Whittington et al. [151]). CloudSim has also been extended to produce many alternative simulators such as NetWorkCloudSim (Garg and Buyya [70]), DynamicCloudSim (Bux and Leser [42]), DartCSim (Li et al. [107]) and Cloud2Sim (Kathiravelu and Veiga [89]) in order to provide more features and properties.

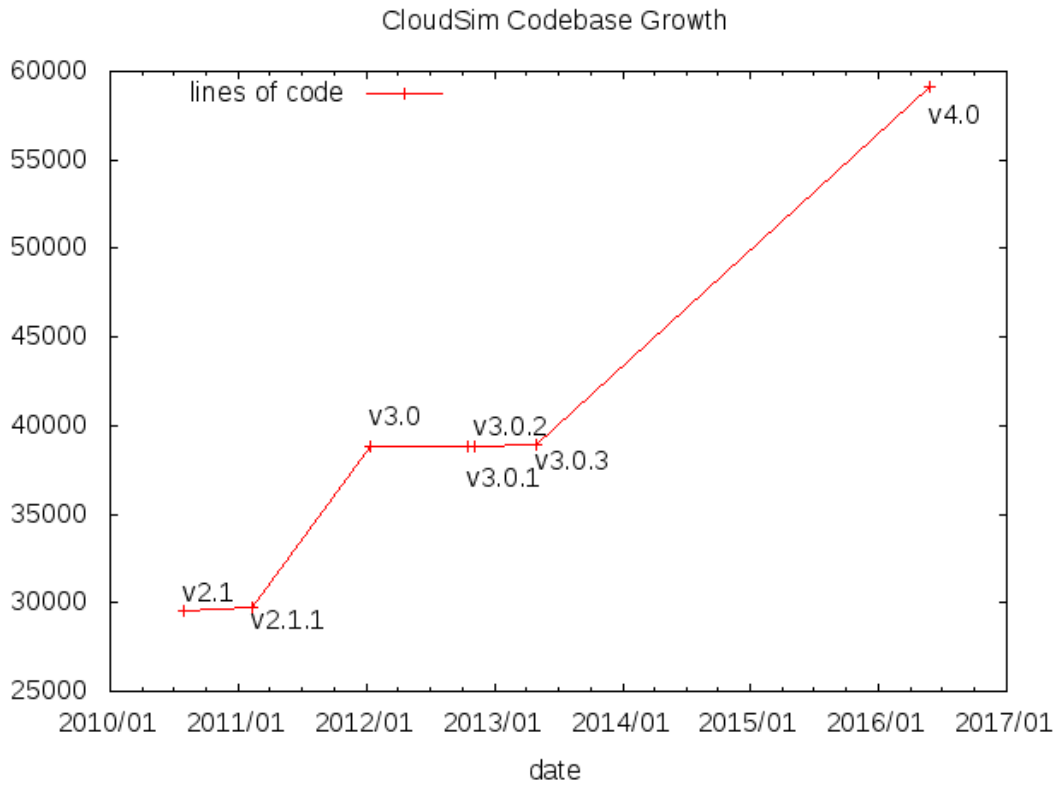

Figure 2.6: *Growth of CloudSim in Terms of Number of Lines of Code*

2.3.2 Mininet

As a network emulator, Mininet (Min [8]) runs a collection of end-hosts, switches, routers, and links on a single Linux kernel. Capitalising on the cryptographic network protocol Secure Shell (SSH), hosts on Mininet are able to be connected as if they were real hosts. Furthermore, Mininet hosts as described in De Oliveira et al. [56] can run real workloads for web servers, an example being the HTTP benchmarking tool Apache Bench (ABa [1]), iwhich is used in the current research. Mininet also permits the installation of several profiling tools, including iperf and perf, which can be run to provide measurements of application performance. As Mininet affords the possibility of creating a realistic virtual network that resembles a hardware network based on several features, Mininet overcomes the complexity and overheads that can be caused by the use of virtual networks or containers on physical machine as research tools (Lantz and O'Connor [104]).

In this research, a Python script that models the ab benchmark was developed. This Python script can be run on Mininet and the ensuing results can be comparatively ex-

amined against actual performance using the Raspberry Pi data centre, with the latter running actual benchmarks. In this research, Mininet was run on a machine with an intel Core i5-4690 processor with 16GiB of memory and a 500 GB disk.

2.3.3 GreenCloud

As a packet-level simulator, the fundamental use case for GreenCloud relates to the domain of energy-aware cloud computing data centres (Kliazovich et al. [100]). For the most part, the data centres GreenCloud targets are associated with cloud communications. It is thus important to recognise that the GreenCloud simulator represents an extension of the widely-used NS2 network simulator, as released by a General Public License Agreement. One of the strong points of the GreenCloud simulator is the potential it affords in modelling levels of energy consumption associated with information technology equipment within a data centre. The equipment that GreenCloud is most effective at modelling in this respect includes computing servers, communication links, and network switches. Figure 2.7 shows the GreenCloud Simulator's architecture.

GreenCloud can be used to develop new techniques for resource allocation and workload scheduling, and it has proven its value in the optimisation of communication protocols and network infrastructures, the latter two applications being closely linked to the efficient power management of data centre infrastructure. In the present research, the features of the Raspberry Pi data centre (which runs MPI) were fed into GreenCloud, including MIPS, algorithms, and execution time. This was done because it enabled a comparative examination of the simulated results against real measurements of the serversâĂŹ energy consumption. The focus was thus on the server segments of the simulated results. The rationale for this stems from the fact that, within the selected infrastructure, only measure the energy consumption of the Raspberry Pi nodes could be measured.

2.4 The Glasgow Raspberry Pi Cloud

A large-scale cloud computing infrastructure is difficult to use for experiments in cloud computing research due to high costs and the time-consuming nature of such use. Repeat-

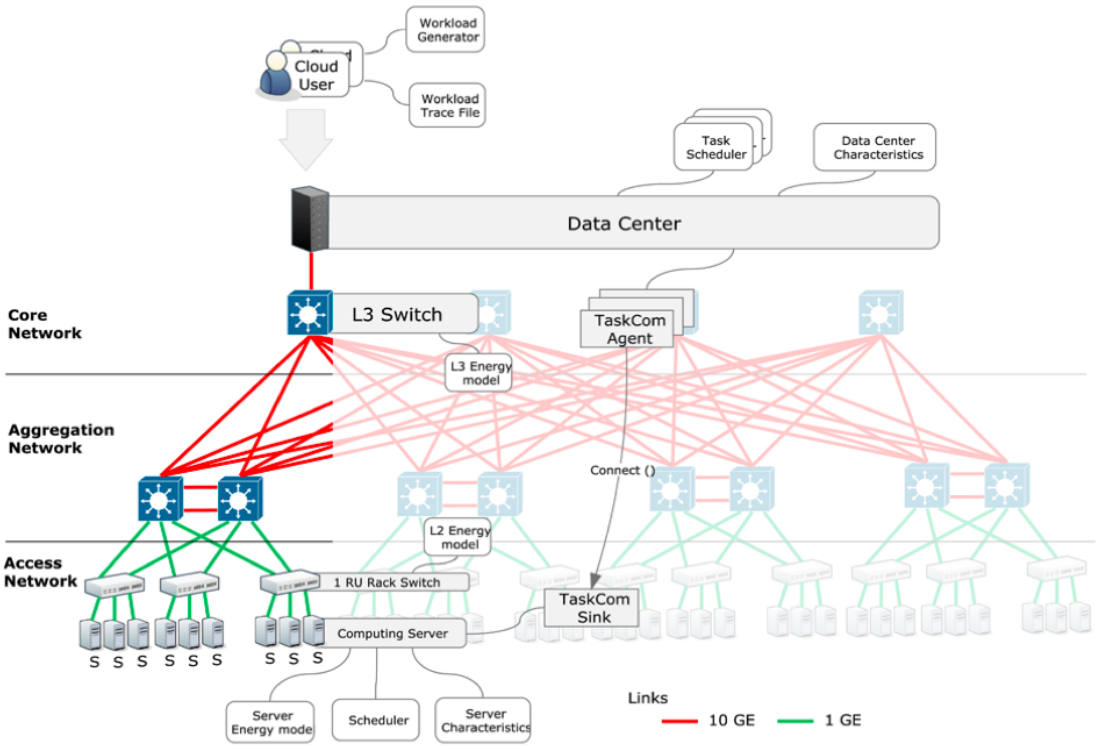

Figure 2.7: *The GreenCloud Architecture, reproduced from Kliazovich et al. [100]*

ing experiments is possible using real cloud computing providers' infrastructure, but it is extremely difficult work. However, the limitations of cloud computing simulators can affect the results of experiments that are run with a simulated environment. Low- cost Raspberry Pi devices thus offer the opportunity to build scale-model cloud computing environments such as data centres, virtual machines, and networks to examine their behaviours; these can be tested as network nodes and offer valid results for such experiments.

Tso et al. [144] developed a cloud computing test-bed containing 56 Raspberry Pi devices that are used to create a small data centre. This is called the Glasgow Raspberry Pi Cloud, and they have provided the design and architecture of the network to the public. The Glasgow Raspberry Pi Cloud is a part of Picloud, running Linux from a Sandisk 16GB SD card storage device. As a cloud computing environment requires a server for the data centre, authors have tended to use Raspbian as a server in the cloud infrastructure. It is clear that Raspberry Pi has several advantages, such as its minimal power and cooling needs, in terms of running any cloud computing experiment repeatedly. The Glasgow Raspberry Pi Cloud has thus been structured as a scale model of a cloud computing

data centre to allow authors to try to avoid the drawbacks of both real environments and simulations of the cloud.

Avoiding the drawbacks of real cloud providers in terms of costs is a clear driver, and the reasons for using Raspberry Pi devices as a substitute are obvious. However, simulator systems have been used in a great deal of published research, and the exact limitations of this approach are less clear. It would therefore be advantageous if real experiments were done comparing the same workload attributes in all three cloud computing environments: a cloud computing provider's 56-machine test-bed, the Glasgow Raspberry Pi Cloud, and an appropriate simulator system. Work on monitoring clouds, intrusion detection systems, and power consumption of the cloud has been done to some extent in the work of Tso et al. [144], who focused on replacing the virtualisation container, Linux Containers LXC, with Docker and applying Spark Hadoop instead of Hadoop in order to increase the performance of the data centre. Raspberry Pi version 2 from the University of Glasgow supports existing small data centres becoming more powerful, however, as the Raspberry Pi 2 has several enhanced features such as a 4-core CPU, 1 GHz clock speed, and 512 MB of storage. With version 2 of Raspberry Pi, the small-scale data centre can be used to investigate a new framework for an algorithm that matches three input rules (power consumption and cost for cloud providers, quality of service attributes for consumers, and green cloud computing requirements).

The low power consumption of Raspberry Pi devices leads to lower running costs, while the capacity of version two Raspberry Pi provides better performance for different types of workloads. In terms of green computing aspects, it is also clear that the way that Glasgow Raspberry Pi Cloud built does not necessitate a cooling system, which reduces the impact to the environment and lowers costs.

Figure 2.8 shows the system architecture of the Pi cloud and Figure 2.9 presents the Pi Cloud software stack.

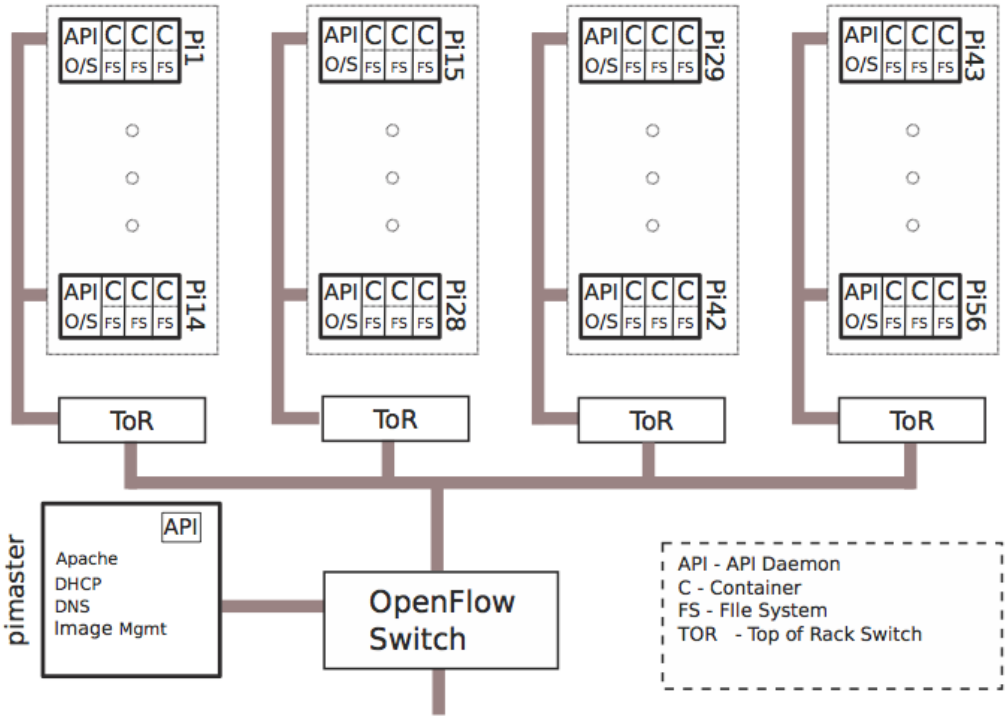

Figure 2.8: *System Architecture of the Pi Cloud, reproduced from Tso et al. [144]*

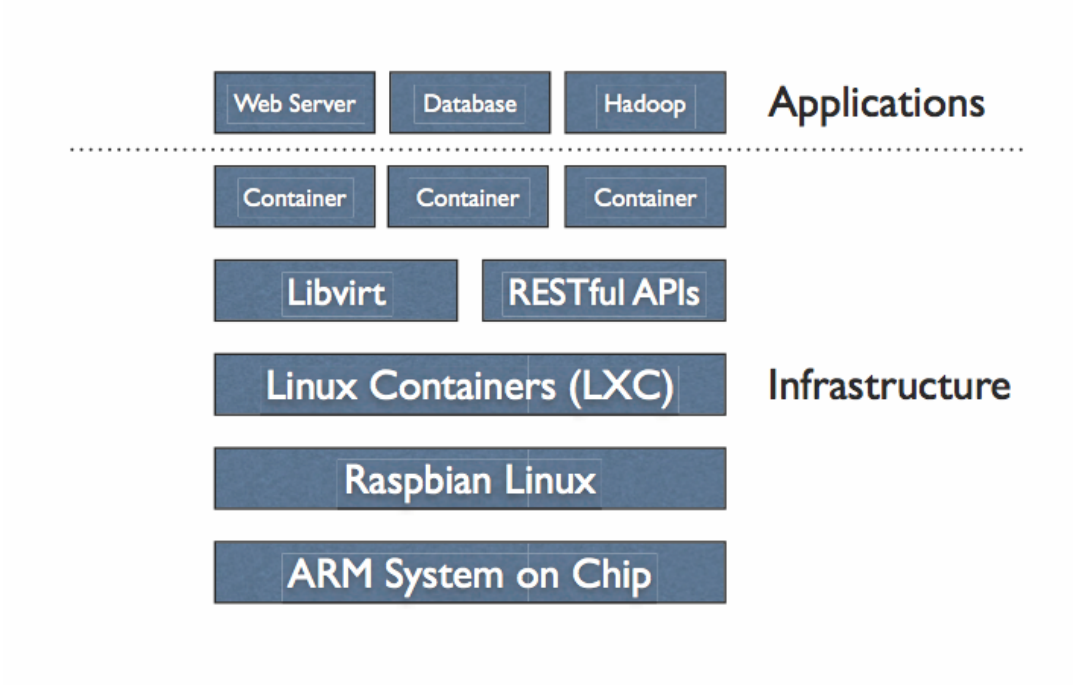

Figure 2.9: *The Pi Cloud software Stack, reproduced from Tso et al. [144]*

2.5 Verification and Validation Computer System Simulations

According to the work of Beckert et al. [37], the verification and validation of computer system simulators is a checking process intended to ensure that various requirements have been met. These verification processes are conducted during the preliminary stages of software development and can be described as a static way to check the documentation of the software by means of experts reviewing the software design. Validation is the process of checking the output and the results of software against the expected outcomes, which can be done either by the developing team or the end users. Once the final version of the software has been produced, a further validation phase is conducted by testing the capabilities and features of the software to ensure they meet requirements.

As indicated in Sargent [131], the verification of computer system simulations is not a well-investigated topic in the literature, and specifying particular techniques or algorithms to be applied for any computing simulators to determine the level of accuracy is not adequately explained.

Validating simulators assesses the accuracy of the simulator for a particular experiment; it is not possible to generalize how accurate the simulator is in general based on a specific experiment. Validating the overall level of accuracy for simulators can thus be conducted by either comparison with the real system, comparison with the analytic model, or by applying engineering judgment (Lilja [108]). In the current research, testing was performed different types of workloads and different attributes measured in various cloud computing experiments in order to determine the level of accuracy achieved by the model within different types of experiments.

A Multi-Stage Verification (MVS) approach was identified in Kleijnen [99]; this is a combination of three stages that allows the computing model to be verified in its entirety. The three stages for MVS are as follows:

1. Study the formulation and the core to determine whether the simulator for set hypotheses describes the behaviour of the system modelled in the simulation

2. Produce test for the hypotheses in the system analysis section, thereby verifying the models regarding the limitations of the existing statistical tests

3. Test the models' ability to predict the computer system under study.

According to Twomey and Smith [146] four other basic approaches for validation and verification of computer simulation tools are as follows:

1. Validation experiments are conducted by the developing team

2. If the team is small, the validation work can involve end users

3. Independent validation can be carried out by a third party to increase the objectivity of the judgements; prior research generally agrees that the first two approaches generate somewhat subjective results.

4. 4. The simulation can be tested on various aspects of various experiments and the overall result for the different tests can then be calculated to generate what is known as a Scoring Pass.

There is a need to test multiple scenarios and different types of experiments on each specific simulation tool in order to generally state the validation and accuracy level of any tools used. As the application of verification analyses can be used to evaluate any simulator by comparing its outcomes with historical observations of either existing valid simulators or actual test-beds, in the current research, analysis of different simulation and prediction tools was applied by means of comparative studies against a micro data centre performance.

2.5.1 Validating and Verification Techniques

Several techniques for validation and verification are collected in Sargent [130] and these are defined as objective or subjective methods. Objectivity in this context refers to being mathematically and statistically tested via an existing validity test such as hypothesis tests or confidence intervals. These techniques for validation and verification include

- Animation: Validating the simulation graphically through the duration of running the simulation by changing some factors inside the model.

- Comparison to Other Models: Comparing the outputs of the simulation with existing valid results from another model, such as a real test-bed or a valid simulation.

- Degenerate Tests: choosing an appropriate selection of values for the input and internal parameters to validate the simulation behavior.

- Event Validity: Comparing events or occurrences in the simulation model with the real system.

- Extreme Condition Tests: checking the models' output to be plausible for any extreme and unlikely combination.

- Face Validity: checking the behavior and logic of the system reasonableness by asking individuals knowledgeable about the system.

- Historical Data Validation: using the existing data about the performance of the system to test the new outcomes.

- Historical Methods: these are rationalism, empiricism, and positive economics. Rationalism is used when the assumptions of the system are recognized as true. Empiricism requires every assumption of the system to be empirically validated. Positive economics focuses only on the ability of predicting the future without concern for the structure correctness and logic of the system.

- Internal Validity: running several experiments on the system to check the stability of the outcomes graph.

- Multistage Validation: linking the three historical methods in one method in three steps: 1) checking the assumption and logic of the system through its documentation; 2) validating the assumptions of the system empirically; 3) comparing the inputs and outputs of the system with the real system.

- Operational Graphics: checking the values of variables measurements through running time graphically to ensure that the software meets the expected performance in a specific period of time.

- Parameter Variability and Sensitivity Analysis: applying signification changes in the inputs of the system in order to check the changes on the outputs to be compared with different statues statistically.

- Predictive Validation: predicting the systems' behavior and subsequently comparing it with the real system.

- Traces: tracing the behavior of the system on different specifications to check if the models' logic is correct.

- Turing Tests: checking with individuals who are experts in the system to see if they can recognize the outputs from the model and differentiate them from the real system output.

Sargent [130] has also also presented the modeling process and the validation and verification steps for a simulation against the real world, as seen in Figures 2.10 and 2.11. The main verification techniques arising from the existing research are:

- User modular programming concepts

- Testing entities into Model

- Checking the simulators based on arrays structures and logic

- Directional Analysis (the relationship between expected actions)

- Checking if the entities reach all parts of the module

The validation methods can be classified in three ways:

- Asking individuals to critically review the entire model.

- Asking individuals about the results in terms of which results are collected from the modules and which are collected from the real system.

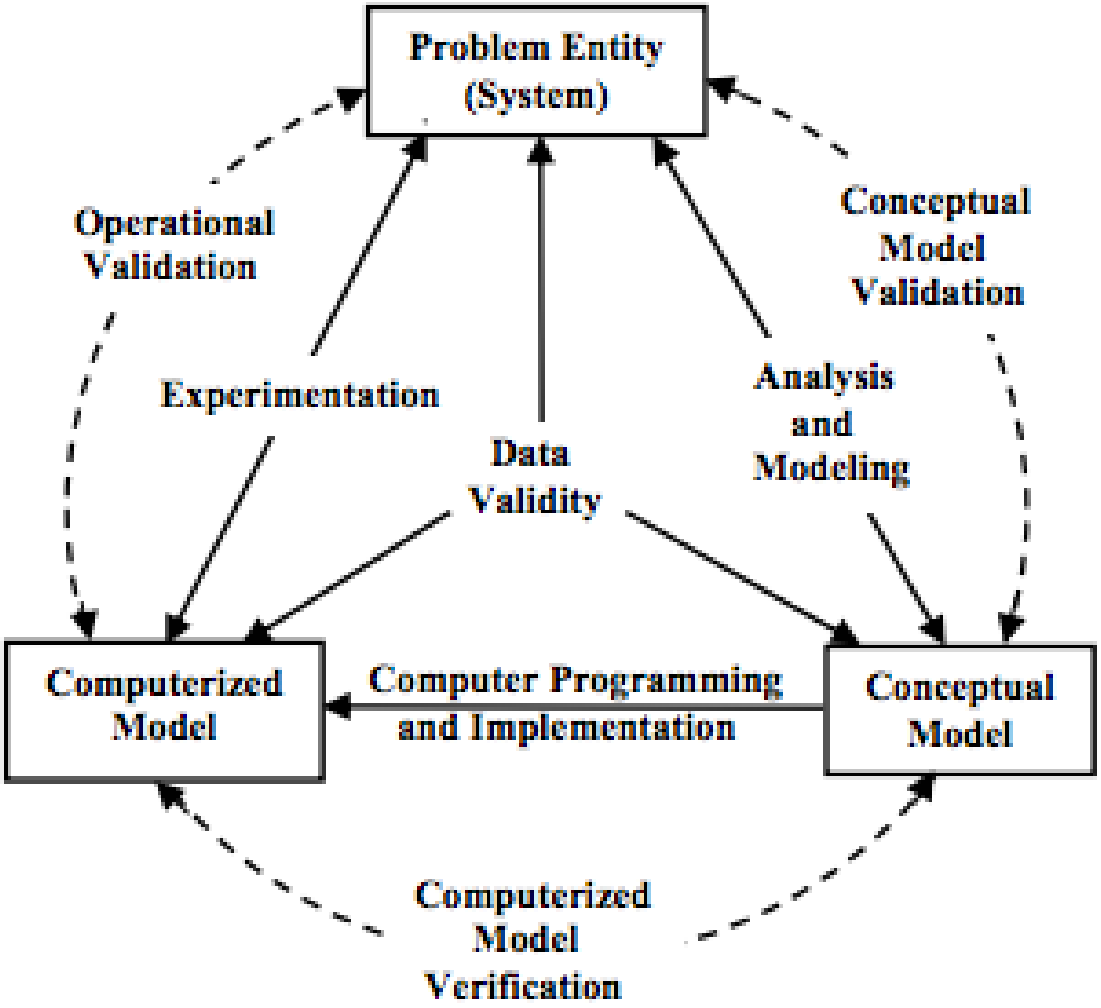

Figure 2.10: *Simplified Version of the Modeling Process Sargent [130]*

- Statistically analyzing the outputs from the simulators and comparing with the real system.

In this work, for the verification sections, the entities and models for each simulation tool were checked before the model was simulated on the Raspberry Pi cloud to make sure that it could be modelled without any modification to the simulator's cores concepts. Moreover, statistical tests were conducted on the outputs of each simulator and the results compared with real test-bed outputs, a method listed as one of the validation techniques in Sargent [130].

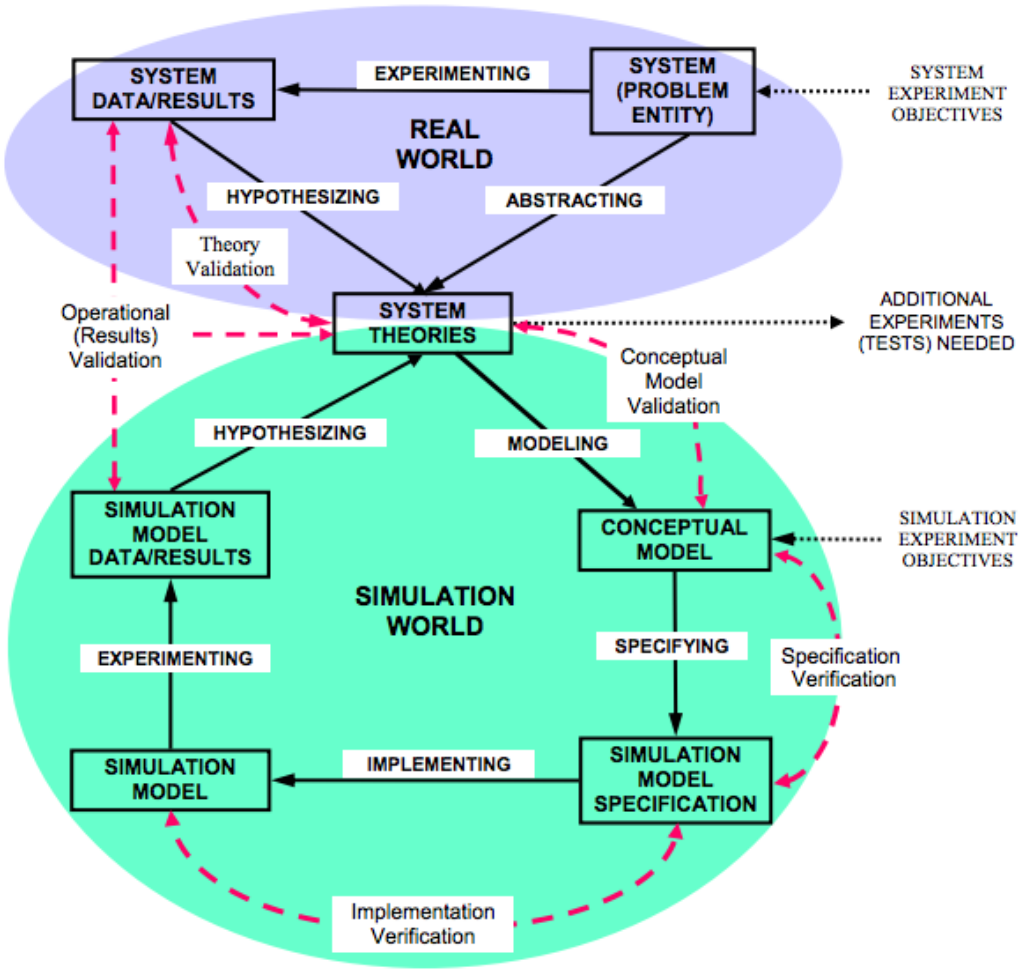

Figure 2.11: *Real World and Simulation World Relationships with Verification and Validation Sargent [130]*

Chapter 3

Literature Review

3.1 Introduction

This chapter provides an overview of existing methods and techniques for modelling Cloud Computing experiments, including simulators and machine learning models. The opportunity offered by using micro data centres and low cost and power computing resources is also explored in the chapter, which highlights the usability of Raspberry Pi devices in the field of Cloud Computing. This chapter also presents related work in terms of performance evaluation and the prediction modelling tools for cloud computing and how these relate to the current research.

3.2 Performance Analysis Goals

Measuring performance analysis for computer systems offers the possibility of comparing alternatives, determining the impact of features of a system, and tuning and setting expectations for future ideas of system capability (Lilja [108]). Computer system performance can be evaluated by different methods that are delivered by Bukh [41] and Lilja [108] as:

- Predictive Modeling
- Real Time measurement.

Researchers who aim to conduct research on cloud computing will choose either simu-

lation tools, mathematical models in case of using predictive modeling methods, or actual infrastructure for the real time measurement methods. However, there are different levels of accuracy and costs associated with using these different models. Lilja [108] claims that simulation accuracy, simulation time, and the non-functional requirements of simulation tools lead to complexity and decrease efficiency; thus, cloud computing simulators should be evaluated where functionally correct, but this does not meet the expectations of a real experiment.

3.3 Existing Cloud Computing Simulators

The use of cloud computing has rapidly increased, with many applications arising for diverse types of deployment. As it is difficult to measure and test application performance in a real cloud environment, simulators have been employed to evaluate algorithms or policies with more ease (Ahmed and Sabyasachi [21]; Senyo et al. [132]). Selecting the appropriate simulation tool for an experiment can prove problematic for researchers, however. Although there are several simulation tools available, as discussed in Maarouf et al. [113]; Kaleem and Khan [87]; [161]; Ahmed and Sabyasachi [21];Byrne et al. [45] there is little guidance or information provided for researchers in terms of suitability or accuracy for a specific function in each case. Moreover, cloud computing simulators are ill-equipped for critical points of study in cloud systems, as they cannot model every case in the system and are generally developed with the objective of duplicating the non-functional features of the cloud system (Souri et al. [139]).

Ahmed and Sabyasachi [21] analysed existing simulator information and divided it into different categories:

- Underlying Platform : some of the simulators are underlying of SimJava such as CloudSim (Calheiros et al. [46]) and SimIC (Sotiriadis et al. [138]). Other simulators are extensions from others simulator such as NetworkCloudsim (Garg and Buyya [70]) which is an extension of CloudSim Calheiros et al. [46].

- Availability : most of the simulators are open source. However there are some

simulators are not available yet -till the time of writing this thesis- such as SimIC (Sotiriadis et al. [138]) and MR-Cloudsim (Jung and Kim [85]).

- Programming Language: Java, C++ and otcel
- Cost Modelling: CloudSim (Calheiros et al. [46]), CloudAnalyst (Wickremasinghe et al. [152]), iCanCloud (Núñez et al. [121]), NetworkCloudSim (Garg and Buyya [70]), EMUSIM (Calheiros et al. [47]), MR-CloudSim (Jung and Kim [85]), DCSim (Tighe et al. [142]) and SimIC (Sotiriadis et al. [138]) provide opportunities to model and simulate the cost of cloud computing architecture.
- Graphical User Interface GUI: Most of the simulators that been provided in this paper do not support the GUI except CloudAnalyst and some limited GUI in GreenCloud Kliazovich et al. [100] and GroudSim Ostermann et al. [122].
- Communication Model: GreenCloud (Kliazovich et al. [100]), iCanCloud (Núñez et al. [121]) and NetworkCloudsim (Garg and Buyya [70]) can fully model the communication action on actual data centre infrastructure.
- Simulation Time: Seconds are likely to be the unit of execution time granularity for simulated jobs and workloads. GreenCloud (Kliazovich et al. [100]) and DCSim (Tighe et al. [142]) use minutes to calculate the time execution of the task in case of intensive workload and big data models.
- Energy Model: Energy model is so important to be simulated as it is clearly known that data centre infrastructures are consuming power and energy. This survey shows that iCanCloud Núñez et al. [121], GroudSim (Ostermann et al. [122]) and DCSim (Tighe et al. [142]) do not support the energy modelling.
- Federation Policy: CloudSim (Calheiros et al. [46]) and its extension have been shown as simulators that can model the federation cloud. SimIC (Sotiriadis et al. [138]) can also model the federation policies.

Kaleem and Khan [87] and Zhao et al. [161] provide descriptive studies of some existing simulators, briefly describing them in terms of their underlying platform programming

language, availability, supporting communication modeling, and simulator type (event based or packet level). However, these works lack investigation into the structure and the level of accuracy, or ways to select the optimum simulator based on features. Maarouf et al. [113] provides details of existing simulators and their features and characteristics, thus enabling researchers to gain an overview of each simulator prior to deploying them in their experiments.

One of the existing simulators in the field of cloud computing is called DISSECT-CF (Kecskemeti [91]) which was developed to overcome existing limitations in the simulation of cloud computing system based on performance issues in scheduler techniques. This simulator supports the energy evaluation of IaaS as applied in different scheduling policies. It provides resource sharing models and complete IaaS stack simulations in order to easily simulate the management of resource and data centre performance. However, as it focuses on infrastructure cloud schedulers, it has limitations with regard to performance accuracy and does not support packet level simulation. It has been validated by comparing its simulated result against a small scale data centre for some workloads and by means of a comparison study of its performance against that of other simulators (CloudSim and GroudSim). However, DISSECT-CF has not been evaluated by testing different types of workloads.

3.4 Modelling the Performance of Cloud Computing Via Machine Learning Techniques

According to Alpaydin Alpaydin [27], machine learning (ML) can be defined as "the solution of detecting certain patterns from large data and is a way to predict future actions and performance based on past data and experience". Prediction Modelling has been employed in many diverse research projects to resolve complex issues and overcome the difficulty of conducting experiments on real infrastructure (Islam et al. [82]; Kim et al. [97]).Machine learning applications can be divided and classified into supervised learning, unsupervised learning, and reinforcement learning. Each kind of machine learning application has unique methods that generate a model based on the concept behind the

application such as classification, regression, clustering, and so on. In supervised learning, machines deal with input data that is already mapped to an output class by a supervisor, while unsupervised learning uses only unlabelled input data; the unsupervised learner must thus identify regularities in the input. Clustering is used in unsupervised applications to group input data based on specific features.

Regression models have been applied by Khoshkbarforoushha and Ranjan [95]; Lee and Brooks [105]; Martens and Dardenne [118] and Ozisikyilmaz et al. [123], in order to predict the performance and power of the infrastructure of the data centre.

The Support vector machine is a machine learning algorithm that has been used in existing works by Veni and Bhanu [149] and Ajila and Bankole [23], It is reported to be more effective than linear regression and neural networks.

The work in Kim et al. [97] delivers a review of the existing workload predictors that are used in resource scalability for cloud computing infrastructure. It is focused on the predictors accuracy, efficiency and achieving the best cost and performance. Kim et al. [97] evaluated the performance of the selected predictors via a simulator called PICS (Public IaaS Cloud Simulator).

There are four styles of predictive scaling that have been applied in the evaluation:

- RR (Scale-Out: Reactive + Scale-In: Reactive)
- PR (Scale-Out: Predictive + Scale-In: Reactive)
- RP (Scale-Out: Reactive + Scale-In: Predictive)
- PP (Scale-Out: Predictive + Scale-In: Predictive)

Machine learning models have been used for different prediction tasks in the context of cloud computing. We have divided them as to predict the workload execution time and resource allocation. Overall, 21 predictors have been reviewed in the work (Kim et al. [97]) and they were grouped into four groups as below:

- **Workload Execution Time:**

1. **Naive:** predicts the next arrival job based on the mean of the previous job and it can be applied on mean or recent mean, which only focuses on the recent jobs.

2. **Regression:** there are two types of regression: (a) Global Regression which predicts the next arrival jobs by generating a linear or polynomial regression model using elements such as all prior job arrival times; and (b) Local Regression where in the predictive model there is a kernel function to choose samples of job arrivals and subsequently create a linear or polynomial regression model based on those samples.

- **Resource Allocation:**

1. **Temporal (Time Series):** there are seven methods that have been used previously for cloud computing resource allocation research such as exponential smoothing ES, auto regressive AR, auto regressive and moving average ARMA, and auto regressive integrated moving average ARIMA.

2. **Non-temporal:** methods which have not previously been used for resource allocation research in cloud computing: support vector machines SVMs, decision tree, and ensemble methods.

The outcomes of the evaluation on these styles shows that workloads patterns (Growing, On/Off, Burst, and Random) should be considered before choosing the right predictor. Furthermore, PP is the optimum style of predictive scaling as it has an overall better performance than the others, with costs 30% lower, and an 80% lower rate of missed deadlines than RR or RP. It also costs 14% less and has a 39% lower deadline miss rate compared to PR.

In our research, in chapter 8, we consider the regression from the machine learning predictors. From the scaling approaches, we use the PP approach as it is listed to be the most accurate approach for predicting workload deadlines as we focus on workload execution time. According to Zhai et al. [158], there are two approaches for predicting the performance of parallel applications in large scale data centre; the first approach is building

an analytic model based on historical data sets on the target platform, while the second approach is to use a system simulator to model the behaviour of the target system based on its characteristic and the functionality of the simulated system and applications. Analytic modelling suffers from difficulty in generalising the predicted results to all applications on the target platform, as each application behaves differently. Simulation tools also suffer from a lack of accuracy in simulating large data centres; even though there is a possibility that a simulator can predict large data centres accurately, this would be more expensive in terms of CPU and memory requirements than analytical modelling, such as regression based models, which predicts the execution time for workloads on particular processors based on measurements of the workload on various processors.

Phantom predicts the execution time in three steps. It first generates traces by collecting the communication time for a parallel application and sets the computation between the communication operators as sequential to apportion the total time between communication and computation operations. The communication time is collected via profiling the message characteristics such as message type, size, source, and destination. It then gathers the sequential computation time for each processor on a single node from the target architecture and updates the traces on the first step with the computation times recorded on a single node. Finally, it feeds the traces results into a simulator called SIM-MPI Zhai et al. [157] which is a trace driven simulator that can read the traces generated in step 1 and 2 against a particular network configuration in the simulator to predict the execution time for the MPI application on the target data centre. Phantom has been evaluated by means of a comparison study against traces produced from the SIM-MPI simulator and it showed better accuracy than regression-based modeling as it takes into account the impact of differences in communication time between nodes in parallel ap- plications. However, it would be advantageous for Phantom to test various types of parallel applications as well as considering the MPI application, as some parallel applications differ based on their running status. For Example, some applications run on the native platform of the server and some run on top of a JVM. Moreover, the memory footprint required for each parallel application can be different based on the parallel application's behaviours such as random access to memory and shared memory.

Neural networks have been used for load demand prediction for cloud computing data centre Prevost et al. [127]; de Almeida et al. [55]). Prevost et al. [127] and the use of neural networks to predict future resource requests for distribution applications. These different approaches can be either stochastic or discrete modeling approaches: if the distribution time between request arrival time and the time for the server transition is adaptable in the model, this is a stochastic model, whereas in a discrete model, the client/server transfer time is fixed. Linear predictors show good accuracy, as the experiments exhibit low root mean square errors (RMSE). However, it would be advantageous to study different types of real time workloads in addition to the URL resource requests on a WWW server at NASA and a WWW server at EPA in order to show the performance of both neural networks and linear predictors with respect to multiple types of applications. These attributes are considered in the extension work on the produced prediction model in Chapter 8

3.5 Existing Micro Data centres in The Field of Cloud Computing

According to Varghese and Buyya [147],low cost and low power servers such as Raspberry Pi devices are likely to be applied more frequently in future generations of Cloud Computing data centres for the decentralisation of data centre between multiple providers and to deploy applications at lower levels. Micro data centres have also been used as parts of the infrastructure of larger mega data centres to increase data centre performance and to decrease power consumption.

The following subsections show the relative usage of micro data centres in the field of Cloud Computing. To be more focused, we present details and statues of existing works in the literature on Raspberry Pi devices. For example works by Huynh and Huh [81] and Johnston et al. [84] showed the feasibility of using Raspberry Pi devices in maicro data centre and proved the benefits of micro data centres.

3.5.1 Feasibility of Using Raspberry Pi Devices in Micro Data centres

Micro data centres have been used in different areas of the field of cloud computing for many years. Abrahamsson et al. [19] focused on the delivery of a micro data centre built using Raspberry Pi devices. That work did not, however, profile the performance of the data centre, which is the focus of the current research, and providing information about the performance of the data centre will allow further research to investigate the introduction of new techniques or algorithms to the field of cloud computing by providing a baseline for such effects' measurement. It is also clear that while the implementers of the system managed to configure and fully set up a data centre based on 300 nodes using Raspberry Pi devices, the workloads run on this infrastructure were not fully explained and were not profiled to demonstrate Raspberry Pi cloud performance.

Huynh and Huh [81] delivered a new technique for micro data centres that allows the prediction of performance based on historical data sets using analysis of seasonality forecasts and time analysis. This provides predicted resource needs that allow the allocation of workloads in micro data centres by using a method that determines the seasonal period then develops a moving average forecast by finding the ratio of each observation, finding the average of the ratio for each season or periodic unit, and dividing each of these ratios by the average of the ratios.

Micro data centres are known to deliver the following benefits to the field of cloud computing:

- Less heat waste generated.
- Lower energy consumption.
- Reducing costs due to distributing the workload among different regions based on the cost of use.

Zhao et al. [162] delivered an evaluation of micro data centre performance in a comparison study with conventional data centres. The micro data centre used in that work was built on Intel Edison devices and this was compared to Dell servers. The authors stated

that micro data centres could be used instead of conventional data centres to reduce power consumption, as increasing the number of nodes can significantly improve performance while keeping energy consumption lower than that of a conventional data centre. Moreover, this research also proved the possibility of deploying two intensive workloads on a cloud service in a micro data centre. Workloads that were implemented for that work included web service applications and big data analytic benchmarks using the Hadoop framework. However, the authors claimed that micro servers were not a great solution for intensive computational workloads.

Toosi et al. [143] delivered work on Raspberry Pi devices connected as a research platform for cloud computing research. Their work focused on research on SDNs and was limited to a small number of nodes and switches as they tested the scalability of the device by conducting an experiment on networking SDNs. An examination of existing micro data centres in the field of cloud computing can be used to justify the current choice of the Glasgow Raspberry Pi Cloud as a realistic data centre for the evaluation of existing Cloud Simulators.

Johnston et al. [84] stated that, alongside the advantages of single board computers built on a single circuit board being less expensive and more portable, single board computers are known to be able to run common operating systems and handle a variety of different workloads. Therefore, single board computers are acceptable devices for fog/edge computing and Internet of things and can be used educational environments for teaching computing parallelism concepts at low cost. Moreover, Johnston et al. [84] surveyed existing projects and case studies using single board computers and showed the feasibility of using Raspberry Pi devices in the research community, quoting such works as Turton and Turton [145]; Doucet and Zhang [61]. In the current research, the feasibility of the use of Raspberry Pi devices running intensive real workloads is confirmed, as these are used for the purpose of investigating the level of accuracy of existing cloud computing simulators in educational work.

3.5.2 Raspberry Pi Devices in Fog Computing

Fog computing is a new paradigm for cloud computing that is intended to improve the performance of cloud computing systems by reducing the latency of applications run on the fog infrastructure as compared to deploying them on a standard cloud infrastructure. Fog computing generally uses micro servers, as they consume less power and can distribute certain applications more effectively among a number of servers to achieve optimal performance (Krishnan et al. [102]).

Jalali et al. [83] further raised the possibility of using a micro data centres alongside a centralised data centre to reduce energy consumption by reducing the number of hops between end users and the centralised data centre. This work was conducted using Raspberry Pi nodes running an Internet of Things application.

Barik et al. [35] presented a state-of-the-art fog assisted cloud computing application for Internet-of-Things in terms of its systems, architecture, and applications. This work stated that Raspberry pi devices can be used in small scale data centres for fog computing technologies. He et al. [75] also conducted work to evaluate analytic benchmarks experiments using Raspberry Pi devices to produce workload models of various analytic jobs for a multi-tier fog computing model.

Elkhatib et al. [63] confirmed the feasibility of using micro servers by evaluating Raspberry Pi devices using four different experiments. The first experiment measured the latency for an application running on a fog cloud; the second checked the possibility of hosting multiple cloud containers on a single host Raspberry Pi; the third profiled the I/O overhead, though this experiment showed relatively slow write speeds for the flash memory used in Secure Digital (SD) cards; and the final experiment examined the boot procedures of various Raspberry pi devices as compared with a t2.small EC2 instance running Ubuntu.

3.6 Power Consumption of the Cloud

Power consumption is one of the most important aspects of cloud computing. Saving energy, reducing costs, and being environmentally friendly by reducing the impact of

computing infrastructures are widely touted as positive aspects of using cloud computing technologies around the world. Providers of cloud computing infrastructures are, however, usually focused on making profits in by reducing power costs and the response time of their infrastructures.

Zhang et al. [160] showed a model for accurate estimation of cloud power usage, defining hardware and software configurations that can affect the power consumption of data centres. Hardware configurations include CPU frequency, memory frequency, and hard drive rotation speed; indeed, all hardware and software configurations can qualitatively and quantitatively impact system-wide power consumption and performance.

Hoelzle and Barroso [77], noted that real data centre computing infrastructures have air-conditioning equipment and cooling systems. These two factors considerably affect the QoS and SLA between providers and consumers in case of any emergency actions in the data centre. Moreover, just as the execution time of parallel tasks depends on the number of nodes for three levels of communication intensity which are high communication, medium communication and light communication. So increasing the number of nodes increases the power consumption for the data centre. Simulators and cloud computing models must thus be able to identify techniques and tools that can apply algorithms to assess the power consumption of each application, based on the equipment required to execute a given type of application.

The work in Bohra and Chaudhary [39] provided a model called VMeter to predict the instantaneous power consumption of an individual VM hosted on a physical node as well as the full system power consumption. They validated their model using industry-standard and computationally diverse benchmarks. According to "Makaratzis et al. [115],simulators of cloud computing can be used in power consumption modelling, and in the current research, power consumption was calculated using these models by running an experiment on GreenCloud Kliazovich et al. [100] based on the standard power consumption algorithms described in Beloglazov and Buyya [38]. The power consumption of the simulation models was validated by running simulations of the power consumption of a real-world workload on a small-scale data centre based on existing tools such as the OpenEnergyMonitor system and a power meter device, which measure the energy use and costs of running an

experiment. The output results from previous experiments were compared using statistical analysis to provide an appropriate level of accuracy for the GreenCloud simulator.

3.7 Critical Appraisal of Existing Techniques for Cloud Performance Modelling

Maenhaut et al. [114] provided a general approach for validating cloud resource allocation techniques. Simulations are often used to avoid the difficulties in running experiments on actual test beds such as costs, dealing with underlying infrastructures, and time consumption. Maenhaut et al. [114] that performance of simulation tools depends on the design of the code, the structure of data-set used in validating the simulation results, and the type of proposed workload in the simulation. To avoid such obstacles in both simulations and real infrastructure, adapting the experiments to small scale data centre help validate experiments before the work is applied to the intended infrastructure. The use of Raspberry Pi in this case was to implement the designed costumed cluster management approach for testing interconnections between a Master node and worker nodes. Another way to implement this was tested on an open-stack private cloud. The work thus provides an approach and a well-designed testing cluster adapter and validation strategies that focus on resource management in cloud environments. However, it lacks a means of implementing real time workloads and testing Raspberry Pi performance. The approach is similar to that taken in the current research, though the latter posits that performance of small scale data centres can be used to predict the performance of large data centres and to validate existing simulators.

Ahmed and Sabyasachi [21] classified some of the available cloud simulators by explaining the important features of each. Their research gives a brief description of a number of simulators, offering an informal first step for cloud developers looking to identify a suitable simulator. Each simulator is described based on its unique features, with a brief description of its capabilities in simulating different research areas within cloud computing. However, Ahmed and Sabyasachi [21] did not characterise the level of accuracy or categorise the characteristics for the surveyed simulator, which the current research redresses.

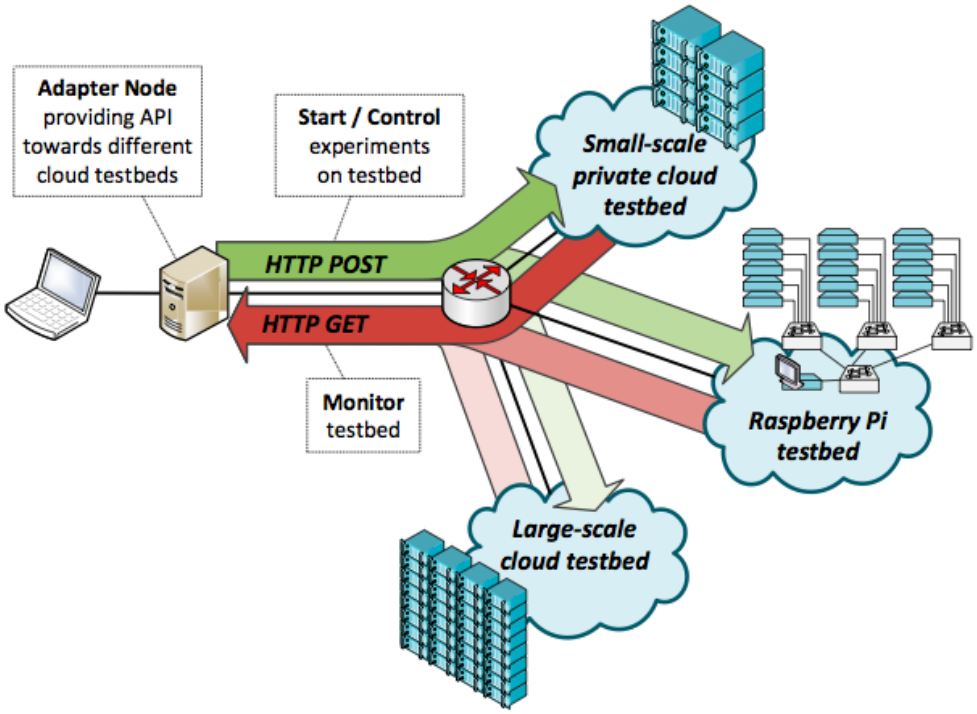

Figure 3.1: *A General Approach for Validating Cloud Resource Allocation Techniques Maenhaut et al. [114]*

Antonescu and Braun [31] delivered a validation methodology for modelling complex distributed systems using CloudSim (Calheiros et al. [46]). The validation experiment was done with a small-scale distributed test-bed. However, the accuracy of evaluation for either CloudSim or the small-scale distributed test-bed is not reported. Moreover, the small scale data centre used in the experiment is not clearly described, limiting reproducibility and failing to quantify the level of accuracy. The current work, however, compares the performance simulation workload with a verified small scale data centre (the Glasgow raspberry pi cloud). A machine learning model is added as an alternative cloud computing model to allow the same experiment to be run again in order to provide further cross-validation land allow meta-analysis of different types of cloud computing models.

NetworkCloudSim (Garg and Buyya [70]) is an extension of CloudSim that models the concept of networking communication, it has been developed and integrated with CloudSim version 3.0.3, and it models Message Passing Interface (MPI). Our work attempts to validate CloudSim by comparing the result of running MPI benchmark DT on either a single node or multiple nodes of Raspberry Pi. Iridis-pi designed by Cox et al. [52], has delivered a small scale data centre based on a Beowulf cluster design, built using 64 Raspberry Pi, model B nodes with 700 MHz ARM processor. They give a brief description of the clustering distribution system in implementing Hadoop job on their data centre, but our work is to profile and report the performance of the benchmarks on two different clusters of Raspberry Pi v2 and v3.

Statistical models have been used to predict resource requirements for cloud computing applications. Ganapathi et al. [69] modelled a framework for a Hadoop [6] application with job scheduling, resource allocation and workload management using a ML technique called Kernel Canonical Correlation Analysis (KCCA). In this work, we provide a multiple machine learning models to predict the execution time for different types of workloads with respect to each node, considering resource availability and average response time for a particular workload. This would ensure the results of my predictive model for cloud computing are comparable with the results from the work in Chapter 8.

Khoshkbarforoushha et al. [96]offered a statistical technique which uses mixture of network topology and mixed models in order to provide a predictive model of resource

usage based on probability density functions. However, their work was limited to the investigation of CPU and memory usage, while the current ML model is based on the features of an available small-scale data centre test bed. The model can thus be validated by running cross-validation experiments to test its accuracy, as compared with the work by Tso et al. [144].

The current research aim differs from that in the works of Ahmed and Sabyasachi [21], Kaleem and Khan [87], Zhao et al. [161], and Maarouf et al. [113], which offer only descriptions of different types of simulators in that it offers experiments in different types of test environment in order to ascertain the level of accuracy in each test environment for developers. There is a lack of work on the accuracy of simulators and little discussion of any experimental testing except in research that delivers validation descriptions from the developers of the simulators as in Kecskemeti [91]. It is thus advantageous for these cloud computing simulators to be validated by either a third party or the end users in order to provide results regarding the validation approaches seen in Twomey and Smith [146]. There is currently relatively little research evaluating and comparing the accuracy of simulators relative to real world benchmarks, and the current work thus focuses on evaluating the tools used to evaluate cloud computing systems by deploying different types of workloads to conduct an evaluation based on a diversity of types of workload; this is important, as different workloads or scenarios cause systems to behave differently, and this make the approach that the current research follows more appropriate to developing future work.

To sum up, as mentioned above that works Maarouf et al. [113]; Kaleem and Khan [87]; [161]; Ahmed and Sabyasachi [21];Byrne et al. [45] briefly describe the existing simulators in the field of Cloud Computing, it is still not clear for researcher to choose the most reliable tool. So our critical investigation in the literature leads to summaries the features and characteristics of existing simulators in order to answer the research questions: how well existing simulators deliver the required modeling features of Cloud Computing experiments and how well accurately model the experiments.

Chapter 4

Research Methodology

4.1 Introduction

This chapter outlines the two methods used in this research. The first method categorised existing simulators into four categories based on their common salient features. This method focuses on gathering together the features of the extensions of each simulators in order to provide the opportunity to merge multiple simulators. The second method evaluates existing cloud computing simulators by performing an experimental comparison study against the actual performance of a Raspberry Pi data centre. Chapter 5 lists the outcomes of examining existing simulators that are extensions of the same simulator, while chapter 7 assesses the level of accuracy of the three simulators described in chapter 2.

4.2 Method to Select the Optimal Cloud Computing Simulator

This section describes an approach that can be used to select the optimal cloud simulator by utilising a guide based on a systematic review of existing simulators' results. This was achieved by collecting data about cloud computing simulators, and then creating classification models that captured their salient features.

4.2.1 How Do We Characterise Simulation Capability?

For simulator characterisations, the cloud computing characterisations listed in Mell and Grance [119] should be considered; these are On-demand self-service, Broad network access, Resource pooling, Rapid elasticity, and Measured service. This work focuses on the simulators' capabilities available over the network and accessed by heterogeneous client platforms. Capabilities can be elastically provisioned and controlled automatically dependent on the type of service (storage, processing, bandwidth, and active user accounts) (Mell and Grance [119]). To test simulators' capabilities, the selected simulators are categorised based on the target area of cloud computing data centre infrastructure under focus, including Networking, Quality of Service, and Power consumption.

The characterisations of software presented in Losavio et al. [111] were also investigated in order to categorise existing simulators. The major characteristics of software are functionality, reliability, usability, efficiency, maintainability, and portability. The method used in this research focused on functionality and usability, however, with the validity characteristic added only in order to make the process of selecting a simulator more straightforward.

Functionality refers to the degree of performance of the software as measured against its intended purpose. Usability refers to the extent to which the software can be used with ease. Validity refers to the level of accuracy of the simulator as compared with an actual environment; in simple terms, it refers to the ability of the software to simulate the performance properly on software platforms with a low level of relative errors.

4.2.2 Performance Evaluation Methods for Simulation Tools

A range of evaluation methods have been applied to simulation packages in the literature within the different areas that use simulation models for research purposes. Alomair et al. [26] delivered a review of existing methods such as evaluation criteria, multi-criteria decision making (MCDM), hierarchical frameworks, SimSelect, Smart Sim Selector, scenarios, two-phase evaluation, and selection methodology and guidelines. The current method is notable in that it unifies the advantages associated with scenarios, guidelines, and two-

phase evaluation methods into a single system. It first analyses the capability of each simulator within the proposed experimental scenario; following this, the features of the selected simulators are examined in terms of their usability and availability; finally, the level of accuracy for selected simulators is identified to determine whether there exists a set of simulators that can model the proposed experiment, a step that is part of the two-phase evaluation method.

4.2.3 Proposed Method for Selecting A Simulator from CloudSim Extensions

The method is designed as a set of steps that begin with conducting a systematic review as advocated by Kitchenham et al. [98], based on work on the existing surveys on cloud computing simulators within the family of CloudSim extensions. The current work focuses on providing details about the current CloudSim family of simulators, as these simulators are built on the same platform; the research thus differs from other surveys in its focus on categorising simulators based on their similarities and differences within the same family of simulators. Individual simulators were selected as candidates for the review from two distinct sources.

- We examined the public list of CloudSim extensions (known as *related projects*) on the official CloudSim website Clo [2], on 1st April 2016. The investigation was restricted to extensions that are associated with at least one peer-reviewed publication available via Google Scholar.

- We identified other CloudSim extensions by searching for the single keyword 'CloudSim' on the IEEE Xplore portal, on 1st August 2017. We examined the first 50 links to look for extension projects.

The method used categorises the CloudSim extensions based on the nature of the additional features of each extension, as reported through associated peer-reviewed publications. Finally, the ability and features of each of the cloud computing simulators were divided into Functionality, Usability, and Validity, and they were thus categorised.

In terms of functionality, the capability of the simulators was checked using work based on common criteria for cloud computing experiments as in Zhang et al. [159] , such as network modelling, power consumption modelling, and ability to simulate different types of workload (Batch, Transactional processing, or Work flow) or types of infrastructure (Heterogeneous, Homogeneous, Hybrid, or Micro data centre), and by modelling the quality of service features.

With regard to usability, the simulators' abilities to support graphical user interfaces and generate graphs for the simulated output were examined. Finally, the validity of the simulators was tested in a manner that has been validated by running different types of experiments, using a comparison study against real infrastructures or existing workload traces on actual public infrastructure. Following the algorithm generated by of this method, a suitable simulator can be selected or, where no existing simulator is compatible with the experimental requirements, a new type of simulator can be developed. This is illustrated in 4.1.

4.3 Method for Evaluating Cloud Computing Simulators

The simulation models were based on the Glasgow Raspberry Pi Cloud (Tso et al. [144]) and developed in CloudSim, GreenCloud, and Mininet by means of customizing the code and configurations that characterize the compute nodes, network connectivity, and virtual machine environments of the data centre instance to be simulated. In this section, two different types of application (batch and transactional processing) are examined. Testing different types of workloads supports the generalisation of the results for different types of workloads. Different workloads were thus tested on the Raspberry Pi cluster to support their availability to be used in experiments as profiled on Raspberry Pi devices or modelled directly on CloudSim, GreenCloud and Mininet or by using an extensions. Standard benchmarks were thus used for message passing interface (MPI) applications and a web server benchmark called Apache Bench AB was used for that function. These workloads were modelled in the simulators with appropriate characterisations by following the steps

of the approach as outlined in Figure 4.2.

The benchmarks were run on Raspberry Pi v2 and Raspberry Pi v3 with different numbers of parallel nodes that effectively formed one rack of the micro data centre; this is explained further in Chapter 5. The experiments were repeated 30 times on each configuration, to allow the elision of error bars.

The testing deployed 22 nodes of Raspberry Pi v2 and 15 nodes of Raspberry Pi v3 and four different workloads were run: high performance computing, databases, big-data, and web-server applications.

The reason for using these numbers of nodes was the availability of the nodes for the research. The Glasgow Raspberry Pi data centre is used by several other researchers and thus the number of nodes available at the time of conducting the evaluation was limited. However, small scale data centres are used extensively in the literature for evaluating and validating different techniques and simulators in the field of cloud computing (Zhao et al. [162]; Garg and Buyya [70]; Kecskemeti et al. [92]).

The work detailed in chapter 7 was conducted by applying an experimental method to measure the performance of an existing small-scale data centre and modelling the small-scale data centre in three simulators in order to evaluate them as compared to the performance of the actual centre. Two different workload types (batch processing and transactional processing) were run on the Raspberry Pi cluster, based on the following requirements for the evaluation process:

- Open source workloads for the reproducibility of research.
- Able to be profiled on Raspberry Pi devices.
- Able to be modelled on CloudSim, GreenCloud and Mininet.

Figure 4.3 shows the approach followed to evaluate the selected simulators.

This research applied a method designed to check the fidelity of any simulation models used for performance prediction of any realistic infrastructure. The cross-validation of simulators with large scale infrastructure is difficult; thus, a sanity check cross-validation of each simulator with a cut-down micro scale infrastructure is performed. If there are

discrepancies between the simulator and the micro infrastructure, the simulator is deemed to be inaccurate.

Finally, this research compared the results of the actual performance when running workloads on a Raspberry Pi data centre (Tso et al. [144]) with the simulated performance of each model sharing the same features in terms of workloads and infrastructure. The low level properties of the Raspberry Pi data centre were gathered via standard Linux monitoring tools such as perf (per [16]) and iperf (ipe [14]). These properties were then fed into the simulators as parameters for modelling the equivalent Raspberry Pi Cloud infrastructure in each simulator. More details regarding these workloads, cluster characteristics, and profiling tools are given in Chapter 6

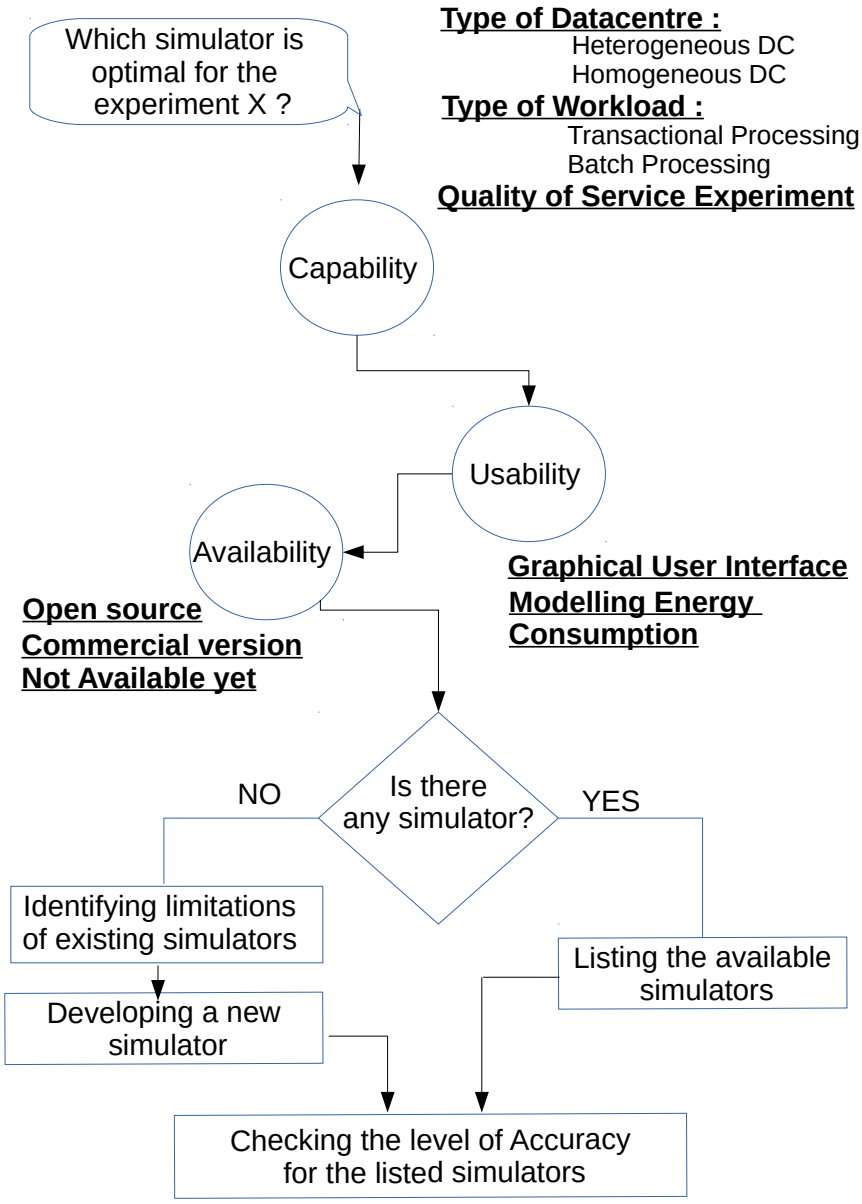

Figure 4.1: *Method for Investigating Simulators Based on Their Capacity*

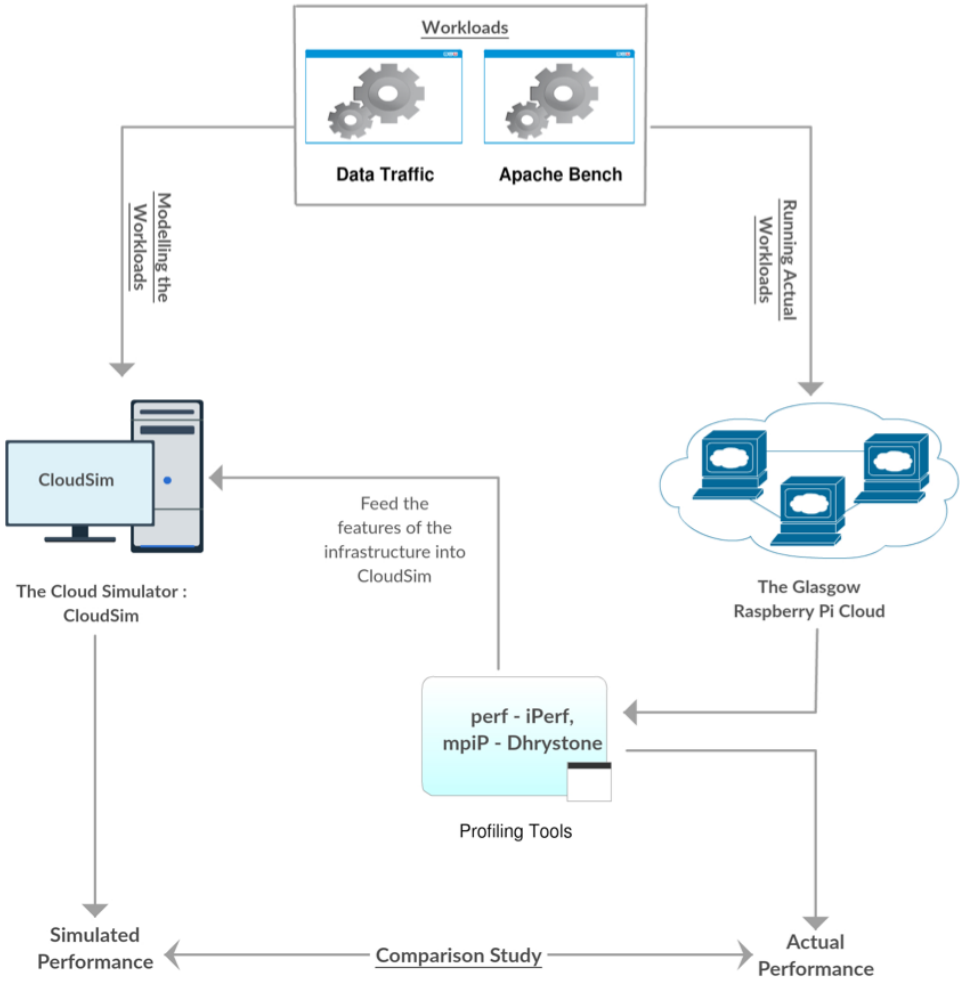

Figure 4.2: *Cross-validate Performance Metrics for CloudSim and The Glasgow Raspberry Pi Micro Data Centre*

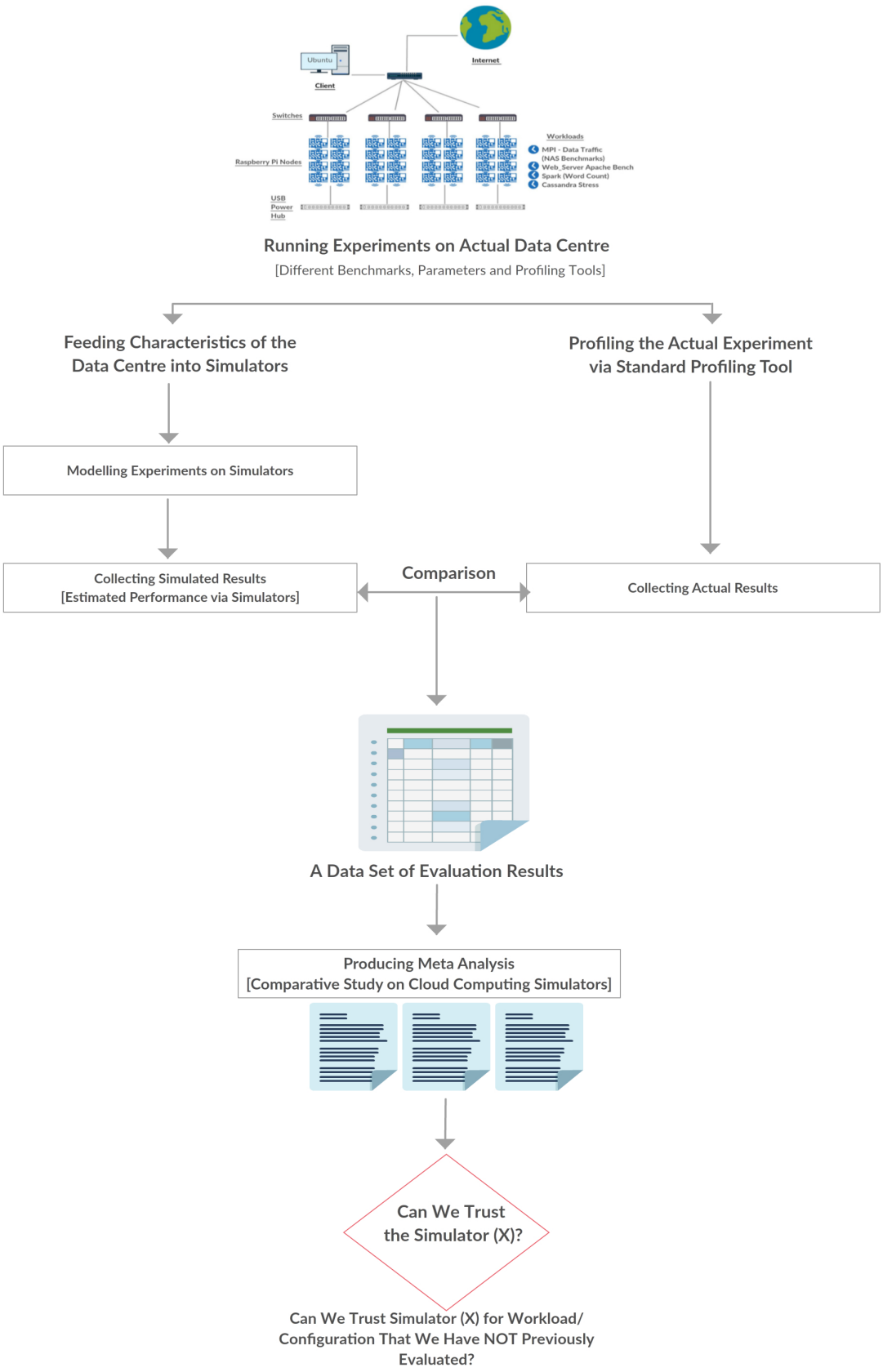

Figure 4.3: *Overview flowchart of the methodology*

Chapter 5

Qualitative Study of Existing Cloud Computing Simulators

5.1 Introduction

A cloud computing data centre is a distributed system with a very complicated architecture created by layer abstractions and the virtualisation migration model that emerges between nodes. Therefore, researchers find it difficult to run experiments on actual data centre infrastructures when developing new algorithms and techniques because of the time and cost implications. Researchers require an environment that provides the opportunity to repeat experiments and to control variables with respect to the architecture of real data centre environments. All simulators have their own features and differ from each other based on the purpose behind their development and the aim of the simulation. Developers and cloud computing researchers thus have a wide variety of types of simulators available to use in their research work, but there are still several limitations and obstacles in terms of tackling actual data centre issues.

Developers of cloud computing systems also choose from a wide variety of simulators for their empirical research (Barker et al. [36]; Li and Deng [106]). There are many limitations and trade-offs inherent in investigating data centre system issues by means of simulation, which offers an explanation for the vast number of simulators available. Ahmed and Sabyasachi [21] noted that many researchers opt to use and extend open-source cloud

simulators such as CloudSim and GreenCloud.

By applying the method for selecting the optimal simulator for any type of experiment in Cloud Computing developed in Chapter 4, it is, however, possible to categorise a set of simulators, which is done in this chapter based on their features to model netowrking, quality of service and power consumption for Cloud Computing data centres.

This chapter thus provides an overview of the existing extensions of CloudSim, GreenCloud, and Mininet. In particular, it delivers a systemic review of the existing extensions for CloudSim, as this is one of the most popular simulators in the field of Cloud Computing (Byrne et al. [45]) based on the number of citations on Google Scholar at the time of writing this chapter. These are categorised based on their features and benefits. Section 5.2 shows how the observed extensions of CloudSim are categorised into four groups (Networking, Graphical User Interface, Quality of Service, and Power Consumption Modelling). The contributions and future opportunities offered by this chapter are then given in Section 5.3

With regard to GreenCloud and Mininet, the same procedural systematic review as followed for CloudSim extensions was used to find and categorise extensions of GreenCloud and Mininet. However, there is currently a lack of extensions of GreenCloud and Mininet that provide new simulators.

For Mininet, an extension called MiniNAM (Khalid et al. [93]) is the main option; this was developed to improve the usability of Mininet by providing a graphical user interface. This extension therefore allows users to debug code without typing into the source code of Mininet. The main advantage of this extension is thus the provision of the GUI for the purposes of learning and teaching computer networks.

For GreenCloud, only one simulator has been developed as an extension; this is called QoS-based GreenCloud simulation (Zhihua [163]) , and it was developed for modelling the concept of Network as a Service (NaaS) which has recently been introduced into the Cloud Computing field. The QoS-based GreenCloud simulator is supported by QoSbox (Christin et al. [50]), which configures the IP routers to monitor loss, delays and throughput of traffic in the network.

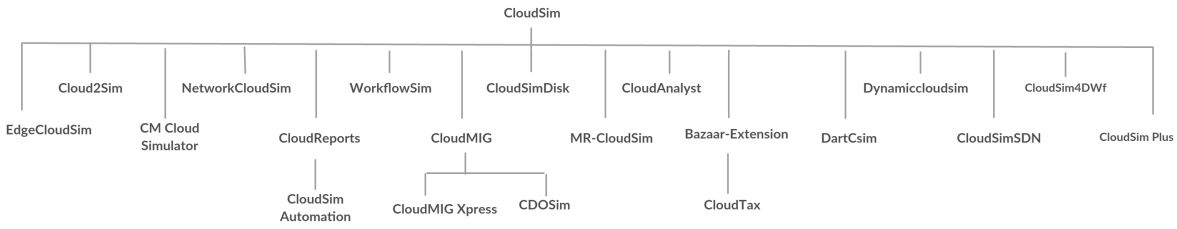

Figure 5.1: *Inheritance Tree for Surveyed Extensions of CloudSim*

5.2 Existing Extensions of CloudSim

Byrne et al. [45] a review of existing cloud computing simulators in terms of their general high-level features and high-level technical aspects. The features of the simulation platforms were surveyed and the outcome of the survey illustrated that CloudSim can act as a *de facto* base platform for simulation development and research; in all, over 20 extensions of CloudSim are used in various different areas of research. This chapter thus delivers a systematic review focused on these extensions of CloudSim in order to explore the similarities and differences between the various extensions.

Figure 5.1 shows the relationships between various extensions of CloudSim, in terms of the provenance of their source code bases. A relationship $A \rightarrow B$ denotes that system B directly extends system A, either by adding to or modifying components of A.

The motivation for this section of the research is to provide a review of existing simulators that have been developed based on a particular platform for future use by researchers in cloud computing. Extensions of CloudSim support additional features such as heterogeneous data centres, graphical user interfaces and more complex distributed applications. Therefore, in some cases, it may be more efficient for researchers to conduct cloud computing experiments with these extensions rather than modelling the required features directly in CloudSim.

Various researchers have extended the original CloudSim system to model additional data-centre features, to measure additional non-functional properties, and to enhance simulator usability. In this survey, 20 such extensions were examined; these were selected on a systematic basis as outlined in Section 4.2.3. Four sets of functionality were first identified for CloudSim extensions, and each extension can thus be characterised in terms of its membership of these sets, as outlined in Figure 5.2 which also shows how some

extensions belong to more than one set as they satisfy multiple orthogonal criteria. In the detailed description, the precise set membership of each extension is thus provided.

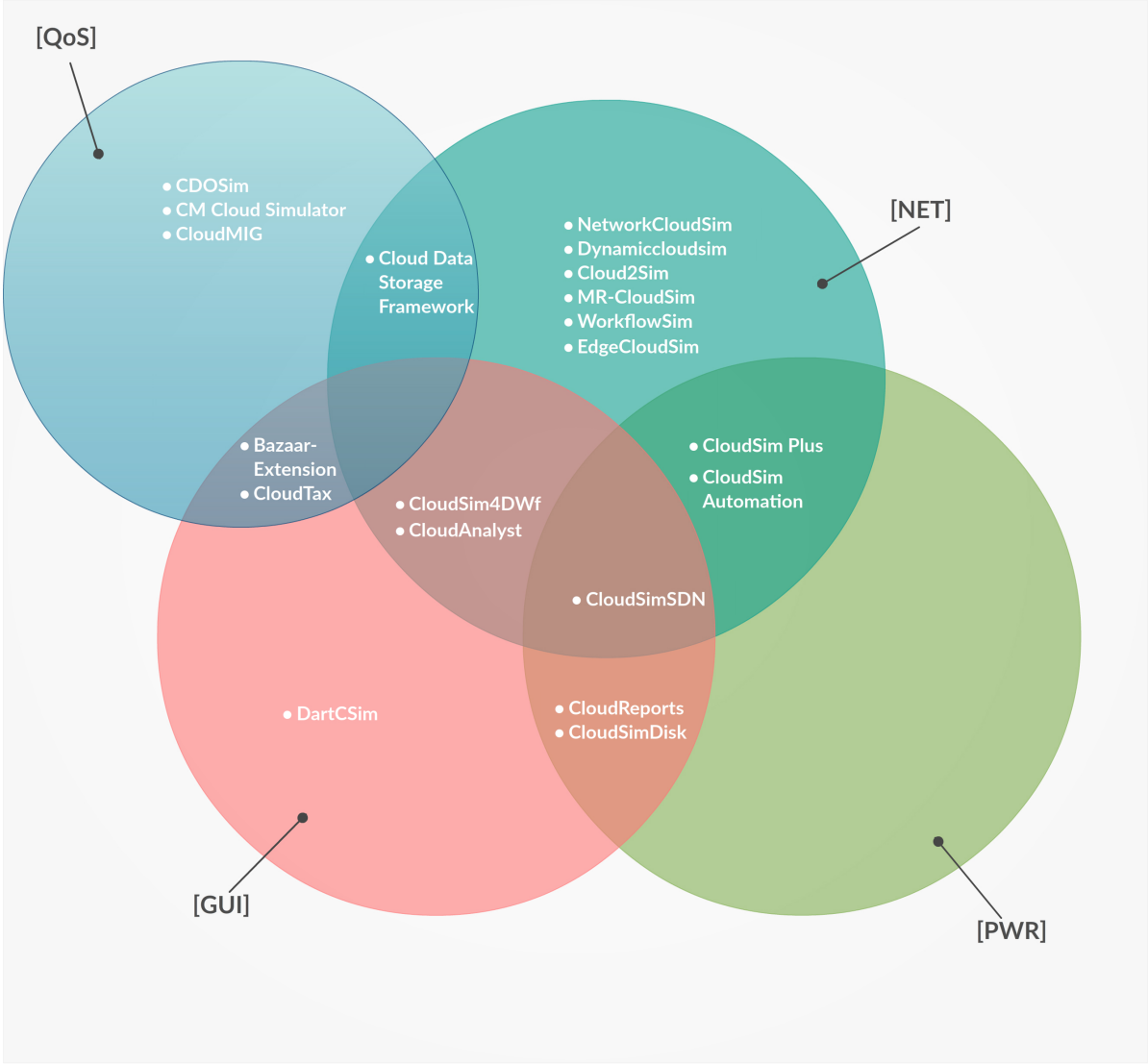

Figure 5.2: *The Main Four Categories for CloudSim Family of Extension*

- Modeling distributed applications and complex network configurations: [NET]
  There are clear distinctions in terms of data communication and infrastructure between cloud and grid computing (Li and Deng [106] Myerson [120]). Experiments to model cloud computing applications and infrastructure need to be carried out on reliable simulation environments (Garg and Buyya [70]). A number of simulators that have extended CloudSim to model distributed applications and complex network configurations:

1. NetworkCloudSim Garg and Buyya [70].
2. DynamicCloudSim Bux and Leser [42].
3. Cloud2Sim Kathiravelu and Veiga [89].
4. MR-CloudSim Jung and Kim [85].
5. Cloud data storage framework Long and Zhao [110].
6. CloudAnalyst Wickremasinghe et al. [152].
7. WorkflowSim Chen and Deelman [49].
8. CloudSimSDN Son et al. [136].
9. CloudSim Automation Silva Filho and Rodrigues [134]
10. CloudSim4DWf Fakhfakh et al. [64]
11. EdgeCloudSim Sonmez et al. [137]
12. CloudSim Plus Silva Filho et al. [135]

Table 5.1 shows the advantages and drawbacks of extensions mainly developed for the purpose of modelling network behaviour in Cloud Computing.

Simulator	Validation	Advantages	Disadvantages
NetworkCloudSim	Modelling a realistic application with communications.	- Modelling data centre bandwidth.	Random access to memory in shared nodes is not applied
Cloudanalyst	Simulating different scenarios for large date centre workloads	Modelling distributed systems on different regions.	Needs to provide more mechanisms and algorithms for resource management for large scaled DC.

MR-CloudSim	Simulating different experiments,without a comparison to actual system.	Modelling MapReduce for big-data computation	Lacks allocation algorithms and validation work.
Cloud Data Storage	Not Explained	Supporting stripping file and replica strategies for storage distribution	Needs to be validated via validation method and to consider the energy consumption and cost of the data centre infrastructure during the distribution of storage among nodes
Workflowsim	Compared with real traces generated by the Pegasus Work flow management system	The provision of modelling transient failures via a failure generator and a failure monitor.	Limited to a specific number of work flow techniques.
Cloud2Sim	Not explained	Simulating storage distribution by integrating Hazelcast and Infinispan clusters.	Needs to provide an experimental work to validate the simulator by comparing its simulated performance against actual test-bed environment

DynamicCloudSim	Compared to real cloud infrastructure on EC2	Modelling heterogeneous data centre in the performance of computational resources. - Modelling dynamic changes to the performance and status of VMs during execution time.	Needs to consider the data locality for dynamic work flow.
CloudSim4DWf	Comparison study against WorkflowSim	Modelling the dynamic adaptation actions at work flow instance level (Running Time)	Needs to consider federated cloud infrastructures, and energy consumption
EdgeCloudSim	Limited comparison study	Modelling Edge Computing Architecture	Lacks supporting different parameters for the decision maker for how and where the task will be done.

Table 5.1: Comparison of Extensions of CloudSim for Modelling Network in Cloud Computing

- Graphical user interface for simulation: [GUI]

  The original CloudSim framework is configured programmatically, and thus variables must be modified in the code before the project is recompiled to configure a new simulation. Further, the results of a simulation run are dumped as raw text to the standard output based on *printf* statements. Several extensions of CloudSim incorporate more user-friendly front-ends for setting up simulations and reporting results. These are generally GUI style front-ends.

1. CloudAnalyst Wickremasinghe et al. [152].
2. CloudReports Teixeira Sá et al. [141].
3. Bazaar-Extension Pittl et al. [126].
4. DartCSim Li et al. [107].
5. CloudTax Pittl et al. [125].
6. CloudSimSDN Son et al. [136].
7. CloudSim4DWf Fakhfakh et al. [64].
8. CloudSimDisk Louis et al. [112].

Table 5.2 shows the advantages and drawbacks of the extensions that are supported by graphical user interfaces.

Simulator	Validation	Advantages	Disadvantages
DartCSim	Not explained	Improving the usability level of CloudSim with GUI and providing power consumption classes	Built on basic functionality of CloudSim without any integrating of any extension of CloudSim.
CloudSimSDN	An empirical study compared with Mininet Min [8]	Modelling of software-defined networking SDN - Managing resource allocation dynamically.	Lacks supporting multiple workloads on a same link between resources (Hosts).

Table 5.2: Comparison of Extensions of CloudSim with Graphical User Interface

- Applying cloud deployment options and Quality of Service Reliability Mechanisms: [QoS] Moving to Cloud depends on how trust, managerial capability, and technical capability can be developed by both the end clients and vendors of infrastructure who

seek to achieve their goals by deploying cloud computing (Garrison et al. [71] Al-Ghuwairi [24]). The following extensions of CloudSim thus model the concepts of cloud deployment options and quality of service reliability:

1. CDOSim Fittkau et al. [65].
2. CloudTax Pittl et al. [125].
3. CloudMIG Fittkau [66].
4. Bazaar-Extension Pittl et al. [126].
5. Cloud data storage framework Long and Zhao [110].
6. CM Cloud Simulator Alves et al. [30].

Table 5.3 shows the advantages and drawbacks of extensions are that mainly developed for modelling the quality of service in Cloud Computing.

Simulator	Validation	Advantages	Disadvantages
CloudMIG	Providing different violations on different system to evaluate the simulation	Providing opportunities to exam the existing cloud providers for a specific system.	Needs to be validated on experiments on different cloud providers for a specified system.
CDOSim	Incorporating Eucalyptus and Amazon EC2 providers.	Providing evaluation of cloud deployment options (CDOs)	Lacks supporting distributed applications among different data centres.at run time
Bazaar-Extension	Not explained	Evaluating different negotiation strategies for cloud deployments.	Lacks of possibilities in modelling different strategies for negotiations.

CM Cloud	Testing different scenarios	Modelling the cost of different cloud providers.	Needs to add more types of services to make broader comparisons.
CloudTax	Not explained	Modelling different tax systems	Needs to consider the possibility of distributing VMs among different regions in large data centre with live migrations of VM during execution time

Table 5.3: Comparison of Advantages and Validation for Extensions of CloudSim for Modelling Quality of Service

- Power consumption modelling :[PWR]

  A key motivation for resource consolidation is reduced energy demands. The efficiency of cloud computing data centres is thus a key metric (Al-Ghuwairi [24]), and while the original CloudSim framework does not incorporate models for energy estimation, several extensions have added this capability.

  1. CloudReports Teixeira Sá et al. [141].
  2. CloudSimDisk Louis et al. [112].
  3. CloudSimSDN Son et al. [136].
  4. CloudSim Automation Silva Filho and Rodrigues [134]
  5. CloudSim Plus Silva Filho et al. [135]

Table 5.3 shows the advantages and drawbacks of the extensions mainly developed for modeling the power consumption in Cloud Computing.

Simulator	Validation	Advantages	Disadvantages

CloudSimDisk	Not explained	Providing models to simulate power consumption on different level in cloud data centre. - Modelling hard disk (HDD) storage in both active or ideal modes.	Has not been validated in comparison with actual test-bed infrastructure.
CloudReports	Running a simulation experiment based on data collected from the Google Cluster Data project	Power Consumption Modelling Providing opportunities for researchers to extend the existing entities of CloudSim core.	Lacks different resource allocation algorithms for VMs.
CloudSim Plus	Limited comparison study	Improving maintainability, reusability and extensibility of CloudSim original source code	Lacks simulating different workloads scenarios
CloudSim automation	Not explained	Improves the readability of the simulation scenario using YAML file	Lacks supporting different resources distributions.

Table 5.4: Comparison of Advantages and Validation for Extensions of CloudSim for Modelling Power Consumption

Table 5.5 lists the CloudSim extensions investigated in this work in order of their availability and year of release. The differences between the selected simulators in terms

Table 5.5: Availability of CloudSim Family of Simulators.

Simulator	Year	Availability	Number of versions	Reference	LICENSE
CloudSim	2010	Open Source	4	Calheiros et al. [46]	GNU LESSER GENERAL PUBLIC LICENSE
CloudAnalyst	2010	Open Source	1	Wickremasinghe et al. [152]	-
NetworkCloudSim	2011	Open Source	1	Garg and Buyya [70]	Integrated with CloudSim3.0.3 release
CloudMIG Xpress	2011	Open Source	1	Fittkau [66]	Apache License V2.0
CDOSim	2012	Open Source		Fittkau et al. [65]	
WorkflowSim	2012	Open Source	1	Chen and Deelman [49]	The Globus Toolkit Public License
MR-CloudSim	2012	Not Open Source	-	Jung and Kim [86]	-
CloudReports	2012	Open Source	1	Teixeira Sá et al. [141]	GNU GENERAL PUBLIC LICENSE
DartCsim	2012	Open Source	1	Li et al. [107]	
Cloud data storage sim	2012	Open Source		Long and Zhao [110]	
DynamicCloudSim	2013	Open Source	1	Bux and Leser [42]	GNU LESSER GENERAL PUBLIC LICENSE
CloudSim Automation	2014	Open Source	2	Silva Filho and Rodrigues [134]	GNU GENERAL PUBLIC LICENSE
Cloud2Sim	2014	Open Source	1	Kathiravelu and Veiga [89]	
CloudSimSDN	2015	Open Source	1	Son et al. [136]	GNU GENERAL PUBLIC LICENSE
CloudSimDisk	2015	Open Source	1	Louis et al. [112]	GNU LESSER GENERAL PUBLIC LICENSE
Bazaar-Extension	2016	Not Available	-	Pittl et al. [126]	Not Available
CloudTax	2016	Not Available	-	Pittl et al. [125]	Not Available
CM Cloud Simulator	2016	Open Source	-	Alves et al. [30]	
EdgeCloudSim	2017	Open Source	1	Sonmez et al. [137]	
CloudSim Plus	2017	Open Source	-	Silva Filho et al. [135]	
CloudSim4DWf	2017	Not Available	-	Fakhfakh et al. [64]	

of their functionality, availability, advantages, drawbacks and validity are then offered.

Tables 5.1 – 5.6 deliver an overview of the functionality, usability, validity and drawbacks associated with the existing extensions of CloudSim. It is expected that this table can act to guide researchers when they select suitable simulators for their experiments, especially in terms of examine the respective capabilities of simulators to conduct comparative examinations.

The various CloudSim extensions provide additional features, of which many are orthogonal. Thus, it remains difficult to select the optimal extensions for a particular experiment in all cases, as this would require manual source code investigation and documentation research.

5.2.1 NetworkCloudSim

NetworkCloudSim was implemented by Garg and Buyya [70] as an exercise to evaluate the workload and scheduling policies in CloudSim for resource allocation in a more reliable environment for applications with communicating tasks. To run any networking work flow experiment, NetworkCloudSim simulates networking topology and architecture. NetworkCloudSim simulator enhances CloudSim Calheiros et al. [46], to model a realistic application with communication elements such switches, routers and parallel application work-flow. Developers of NetworkCloudSim have presented an evaluation work that shows the importance of modelling the network architecture to test the scheduling policies and

Table 5.6: Comparison of Features of CloudSim Family of Simulators.

Simulator	Networking Typologies	Graphical User Interface	Energy Consumption
CloudAnalyst	Yes	Yes	Limited
NetworkCloudSim	Yes	No	Limited
CloudMIG Xpress	Limited	No	No
CDOSim	Yes	No	Limited
WorkflowSim	Yes	No	No
MR-CloudSim	Yes	No	No
CloudReports	yes	Yes	Yes
DartCsim	Limited	Yes	Limited
Cloud data storage sim	Yes	No	No
DynamicCloudSim	Yes	No	No
CloudSim Automation	Yes	No	Limited
Cloud2Sim	Yes	No	No
CloudSimSDN	Yes	Yes	Limited
CloudSimDisk	Limited	Yes	Yes
Bazaar-Extension	Limited	Yes	Yes
CloudTax	Limited	Yes	Yes
CM Cloud Simulator	Limited	No	Limited
EdgeCloudSim	Yes	No	No
CloudSim Plus	Yes	No	Yes
CloudSim4DWf	Yes	Yes	No

algorithms. There are new classes that model the work flow of the actual data centre network topology, such as Switch and NetworkPacket, and there are also modelled classes for application modelling such as NetworkCloudlet and AppCloudlet.

5.2.2 CloudSimSDN

CloudSimSDN is developed to extend network simulation features in CloudSim to support software-defined networking (Son et al. [136]). The task of measuring how the network and host capacity perform is facilitated by this extension as it affords researchers a simulated medium for modelling cloud data centre. In addition, the resource management policies that are relevant to the management of the cloud data centre based on SDN are assessed by CloudSimSDN. The manner in which software-defined networking behaves can be effectively outlined with the help of the new classes (e.g. Switch and Network Operating System classes) offered by CloudSimSDN. Software-defined networking behaves can be effectively outlined with the help of the new classes (e.g. Switch and Network Operating System classes) offered by CloudSimSDN. Furthermore, the transmission delay associated with data transmitted among elements of data centres can be modelled and estimated

with the help of CloudSimSDN as well. Due to the fact that a channel is made up of more than one link, its bandwidth is used in CloudSimSDN to assign the channel to the data amongst two hosts. In the context of CloudSimSDN, the user request is understood to refer to abstraction calculations of computing processing with transmissions of packets. In order to allow users to establish the input specifications and output results, the visualisation of data centre components is made possible by CloudSimSDN through a graphical user interface.

5.2.3 Cloud2Sim

According to Armbrust et al. [32], running between distributed nodes and extensive clusters, cloud computing applications are oftentimes of substantial size and display a high level of complexity. In the case of such large applications, shared memory access and distributed execution processing are not supported by CloudSim (Calheiros et al. [46]). In response to this limitation, Cloud2Sim has been created as an extension of CloudSim that can enable simulation of the storage distribution of VM, Cloudlet, and data centre objects Kathiravelu and Veiga [89]. What is more, the simulation of the concurrent of the execution to the cluster instances is an additional property of Cloud2Sim.

Running on top of a cluster underpinned by Hazelcast (haz [13]) and Infinispan (Inf [7]), Cloud2Sim represents a concurrent and distributed cloud simulator. More specifically, the distributed applications for the java.util.concurrent package are supplied by Hazelcast, while Infinispan represents a distributed key/value data grid serving as a cluster-aware data-grid across more than one node. Applications that cannot be run on just one node/computer because of resource insufficiency can be implemented with the help of Infinispan (Kathiravelu and Veiga [89]). The incorporation of Hazelcast haz [13] and Infinispan Inf [7] cluster into the simulator is made possible by the two specifically altered packages CloudSim.hazelcast and CloudSim.infinispan that are contained in Cloud2Sim.

5.2.4 MR-CloudSim

For purposes of management of extensive data calculations in sizable clusters and servers, the CloudSim extension MR-Cloud-Sim has been designed for modelling *MapReduce* (Jung and Kim [85]). Commonly employed for ample data and distributed application between cloud data centres, MR-CloudSim is intended as an instrument of simulation for map-reduce instructions. The present model encompasses a number of map-reduce attributes, as that the workload may be independent of the size of file input, inclusion of one reduce operation in every map output and the map operation is undertaken before the reduce operation. Key alterations have been brought to the primary elements of CloudSim upon incorporation of the MapReduce stages into the CloudSim core.

5.2.5 CloudAnalyst

CloudAnalyst is designed by Wickremasinghe et al. [152] to help understand how a sizable Internet application behaves in the context of a cloud environment. The classes and several novel functionality included in the CloudAnalyst facilitate a number of tasks and procedures. These include simulation of the realistic behaviour model the same way as traffic generators do, network delay with regard to the data that are transmitted over the Internet, management of user requests amongst more than one data centre, as well as provision of an extensive graphical user interface (GUI).

The major elements that make up the CloudAnalyst are outlined in Region, Internet, Cloud Application Service Broker, User base, Internet cloudlet, Data centre controller, VM load balancer and GUI.

5.2.6 DynamicCloudSim

When it comes to creation of models of dynamic work flow and of the non-homogeneous nature of an actual cloud configuration, CloudSim (Calheiros et al. [46]) presents some shortcomings. More specifically, among the cloud computing factors of instability that CloudSim as a tool-kit is unable to model are heterogeneity, run time dynamic modifications, straggler VMs, as well as failure task executions Bux and Leser [42]. In order to

overcome these shortcomings, DynamicCloudSim is created as a CloudSim extension with the ability to achieve the simulation of all of the above mentioned cloud computing factors of instability Bux and Leser [42]. The two researchers accomplished work flow production in the context of DynamicCloudSim by employing the instrument of montage in the empirical work they conducted. Each of their four experiments focused on one particular cloud computing factor of instability, namely, heterogeneity (Het), dynamic changes at run time (DCR), straggler VMs and failure task (SaF), and extreme parameters. A different extension of CloudSim designed for simulating the network typologies of the data centre configuration of a real cloud infrastructure as well as the node failures and the work flow delay passing is WorkflowSim (Chen and Deelman [49]).

5.2.7 WorkflowSim

The creation of WorkflowSim (Chen and Deelman [49]) has been prompted by the need to have a model of work flow management that could help address certain problems faced by CloudSim, namely, the absence of support for distributed work flows and the overhead associated with data flows in a non-homogeneous system. Since the task clustering method targets failures and delays that take place at different levels of the systems of work flow management, models with multiple layers are supported by WorkflowSim. This CloudSim extension is made up of work flow mapper, clustering engine, work flow engine, work flow scheduler, and task execution. WorkflowSim takes the form of directed acyclic graphs (DAGs), the XML-formatted DAG files being imported by the workflow mapper and allocated to the execution site in accordance with the list of tasks that this mapper produces. Meanwhile, to decrease the scheduling overhead, tasks are combined into jobs at the same horizontal levels by the WorkflowSim clustering engine. The connections amongst job executions are managed by the workflow engine. Furthermore, the workflow scheduler is responsible for establishing correlations between jobs and the worker nodes.

5.2.8 CloudSim4DWf

CloudSim4DWf (Fakhfakh et al. [64]) is developed to simulate work flows while they are being executed to investigate the impacts for unusual situations for tasks during execution time. This extension has extended CloudSim with a new resource provisioning policy for dynamic work flows. It has provided with components which are responsible for triggering and monitoring the simulation events. Moreover, it has a graphical user interface that allow users to evaluate their experiments on easier tool than coding platform.

5.2.9 Cloud Data Storage Framework

Cloud Data Storage Framework (Long and Zhao [110]) is an extension of CloudSim which has been developed for conducting experiments on the file striping function on CloudSim. Cloud Data Storage simulator has a layer called Data Cloud layer with the original layers of CloudSim. This layer allows researchers to implement their own layout strategy, file striping strategy and replica management strategy for storage distributions among nodes in a data centre or even multiple data centres.

5.2.10 DartCsim

CloudSim forms the basis for the cloud computing simulation platform that is known as DartCSim (Li et al. [107]) and that presents user-friendly features. This platform affords users a visual approach to utilisation of the CloudSim, with no need for coding to visually configure the simulation experiments. Thus, researchers do not have to concern themselves with trying to understand and creating models of the CloudSim source code, being free to focus instead on their empirical work. This is what makes DartCSim so advantageous. The interface for project development that is provided by this platform is very easy and straightforward to use, as it is not encumbered with technicalities and fine points about the simulation. Configuration of network topology can be efficiently achieved by following the straightforward steps specified by DartCSim (Li et al. [107]). Users can take advantage of DartCSim not only to construct their network or resource configurations on XML, but also to export them, or to import them if existing configurations are employed, without

any difficulty.

5.2.11 CloudSimDisk

The modelling and simulation of energy-aware storage in the cloud system can be carried out with the help of the CloudSim extension known as CloudSimDisk (Louis et al. [112]). The hard disk drive (HDD) manufacturing specification is the aspect that this extension is most concerned with. The elements that make up the HDD comprise a mixture of read/write head traversal movement and platter rotational movement. Current classes in CloudSim are enriched with new classes and parameters that are provided by the CloudSimDisk, including requiredFiles and dataFile, which are capable of determining the transaction time and have been incorporated into the Cloudlet model in CloudSim. The data introduced in the data centre storage are indicated by the dataFile, while the extracted data are indicated by the requiredFiles. To satisfy the specifications of the power model, several alterations have been made to the simulation process in CloudSimDisk.

5.2.12 CloudReports

Researchers can carry out empirical work more easily by employing the graphical user interface that is provided by CloudReports (Teixeira Sá et al. [141]). Being particularly useful in the field of energy-aware cloud computing environments, CloudReports enables researchers not only to document the results of their empirical work, but also to produce and amend policies via the application programming interface (API). CloudReports has introduced to CloudSim including the following components a:

- Persistence layer: This module makes sure that the entirety of the data related to the application and simulation are stored in just one SQL database file for each environment;
- Reports manager: This module makes use of the data from the database file created in the persistence layer in order to produce the simulation report;
- Graphical user interface: The purpose of this module is to provide a GUI written

with the Swing Java GUI widget tool-kit in order to help users to manage and monitor their simulations.

5.2.13 CloudSim Automation

Both CloudSim (Calheiros et al. [46]) and CloudReports (Teixeira Sá et al. [141]) support the CloudSim extension known as CloudSim automation (Silva Filho and Rodrigues [134]). Researchers can gain a more comprehensive understanding and awareness of the overall cloud environment of significant complexity by using this extension, which improves the readability of the simulation files by enabling automation of the simulation scenario related to a YAML file. In spite of this advantage, CloudSim automation presents a significant limitation as well, namely, that it does not support different distributions of resources, VM scheduling, or the power usage associated with the simulation.

5.2.14 CloudMIG

CloudMIG facilitates the exploration of a range of cloud provider options and negotiation of the service level agreement (SLA) (Fittkau [66]). To make sure that the most suitable cloud profile candidate is selected, CloudMIG is designed to compare various cloud environments (Frey and Hasselbring [67]). A comprehensive discussion with regard to the attributes of CloudMIG has been provided in Frey and Hasselbring [67] and Frey and Hasselbring [68]. CloudMIG delivers the following features to CloudSim:

- Retrieval of the architecture and use models of an existing system that is provided by software as a service (SaaS);
- Selection of the most suitable cloud profile candidate;
- Creation of the target architecture and mapping models;
- Adaptation of the systems in order to ensure that they are compatible with the target architecture;
- Verification and simulation of the architecture and performance of static analyses prior to system conversion;

- The culmination of the preceding steps is the conversion of the system into the desired and selected cloud environment.

5.2.15 CDOSim

Selection of a suitable cloud provider that effectively meets the established specifications is a significant challenge facing cloud computing users Ahmed and Sabyasachi [21]. The modelling of cloud deployment options can be successfully carried out with the help of the simulation environment that is supplied by Fittkau et al. [65]. CDOSim (Fittkau et al. [65]) is intended not for cloud providers but for cloud users. It is capable of determining the extent to which the general performance of an application may be influenced by the ignorance of cloud users with regard to the structure of the cloud platform, conducting this assessment with the help of a benchmark.

5.2.16 Bazaar-Extension

Service level agreements (SLAs) and quality of service (QoS) are important considerations for both provider and consumer in cloud computing environments. Bazaar-Extension (Pittl et al. [126]) is an extension of CloudSim that provides a simulation mechanism for the negotiation process between providers and consumers based on resource allocation using the offer-counteroffer negotiation protocol.

Bazaar-Extension characterizes VMs as resource by their processing power, storage, RAM and prices. In this extension, there is a mechanism that used for resource allocation called *supermarket* which lets providers sell their offer to consumer without customizing the offer. Bazaar-Extension consists of three main component : Communication support between parties in negotiation, negotiation strategies and visualization of ongoing negotiations and their outcomes.

5.2.17 CM Cloud Simulator

As is explained in Armbrust et al. [32], Cloud Computing services can be divided into three categories: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure

as a Service (IaaS). All of these types of service need to be investigated in any research aiming to make improvements to services of benefit to both providers and consumers of cloud computing services. When users apply for services, different providers have different costs based on their specifications and conditions of service. However, there is no method for evaluating the relative costs of services from providers as it is difficult and expensive to evaluate costs for a specific cloud computing environment. As a result, CM Cloud Simulator (Alves et al. [30]) has been delivered as an extension of CloudSim to overcome the difficulties in running experiments on different cloud providers infrastructure.

5.2.18 CloudTax

CloudTax (Pittl et al. [125]) is en extension of Bazaar-Extension which is an extension of CloudSim to investigate the negotiation process between providers and consumers. CloudTax has added the feature of cost of tax based on basic economical literature to analyse the impact of tax on market. Modelling the tax impact is important as they influence strongly the efficiency of negotiation processes.

5.2.19 EdgeCloudSim

EdgeCloudSim is built upon CloudSim in order to provide the functionality and features of edge computing infrastructure. It is clear that in Sonmez et al. [137] there are new classes and modules for edge computing systems have been added to CloudSim such as a Load Generator Module, Transmission delay in the WLAN and WAN, Edge Orchestrator to handle the incoming request from client where to be processed and mobility module to specify the location for the mobile device connected to the edge computing. to Evaluate EdgeCloudSim, there is a comparison study with expected results for face recueqntion simulated experiments on three different architecture for the simulated fog computing system as single-tier, two-tier, and two-tier with edge orchestrator. It would be advantageous if there was a comparison study against a real data traces of mobility computing experiments.

5.2.20 CloudSim Plus

CloudSim Plus (Silva Filho et al. [135]) is an extension of CloudSim framework to improve its maintainability, reusability and extensibility in terms of adding classes for distributions, network an power. These classes allow users to use more functionality rather than the basic functions on CloudSim and having a fast implementation of simulation scenarios The most important features have been added to CloudSim in This extensions are Extensible Improvements , Reduced Code Duplication and Tests and Code Coverage. Extensible improvements is conducted by allowing users to inherit interfaces. It is reported that CloudSim Plus has scale down the number of duplicated lines of code from CloudSim 3.0.3 in 30% as the duplicated code well known in the literature to be major complication for software maintenance and characteristic.

5.3 Conclusion

This chapter reviewed software simulators for cloud computing called CloudSim, Greencloud, and Mininet, and succinctly categorised a set of publicly available extension projects, outlining their features and benefits. The main goal of this section of the work is to provide a survey of the current CloudSim software ecosystem, which is intended to be useful for researchers who may want to (a) use CloudSim or one of its extensions for an empirical investigation, or (b) develop a new extension for CloudSim that addresses a factor that has been neglected so far. The main features of the frameworks were presented in this chapter based on their extensions' advantages, limitations, and validations. These should allow researchers to explore the similarities and differences between the various extension in order to determine the most appropriate tool to use or extend for a particular experiment. For instance, by applying the method for selecting a simulator to the set of simulators that can provide an environment for workflow experiments, it becomes clear that four extensions of CloudSim exist that support the modelling of the workflow process in cloud computing. These extensions are supported by different features that may improve the capability and usability of the simulation environment. While CloudSim4DWf (Fakhfakh et al. [64]) supports a graphical user interface and dynamic changes, DynamicCloudSim

([42]) affords the possibility of modelling a heterogeneous data centre.

Ahmed and Sabyasachi [21] stated that cloud computing faces some challenges such as security, cost modelling, energy management, and virtual machine migration, and researchers are working to solve these obstacles by trying new algorithms and techniques in cloud data centres.

Therefore, based on the features a researcher requires, along with the available data about the simulators presented in this study, the most suitable extension can be straightforwardly identified by applying the method. End users can also generate a mind map based on the method's steps for the selection of a suitable simulator based on the researchers' requirements.

5.3.1 Contributions

This work enables researchers to identify suitable simulation tools for their experiments. In particular, CloudSim and its extensions have been examined with regard to their diverse features and relative limitations. This involved the categorisation and explanation of existing features that have been added to the CloudSim extensions under investigation. Several common cloud computing research challenges are not modelled explicitly in CloudSim or its extant extensions, however, and developers of CloudSim can therefore easily identify missing features among the reviewed extensions of CloudSim, and may be encouraged to contribute new extensions. For example, security risk is one of the highest concerns in Cloud Computing (Khan et al. [94]), and although several projects have used CloudSim to investigate security (Sun et al. [140]; Zardari et al. [156]; Karthik and Shah [88]), it would be more helpful if there were a security-focused extension of CloudSim that could model different aspects of monitoring tools and security risk management.

5.3.2 Future Opportunities

The various CloudSim extensions provide additional features, many of which are orthogonal. However, it is difficult to combine the features of multiple extensions into a single coherent simulator, as this would require manual source code merging, conflict resolution,

and extra integration work. The field of cloud computing research needs more capable simulators, and it would therefore be advantageous for future CloudSim work to develop an extension framework (perhaps resembling the Eclipse plugin interface) to enable combinations of extensions.

For example, DynamicCloudSim and WorkflowSim have similar features in terms of providing the opportunity to model a heterogeneous data centre with a fault tolerance mechanism. Further, there is still a clear need to investigate simulator accuracy and reliability by reproducing existing simulated experiments on actual test-beds and comparing the results. This is a major challenge for the cloud simulation community, however, and it is not exclusively restricted to the CloudSim ecosystem.

Chapter 6

Performance of Benchmarking a Micro Data Centre

6.1 Introduction

This chapter offers an assessment of the performance of a prototypical micro data centre operating on a range of intensive workloads. A series of experiments were carried out in order to evaluate the scalability of Raspberry Pi devices running intensive workloads in a cluster configuration. According to Bondi [40], scalability is the ability of a system, network, or process to cope with an increasing amount of work, or its potential to be expanded to handle such volumes of work. The acquired data and the outcomes of this chapter should thus be valuable when utilised in the context of evaluating the performance of single board computers and micro data centres. The automated scripts presented in this evaluation deliver clear and detailed steps to allow data centre infrastructure to be completely configured and benchmarked, offering a highly beneficial tool for researchers to adopt during research projects. Using the scripts, written in Bash (a UNIX shell and command language), as a starting point, fully setting up 30 nodes of Raspberry Pi in an automated manner was accomplished in a straightforward manner during this research. The workloads that were then run on top of the infrastructure are fully explained below; these have been profiled to illustrate the Raspberry Pi cloud performance.

This chapter further describes the infrastructure and benchmarks already mentioned in

the methodology chapter. This infrastructure is set up for comparison validation work as seen in chapters 7 and 8, and the current chapter mainly links the methodology chapter and the results of the accuracy check of the simulators given in Chapter 7. It is also delivered to present the source of the data set used to develop a machine learning predictive model seen in chapter 8.

Novelty: Generating a data set to examine the performance of a micro data centre running different types of workloads on Raspberry Pi devices generates a performance evaluation that can be used for the validation of different prediction modelling tools.

6.2 Performance of The Glasgow Raspberry Pi Data Centre

This section describes the performance of the Glasgow Raspberry Pi data centre in terms of running micro benchmarks and other existing benchmarks for the Map Reduce application and key value database application. The main idea behind this section of the dissertation is to provide a performance evaluation of actual workloads run on a physical data centre to act as an open source data set for researchers to use for micro data centre performance evaluation. These experiments profiling on the Glasgow Raspberry Pi Data Centre are also used to evaluate the selected simulators and the predictive model in a set of cross validation experiments in chapters 7 and 8.

6.2.1 Dataset

This data describes the performance of the Glasgow Raspberry Pi Cloud in terms of the number of instructions that different workloads manage during the running time on different cluster configurations, alongside measures of the network performance. The infrastructure and workloads have different features and specifications that are also delivered in this data.

Graphs 6.3 - 6.14 show the number of instructions and the execution time for running these workloads on different cluster sizes in the Raspberry Pi data centre. The data

was gathered by calculating the arithmetic mean of running each workload 30 times in each size of cluster. No inappropriate format type were allowed, based on errors and inconsistent units; this was achieved by looking at logs files and monitoring tool results for each workload, and matching these with the concepts modelled in the simulators in the evaluation work presented in chapter 7. The approach for data collection takes a set of nodes running these workloads and profiles a single node from the selected nodes by using the arithmetic mean of the node measurements.

6.2.2 Experimental Design, Materials and Methods

A micro data centre was built on top of multiple Raspberry Pi devices. The specifications of the Raspberry Pi devices are given in table 6.1 and Figures 6.1 and 6.2 show the design and schematic architecture of the Raspberry Pi data centre. Different types of workload (Batch Processing and Transactional Processing) were then selected and profiled. The data shows the number of instructions per node based on profiling a single node in the data centre. The averages of running the profiling tools on multiple nodes show relatively similar results on each node. The result for performance is different from other works such as Abrahamsson et al. [19]; Cox et al. [53])however, as it includes the total execution time and number of instructions to demonstrate scalability in the micro data centre's size and a wide range of workloads was run, as shown in table 6.3. The following list describes the steps used to conduct the performance evaluation experiments on the Raspberry Pi cloud.

- Raspberry Pi v2 and Raspberry Pi v3 clusters were generated, as the existing clusters of Raspberry Pi v1 lacked the ability to profile the devices using the Perf profiling tool.

- Various workloads (Web Servers, MPI framework, Big Data application and Key value Data base) were installed.

- The network topology was set as a single level tree with open flow standard Ethernet switches.

- A stock router was used to connect the nodes.

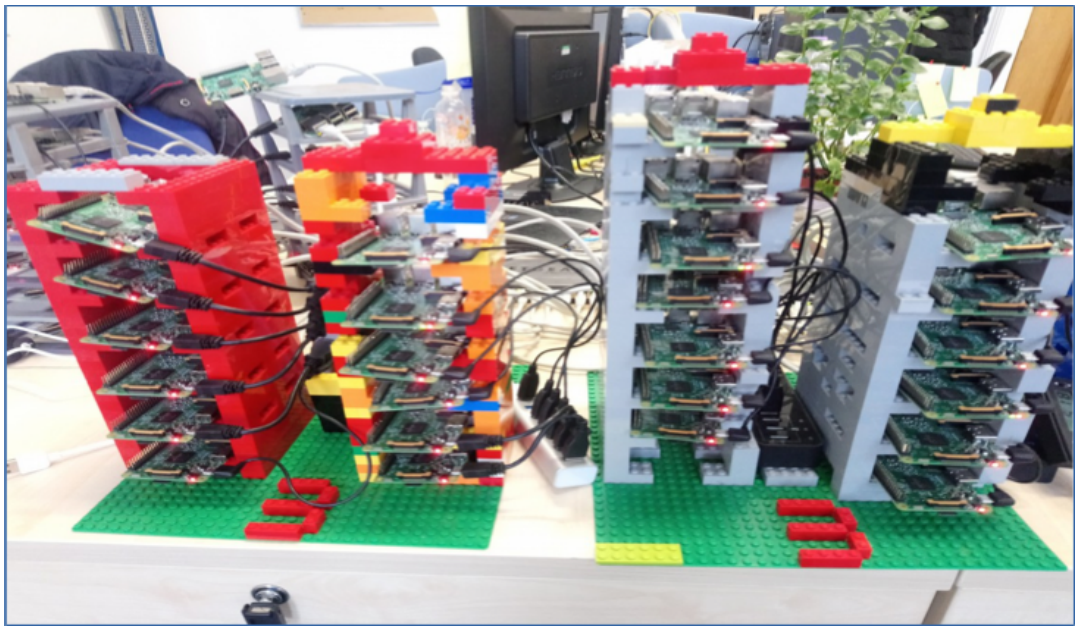

Figure 6.1: *Raspberry Pi Nodes in the Micro data centre*

- Various different profiling and managing tools were installed to capture performance.

- The protocol Secure Shell (SSH) was utilised to open communication between nodes as described in SSH (ssh [17]). SSH is a cryptographic network protocol that allows remote login and other network services to operate securely over an unsecured network (Ylonen and Lonvick [155]).

- ClusterSSH (Rankin [128]) towas used to manage the installation of the workloads among the clusters and Ansible was used to script the steps required for of installing the Spark and Cassandra installations.

All profiling tools and managing cluster tools are shown in Table 6.2 , and these were used to gather the features of the Raspberry Pi Cloud as illustrated in table 6.1. These are the features used to customize the modeling on CloudSim, GreenCloud, and Mininet simulation tools and to build the predictive modeling tool for the Raspberry Pi Cloud in this research.

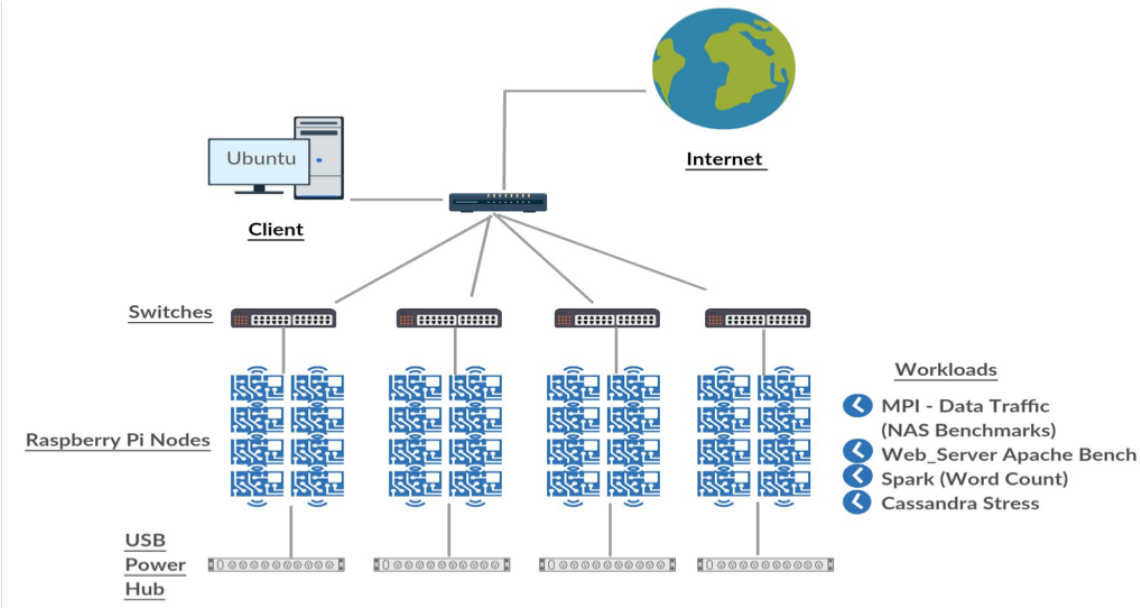

Figure 6.2: *Schematic Architecture of Raspberry Pi data centre*

Table 6.1: Features of The Raspberry Pi Cloud Infrastructure

Node	Raspberry Pi 3	1.2 GHz , 64bit quad core ARMv8
		4 Cores CPU
		1 GB RAM
	Raspberry Pi 2	900 MHz, 32 bit quad core ARM CortexA7
		4 Cores CPU
		1 GB RAM
Storage	SD Cards	16 GB
Network	Topology	Tree
	Switches	Standard ethernet
	Bandwidth	100 Mbit/sec

6.3 Actual Workloads and Benchmarks Run on The Raspberry Pi Cloud

In this section, we provide details with regard to the results of my experiment. We run a variety of jobs on a Raspberry Pi cluster. Table 6.3 outlines the workloads that have been profiled on the Raspberry Pi cloud.

Table 6.2: Profiling and Managing The Raspberry Pi Cloud Tools

Tool	Used for
perf	Instruction Count
Dhrystone	Machine Capacity
ntop	Network Monitoring
iperf	Network Bandwidth and Latency
ClusterSSH	Cluster Management
HAproxy	Load Balancer
Ansible	Automated Scripting on multiple nodes
iptables	Automated Scripting for network performance measurements

Table 6.3: Workloads Tested on Raspberry Pi Cloud

Type	*Benchmark*	*Profiling Tool*
High Performance Computing	Data Traffic (DT) from NAS Bailey et al. [34]	mpiP
Web Server Application	Apache Bench ABa [1]	iptables, iperf and perf
Database Application (NoSql)	Cassandra-Stress Abramova and Bernardino [20]	perf
Big-Data Application	Spark - Word Count Dean and Ghemawat [57]	Spark Log, iptables and perf

6.3.1 Message Passing Interface: NAS Benchmark – Data Traffic DT

Parallel Benchmarks (NPB) are micro benchmarks that design to evaluate parallelism for super computers. There are different types of benchmarks of NAS for different purposes such as kernels, pseudo-applications, unstructured computation, parallel I/O, and data movement Bailey et al. [34]. We run and profiled a benchmark for data movement (Data Traffic benchmark DT), with input size set for Class: A on both types of infrastructure Raspberry Pi 2 and 3. We choose and specified the DT benchmark as the other benchmarks from NAS require large footprint of memory to be run on Raspberry Pi 2 and 3. We test different sizes of high-performance computing applications on the cluster in order to gather execution times for each application on a particular node. Figure 6.3 and 6.5 show the number of instructions per node for the DT benchmark on Raspberry pi 2 and Raspberry Pi 3 whereas, figures 6.4 and 6.6 show the total execution time.

6.3.2 Web Server: Apache Bench AB

Web servers are widely deployed in cloud computing data centre. A web server processes requests via Hypertext Transfer Protocol (HTTP). We aim to profile the network per-

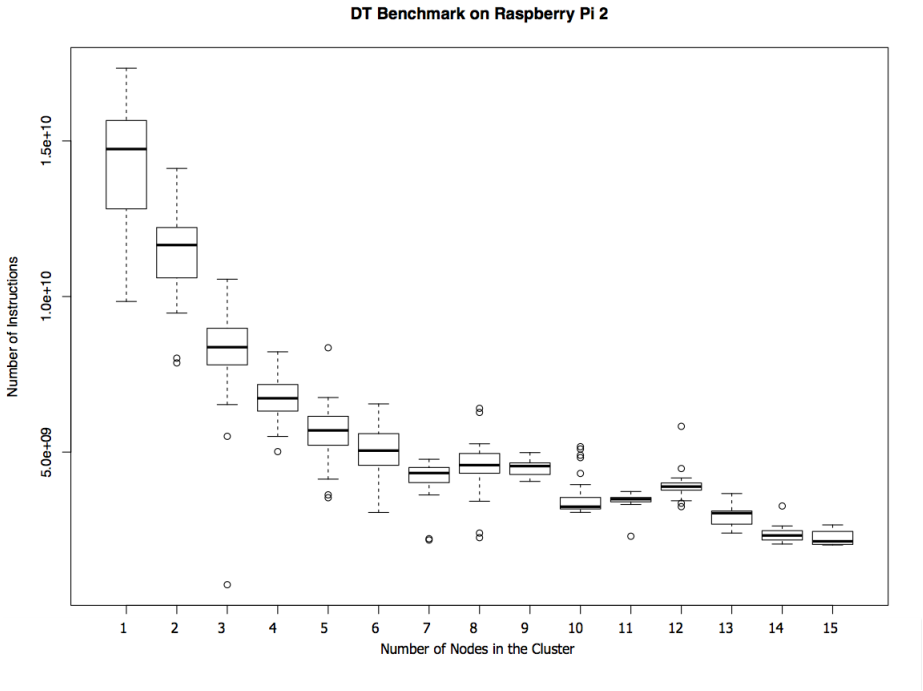

Figure 6.3: *Number of Instructions for DT Benchmark on Raspberry Pi 2*

formance of our Raspberry Pi data centre implementing the Apache Bench ABa [1] that used to measure concurrent HTTP requests between a client and server. We configure the benchmark with 10000 concurrent requests from the client user to multiple servers in the Raspberry Pi data centre. I have used the perf tool in order to count the number of instructions per node on the server devices and showing the performance during scaling up and down the number of servers responding to the client request. Figures 6.8, 6.7, 6.10 and 6.9 present the performance of Raspberry Pi 2 and 3 for Ab benchmark in terms of Number of instructions and total execution time.

6.3.3 Spark: Word Count

Map reduce is widely used in large and commercial data centre like Google Cluster and Amazon, It has been evaluated in the work Dean and Ghemawat [57]. Map reduce provides the advantages for scalability in dividing computation among cluster without much consideration about how it distributes the workload from the end user thinking.

For a representative big data application we test the performance of Raspberry Pi devices running map reduce job for a word count benchmark. My script for starting the spark

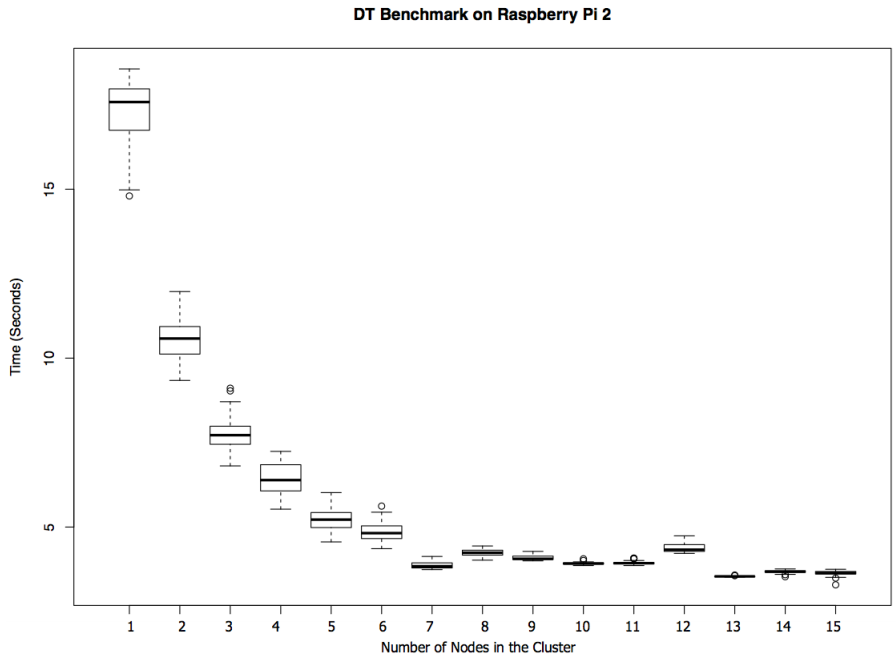

Figure 6.4: *Execution Time for DT Benchmark on Raspberry Pi 2*

service and testing 3 GB text file for mapping and reducing the count over multiple nodes is designed to be automatically run. Graphs 6.12 and 6.11 show the performance of the benchmark in terms of the total execution time on multiple nodes and the number of instructions per node.

6.3.4 Cassandra: Cassandra-stress

In this workload we profile the performance of Raspberry Pi data centre running NoSQL database management system called Cassandra[12]. We choose the benchmark Cassandra-stress to test writing and reading on and from large database that has been stored on our Raspberry data centre. We install and test Cassandra service on Raspberry Pi data centre by costmised Bash script to be publicly available for anyone to use for Linux based servers on data centres. Figure 6.14 and 6.13 show the performance of Raspberry Pi 3 running Cassandra stress with the configuration that is stored on (GitHub)

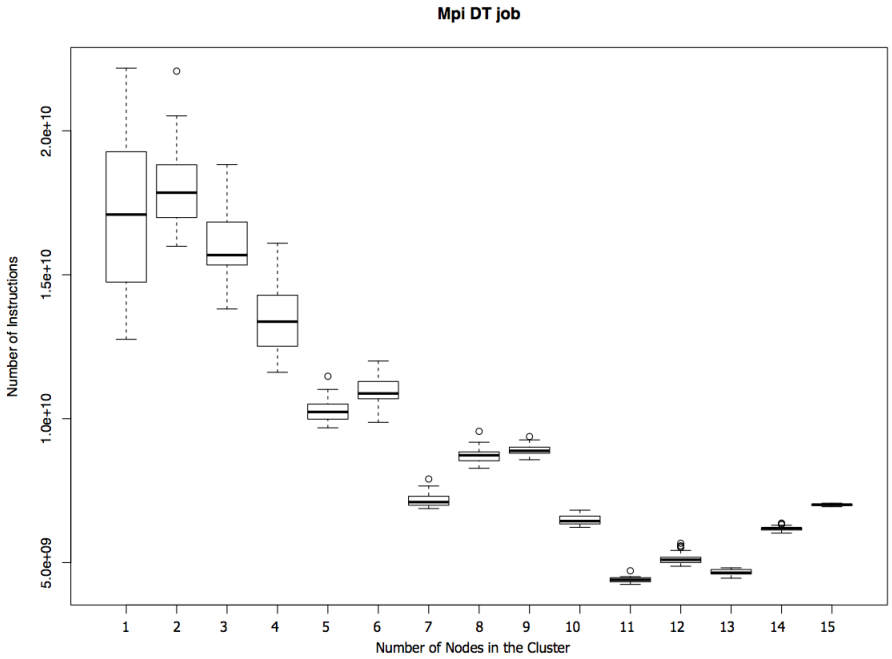

Figure 6.5: *Number of Instructions for DT Benchmark on Raspberry Pi 3*

6.4 Profiling Tools: Perf, iperf and mpiP

The infrastructure specifications were collected via different profiling tools. All workloads run on the data centre were profiled on a per-node basis using standard Linux performance monitoring tools, while a Bash script including *perf* (per [16]) was implemented to generally profile any workloads run on the Raspberry Pi cloud. The perf record was run as it identifies the specific processor ID when gathering its hardware events. Another tool, *iperf* (ipe [14]) was also used in order to measure and specify network features such as latency, throughput, and bandwidth. The results of running iperf provided the bandwidth and the throughput of the network, as it is a tool for active measurement of the maximum achievable bandwidth on IP networks. It thus supports the tuning of various parameters related to timing, buffers, and protocols (TCP, UDP, SCTP with IPv4 and IPv6). For each test, the bandwidth as well as other specific parameters was recorded.

Moreover, *mpiP* (mpi [15]) was used as a profiling tool in order to profile cluster performance by measuring the execution time of the jobs on each node in the cluster. MpiP is a lightweight profiling library for MPI applications that collects statistical information regarding MPI function-calls.

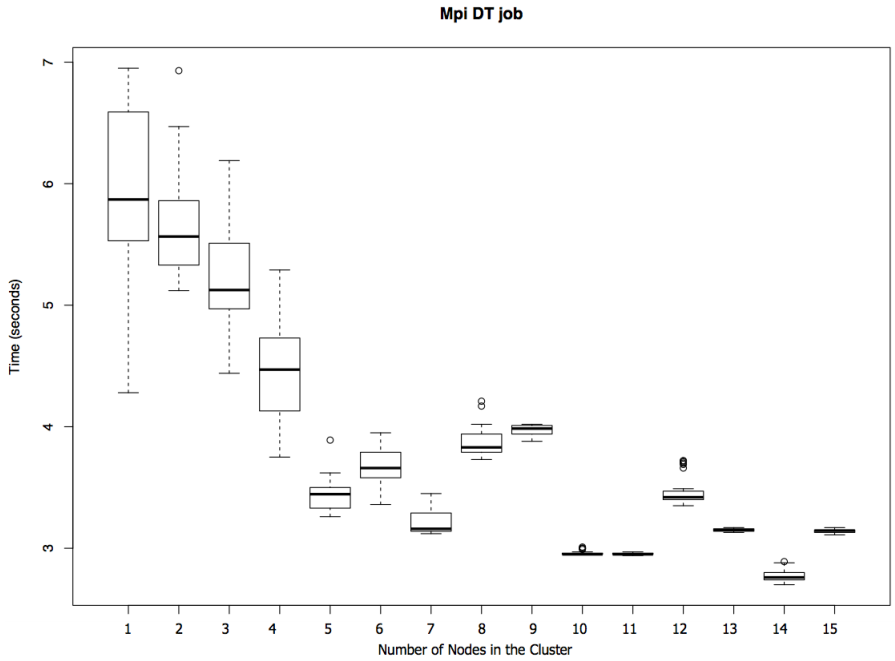

Figure 6.6: *Execution Time for DT Benchmark on Raspberry Pi 3*

Dhrystone (Dhr [4]; Dhr [5]) measures the performance of nodes in the cluster in terms of million instructions per second (MIPS).

The protocol Secure Shell (SSH) (Ylonen and Lonvick [155]) was also applied in the experiments to open communication between nodes, as described in SSH (ssh [17]). SSH is a cryptographic network protocol that allows remote login and other network services to operate securely over an unsecured network.

For cluster management, a customised version of *Ansible* by Hochstein and Moser [76] , self-written scripts, and the *ClusterSSH* tool (Rankin [128]) were used for monitoring and deploying packages and required libraries in an automated manner. For load balancing, *Haproxy* (Kaushal and Bala [90]) was used.

In terms of network monitoring, an open source tool called *ntop* (Deri et al. [59]; Deri and Suin [58]) was used to capture the throughput and total amount of data distributed between nodes in the data centre during the execution time. This gives more details about the network than iperf. It is a web-based application that reports the network traffic, and which is able to conduct a traffic analysis for different types of data in the network such as the HTTP server for web access. Other advantages of ntop include the fact that it

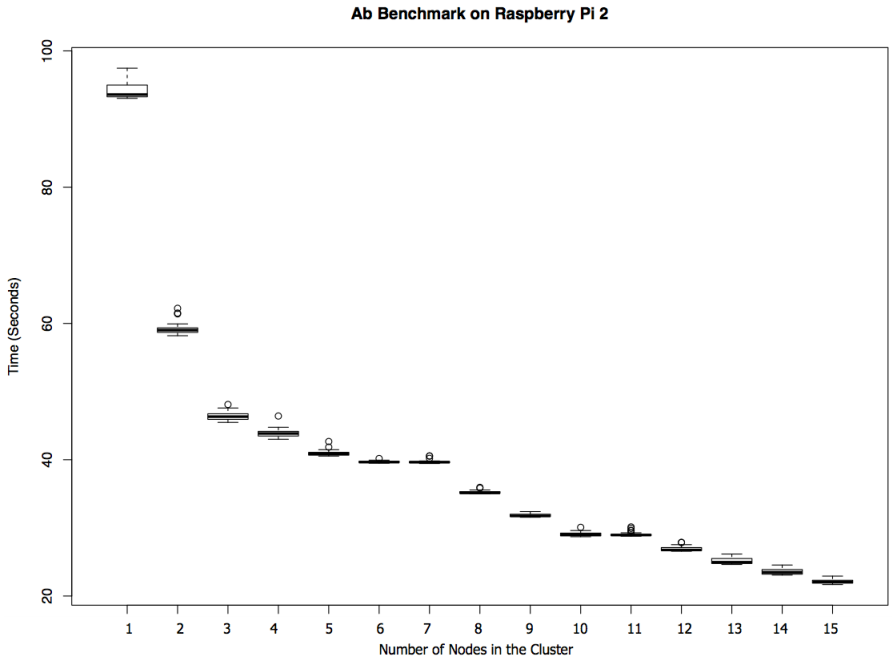

Figure 6.7: *Execution Time for AB Benchmark on Raspberry Pi 2*

supports various network media types, and it can store traffic information in an SQL database. Thus, ntop and iptable were used in the research to measure the amount of data transferred during the execution time of the workload, run and profiled in automated manner via Bash scripts. The idea behind using both was to confirm the distribution amount of data transferred in all nodes involved in running each workload.

There are several main approaches to traffic accounting:

- Determine and count traffic flows (netflow, openflow).
- Intercept (catch) traffic data at the network interface (bpf, tcpdump).
- Use kernel iptables (ip accounting).

Of these, the iptables method was chosen as is it does not require additional equipment or complex software. All required programs are included in every modern Linux distribution, as they are universal tools used in many different cases; iptables is a standard tool for managing internal (kernel-based) Linux IPv4 firewalls, and it is thus installed and enabled by default. It is used for adding permissive rules for interesting ports and interfaces and the kernel then counts all packets corresponding to this rule. For the current case, when

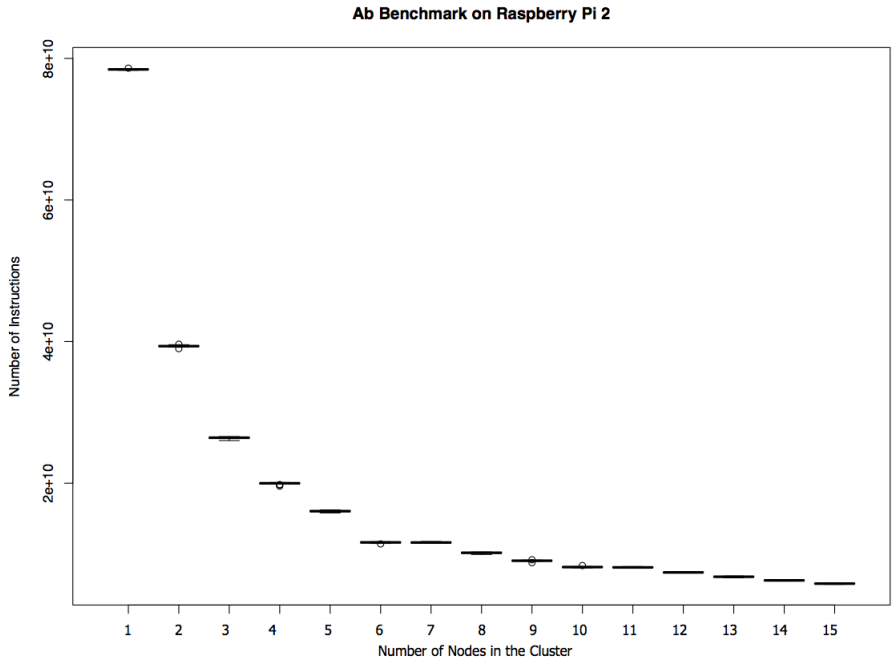

Figure 6.8: *Number of Instructions for AB Benchmark on Raspberry Pi 2*

there is a single application at the node at a very specific port and this application starts to perform actions, IP accounting is the best and simplest measure. To perform a 'counter reset', it sufficient to flush the table, as the firewall is fully open. To cut the required data from both scripts for perf or iptables, scripts collect the required data by commanding the output using text processors: To cut required data from both scripts in my research for perf or iptables, the scripts collect the required data by commanding the output using some text processors such as:

- **grep**: To filter strings contains data for required interface and port;
- **awk**: To format output data according to conditions;
- **tr**: To remove end-of-line and return signs from the data.

Base system soft data was used to make a timestamp. These scripts generate reports that are more ergonomic, human-readable, and in a standard CSV format.

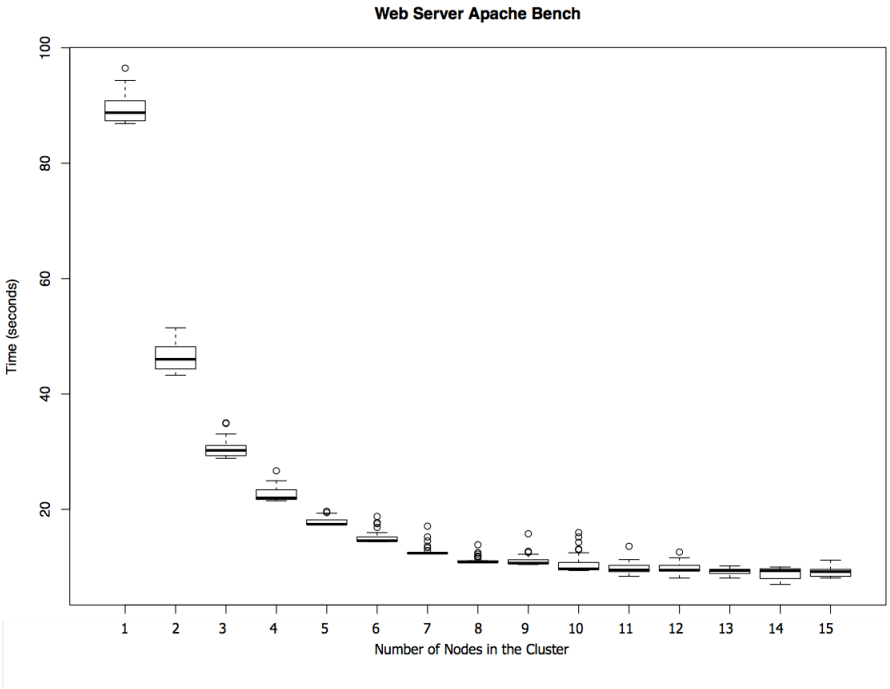

Figure 6.9: *Execution Time for AB Benchmark on Raspberry Pi 3*

6.5 Discussion

The measurement of three different factors within the dataset associated with the performance of a realistic micro data centre is made possible by the fact that the profiling of every workload was carried out using two kinds of machines, Raspberry Pi v2 and Raspberry Pi v3. In order to determine the execution time, the approach adopted involved the implementation of the command (time) in a BASH script. The unit of measurement in which the calculated execution time was expressed was seconds. The BASH script written for the implementation of the command (time) also comprised perf for the measurement of the number of instructions in every workload during run-time. The measurement of the volume of data transferred between servers during the execution time was undertaken using the software program iptables, which was also included in the BASH script.

The execution times (S) associated with the DT benchmark for the MPI job on the two kinds of machines employed (Raspberry Pi v2 and Raspberry Pi v3) are indicated in Figures 6.4 – 6.6. These figures clearly show that an increase in the number of nodes up to seven nodes within the cluster led to a marked reduction in time; subsequently however, the compromise between the computation time and the communication time for

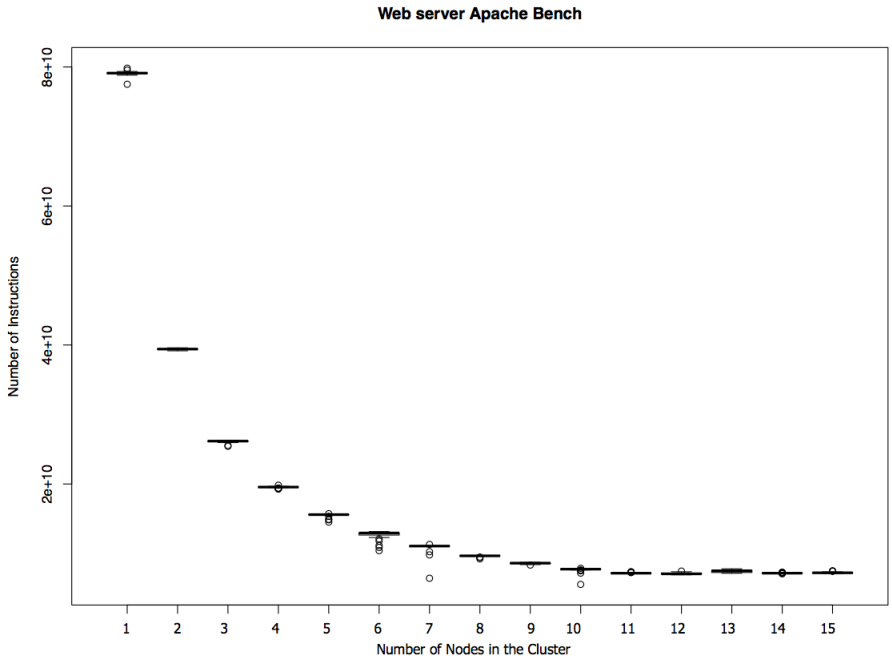

Figure 6.10: *Number of Instructions for AB Benchmark on Raspberry Pi 3*

the application caused a gradual increase in time up to 15 nodes. It must be noted that, in the case of Raspberry Pi v3, the overall execution time associated with the workload was impacted by the fact that the number of instructions for each node was not constant.

Meanwhile, as seen in the Figure 6.9, the distribution exhibited by the benchmark AB during execution time was balanced. The reason for this balanced distribution was that the benchmark was quite small, owing to the compression work carried out on Mininet as described in the sixth chapter, which necessitated the running of a small size system. The number of instructions and the execution time were correlated as they achieved a balanced distribution within the cluster, as definitively proven by the additional workloads, such as DT, Cassandra Stress and Spark Word Count. The correlation between the number of instructions and the number of nodes related to the Spark Word Count workload is shown in the fourth figure below. The explanation for the existence of a correlation between the number of instructions and the number of nodes is that, after the cluster expanded to more than seven nodes, the intensity of the work on the memory in every node during the execution time is diminished. As a result, the workload exhibited a steady performance with a similar execution time. Furthermore, the number of instructions and the execution

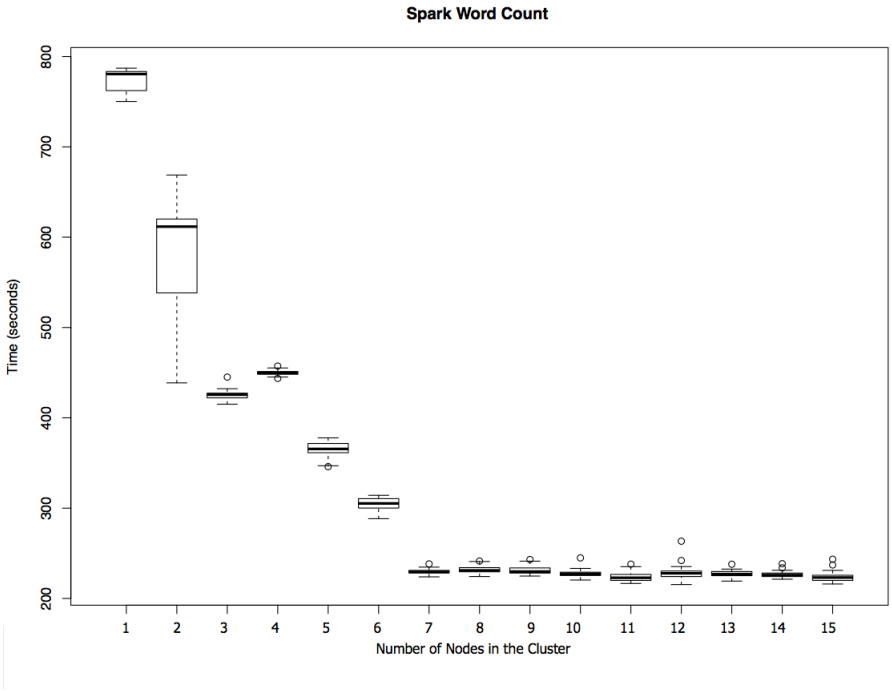

Figure 6.11: *Execution Time for Spark-Word Count Benchmark on Raspberry Pi 3*

time were directly and clearly correlated. What is more, this correlation existed despite the fact that the communication time between the nodes in every cluster had an impact. In spite of this impact of the communication time, overall computation time was not restricted to communication time between the nodes, as attested to by the fact that it showed up during the execution time associated with the workloads during the profiling of the latter on nodes within the cluster that ranged in size from one to five.

The similarity of outcomes with regard to the number of instructions became even closer when the number of nodes running the workloads was increased. This can be clearly seen in Figures 6.3 – 6.14, which are related to the various benchmarks examined on the two types of machines; these include the Mpi DT job, the AB benchmark, and the Cassandra-Stress benchmark as well as the Spark Word Count. This observation was made possible by the fact that the same experiment was carried out a thirty times, with the number of instructions being gradually reduced; each node had more or less the same number of instructions, which did not fluctuate, but remained stable and balanced.

Regarding the amount of data transferred between nodes in the infrastructure, the total amount of data for each workload is presented in Graphs 6.15 – 6.22. The amount

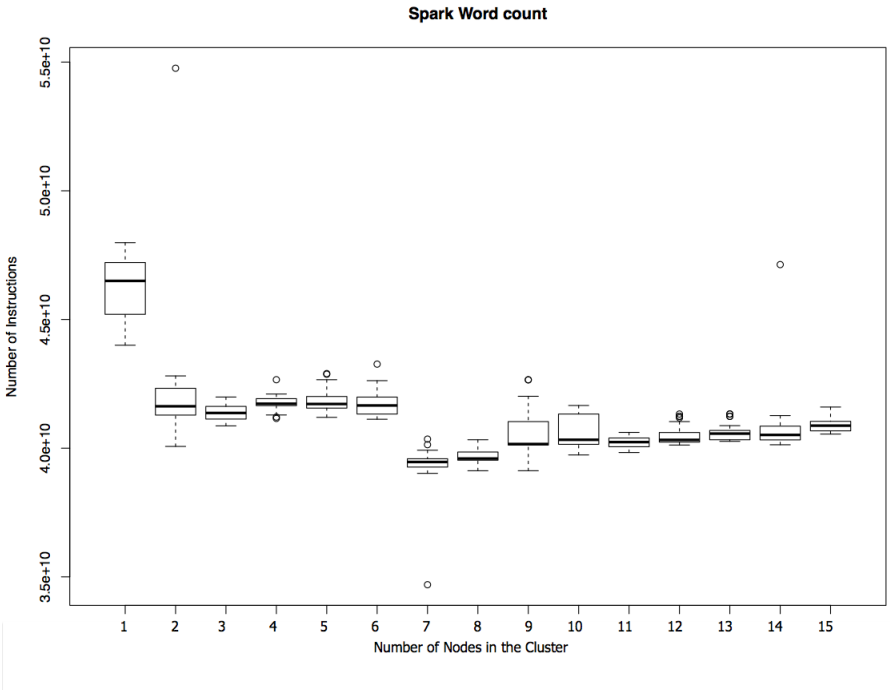

Figure 6.12: *Number of Instructions for for Spark-Word Count Benchmark on Raspberry Pi 3*

of data decreased when the number of paralleled nodes running the workload increased. However, in the case of the Spark - Word Count benchmark, there was an increase in the amount of data when the number of nodes increased after four nodes, due to the exchange of system (hidden) data required for map reduce jobs.

6.6 Conclusion

Providing a report on the performance of the data centre is likely to be of benefit to researchers in the future as they investigate either new techniques or algorithms in the field of Cloud Computing by comparing the results of these evaluation against their own work. In this chapter, the focus was placed on the number of instructions and network performance in terms of describing the workloads. The number of instructions is widely used as a metric for representing workloads in simulators such as CloudSim, GreenCloud, iCanCloud, and extensions of CloudSim.

Different simulation tools are developed for particular types of experiments, though by systemically comparing the results of different types of workloads on each particular simulator with actual physical infrastructure, it is possible to present the level of accuracy

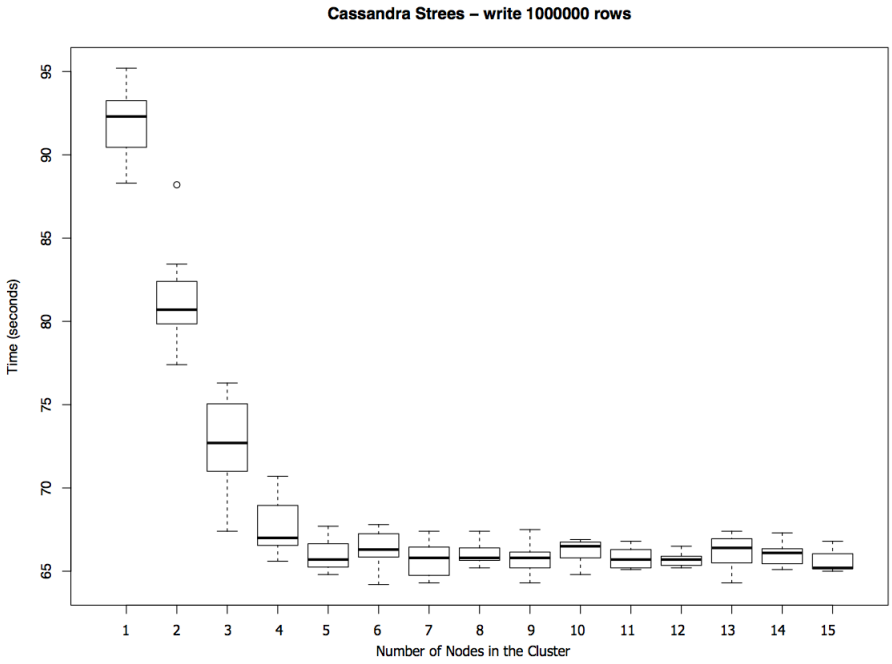

Figure 6.13: *Execution Time for Cassandra-Stress Benchmark on Raspberry Pi 3*

for each simulator. Moreover, by using a range of workloads performance compared with actual infrastructures, the level of accuracy can be determined and the simulator can be updated based on the requirements of the type of experiments under consideration.

Researchers using any of the three simulators investigated would find it advantageous to utilise the data centre performance report from this work as a guide for their own work due to the fact that it provides a step-by-step instruction process, and also as a comparative tool in order to ensure the legitimacy and relevancy of their own data as their projects progress. This data and evaluation can be used to sanity check any blatantly false results, and to validate any further simulation tools for cloud computing by modeling the characteristics of the different types of presented workloads based on their characteristics and features, comparing the performance against the actual results generated by the Raspberry Pi data centre.

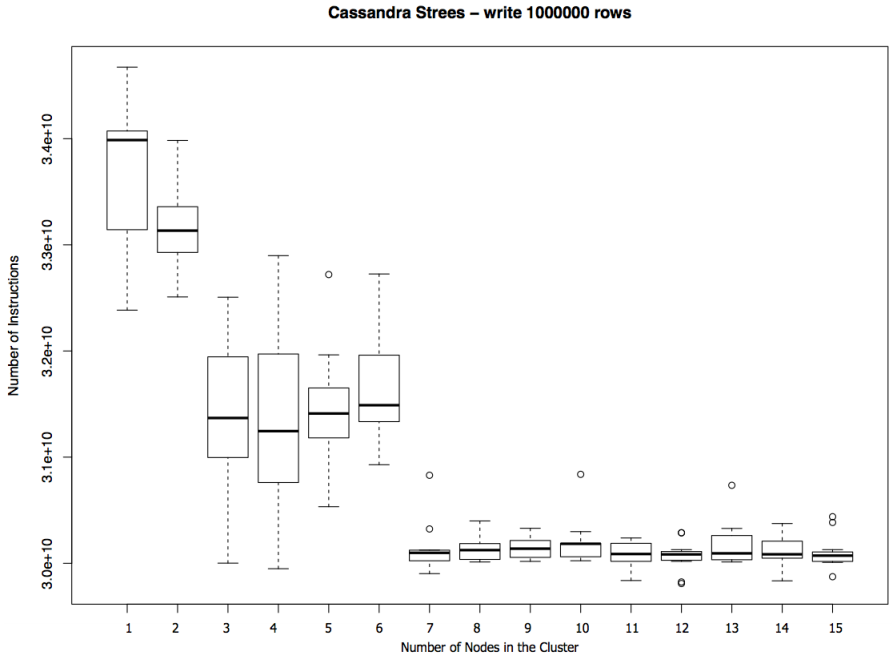

Figure 6.14: *Number of Instructions for for Cassandra-Stress Benchmark on Raspberry Pi 3*

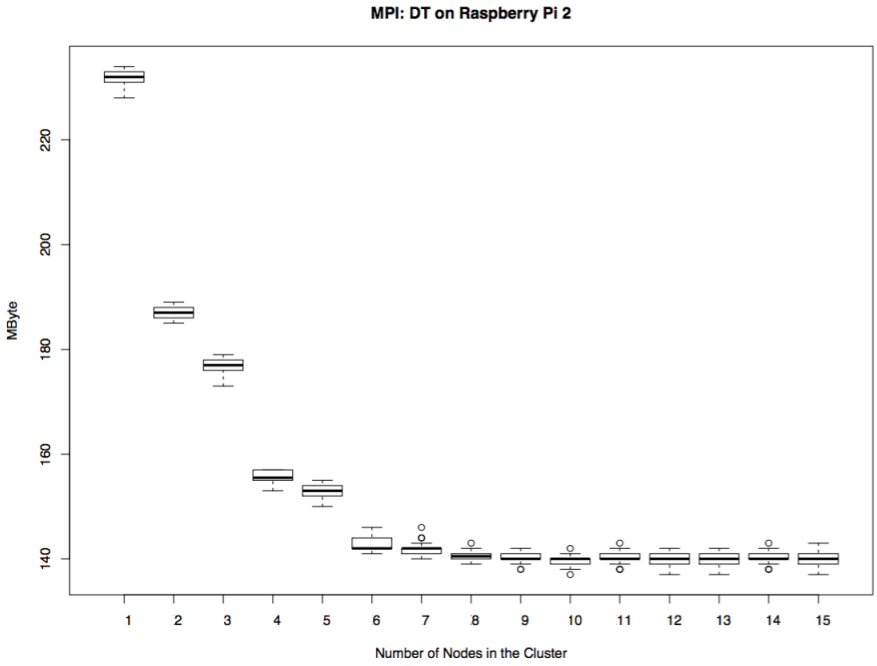

Figure 6.15: *Amount of Data Transferred in the Cluster for DT Benchmark on Raspberry Pi 2*

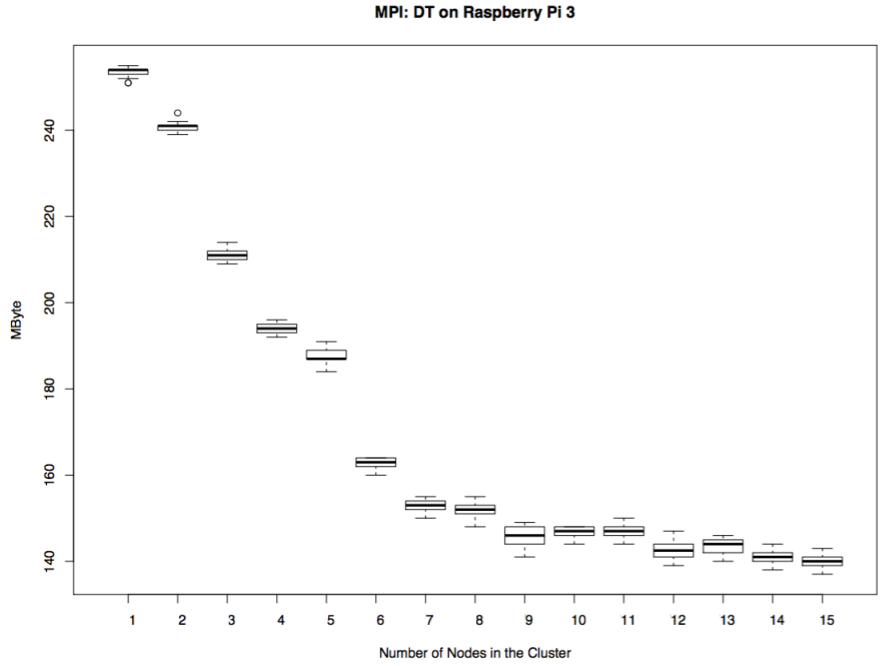

Figure 6.16: *Amount of Data Transferred in the Cluster for DT Benchmark on Raspberry Pi 3*

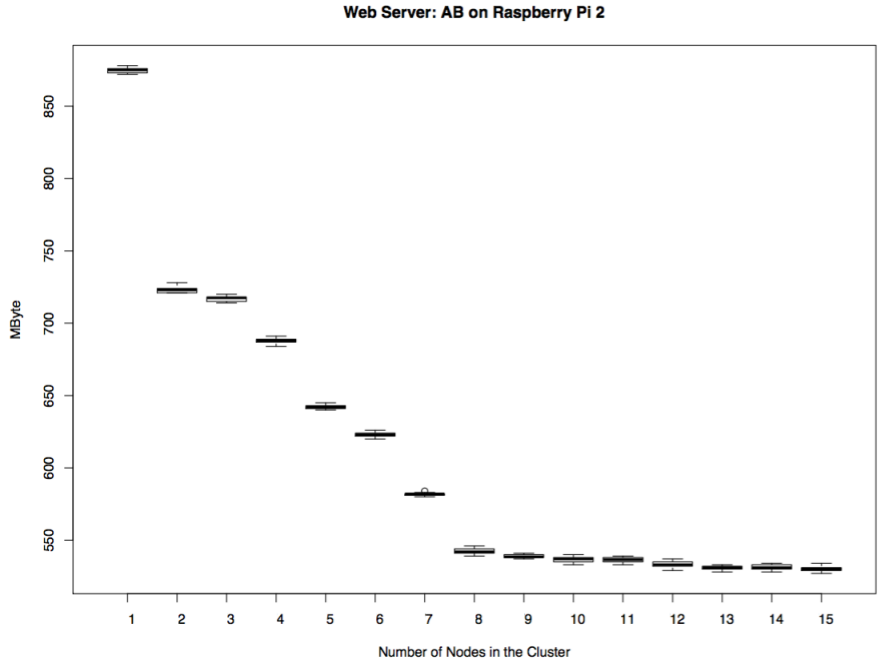

Figure 6.17: *Amount of Data Transferred in the Cluster for AB Benchmark on Raspberry Pi 2*

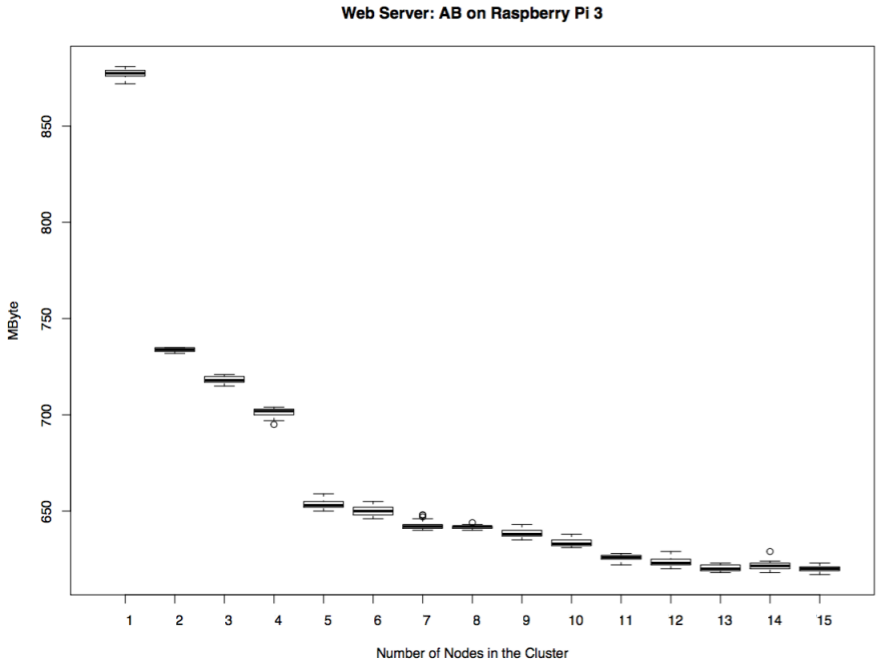

Figure 6.18: *Amount of Data Transferred in the Cluster for AB Benchmark on Raspberry Pi 3*

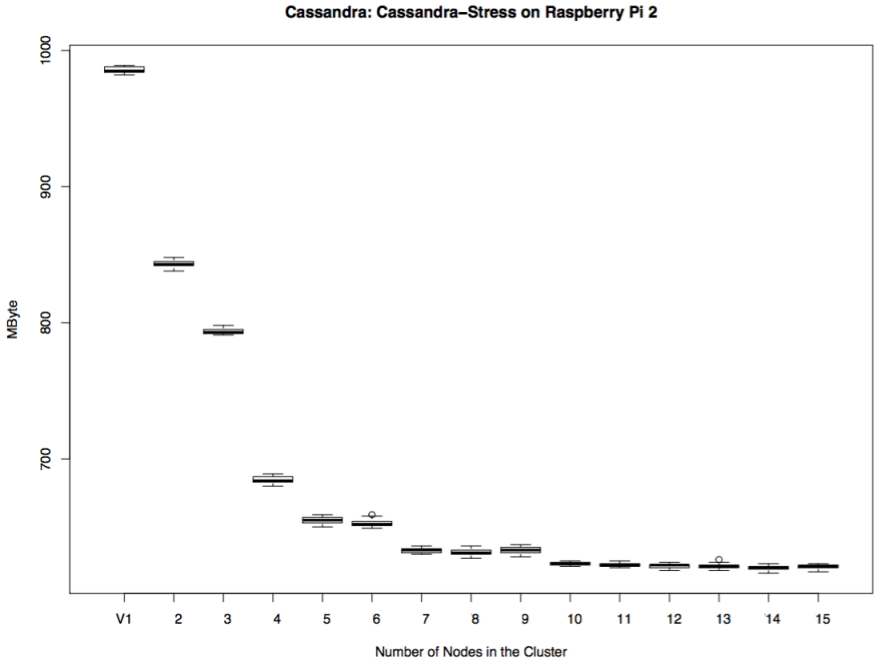

Figure 6.19: *Amount of Data Transferred in the Cluster for Cassandra-Stress Benchmark on Raspberry Pi 2*

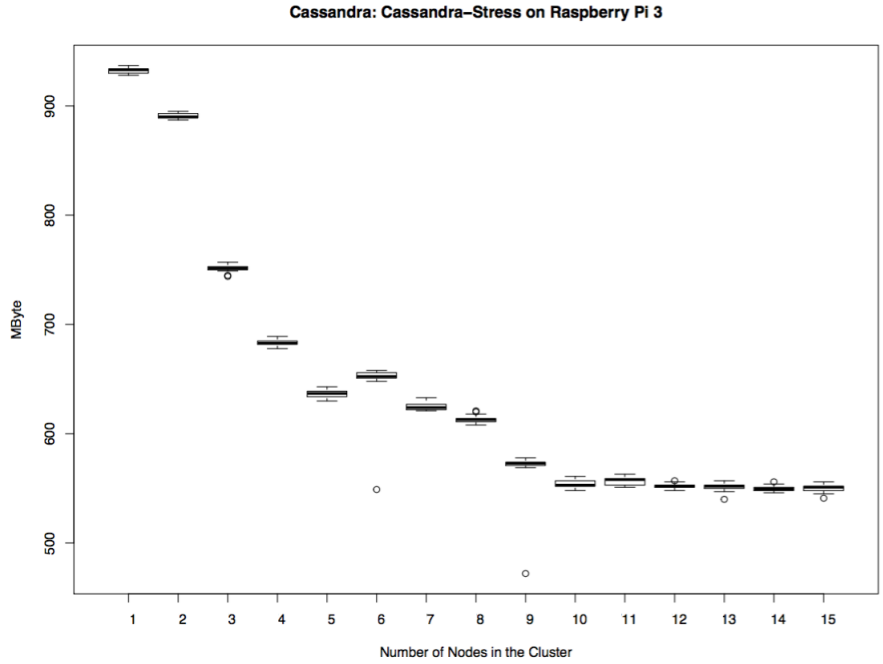

Figure 6.20: *Amount of Data Transferred in the Cluster for Cassandra-Stress Benchmark on Raspberry Pi 3*

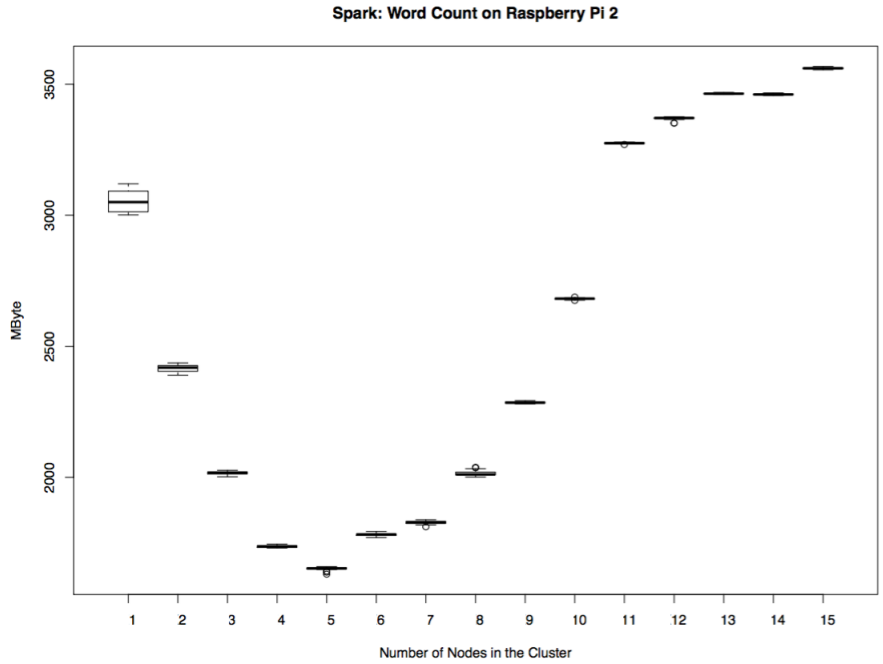

Figure 6.21: *Amount of Data Transferred in the Cluster for Spark-Word Count Benchmark on Raspberry Pi 2*

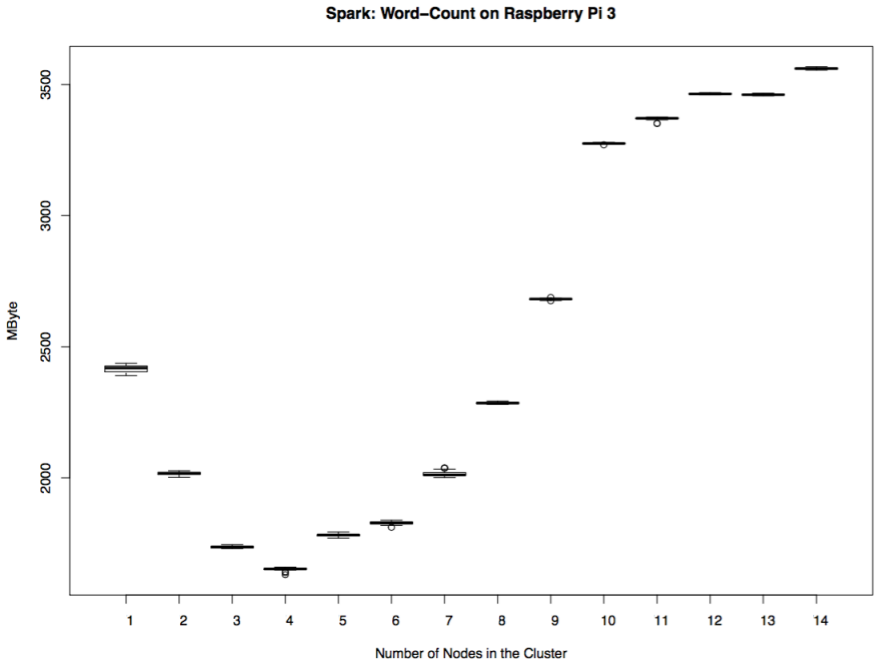

Figure 6.22: *Amount of Data Transferred in the Cluster for Spark-Word Count Benchmark on Raspberry Pi 3*

Chapter 7

Evaluation Method for Cloud Computing Simulators

7.1 Introduction

This chapter presents the results of applying a cross validation method to evaluate cloud computing simulators, namely CloudSim, GreenCloud and mininet. This chapter demonstrates the methodology as a comparative performance evaluation of real-world workloads on physical infrastructure against corresponding models of the workloads and infrastructure on the selected simulators. The level of accuracy for each tool is presented based on the results of a sensitivity analysis, which study the Root Mean Square Error (RMSE) for the relative and absolute error between the simulated and actual execution time and energy consumption.

In this chapter, we use the results of the earlier performance evaluation of the Glasgow Raspberry Pi Cloud from Chapter 5, to be a source of the inputs that we feed into the selected simulators as workload and platform characteristics for the data centre.

Novelty: This is the first empirical comparison of multiple cloud simulators against a scalable micro data centre system, to assess their relative consistency over a range of non-functional metrics.

7.1.1 Results for the Accuracy of CloudSim

Evaluation of MPI Application

CloudSim v 3.0.3 is designed to model the MPI application and the communication between tasks in the data centre as it contains the following Java classes:

1. `Taskstage` represents various stages a `networkCloudlet` can have during execution. Four stage types which are possible(EXECUTION, WAIT-SEND, WAIT-RECV and FINISH).

2. `AppCloudlet` class to represent an application which user submit for execution within data centre. It consist of several `networkClouds`.

3. `Workflowapp` is also a class of `AppCloudlet` having communicating tasks Calheiros et al. [46].

We measure the execution time for the NAS MPI Data Traffic benchmark (DT) Bailey et al. [34] , Class:A on the actual test-bed. We have chosen and specified the DT benchmark as the other benchmarks from NAS require a large footprint of memory than can be run on Raspberry Pi 2 and 3. We tested different sizes of high-performance computing applications on the cluster in order to gather execution times for each application on a particular node. We obtain the output of each application using mpiP mpi [15] to generate the profiling file for each benchmark running time on the cluster in order to be compared with CloudSim.

Figures 7.1 and 7.2 show the execution time on Raspberry Pi v2 and v3 and the simulated execution time calculated by CloudSim. The points on the lines represents the mean of the execution time of running the benchmark (DT) 30 times on Raspberry Pi nodes and the simulated execution time the model of the benchmark on CloudSim in tandem with the increasing number of nodes.

Evaluation of Web server Application

In order to run Apache Bench (AB) ABa [1] on our Raspberry Pi data centre, we use an external Ubuntu machine as a client for the web servers. We also use HAproxy Kaushal

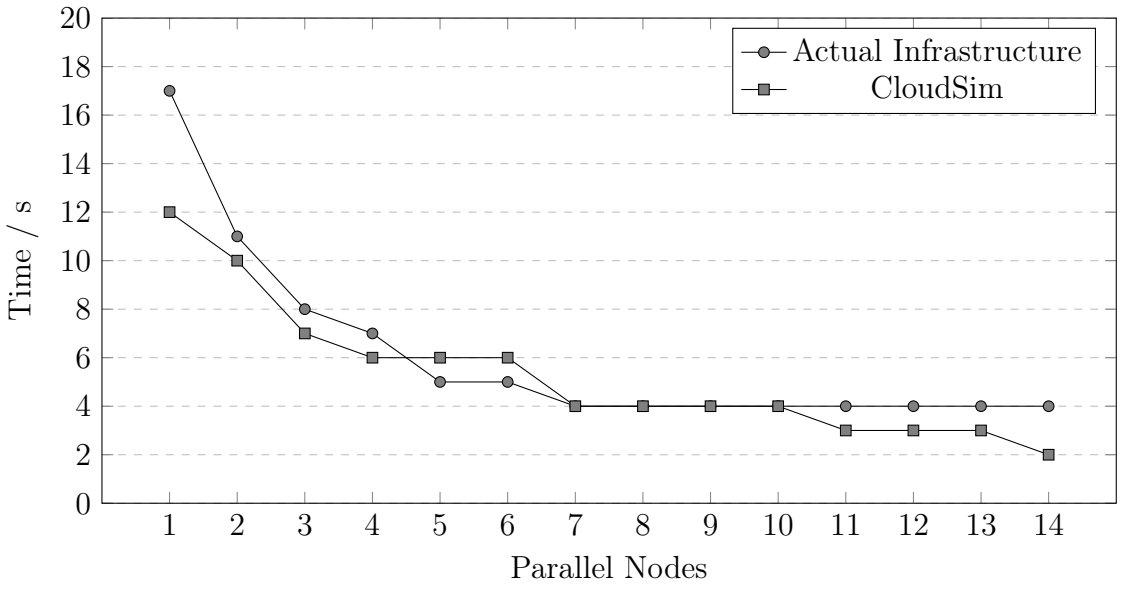

Figure 7.1: *Actual and Simulated Performance of DT Benchmark on Raspberry Pi 2 and CloudSim*

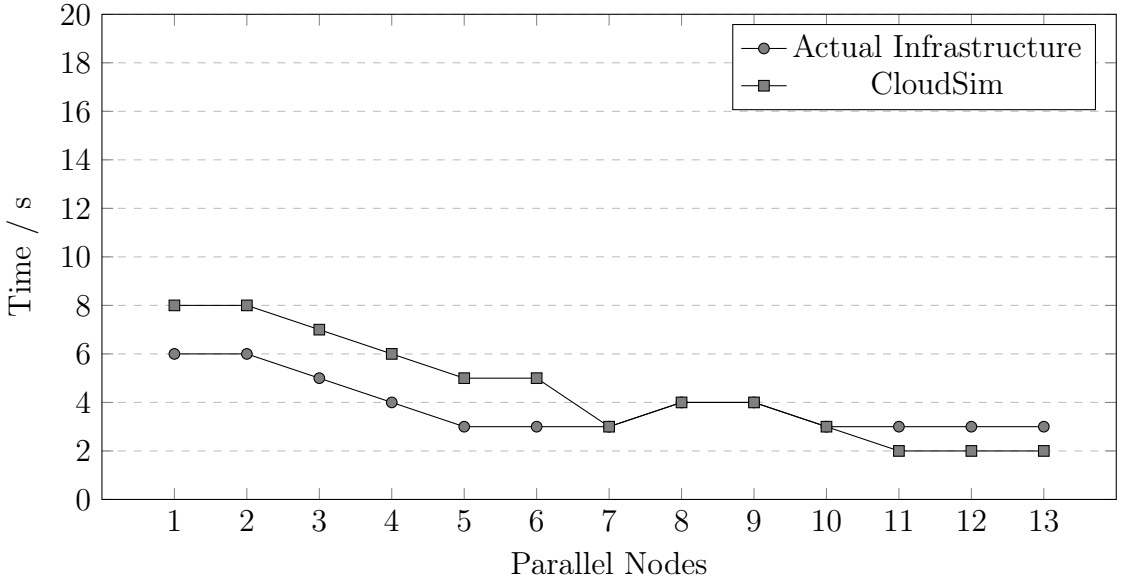

Figure 7.2: *Actual and Simulated Performance of DT Benchmark on Raspberry Pi 3 and CloudSim*

Table 7.1: Root mean square (RMS) for absolute and relative error of the simulated performance on CloudSim

DT Benchmark - MPI Workload

	Raspberry Pi2	Raspberry Pi3
RMS for Absolute Error	1.46	1.36
RMS for Relative Error	32.03 %	17.95 %

AB Benchmark - Web Server Workload

	Raspberry Pi2	Raspberry Pi3
RMS for Absolute Error	56.81	21.10
RMS for Relative Error	75.80 %	60.96 %

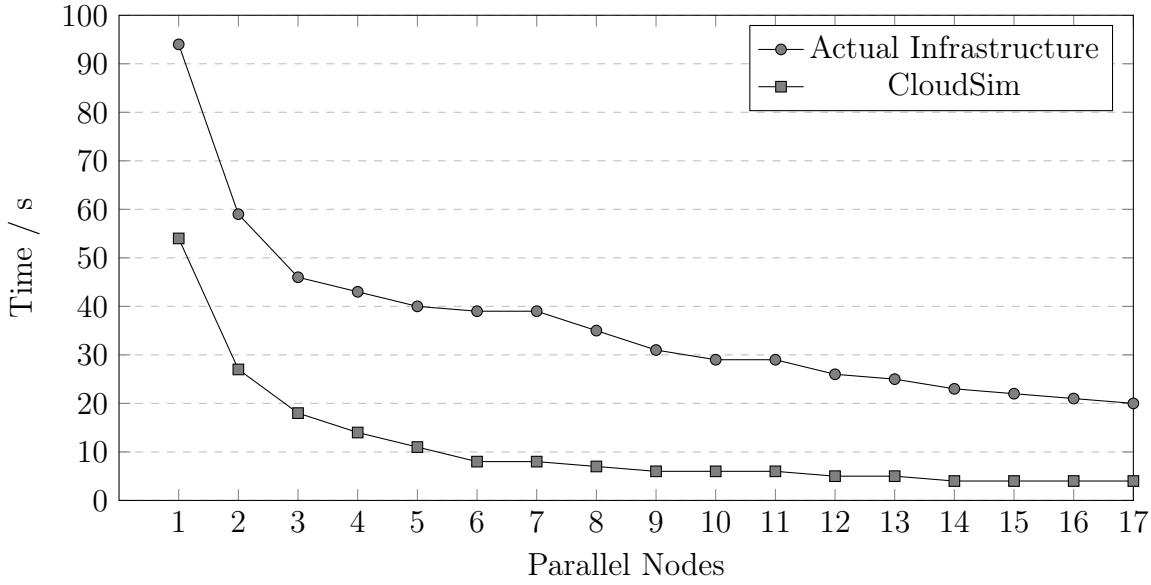

Figure 7.3: *Actual and Simulated Performance of AB Benchmark on Raspberry Pi 2 and CloudSim*

and Bala [90] in our data centre as a load balancer to distribute the requests between the Raspberry Pi nodes. Using HAproxy allowed us to expand the number of nodes serving the request from the client. Figures 7.3 and 7.4 show the performance of the benchmark AB on both actual tests, which were on Glasgow Raspberry Pi Cloud and the simulated environment of CloudSim.

Table 7.1 shows the errors for the simulated environment on CloudSim with respect to the execution time on Raspberry Pi.

Figures 7.1 – 7.4 show the execution time for the DT and AB benchmarks on both Raspberry Pi v2 and Raspberry Pi v3 with the simulated execution time on CloudSim for the model of the benchmark. The first point on each line represents the mean of the

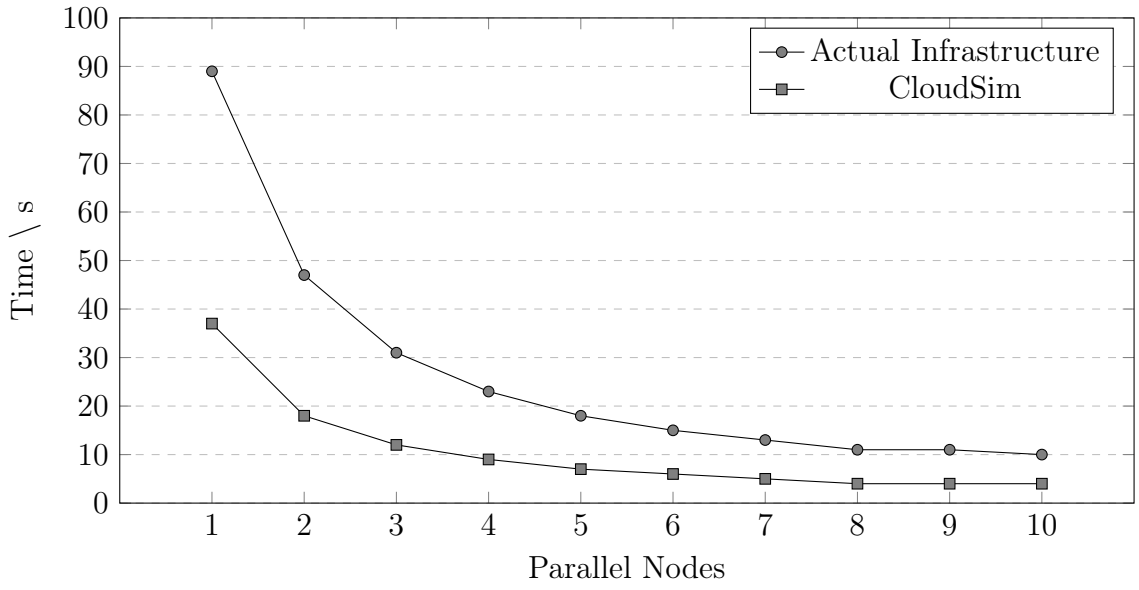

Figure 7.4: *Actual and Simulated Performance of AB Benchmark on Raspberry Pi3 and CloudSim*

execution time of running the benchmark 30 times on a single nodes, and the second point represents the mean of running the benchmark 30 times on two nodes, and so on for the rest of the points on the line in increasing the number of nodes. From the highest point in the lines represents the simulated execution time on CloudSim and the actual infrastructure for the model of the benchmark, there is a clear similarity between actual execution time and simulated execution time, except in the case of running the benchmark on a single node. Running the benchmark on a single node will include the time to manage the execution of the benchmark in terms of the buffer system and application buffer, which causes delays in the task communications Hu et al. [80].

7.1.2 Results for the performance of Mininet and Green Cloud

Our data sets for the actual and predicted performance of all tools followed the normal distribution. Considering this, we determine that using the technique of calculating the root-mean-square error (RMSE) 7.1 was a sufficient way to demonstrate the degree to which the tools used in this research were accurate Chai and Draxler [48]. To be more specific, we applied RMSE to compare the predicted performance of the selected tools against the actual performance of the Raspberry Pi cloud. The formula given in V.1

comparatively examines the difference between the values for the actual execution time in $x_{1,t}$ and the predicted performance in $x_{2,t}$. With respect to the ratio of energy consumption for servers in the data centre (i.e., those built on top of Raspberry Pi devices in Table 7.3), the RMSE results for GreenCloud showed a high relative error when considered in relation to the actual measurements from the Raspberry Pi cloud. For more information, see Figure 7.5. With respect to the performance of Mininet, we conducted an experiment to model the network performance. The rationale for doing so stemmed from the fact that Mininet focuses on communication rather than computation in experimental situations. Based on the collected data, we confirm that Mininet was not able to run the ab benchmark with 10,000 concurrent requests in the benchmark. Therefore, we focus on the latency between nodes in running the benchmark, and we sought to calculate the time via iperf on the actual Mininet experiments. The reader is given a representation of the differences between the two experiments (running the ab benchmark with 1,000 concurrent connections) in Figure 7.6. Given in Table 7.2 are the outcomes from the investigation of the selected tools. Information pertaining to their advantages, drawbacks, and their levels of accuracy when modelling the performance of the Raspberry Pi Cloud is presented.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(x_{1,t} - x_{2,t})^2}{n}}. \tag{7.1}$$

7.2 Calibration in CloudSim for Calculating Link Latency in Cloud Environment

In the Cloud IaaS (Infrastructure as a Service) environment, total execution time for each cloudlet can be calculated. From this, we can decide how many resources such as CPU, memory, network, bandwidth and storage are necessary in order to execute all of the tasks; we can also calculate costs, based on used resources. We implement a mechanism available to calculate latency (time) in the cloud environment. The purpose of calculating the delay is to discover the total execution time required including latency, propagation and execution time. Then, based on this integrated time, cost of resources and data centre

Table 7.2: Features and accuracy results of GreenCloud and Mininet

Evaluation Criteria	GreenCloud Kliazovich et al. [100]	Mininet Min [8]
Area	Energy Consumption	Network
Advantages	Modelling all communications processes on packet level	Running real code including standard Unix and Linux
Drawbacks	Does not consider different architecture for CPU and workloads characteristics.	Limited to the physical resource. Dose not support CPU computations
Actual Experiment	Running MPI job Data Traffic (DT) from NAS Benchmark Bailey et al. [34]. Measuring Real Power consumption by a monitoring tool	Running Apache Bench (AB) on Raspberry Pi Cloud. Using iperf tool to profile network performance.
Modelled Experiment	Simulating Raspberry Pi MIPS and File specifications on the simulator. Selecting Round-Robin algorithm for resource allocation	Developing python script involving AB benchmark ABa [1] and iperf commands line to be run on Mininet.
Accuracy	49.5 % RMSE Relative Error	27.05 % RMSE Relative Error

Table 7.3: Power Consumption for Raspberry Pi Devices Runing MPI Job

	Ideal	Full Utilization
Raspberri Pi 1	1.96 Watt	2.16 Watt
Raspberry Pi 2	1.03 Watt	2.70 Watt
Raspberry Pi 3	1.34 Watt	2.31 Watt

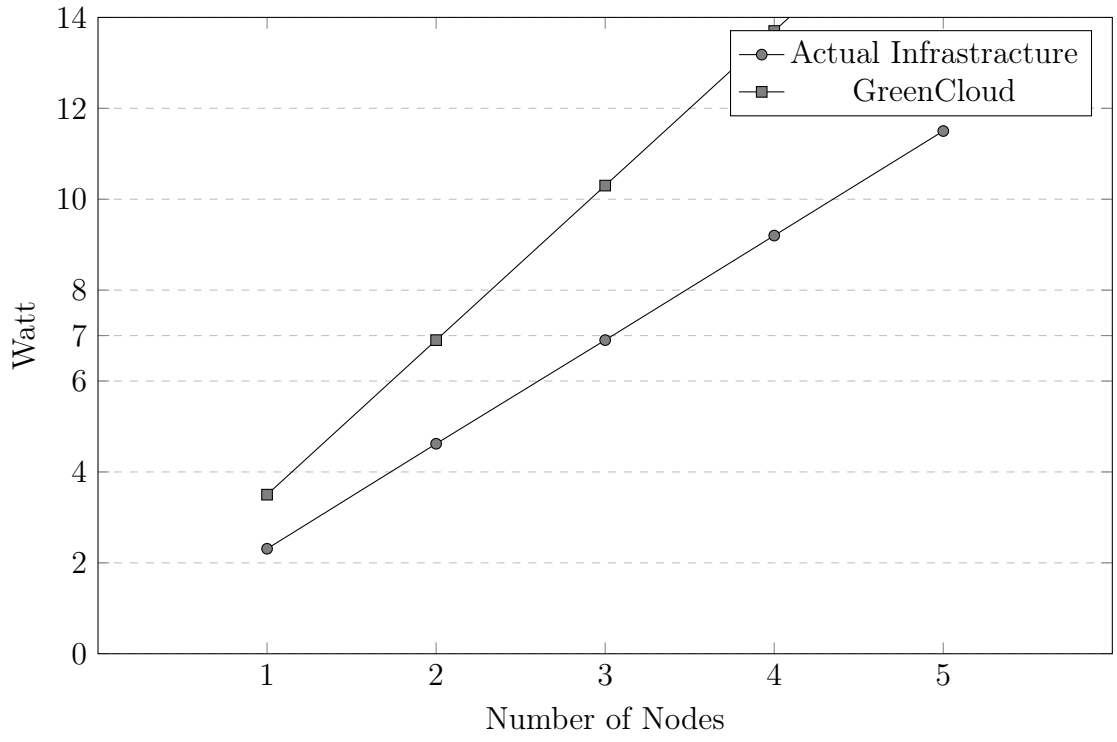

Figure 7.5: *Actual and Simulated Power Consumption (Raspberry Pi 3 Cluster vs GreenCloud)*

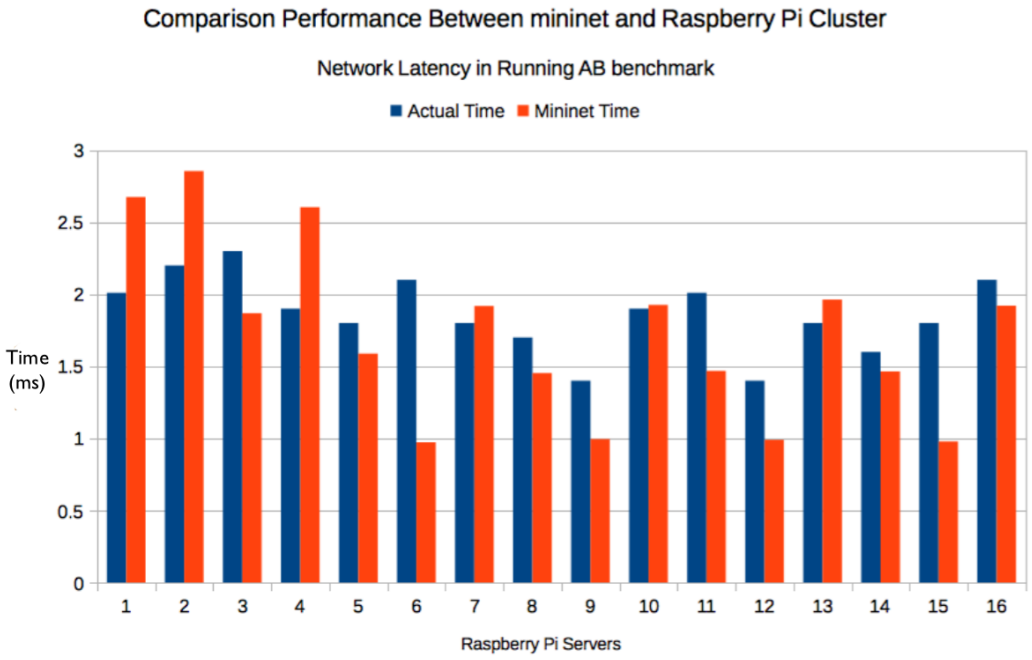

Figure 7.6: *Network Latency Between a Client and Servers (Raspberry Pi Cluster vs Mininet)*

performance can be calculated and measured. There are a number of reasons that make this approach useful in decision making, for example, if total resource costs are increasing, alternative choices should be made. Failing this, schedule changes can be made in an attempt to reduce costs. We successfully apply a new mechanism to CloudSim that allows users to model the network transmission and propagation of the data centre. Links in CloudSim do not consider the effects that the transmission and propagation have on the network, However, we add these parameters as we realised there is missing simulated time in CloudSim from the actual time in my experiment. This is a result of CloudSim only showing the CPU bound time and not the I/O bound time. Parameters to work on:

1. Transmission time.

2. Propagation time.

3. Execution time.

Proposed Mathematical Equation to calculate / Display Latency:

$$\boxed{\textit{Total time} = \textit{Transmission time} + \textit{Propagation time} + \textit{Execution time}}$$

7.2.1 Result of Applying Latency Calculation on Simulated Execution Time on CloudSim

Figure 7.7 shows the comparative result between actual execution time and simulated execution time before and after adding the latency mechanism for transmission delay in networking for Ab benchmark. The $actual_Ex$ data point represents the actual execution time taken to finish the jobs on Raspberry Pi cloud, whereas CloudSim refers to the simulated execution time on CloudSim framework. Additionally, DhCloudSim means the simulated execution time from the customised CloudSim with the latency mechanism, as a result, reliability in DhCloudSim is increased more than just relying on the previous execution time of the CloudSim framework.

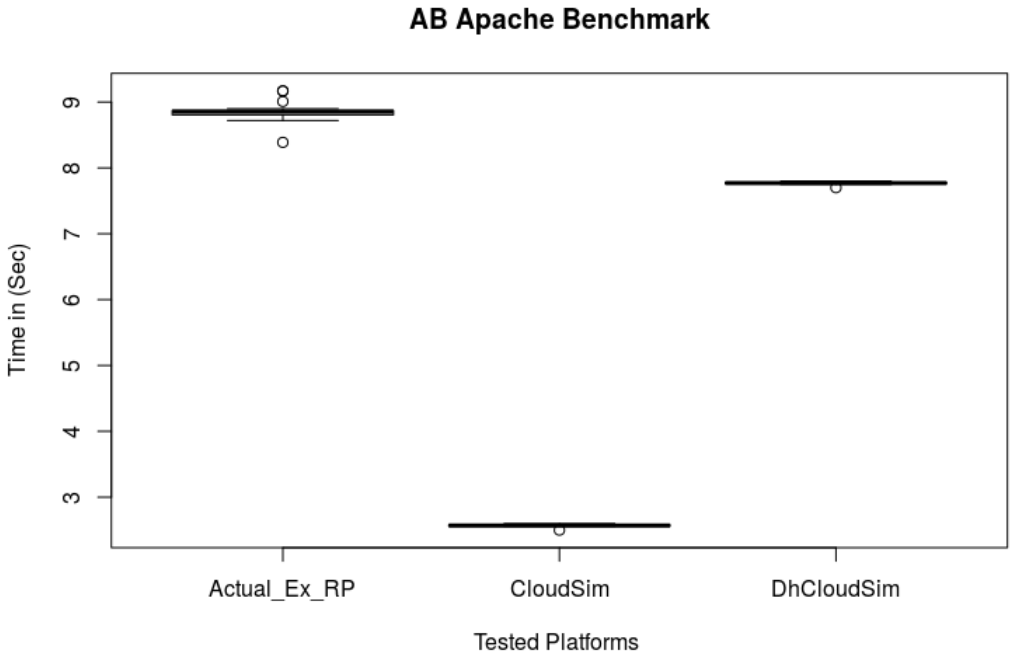

Figure 7.7: *Comparison Graph Between Actual Execution Time and Simulated Time on CloudSim for AB Benchmark*

7.3 Conclusion

This chapter has presented an empirical approach to evaluate the software simulation tools *CloudSim* Calheiros et al. [46], *GreenCloud* and *mininet* through a comparative study with a micro data centre known as the Glasgow Raspberry Pi Cloud Tso et al. [144]. Two workloads have been profiled on a per-node basis using standard Linux performance monitoring tools. I make the following four suggestions based on my experiences with these simulation experiments:

1. CloudSim needs a *richer set of input features* to calculate workload execution time. Currently, CloudSim requires number of instructions for the workload, and an instruction processing rate for the CPU. Our results show that there is clear execution time dependence on the CPU architecture, based on the discrepancies between CloudSim results for data centre nodes featuring Raspberry Pi v2 and Raspberry Pi v3.

2. CloudSim needs a *more complex model of inter-node communication* for distributed applications. Our experiments show high relative error between the actual and

simulated performance for a non-batch-processing workload, i.e. one which is likely to be a representative cloud application.

3. *GreenCloud* needs to be improved to predict the energy consumption for micro server instances, involving the energy consumption for different architectural components such as CPU, RAM, and buses. This stems from the fact that these components have a significant impact on the energy consumption of the servers in a way that is dependent on the nature of the architectures and the nature of the workload type.

4. *Mininet* was associated with a higher level of accuracy when modelling the network performance for Apache Bench when comparatively examined against *CloudSim*. There is 60% relative RMSE in CloudSim performance on Apache Bench, a typical transactional processing workload. However, an important consideration is that *Mininet* is constrained by the capacity of the physical machine that it is installed on, given that it is a full-system simulator based on virtual machine execution.

In conformance with the theoretical tenets of empiricism, the cross-validation method was applied to derive insights about the tools in question, and the collected results indicate that this was a reliable way in which to verify and validate the computer system simulators.

The immediate aim of this chapter has been to illuminate the degree to which a collection of modelling tools can be viewed as accurate. This aim was achieved by designing a methodology that applied sensitivity analysis for the results, specifically relating to the actual and simulated performance of the models.

In respect of this chapter's findings, the simulations are abstractions and can not represent full complexity of sufficient details of real infrastructure configurations or workloads. So it would be more appropriate for researchers to conduct their experiments using alternative tools like prediction models.

Moreover, the following chapter will assess how to predict the performance of cloud computing infrastructure using Machine Learning techniques based on the evaluation measurements of our Raspberry Pi infrastructure. This will lead me to investigate the level of

accuracy of the prediction models by comparing the predicted results with the measurements from the tools that have been evaluated in this chapter.

Chapter 8

Predictive Model for Cloud Computing Experiments

8.1 Introduction

This chapter presents a prediction model for infrastructure performance for the Glasgow Raspberry Pi cloud based on Artificial Neural Networks ANN and Linear Regression Model LM. These prediction models built based on the data set that we generate by testing different types of workloads on top the Glasgow Raspberry Pi cloud which is already presented in Chapter 5. The idea behind this chapter is to compare the level of accuracy of different machine learning prediction models against the level of accuracy of the simulation tools that we already presented in chapter 6. The comparison will be shown the following chapter which is the conclusion of my thesis. This chapter centre on modelling four workloads performance and testing different types of cross validation method namely: Randomly splitting, Leave One Out and the K-Fold Cross Validation methods on both ANN and LM models in order to get the more accurate model.

Novelty: Presenting the level of accuracy of ANN and LM models based on The Glasgow Raspberry Pi Infrastructure and shows how different methods behave differently based on the selected workloads for modelling Cloud Computing experiments.

Table 8.1: The Proposed data set for the predictive model

Infrastructure Type	Raspberry Pi v2 & Raspberry Pi v3 Cluster Size 1-15		
Workloads	Execution Time	Number of Instructions	Amount of Data
DT - MPI Job	Seconds	Instructions	MByte
AB - Web Server	Seconds	Instructions	MByte
Cassandra-Stress	Seconds	Instructions	MByte
Spark-Word Count	Seconds	Instructions	MByte

8.2 Overview of the Proposed Model

Prediction models are often used in various forms of research. They can be successfully employed within investigations into the management of resource allocation, the power consumption of a data centre, the performance of a database application, or even the service level of agreement work Wu et al. [154], Al-Janabi et al. [25].

This experiment attempts to conduct an investigation into how accurate prediction models are at achieving a predicted result that matches the actual result. Within this experiment, the measured performance of a micro data centre will be predicted using prediction models. Once results are obtained, these results will be employed, within this examination, to attempt to predict the performance of other workloads. it would be also used for different applications or may even contain a different size data centre. In addition, this examination has also applied the predictive model to predict the performance of different types of infrastructures. These infrastructures all operate with identical workloads.

Within the diagram 8.1 an overview is presented of the predictive model. This predictive model was created specially for employment within this examination. In addition, this predictive model was used within the examination to predict the performance of any type of infrastructure. Once a general performance level was obtained, this result was then compared and contrasted to the performance results which were previously obtained. Previous performance results involved the performance of two operating systems: the Raspberry Pi v3 and Raspberry Pi v2.

We used the machine learning models due to the nature of the data set we have generated from the performance of the Raspberry Pi data centre as quantitative numeric. The other non-linear model among the predictors created problems for other regression

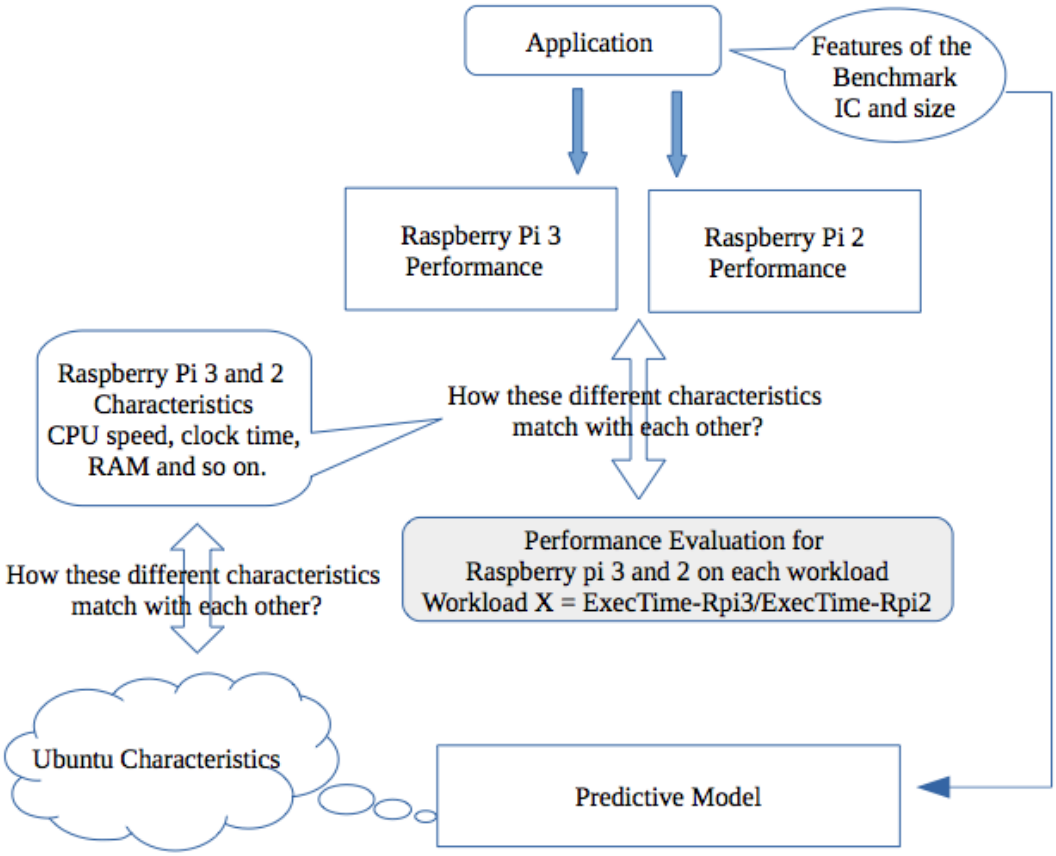

Figure 8.1: *The Prediction Model Overview*

models when attempting to predict computing time. Therefore, the Ann and LM are more flexible models for the unknown underlying function that dictate execution time.

8.3 Data Preparation

As part of this data collection for this prediction models, the following steps were conducted in order to prepare the data set:

- As previously discussed in Chapter 5, four different applications were profiled on two different CPUs.
- A number of specific instructions were executed and the elapsed time for each CPU was recorded.
- Amount of data transferred in the network was recorded.

- The received data was documented and recorded.
- For each separate core that was used, four distinct factors were recorded: CPU, CPU speed, RAM, and MIPS.

The data set we have generated from the work in Chapter 5 consists of 3600 observations, eight variables, seven predictors and one response which is time. The seven predictors are workload categorical (4 levels, 900 observations each), machine categorical (2 levels, 1800 observations each), nodes numeric (15 level). CPU (numeric dependent attribute of machine), MIPS (numeric dependent attribute of machine), instructions (numeric dependent attribute of workload) and amount of data (numeric dependent attribute of workload).

8.4 Prediction Model Based on Machine Learning Algorithms

In order to predict the performance of both of these operating systems, a Linear Regression model LM and Artificial Neural Netwroks Model are used. The linear regression model can be defined as a type of regression analysis in which observational information is obtained through a linear combination of the model parameters. Typically, these model parameters depend on at least one independent variable.

This model is selected by grid searching many different permutations of the model as Hyper-Parameters. In Grid searching, a script is created that varies different models parameters like structure, actavation function and convergence threshold. Then the model is run and tested. Error and execution time are used to determine the best model which provide low error and low execution time.

The Artificial Neural Networks and Linear Regression models are applied with splitting the data randomly into a test 20% of data and training 80% of data set and trained on the training set. Then the trained model is used to predict the test set. The prediction is shown in Figure 8.3 for predicted values vs. actual values. Figure 8.2 shows the selected designed structure of the ANN model showing the input, hidden and output layers of

the model In this ANN, we split machine, application and number of nodes into separate inputs and coded as -1 to 1, applying hyperbolic tangent activation function. Also we match the number of neurons in the single hidden layer with the number of input neurons. Model specifications were:

- Structure: 25-15-1
- Activation Function: Hyperbolic Tangent
- Convergence Threshold: 0.01

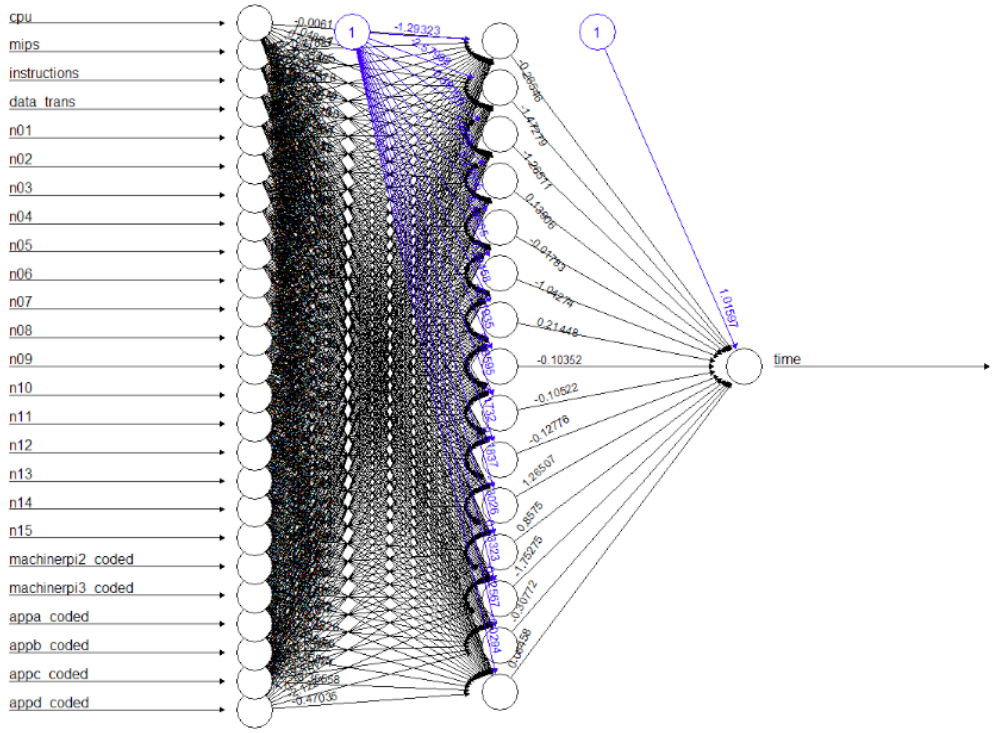

Figure 8.2: *The Diagram of the Artificial Neural Networks Model*

Figure 8.12 shows the predicted performance using Linear Regression Model in the case of testing 20% of the data based on randomly splitting the data into 20% testing data and 80% training data.

8.5 Leave One-Out Cross Validation

When randomly splitting the data into training and test sets the ANN Model performed with good level of accuracy. However, in the leave-one-out (removing each Workload x

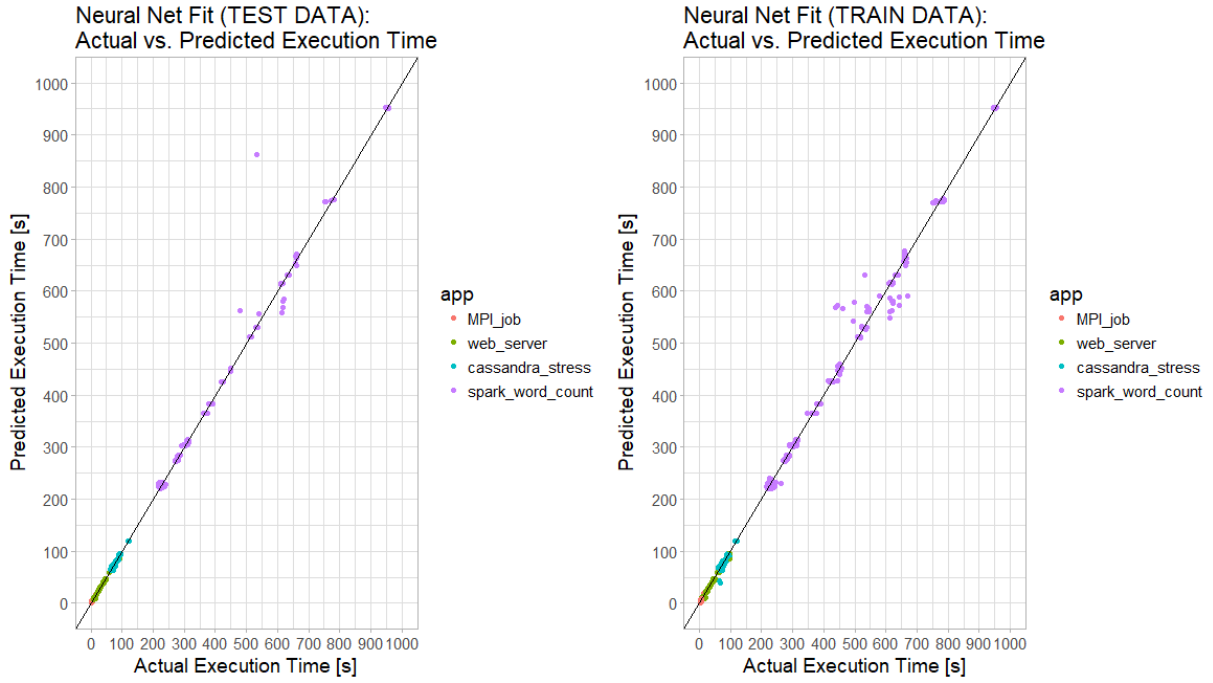

Figure 8.3: *Predicted Values vs. Actual Values - All Data*

MACHINE combination 1-by-1), test Linear Models performed better than The ANN Model. The following points are considered in my finding about the models:

- CPU and MIPS are not very helpful for the model. This because the variables CPU and MIPS have only two levels and they are perfectly correlated with machine this means that they are not 'independent'. Then, since there are only two unique combinations of CPU and MIPS this means that these combined are only basically as useful as the categorical labeling variables that we use for (RPI2 and RPI3).

- RAM and CORE are not useful in any statistical model because they are the same value for RPI2 and RPI3 and thus these values RAM = 1 and CORE = 4 are constants and cannot be included in the model. If we had more machines than 2, and these machines had more unique levels of RAM and CORE, then these variables would be very useful.

Figures 8.4 – 8.11 show the results of the leave one out method based on ANN prediction model, whereas Figures 8.13 – 8.20 show the results of the leave one out method for each workload based on linear regression model.

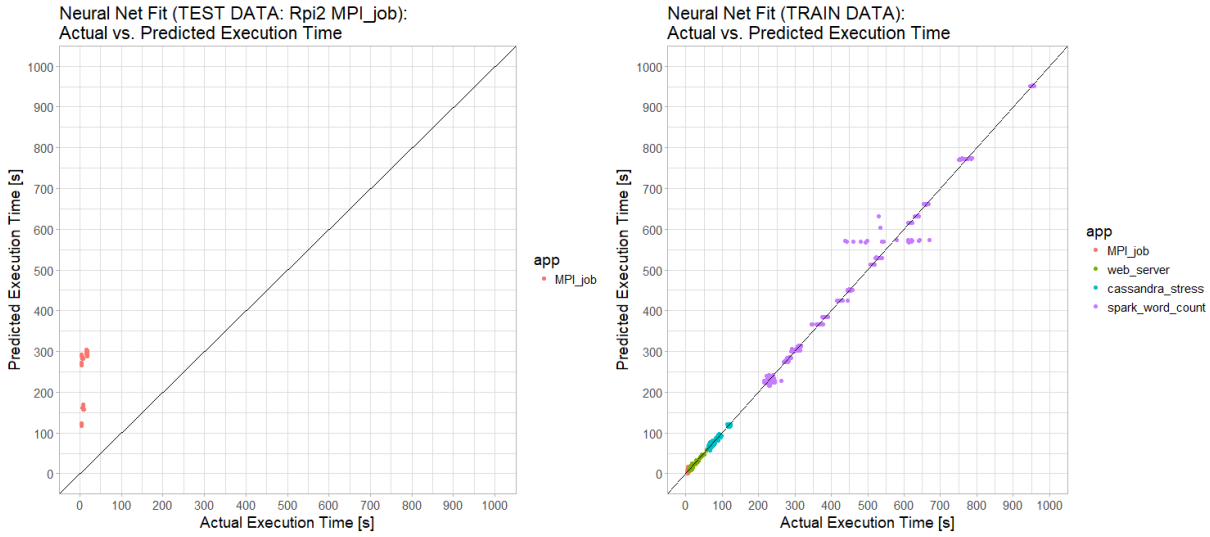

Figure 8.4: *Predicted Values vs. Actual Values for MPI on Raspberry Pi 2 Based on Other Workloads Performance*

8.6 K-Fold Cross-Validation

As shown in 8.5 that both statistical models LM and ANN struggle to predict an unknown machine x app combination in the leave-one-out scenario. However, it was also shown that for randomised splitting of the data into model testing and model training sets, the ANN outperforms the linear model and produces fairly precise predictions. Thus, the goal of this section of the analysis is to take a deeper dive into validation of the ANN model. This was achieved through a technique known as K-Fold cross-validation K-Fold CV. This is the process of randomly sub-setting your overall data set into k subsets. It is common to use k = 5 (5-fold CV) and k = 10 (10-fold CV) Witten et al. [153]. K-Fold CV mainly is used to increase confidence in how the model will generalise to an independent data set (or a new data set that the model has not seen yet). Basically, it is a robust diagnostic tool that helps set expectations for how well the model should work when applied to a new or independent data set. The results of 10-fold CV are listed in the Table 8.3. The data set was randomly split into 10 equally sized subsets (360 data points per subset). For each split the ANN model was trained and tested on each subset for example if the model is trained on subset (2,10), then it is tested on subset (1), similarly if the model is trained on subsets 1 U (3,10), then it is tested on subset 2. This process of training and test is run 10 times sequentially testing on subset 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10. Then this

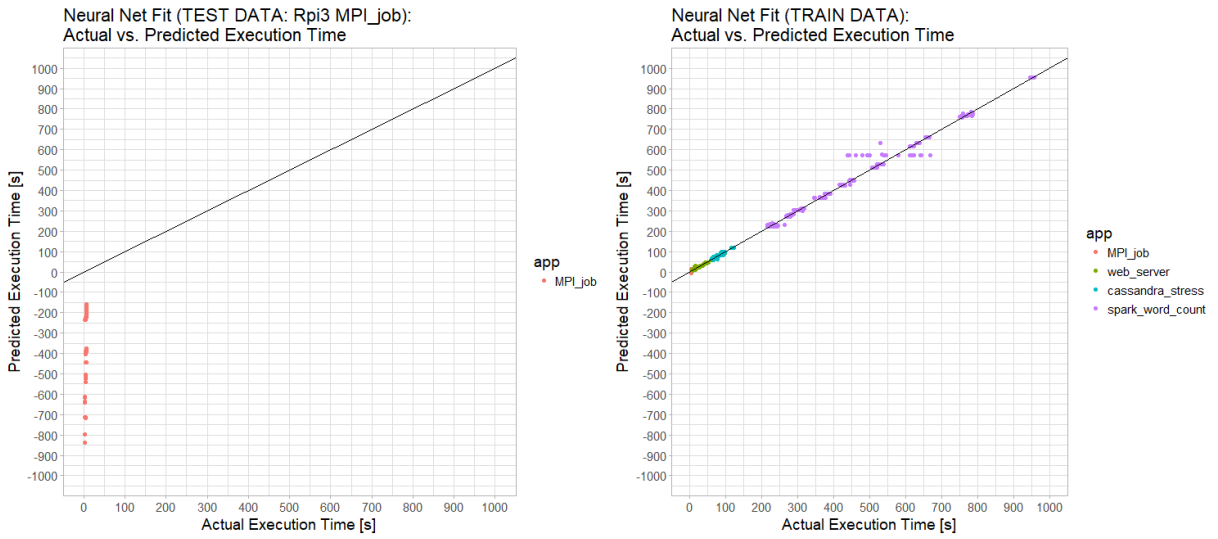

Figure 8.5: *Predicted Values vs. Actual Values for MPI on Raspberry Pi 3 Based on Other Workloads Performance*

process was repeated a total of 3 more times testing at model convergence thresholds of 0.5, 0.1, 0.05, 0.01. Decreasing the convergence threshold takes more computing time, but usually improves the prediction accuracy model until the point of over-fitting Rojas [129]. In over-fitting, the model has been overexposed to the training data, and is modeling its noise (undesirable) in addition to its underlying structure (desirable). With the K-Fold CV diagnostic technique, we can also see over-fitting when it is present Dangeti [54], it is usually evident when the MSE, RMSE, are very low for the training data predictions and very large for the test data predictions Perner [124].

From the 10-Fold CV results shown in Table 8.3, it is clear that a convergence threshold of 0.01 for the ANN model is optimal and that it yields a test mean percent error (MPE) of 10-13% with a standard deviation of 3-5%. This could likely be improved by adding more data to the model, but the current design with 3600 data points is an excellent starting point. From the 5-Fold CV study in Table 8.4, the test mean percent error (MPE) ranged 9-13% with a standard deviation of 2-5%. This is similar to the 10-Fold CV result.

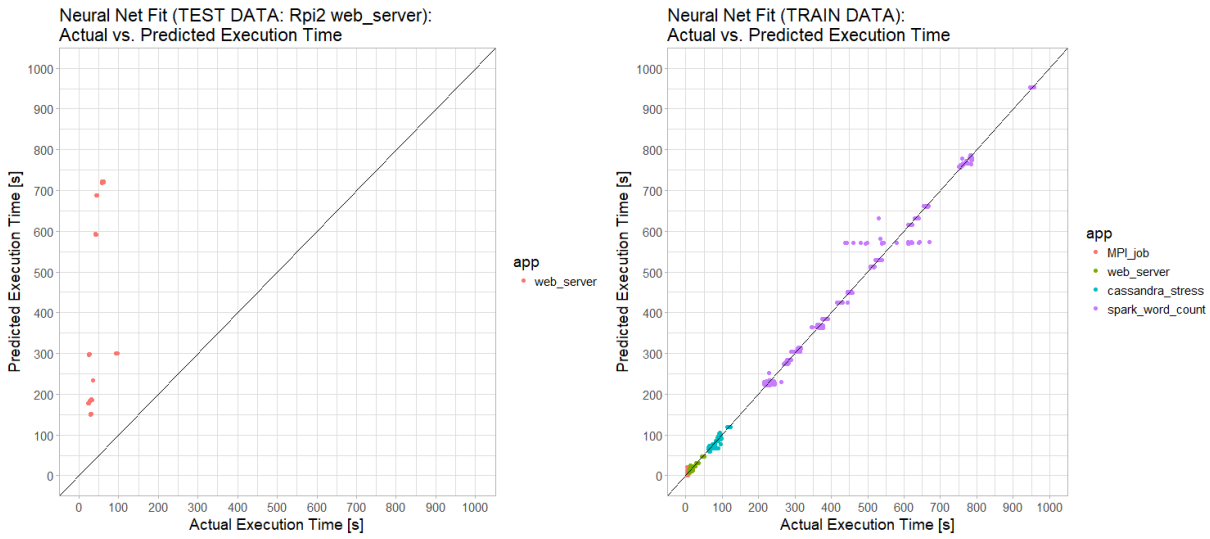

Figure 8.6: *Predicted Values vs. Actual Values for Web-Server on Raspberry Pi 2 Based on Other Workloads Performance*

8.7 Discussion:

Table 8.2 shows the results of the RMSE of all cross validations methods that we conducted to evaluate machine learning models to predict the execution time of different types of workloads on Raspberry Pi infrastructure. With the ANN model, the leave-one-out attempts are very poor predictions. ANN model needs to be built with more continuous variables in our data set, otherwise it will always struggle with the leave-one-out predictions. However, in this research, In the case of the leave-one-out trials, a Linear Model outperforms the model in ANN. ANN is better than LM for TEST data and RANDOM test/training split. LM is better than ANN for TEST data and LEAVE-ONE-OUT test/training splits. In these cases, we consider the RMSE for TEST data for both random and leave-one-out data split, we are not as interested in model performance on TRAIN data, as performance on TEST data represents the scenario of interest where researchers intend to predict an unknown (machine, workload) combination.

Overall, it seems that the ANN model is in this example and in principle a good way to predict cloud computing execution time. The most accurate statistical models arise from the designed experiments, where data is collected thoughtfully within a predetermine structure. These experiments were well designed and thus a good basis for creating an accurate and robust model. However, the ANN or any statistical model for that mat-

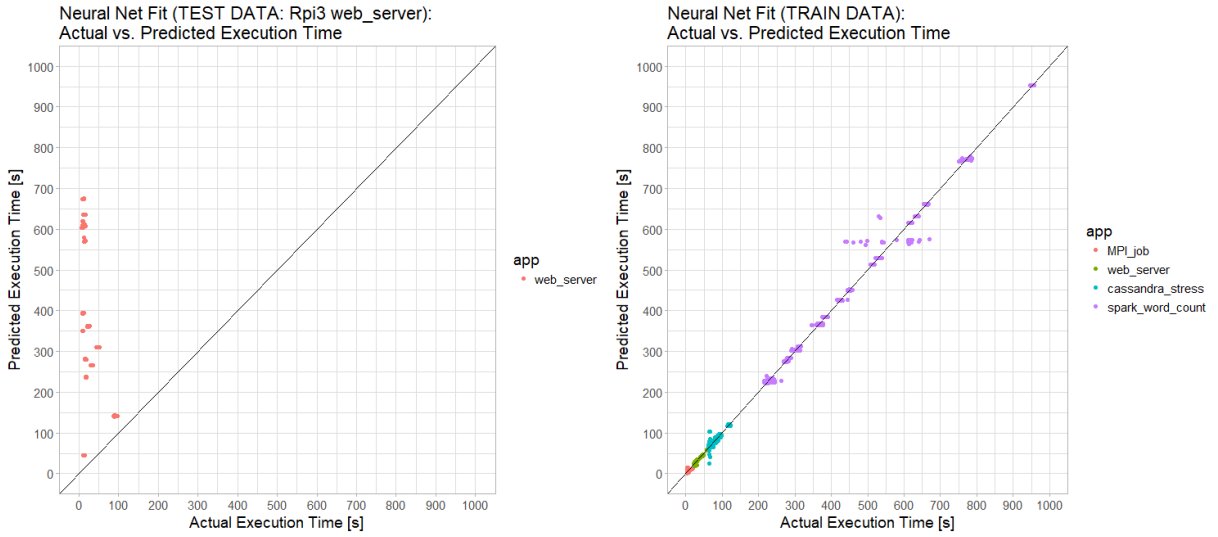

Figure 8.7: *Predicted Values vs. Actual Values for Web-Server on Raspberry Pi 3 Based on Other Workloads Performance*

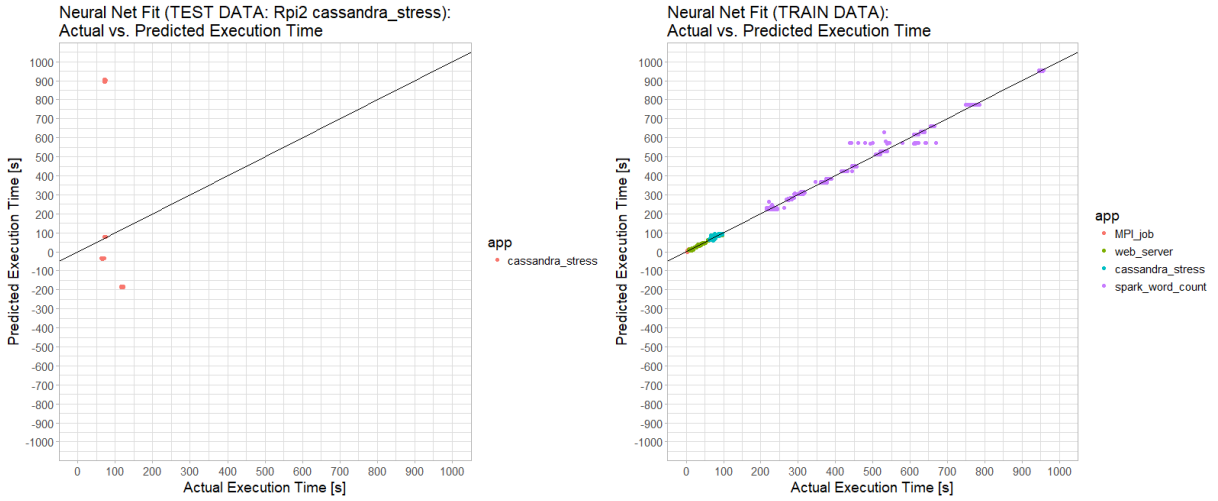

Figure 8.8: *Predicted Values vs. Actual Values for Cassandra-Stress on Raspberry Pi 2 Based on Other Workloads Performance*

ter, does struggle to predict an unknown test data set in the leave-one-out testing scenario (where a single factor level is completely removed to be used as the test data). In essence, this is because the design has a void where the 'unknown' to the trained model data was removed.

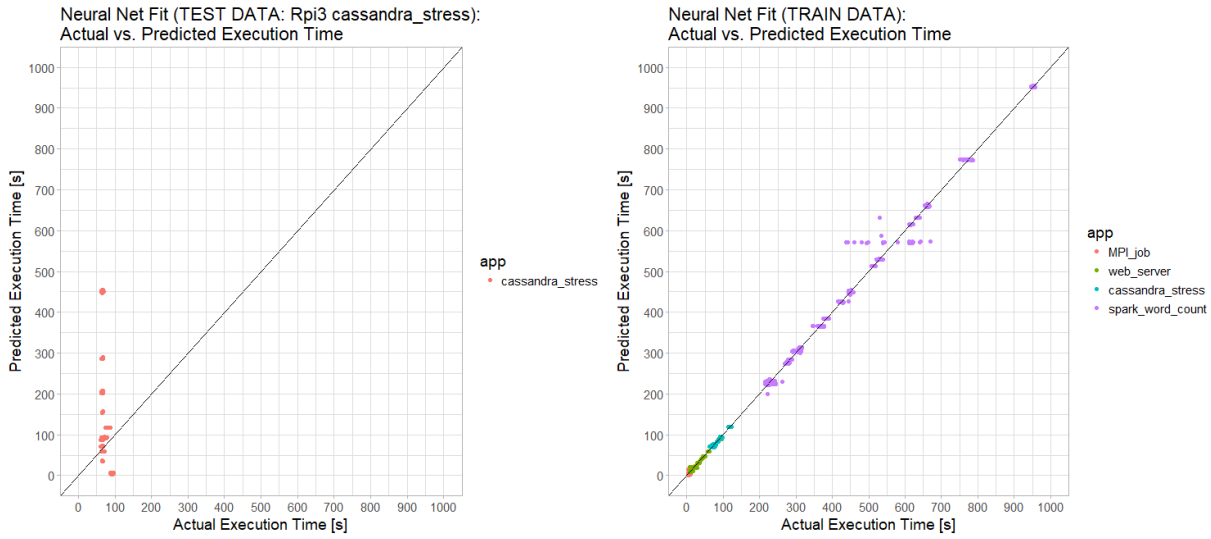

Figure 8.9: *Predicted Values vs. Actual Values for Cassandra-Stress on Raspberry Pi 3 Based on Other Workloads Performance*

Table 8.2: Results of the RMSE on ANN and LM

Split Description			Metric RMSE Test Data	
Split Type	Test Data	Train Data	ANN	LM
Random	Random 20%	Random 80%	**5.87**	10.37
Leave-One-Out	MPI-DT-Rpi2	All other applications	106.44	**38.52**
Leave-One-Out	MPI-DT-Rpi3	All other applications	232.46	**38.10**
Leave-One-Out	WebServer-AB-Rpi2	All other applications	273.7	**16.23**
Leave-One-Out	WebServer-AB-Rpi3	All other applications	1429.8	**17.18**
Leave-One-Out	Cassandra-Stress-Rpi2	All other applications	377.6	**38.35**
Leave-One-Out	Cassandra-Stress-Rpi3	All other applications	95.7	**45.87**
Leave-One-Out	Spark-WC-Rpi2	All other applications	665.4	**148.01**
Leave-One-Out	Spark-WC-Rpi3	All other applications	122.8	**154.81**

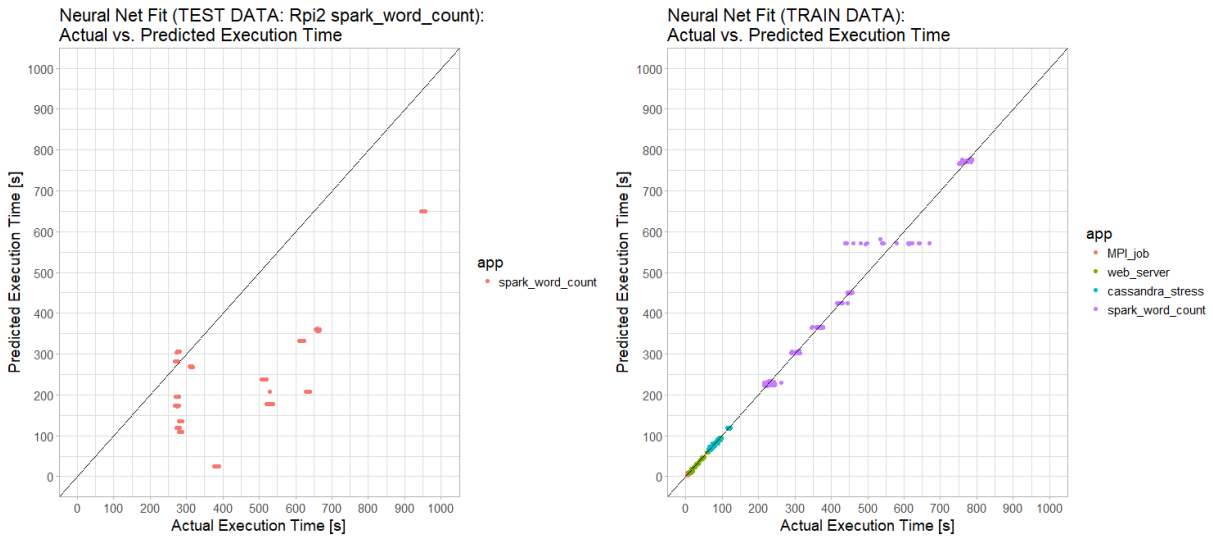

Figure 8.10: *Predicted Values vs. Actual Values for Spark Word-Count on Raspberry Pi 2 Based on Other Workloads Performance*

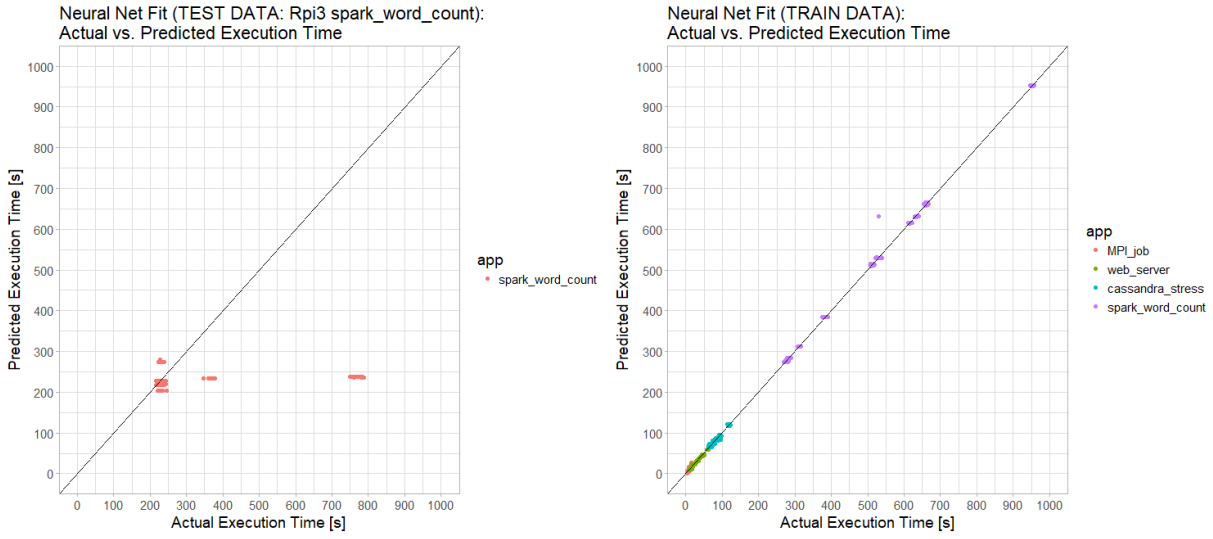

Figure 8.11: *Predicted Values vs. Actual Values for Spark Word-Count on Raspberry Pi 3 Based on Other Workloads Performance*

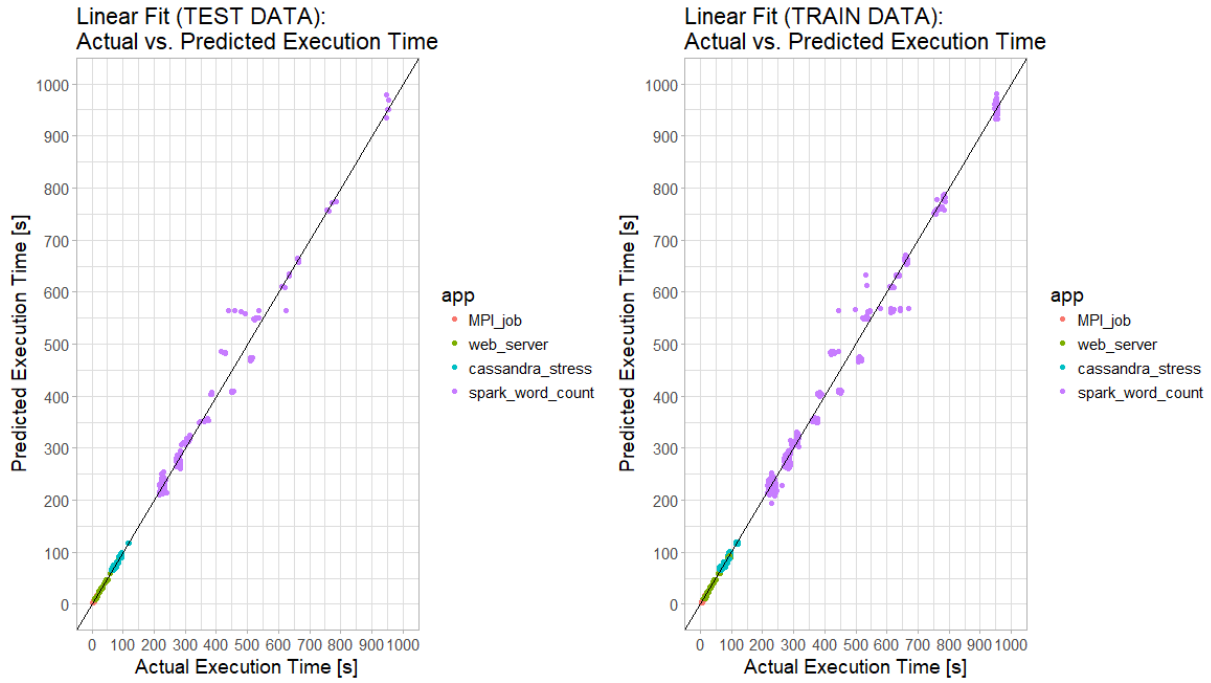

Figure 8.12: *Predicted Values vs. Actual Values for All Selected Workloads on Raspberry Pi 2 and 3 Based on Other Workloads Performance*

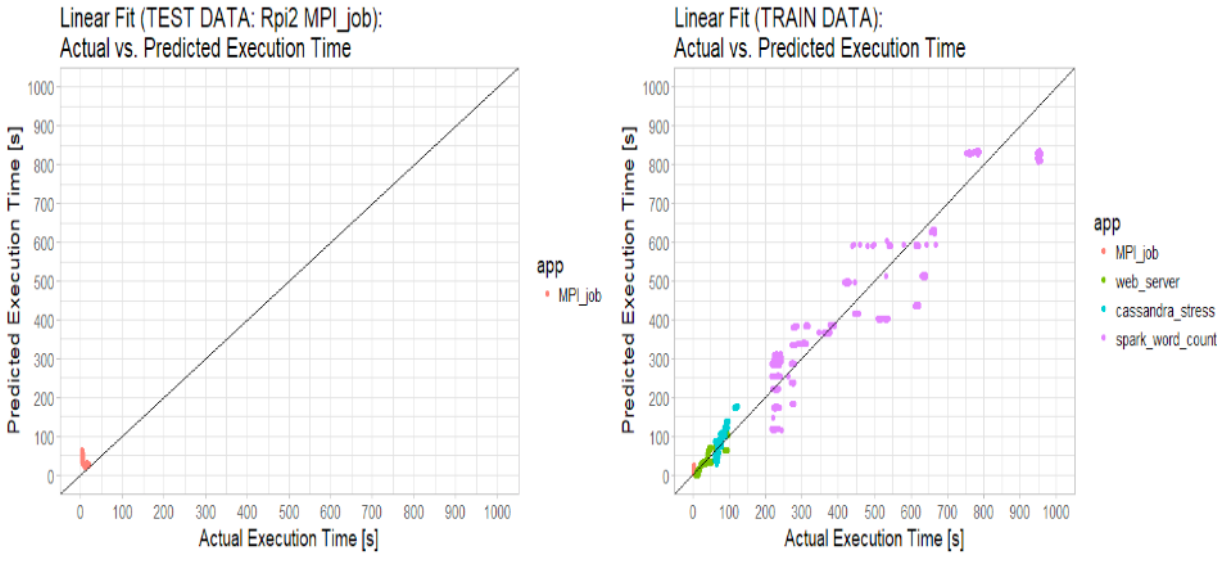

Figure 8.13: *Predicted Values vs. Actual Values for DT Benchmark on Raspberry Pi 2 Based on Other Workloads Performance*

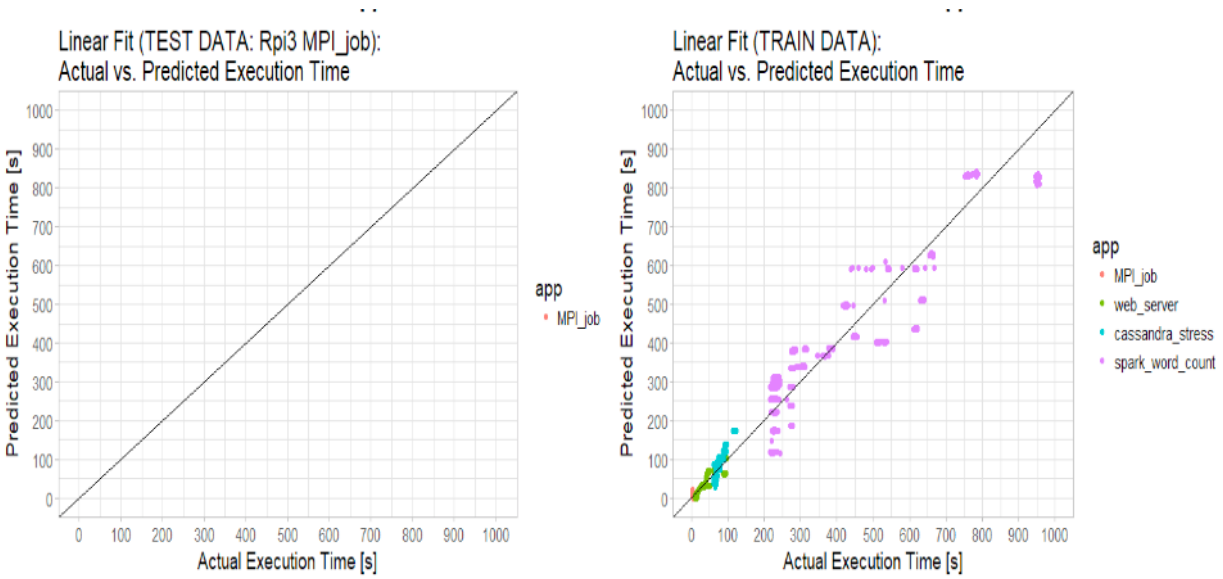

Figure 8.14: *Predicted Values vs. Actual Values for DT Benchmark on Raspberry Pi 3 Based on Other Workloads Performance*

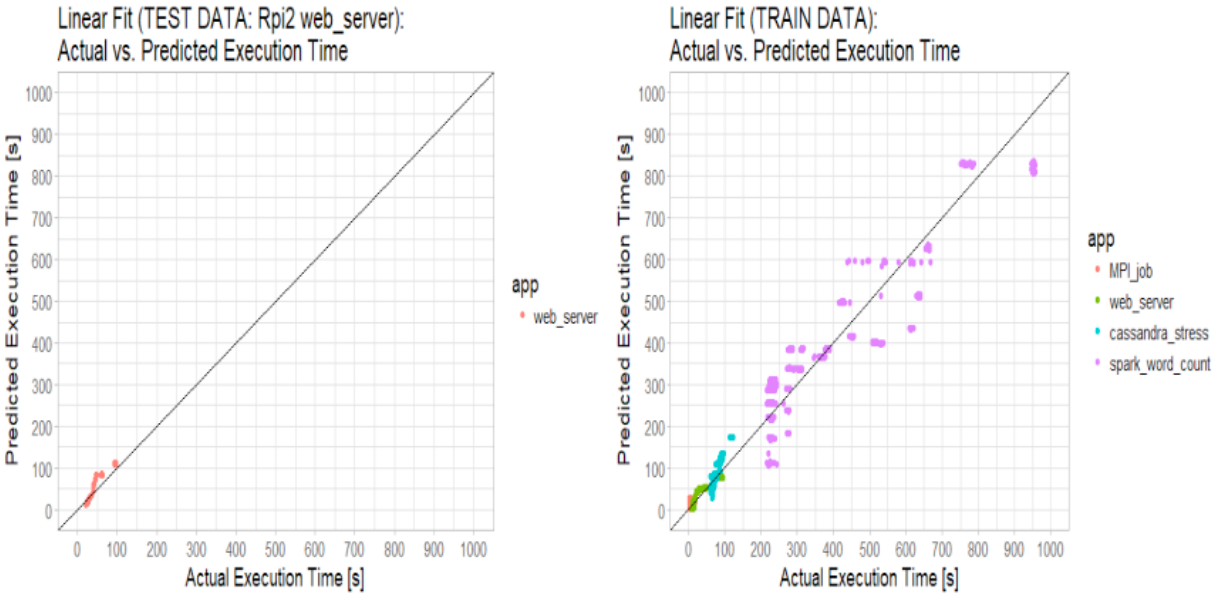

Figure 8.15: *Predicted Values vs. Actual Values for AB Benchmark on Raspberry Pi 2 Based on Other Workloads Performance*

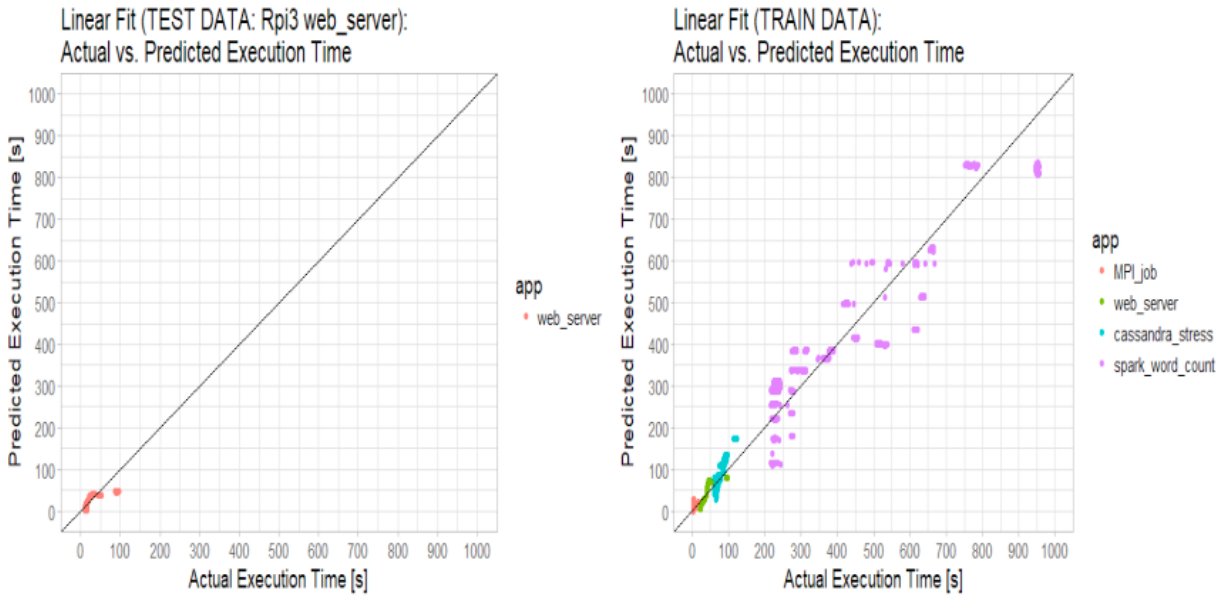

Figure 8.16: *Predicted Values vs. Actual Values for AB Benchmark on Raspberry Pi 3 Based on Other Workloads Performance*

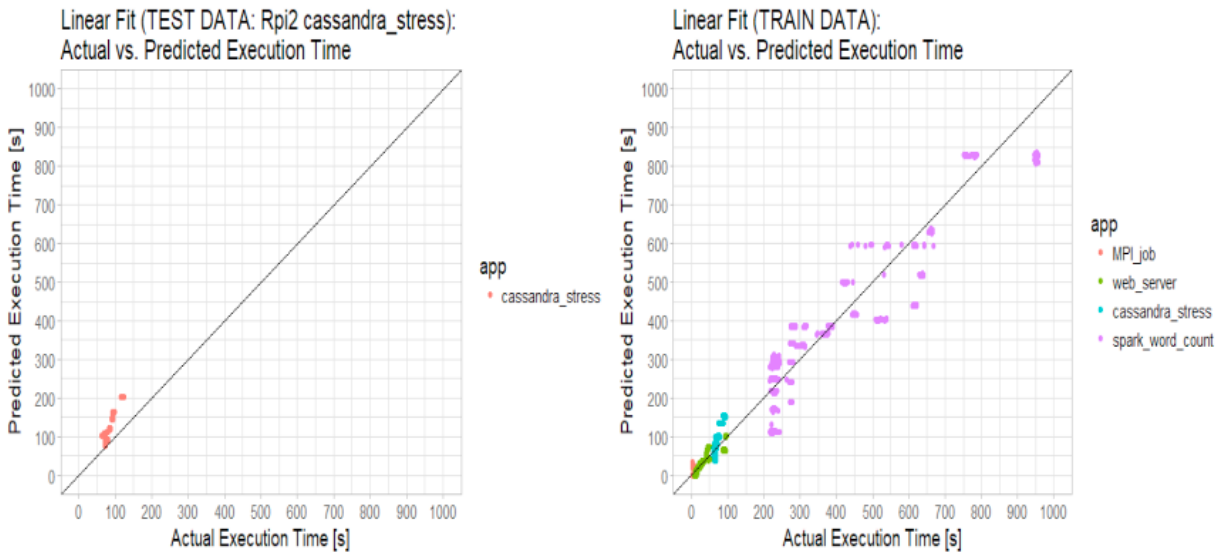

Figure 8.17: *Predicted Values vs. Actual Values for Cassandra-Stress on Raspberry Pi 2 Based on Other Workloads Performance*

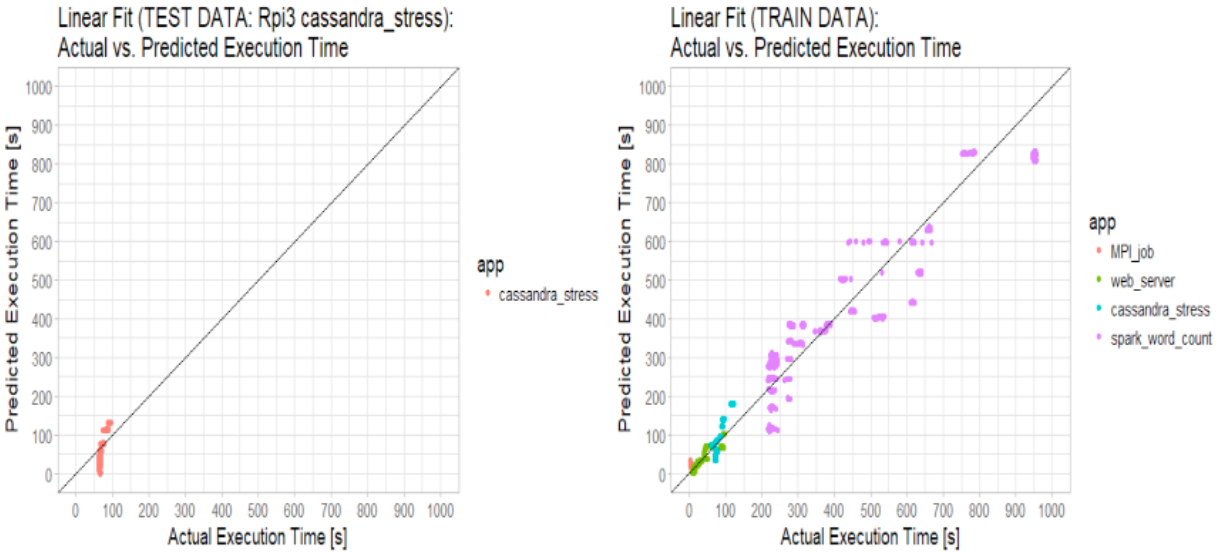

Figure 8.18: *Predicted Values vs. Actual Values for Cassandra-Stress on Raspberry Pi 3 Based on Other Workloads Performance*

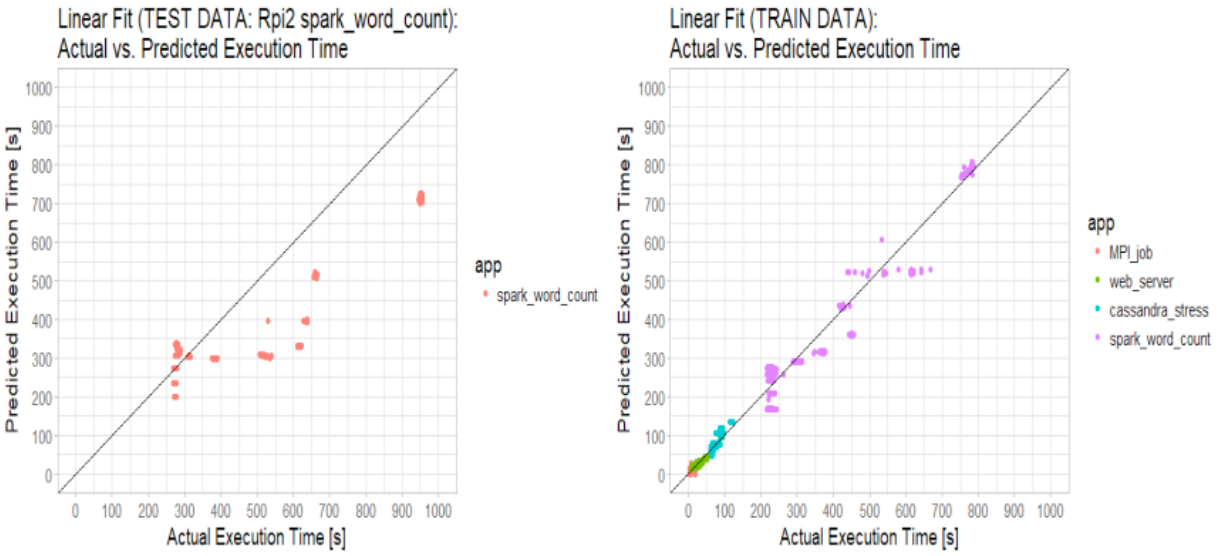

Figure 8.19: *Predicted Values vs. Actual Values for Spark Word-Count on Raspberry Pi 2 Based on Other Workloads Performance*

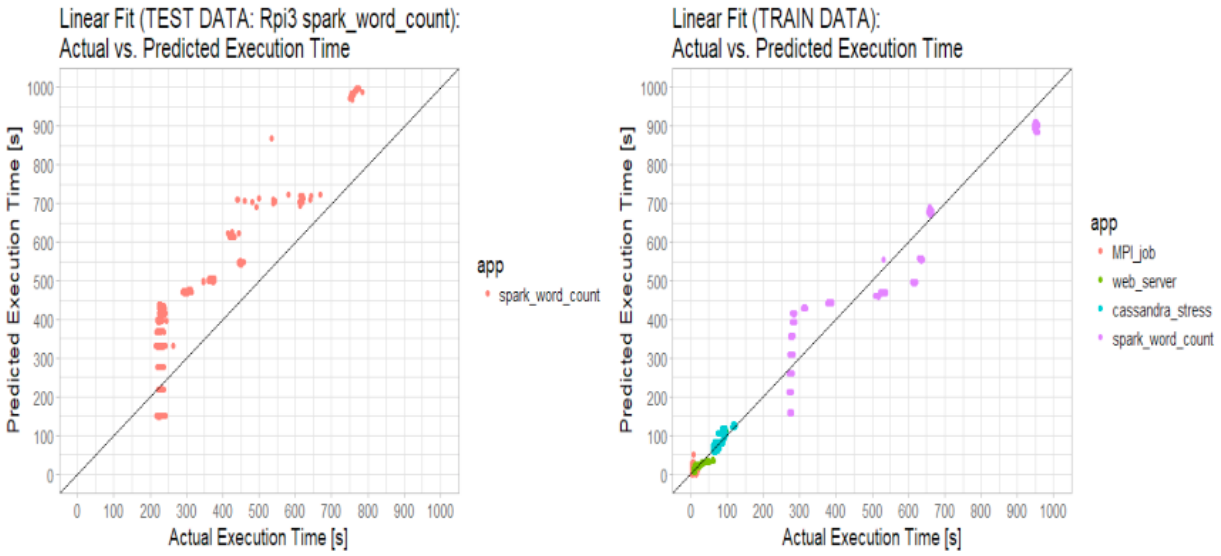

Figure 8.20: *Predicted Values vs. Actual Values for Spark Word-Count on Raspberry Pi 3 Based on Other Workloads Performance*

Table 8.3: Results of the 10 -Fold Cross Validation of ANN Model

10 FOLD CROSS VALIDATION	THRESHOLD=0.5		THRESHOLD=0.1		THRESHOLD=0.05		THRESHOLD=0.01	
CHUNK	TEST RMSE	TRAIN RMSE	TEST RMSE	TRAIN RMSE	TEST RMSE	TRAIN RMSE	TEST RMSE	TRAIN RMSE
1	16.5	17.0	6.9	7.8	7.1	8.5	6.6	6.9
2	12.7	13.0	13.2	11.1	7.0	8.9	4.7	6.8
3	15.1	15.1	7.5	9.6	6.0	7.8	5.0	7.2
4	17.6	11.9	15.0	8.4	11.9	7.0	13.0	5.9
5	18.7	18.2	9.8	7.5	10.2	7.9	9.0	6.4
6	19.4	19.3	9.8	10.6	4.9	7.3	4.6	7.2
7	11.5	10.9	7.5	7.7	6.6	7.1	6.5	7.0
8	26.0	28.2	7.0	7.7	8.2	8.2	8.1	7.4
9	15.0	13.3	10.1	7.3	9.8	7.1	10.2	7.3
10	9.9	11.6	6.6	8.0	6.6	8.3	5.4	7.2
MEAN	16.2	15.9	9.3	8.6	7.8	7.8	7.3	6.9
STDEV	4.6	5.2	2.9	1.4	2.2	0.7	2.8	0.5

Table 8.4: Results of the 5-Fold Cross Validation of ANN Model

5 FOLD CROSS VALIDATION	THRESHOLD=0.01		THRESHOLD=0.01		THRESHOLD=0.01		THRESHOLD=0.01		THRESHOLD=0.01	
CHUNK	TEST RMSE	TRAIN RMSE	TEST RMSE	TRAIN RMSE	TEST RMSE	TRAIN RMSE	TEST RMSE	TRAIN RMSE	TEST RMSE	TRAIN RMSE
1	8.6	6.3	6.0	6.9	4.3	7.6	8.5	6.7	8.1	6.5
2	7.7	7.1	9.0	7.7	5.4	6.8	4.1	6.9	7.1	7.1
3	6.4	7.5	10.8	6.4	9.5	6.2	4.4	7.2	6.2	6.8
4	4.5	7.0	6.6	7.0	9.9	7.3	5.7	7.0	6.1	7.1
5	7.2	6.7	6.5	6.7	4.4	6.9	6.7	7.2	6.6	7.3
MEAN	6.9	6.9	7.8	6.9	6.7	7.0	5.9	7.0	6.8	6.9
STDEV	1.6	0.4	2.1	0.5	2.8	0.6	1.8	0.2	0.8	0.3

For checking the accuracy of the prediction model, there are different approaches such as receiver operating characteristic ROC, precision/recall F measures, root mean square error RMSE and mean absolute percentage error MAPE. Both ROC and precision/recall F measures are used to characterise performance of binary classifiers and our models predict a continuous value (Execution Time). So they are not appropriate for our model. For the the continuous value the RMSE provided in the tables 8.2, 8.3 and 8.4.

8.8 Conclusion

Statistical models are often not particularly useful for interpolating and extrapolating e.g. Coles et al. [51]. If we completely leave out a factor level like the Benchmark DT or the machine Raspberry Pi v2, the model formed in its absence will struggle to predict its behavior. In order to predict the behavior of an unknown factor level or system, three approaches would be more effective in producing an accurate model:

- Create a statistical model, for example a machine learning model, where system elements, the independent variables, application, machine, nodes, or other factors are expressed as numbers. Run a designed experiment where as much as possible the dependent variables are varied and combined in different ways, and the output response execution time is recorded.

- Use varieties of these elements that expand the experimental design space, that is the dependent variables or system elements should include extremes, or represent the outermost boundaries that would be predicted within. This is because interpolation is frequently more accurate than extrapolation in predictive statistical models Coles et al. [51].

- Create a mechanistic or mathematical model relating the application, machine, nodes and other relevant elements of the overall cloud computing system to its execution time. In this type of model, the basic elements application, machine, nodes, and other would be expressed in terms of their attributes, specifically the attributes that effect execution time. These attributes should be expressed as continuous numerical

values. The structure of the model, or way the elements are brought together should reflect the underlying mechanisms in the cloud computing model.

As an answer for the RQ4 about which model for Cloud Computing is more accurate, machine learning prediction algorithms give better accuracy than simulation tools whach have been evaluated in chapter 6. They involve the performance of actual results with different workload behaviours. This is necessary for confirming the accuracy of the predictive model to be evaluated by the cross-validation method, K-fold cross-validation or the leave one out cross-validation method.

Chapter 9

Conclusion

The research findings, contributions to knowledge and areas of prospective inquiry are summarised below. The adoption of a Glasgow Raspberry Pi cloud dataset for testing various workload forms as a means of establishing an estimation model, as well as identifying the simulation techniques' degree of precision, are this study's principal contributions to knowledge. The major experimental processes were threefold. First, the ability to effectively run four rigorous workloads using Raspberry Pi Cloud was tested. Secondly, the Glasgow Raspberry Pi Cloud's system and workload characteristics were tested in terms of their absolute and relative errors, applying the developed model to contrast the simulated outcomes of the three simulation techniques in terms of precision. Thirdly, instead of depending on the identified simulation techniques, enhanced precision was sought through construction of a predictive model based on the initial experiment's dataset of results.

9.1 Research Results Summary

It will be beneficial in future for a particular relevant method to be adopted as a means of investigating various simulation techniques' degree of precision. Nevertheless, appropriate experimental techniques must be identified through a rigorous process, as well as the techniques offering a significant degree of precision, as a means of ensuring that the study results are legitimate. Consequently, this study was initiated with the extant cloud computing modelling techniques being evaluated. Given that the selection technique adopted

was particular to a single platform, namely CloudSim, this specific method of extension contrasted with the approach in other researches. In this regard, the shortcomings of two simulators or more could be averted and their strengths amalgamated for a single platform. Relevant here is that energy consumption apparatus is provided by certain extensions, while networking apparatus is offered by others, which is important for analysts to note when adopting CloudSim as Chapter 5 detailed. Ultimately, different simulation types have strengths that can be utilised to ensure that the platform's existing simulation techniques can be scrutinised and amended, given their identified shortcomings. Additionally, the degree of precision of relevant simulation techniques should be able to be accurately determined, through evaluating a cross-validation technique in relation to a specific test-bed.

Below, the appraisal of cloud computing modelling techniques is undertaken, based on the study results and contributions. The initial stage was to utilise Raspberry Pi apparatus as a foundation for an accurate information data system. In order to provide an overall overview of every workload available via the Glasgow Raspberry Pi Cloud, as well as to expand the dataset's magnitude, Ansible and Bash were used to formulate automated scripts and develop the volume of nodes. Subsequently, the power consumption of nodes, the volume of cluster information, size of instructions relating to the workload, as well as execution duration for the workload, were all tested as a means of determining the data system's efficacy.

Furthermore, numerous workloads were subjected to analysis. Given that micro data centres have not been subject to analysis determining the efficacy of using various systems for running separate workloads, it was significant for this research gap to be filled by assessing a micro data centre. Spark Word-Count, Cassandra-Stress, Apache Bench and Data Traffic were the four workloads that were run. The tying of workloads' numerical commands to the relevant execution durations was effective through the application of CloudSim, as evidenced by the micro data centre analysis results. Evidently, compared to using Raspberry Pi v2 to run the workloads, swifter implementation and a greater volume of commands can be dealt with using Raspberry Pi v3 nodes. Therefore, this study offers an assessment of the degree of precision of Mininet, GreenCloud and CloudSim as

current simulators, while through categorising their specific characteristics and dynamics, establishing guidelines for selecting the simulator with the greatest relevance and planning future experimental studies.

Additionally, an estimative cloud computing system model has been established through this study, drawing on LM and ANN as machine learning algorithms, with the extant simulation techniques being contrasted with this model's efficacy. Three methods were applied for determining the degree of precision of each tool, which enabled the verification of the estimative model. Initially, 80% training and 20% testing data was arbitrarily distinguished within the Glasgow Raspberry Pi Cloud dataset relating to efficacy. Subsequently, the test-bed and test data were contrasted for their degree of precision, with the latter data subjected to a cross-validation technique. The 'leave one out' cross-validation technique was a further precision method applied, which considers different workloads' respective efficacies in order to identify a specific workload performance to delete from the databank. The third technique is the K-fold cross-validation method; five groups are established comprising a fifth of the overall dataset respectively. Following this, the arithmetic mean is calculated for all groups, enabling the estimative model's precision to be determined.

- ***Chapter 5***:

  Informative guidance may be taken from this study for further inquiry into the set of existing CloudSim extensions, by applying the results pertaining to them from this research. By producing a single simulator through the amalgamation of numerous extensions, the simulation techniques may be enhanced in future studies, particularly drawing on the four sets of multiple extensions with matched characteristics identified here.

- ***Chapter 6***

  The creation of a databank that may be applied for authenticating extant simulation techniques, which also offers training data relating to machine-learning models and the characteristics of energy consumption, data volume transmitted across the network, execution duration and number of commands, was developed through the running of myriad workloads, which enabled the Raspberry Pi Cloud's efficacy to

be appraised. Furthermore, Cloud Computing associated micro servers have no relevant benchmark suites offered through SPEC SPE [9] or related companies and their traditional benchmarking apparatus. Consequently, a contribution has been offered through this investigation of an installation and implementation harness system comprising of artificial workloads and benchmark applications, allowing clustered network configurations to profile Raspberry Pi apparatus.

- ***Chapter 7***

  Building on Chapter Six, the Raspberry Pi Cloud's actual measurements were tested against the extant simulators for their degree of precision. The test-bed findings were contrasted with the simulation techniques' outcomes, which was made feasible through system feature and workload modelling. As a means of confirming the degree of precision according to various experimental forms and scenarios, numerous workloads had to be incorporated into every simulator prior to analysis. This means that the assessment of the various kinds of workload underpinned the determination of each simulation's degree of precision.

- ***Chapter 8***

  This chapter utilised a Raspberry Pi-based micro data centre's efficacy databank, as a means of formulating an estimative model. Numerous authentication techniques were adopted to test it, namely K-Fold cross-validation, 'leave one out' cross-validation and cross-validation. An original contribution was developed in this regard, as it was discovered that instead of depending on a simulator for assessing a sole system form's specific workload, various kinds of Raspberry Pi and specific workloads' efficacy could be estimated with greater precision.

Figure 9.1 presents the contributions on this research in a flow diagram to show the relationships between all outcomes.

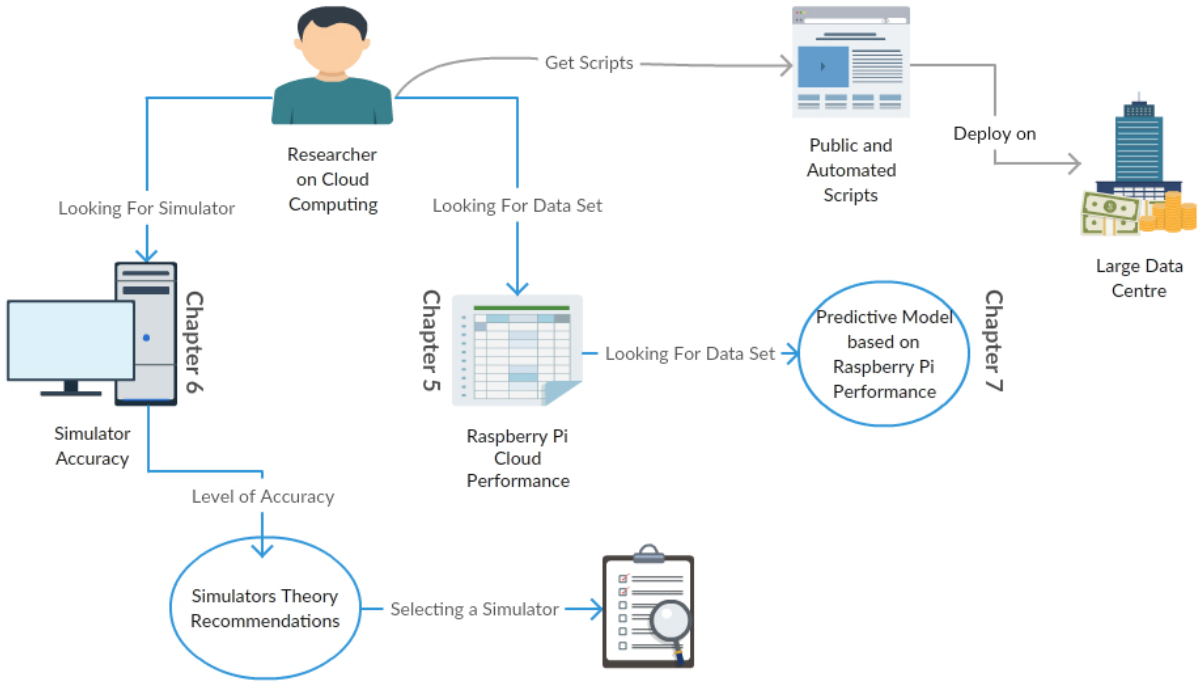

Figure 9.1: *The Outcomes of the Dissertation*

9.2 Shortcomings of the Research

Infrastructure and workload are the two principal areas of shortcomings relating to this study. Infrastructure: The Raspberry Pi Cloud's present system was the sole infrastructure that I could rely on. Although a virtual machine and the University of Glasgow's local network devices were tested for running the workloads, the expenditure entailed with using business infrastructure services for VM, as well as acceptability, both posed problems. Workload: Within the time allowed for this research, actual workloads and different benchmarks could not be tested, given that the infrastructure available was also restricted. Resultantly, as Chapter 6 outlined, evaluation was the emphasis in this study. Furthermore, Yahoo Cloud Serving Benchmark (YCSB) and others were run for Own-Cloud and MongoDB, however, due to their significant memory footprint, such benchmarks were challenging to profile via Raspberry Pi apparatus.

9.3 Prospective Research

Given that the estimation of machine learning techniques' efficacy has been proven as facilitated through this method, increasing the amount of assessed workloads and the cluster size would be beneficial in further studies. Meanwhile, various infrastructure forms and diverse workloads could be collated to establish a bigger databank for testing. Such a big databank would allow various simulators to be tested in relation to various infrastructure forms and the effect of distinct workload categories. Ultimately, a greater volume of incorporated data within a data set allows a cloud computing experimental model to be developed that is characterised by greater precision, thus permitting more exact estimations. Additionally, across the Computer Science discipline, various dynamics and their relevant simulation techniques' results could be estimated through further studies utilising the cross-validation method.

Appendix A

An Appendix

- The scripts and data set for running an profiling the selected workloads in my research are stored in the repository on *https://github.com/Dhahi374/Dataset.git*.

- The Modelling of the Glasgow Raspberry Pi Cloud on CloudSim is stored in: *https://Dhahi374@bitbucket.org/Dhahi374/grponcloudsim*.

Bibliography

[1] ab - Apache HTTP server benchmarking tool . `https://httpd.apache.org/docs/2.4/programs/ab.html`. Accessed: 2016-05-22.

[2] The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. `http://www.cloudbus.org/cloudsim/`, . Accessed: 2016-04-10.

[3] cloudsim-3.0.3. `https://github.com/Cloudslab/cloudsim/releases`, . Accessed: 2016-04-12.

[4] Dhrystone Benchmark Results On PCs. `http://www.roylongbottom.org.uk/dhrystone%20results.htm`, . Accessed: 2015-05-08.

[5] Dhrystone and MIPs performance of ARM processors kernel description. `http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/4160.html`, . Accessed: 2015-05-08.

[6] Welcome to Apache™ Hadoop®! `https://hadoop.apache.org/`. Accessed: 2015-05-08.

[7] What is Infinispan ? `http://infinispan.org/`. Accessed: 2016-04-01.

[8] Mininet Python API Reference Manual. `http://mininet.org/api/annotated.html`. Accessed: 2016-04-10.

[9] Standard Performance Evaluation Corporation . `https://www.spec.org/`. Accessed: 2016-03-18.

[10] AWS. `https://https://aws.amazon.com/`. Accessed: 2018-03-20.

[11] Microsoft Azure - Your vision. Your cloud. `https://azure.microsoft.com/en-us/`. Accessed: 2018-03-20.

[12] Google Cloud. `https://cloud.google.com/`. Accessed: 2018-03-20.

[13] Getting Started with Hazelcast. `http://hazelcast.org/getting-started/`. Accessed: 2016-04-01.

[14] iPerf - The network bandwidth measurement tool . `http://iperf.fr/`. Accessed: 2016-05-12.

[15] mpiP: Lightweight, Scalable MPI Profiling. `http://mpip.sourceforge.net/`. Accessed: 2016-05-22.

[16] PERF tutorial: Finding execution hot spots. `http://sandsoftwaresound.net/perf/perf-tutorial-hot-spots/`. Accessed: 2015-04-12.

[17] Logging into a Rasberry Pi using Public/Private Keys. `https://steve.dynedge.co.uk/2012/05/30/logging-into-a-rasberry-pi-using-publicprivate-keys/`. Accessed: 2016-05-22.

[18] S. Abar, P. Lemarinier, G. K. Theodoropoulos, and G. M. P. OHare. Automated dynamic resource provisioning and monitoring in virtualized large-scale datacenter. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 961–970, May 2014. doi: 10.1109/AINA.2014.117.

[19] Pekka Abrahamsson, Sven Helmer, Nattakarn Phaphoom, Lorenzo Nicolodi, Nick Preda, Lorenzo Miori, Matteo Angriman, Juha Rikkila, Xiaofeng Wang, Karim Hamily, et al. Affordable and energy-efficient cloud computing clusters: the bolzano raspberry pi cloud cluster experiment. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 2, pages 170–175. IEEE, 2013.

[20] Veronika Abramova and Jorge Bernardino. Nosql databases: Mongodb vs cassandra. In *Proceedings of the International C* Conference on Computer Science and Software Engineering*, pages 14–22. ACM, 2013.

[21] A. Ahmed and A.S. Sabyasachi. Cloud computing simulators: A detailed survey and future direction. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 866–872, Feb 2014. doi: 10.1109/IAdCC.2014.6779436.

[22] Sanjay P Ahuja and Karthika Muthiah. Advances in green cloud computing. In *Green Computing Strategies for Competitive Advantage and Business Sustainability*, pages 1–16. IGI Global, 2018.

[23] Samuel A Ajila and Akindele A Bankole. Cloud client prediction models using machine learning techniques. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 134–142. IEEE, 2013.

[24] Abdel-Rahman F. Al-Ghuwairi. *Formal Modeling and Dynamic Verification of Service Level Agreements in Cloud Computing*. PhD thesis, New Mexico State University, Las Cruces, NM, USA, 2013. AAI3574545.

[25] Samaher Al-Janabi, Ahmed Patel, Hayder Fatlawi, Kenan Kalajdzic, and Ibrahim Al Shourbaji. Empirical rapid and accurate prediction model for data mining tasks in cloud computing environments. In *Technology, Communication and Knowledge (ICTCK), 2014 International Congress on*, pages 1–8. IEEE, 2014.

[26] Yasmeen Alomair, Iftikhar Ahmad, and Abdullah Alghamdi. A review of evaluation methods and techniques for simulation packages. *Procedia Computer Science*, 62: 249–256, 2015.

[27] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010. ISBN 026201243X, 9780262012430.

[28] Dhahi Alshammari, Jeremy Singer, and Timothy Storer. Does cloudsim accurately model micro datacenters? In *Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on*, pages 705–709. IEEE, 2017.

[29] Dhahi Alshammari, Jeremy Singer, and Timothy Storer. Performance evaluation of cloud computing simulation tools. In *2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 522–526. IEEE, 2018.

[30] D. C. Alves, B. G. Batista, D. M. L. Filho, M. L. Peixoto, S. Reiff-Marganiec, and B. T. Kuehne. Cm cloud simulator: A cost model simulator module for cloudsim. In *2016 IEEE World Congress on Services (SERVICES)*, pages 99–102, June 2016. doi: 10.1109/SERVICES.2016.20.

[31] Alexandru-Florian Antonescu and Torsten Braun. Modeling and simulation of concurrent workload processing in cloud-distributed enterprise information systems. In *Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing*, DCC '14, pages 11–16, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2992-7. doi: 10.1145/2627566.2627575. URL `http://doi.acm.org/10.1145/2627566.2627575`.

[32] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010. ISSN 0001-0782. doi: 10.1145/1721654.1721672. URL `http://doi.acm.org/10.1145/1721654.1721672`.

[33] Saleh Atiewi and Salman Yussof. Comparison between cloud sim and green cloud in measuring energy consumption in a cloud environment. In *Advanced Computer Science Applications and Technologies (ACSAT), 2014 3rd International Conference on*, pages 9–14. IEEE, 2014.

[34] David H Bailey, Eric Barszcz, John T Barton, David S Browning, Robert L Carter, Leonardo Dagum, Rod A Fatoohi, Paul O Frederickson, Thomas A Lasinski, Rob S Schreiber, et al. The nas parallel benchmarks. *International Journal of High Performance Computing Applications*, 5(3):63–73, 1991.

[35] Rabindra K. Barik, Harishchandra Dubey, Chinmaya Misra, Debanjan Borthakur, Nicholas Constant, Sapana Ashok Sasane, Rakesh K. Lenka, Bhabani

Shankar Prasad Mishra, Himansu Das, and Kunal Mankodiya. *Fog Assisted Cloud Computing in Era of Big Data and Internet-of-Things: Systems, Architectures, and Applications*, pages 367–394. Springer International Publishing, Cham, 2018. ISBN 978-3-319-73676-1. doi: 10.1007/978-3-319-73676-1_14. URL `https://doi.org/10.1007/978-3-319-73676-1_14`.

[36] Adam Barker, Blesson Varghese, Jonathan Stuart Ward, and Ian Sommerville. Academic cloud computing research: Five pitfalls and five opportunities. In *Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'14, pages 2–2, Berkeley, CA, USA, 2014. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=2696535.2696537`.

[37] Bernhard Beckert, Reiner Hähnle, and Peter H Schmitt. *Verification of object-oriented software: The KeY approach*. Springer-Verlag, 2007.

[38] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr. Comput. : Pract. Exper.*, 24 (13):1397–1420, September 2012. ISSN 1532-0626. doi: 10.1002/cpe.1867. URL `http://dx.doi.org/10.1002/cpe.1867`.

[39] A.E.H. Bohra and V. Chaudhary. Vmeter: Power modelling for virtualized clouds. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8, April 2010. doi: 10.1109/IPDPSW.2010.5470907.

[40] André B. Bondi. Characteristics of scalability and their impact on performance. In *Proceedings of the 2Nd International Workshop on Software and Performance*, WOSP '00, pages 195–203, New York, NY, USA, 2000. ACM. ISBN 1-58113-195-X. doi: 10.1145/350391.350432. URL `http://doi.acm.org/10.1145/350391.350432`.

[41] Per Nikolaj D Bukh. The art of computer systems performance analysis, techniques for experimental design, measurement, simulation and modeling, 1992.

[42] Marc Bux and Ulf Leser. Dynamiccloudsim: Simulating heterogeneity in computational clouds. In *Proceedings of the 2Nd ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, SWEET '13, pages 1:1–1:12, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2349-9. doi: 10.1145/2499896.2499897. URL `http://doi.acm.org/10.1145/2499896.2499897`.

[43] R. Buyya, R. Ranjan, and R. N. Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *High Performance Computing Simulation, 2009. HPCS '09. International Conference on*, pages 1–11, June 2009. doi: 10.1109/HPCSIM.2009.5192685.

[44] Rajkumar Buyya and Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and computation: practice and experience*, 14(13-15):1175–1220, 2002.

[45] James Byrne, Sergej Svorobej, Konstantinos M Giannoutakis, Dimitrios Tzovaras, PJ Byrne, Per-Olov Östberg, Anna Gourinovitch, and Theo Lynn. A review of cloud computing simulation platforms and related environments. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, pages 679–691. SCITEPRESS-Science and Technology Publications, Lda Portugal, 2017.

[46] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41(1):23–50, January 2011. ISSN 0038-0644. doi: 10.1002/spe.995. URL `http://dx.doi.org/10.1002/spe.995`.

[47] Rodrigo N. Calheiros, Marco Aurélio Stelmar Netto, César A. F. De Rose, and Rajkumar Buyya. Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications. *Softw., Pract. Exper.*, 43(5):595–612, 2013. URL `http://dblp.uni-trier.de/db/journals/spe/spe43.html#CalheirosNRB13`.

[48] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3):1247–1250, 2014.

[49] Weiwei Chen and E. Deelman. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pages 1–8, Oct 2012. doi: 10.1109/eScience.2012.6404430.

[50] Nicolas Christin, Jorg Liebeherr, et al. The qosbox: A pc-router for quantitative service differentiation in ip networks. *University of Virginia*, 2001.

[51] Stuart Coles, Joanna Bawa, Lesley Trenner, and Pat Dorazio. *An introduction to statistical modeling of extreme values*, volume 208. Springer, 2001.

[52] Simon J Cox, James T Cox, Richard P Boardman, Steven J Johnston, Mark Scott, and Neil S O?brien. Iridis-pi: a low-cost, compact demonstration cluster. *Cluster Computing*, 17(2):349–358, 2014.

[53] Simon J Cox, James T Cox, Richard P Boardman, Steven J Johnston, Mark Scott, and Neil S OâĂŹbrien. Iridis-pi: a low-cost, compact demonstration cluster. *Cluster Computing*, 17(2):349–358, 2014.

[54] Pratap Dangeti. *Statistics for Machine Learning*. Packt Publishing Ltd, 2017.

[55] Filipe Fontinele de Almeida, Areolino de Almeida Neto, and Mario Meireles Teixeira. Resource scheduling in web servers in cloud computing using multiple artificial neural networks. In *Artificial Intelligence (MICAI), 2015 Fourteenth Mexican International Conference on*, pages 188–193. IEEE, 2015.

[56] Rogério Leão Santos De Oliveira, Ailton Akira Shinoda, Christiane Marie Schweitzer, and Ligia Rodrigues Prete. Using mininet for emulation and prototyping software-defined networks. In *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*, pages 1–6. IEEE, 2014.

[57] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[58] Luca Deri and Stefano Suin. Effective traffic measurement using ntop. *IEEE Communications Magazine*, 38(5):138–143, 2000.

[59] Luca Deri, Rocco Carbone, and Stefano Suin. Monitoring networks using ntop. In *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*, pages 199–212. IEEE, 2001.

[60] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: issues and challenges. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 27–33. Ieee, 2010.

[61] Kevin Doucet and Jian Zhang. Learning cluster computing by creating a raspberry pi cluster. In *Proceedings of the SouthEast Conference*, pages 191–194. ACM, 2017.

[62] Werner Dubitzky, Krzysztof Kurowski, and Bernard Schott. *Large-scale computing techniques for complex system simulations*, volume 80. John Wiley & Sons, 2012.

[63] Yehia Elkhatib, Barry Porter, Heverson B Ribeiro, Mohamed Faten Zhani, Junaid Qadir, and Etienne Rivière. On using micro-clouds to deliver the fog. *IEEE Internet Computing*, 21(2):8–15, 2017.

[64] F. Fakhfakh, H. H. Kacem, and A. H. Kacem. Cloudsim4dwf: A cloudsim-extension for simulating dynamic workflows in a cloud environment. In *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 195–202, June 2017. doi: 10.1109/SERA.2017.7965728.

[65] F. Fittkau, S. Frey, and W. Hasselbring. Cdosim: Simulating cloud deployment options for software migration support. In *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the*, pages 37–46, Sept 2012. doi: 10.1109/MESOCA.2012.6392599.

[66] Florian Fittkau. *Simulating Cloud Deployment Options for Software Migration Support.* diplom. de, 2014.

[67] S. Frey and W. Hasselbring. An extensible architecture for detecting violations of a cloud environment's constraints during legacy software system migration. In

Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on, pages 269–278, March 2011. doi: 10.1109/CSMR.2011.33.

[68] Sören Frey and Wilhelm Hasselbring. The cloudmig approach: Model-based migration of software systems to cloud-optimized applications. *International Journal on Advances in Software*, 4(3 and 4):342–353, 2011.

[69] A. Ganapathi, Yanpei Chen, A. Fox, R. Katz, and D. Patterson. Statistics-driven workload modeling for the cloud. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, pages 87–92, March 2010. doi: 10.1109/ICDEW.2010.5452742.

[70] S.K. Garg and R. Buyya. Networkcloudsim: Modelling parallel applications in cloud simulations. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 105–113, Dec 2011. doi: 10.1109/UCC.2011.24.

[71] Gary Garrison, Sanghyun Kim, and Robin L. Wakefield. Success factors for deploying cloud computing. *Commun. ACM*, 55(9):62–68, September 2012. ISSN 0001-0782. doi: 10.1145/2330667.2330685. URL `http://doi.acm.org/10.1145/2330667.2330685`.

[72] Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM computer communication review*, 39(1):68–73, 2008.

[73] Prashant Gupta, A. Seetharaman, and John Rudolph Raj. The usage and adoption of cloud computing by small and medium businesses. *International Journal of Information Management*, 33(5):861 – 874, 2013. ISSN 0268-4012. doi: https://doi.org/10.1016/j.ijinfomgt.2013.07.001. URL `http://www.sciencedirect.com/science/article/pii/S026840121300087X`.

[74] Jens Gustedt, Emmanuel Jeannot, and Martin Quinson. Experimental methodologies for large-scale systems: a survey. *Parallel Processing Letters*, 19(03):399–418, 2009.

[75] Jianhua He, Jian Wei, Kai Chen, Zuoyin Tang, Yi Zhou, and Yan Zhang. Multitier fog computing with large-scale iot data analytics for smart cities. *IEEE Internet of Things Journal*, 5(2):677–686, 2018.

[76] Lorin Hochstein and Rene Moser. *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way.* " O'Reilly Media, Inc.", 2017.

[77] Urs Hoelzle and Luiz Andre Barroso. *The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st edition, 2009. ISBN 159829556X, 9781598295566.

[78] Laura Hosman and Bruce Baikie. Solar-powered cloud computing datacenters. *IT Professional*, 15(2):15–21, 2013.

[79] Fred Howell and Ross McNab. Simjava: A discrete event simulation library for java. *Simulation Series*, 30:51–56, 1998.

[80] J. Hu, U. Y. Ogras, and R. Marculescu. System-level buffer allocation for application-specific networks-on-chip router design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(12):2919–2933, Dec 2006. ISSN 0278-0070. doi: 10.1109/TCAD.2006.882474.

[81] Cong-Thinh Huynh and Eui-Nam Huh. Prediction technique for resource allocation in micro data center. In *Information Networking (ICOIN), 2015 International Conference on*, pages 483–486. IEEE, 2015.

[82] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155 – 162, 2012. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2011.05.027. URL `http://www.sciencedirect.com/science/article/pii/S0167739X11001129`.

[83] Fatemeh Jalali, Kerry Hinton, Robert Ayre, Tansu Alpcan, and Rodney S Tucker.

Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications*, 34(5):1728–1739, 2016.

[84] Steven J Johnston, Philip J Basford, Colin S Perkins, Herry Herry, Fung Po Tso, Dimitrios Pezaros, Robert D Mullins, Eiko Yoneki, Simon J Cox, and Jeremy Singer. Commodity single board computer clusters and their applications. *Future Generation Computer Systems*, 2018.

[85] Jongtack Jung and Hwangnam Kim. Mr-cloudsim: Designing and implementing mapreduce computing model on cloudsim. In *ICT Convergence (ICTC), 2012 International Conference on*, pages 504–509, Oct 2012. doi: 10.1109/ICTC.2012.6387186.

[86] Jongtack Jung and Hwangnam Kim. Mr-cloudsim: Designing and implementing mapreduce computing model on cloudsim. In *ICT Convergence (ICTC), 2012 International Conference on*, pages 504–509, Oct 2012. doi: 10.1109/ICTC.2012.6387186.

[87] MA Kaleem and PM Khan. Commonly used simulation tools for cloud computing research. In *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on*, pages 1104–1111. IEEE, 2015.

[88] S. Karthik and J. J. Shah. Analysis of simulation of ddos attack in cloud. In *Information Communication and Embedded Systems (ICICES), 2014 International Conference on*, pages 1–5, Feb 2014. doi: 10.1109/ICICES.2014.7033841.

[89] P. Kathiravelu and L. Veiga. An adaptive distributed simulator for cloud and mapreduce algorithms and architectures. In *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pages 79–88, Dec 2014. doi: 10.1109/UCC.2014.16.

[90] Vishonika Kaushal and Anju Bala. Autonomic fault tolerance using haproxy in cloud environment. *International Journal of Advanced Engineering Sciences and Technologies*, 7(2):222–227, 2011.

[91] Gabor Kecskemeti. Dissect-cf: A simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory*, 58:188 – 218, 2015.

ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2015.05.009. URL `http://www.sciencedirect.com/science/article/pii/S1569190X15000842`. Special issue on Cloud Simulation.

[92] Gabor Kecskemeti, Wajdi Hajji, and Fung Po Tso. Modelling low power compute clusters for cloud simulation. 2017.

[93] Ahmed Khalid, Jason J Quinlan, and Cormac J Sreenan. Mininam: A network animator for visualizing real-time packet flows in mininet. In *Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on*, pages 229–231. IEEE, 2017.

[94] A. U. Khan, M. Oriol, M. Kiran, M. Jiang, and K. Djemame. Security risks and their management in cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 121–128, Dec 2012. doi: 10.1109/CloudCom.2012.6427574.

[95] Alireza Khoshkbarforoushha and Rajiv Ranjan. Resource and performance distribution prediction for large scale analytics queries. In *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, ICPE '16, pages 49–54, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4080-9. doi: 10.1145/2851553.2851578. URL `http://doi.acm.org/10.1145/2851553.2851578`.

[96] Alireza Khoshkbarforoushha, Rajiv Ranjan, Raj Gaire, Prem Prakash Jayaraman, John Hosking, and Ehsan Abbasnejad. Resource usage estimation of data stream processing workloads in datacenter clouds. *CoRR*, abs/1501.07020, 2015. URL `http://arxiv.org/abs/1501.07020`.

[97] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. Empirical evaluation of workload forecasting techniques for predictive cloud resource scaling. In *Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on*, pages 1–10. IEEE, 2016.

[98] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering - a

systematic literature review. *Inf. Softw. Technol.*, 51(1):7–15, January 2009. ISSN 0950-5849. doi: 10.1016/j.infsof.2008.09.009. URL `http://dx.doi.org/10.1016/j.infsof.2008.09.009`.

[99] Jack P.C. Kleijnen. Verification and validation of simulation models. *European Journal of Operational Research*, 82(1):145 – 162, 1995. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/0377-2217(94)00016-6. URL `http://www.sciencedirect.com/science/article/pii/0377221794000166`.

[100] D. Kliazovich, P. Bouvry, Y. Audzevich, and S.U. Khan. Greencloud: A packet-level simulator of energy-aware cloud computing data centers. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5, Dec 2010. doi: 10.1109/GLOCOM.2010.5683561.

[101] N. Kord and H. Haghighi. An energy-efficient approach for virtual machine placement in cloud based data centers. In *Information and Knowledge Technology (IKT), 2013 5th Conference on*, pages 44–49, May 2013. doi: 10.1109/IKT.2013.6620036.

[102] Y Navaneeth Krishnan, Chandan N Bhagwat, and Aparajit P Utpat. Fog computingâĂŤnetwork based cloud computing. In *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*, pages 250–251. IEEE, 2015.

[103] Ar Kar Kyaw, Hong Phat, and Justin Joseph. Low-cost computing using raspberry pi 2 model b. *Journal of Computers*, 13(3):287–300, 2018.

[104] Bob Lantz and Brian O'Connor. A mininet-based virtual testbed for distributed sdn development. *ACM SIGCOMM Computer Communication Review*, 45(4):365–366, 2015.

[105] Benjamin C Lee and David M Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *ACM SIGOPS Operating Systems Review*, volume 40, pages 185–194. ACM, 2006.

[106] C. Li and Z. Deng. Value of cloud computing by the view of information resources.

In *Network Computing and Information Security (NCIS), 2011 International Conference on*, volume 1, pages 108–112, May 2011. doi: 10.1109/NCIS.2011.30.

[107] Xiang Li, Xiaohong Jiang, Peng Huang, and Kejiang Ye. Dartcsim: An enhanced user-friendly cloud simulation system based on cloudsim with better performance. In *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, volume 01, pages 392–396, Oct 2012. doi: 10.1109/CCIS.2012.6664434.

[108] David J Lilja. *Measuring computer performance: a practitioner's guide*. Cambridge university press, 2005.

[109] S. H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. R. Das. Mdcsim: A multi-tier data center simulation, platform. In *2009 IEEE International Conference on Cluster Computing and Workshops*, pages 1–9, Aug 2009. doi: 10.1109/CLUSTR.2009.5289159.

[110] Saiqin Long and Yuelong Zhao. A toolkit for modeling and simulating cloud data storage: An extension to cloudsim. In *Control Engineering and Communication Technology (ICCECT), 2012 International Conference on*, pages 597–600, Dec 2012. doi: 10.1109/ICCECT.2012.160.

[111] Francisca Losavio, Ledis Chirinos, Nicole Lévy, and Amar Ramdane-Cherif. Quality characteristics for software architecture. *Journal of Object Technology*, 2(2):133–150, 2003.

[112] B. Louis, K. Mitra, S. Saguna, and C. Åhlund. Cloudsimdisk: Energy-aware storage simulation in cloudsim. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 11–15, Dec 2015. doi: 10.1109/UCC.2015.15.

[113] Adil Maarouf, Abderrahim Marzouk, and Abdelkrim Haqiq. Comparative study of simulators for cloud computing. In *Cloud Technologies and Applications (CloudTech), 2015 International Conference on*, pages 1–8. IEEE, 2015.

[114] Pieter-Jan Maenhaut, Hendrik Moens, Bruno Volckaert, Veerle Ongenae, and Filip De Turck. Resource allocation in the cloud: From simulation to experimental validation. In *Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on*, pages 701–704. IEEE, 2017.

[115] Antonios T "Makaratzis, Konstantinos M Giannoutakis, and Dimitrios" Tzovaras. Energy modeling in cloud simulation frameworks. *Future Generation Computer Systems*, 79:715–725, 2018.

[116] Dan C Marinescu. *Cloud computing: theory and practice*. Morgan Kaufmann, 2017.

[117] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing—the business perspective. *Decision support systems*, 51(1): 176–189, 2011.

[118] Harald A. Martens and Pierre Dardenne. Validation and verification of regression in small data sets. *Chemometrics and Intelligent Laboratory Systems*, 44(1):99 – 121, 1998. ISSN 0169-7439. doi: https://doi.org/10.1016/S0169-7439(98)00167-1. URL `http://www.sciencedirect.com/science/article/pii/S0169743998001671`.

[119] Peter Mell and Tim Grance. The nist definition of cloud computing. Technical report, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011.

[120] Judith Myerson. Cloud computing versus grid computing: Service types, similarities and differences, and things to consider. *IBM Research*, 2009.

[121] Alberto Núñez, Jose L. Vázquez-Poletti, Agustin C. Caminero, Gabriel G. Castañé, Jesus Carretero, and Ignacio M. Llorente. icancloud: A flexible and scalable cloud infrastructure simulator. *J. Grid Comput.*, 10(1):185–209, March 2012. ISSN 1570-7873. doi: 10.1007/s10723-012-9208-5. URL `http://dx.doi.org/10.1007/s10723-012-9208-5`.

[122] Simon Ostermann, Kassian Plankensteiner, Radu Prodan, and Thomas Fahringer. Groudsim: An event-based simulation framework for computational grids and

clouds. In *Proceedings of the 2010 Conference on Parallel Processing*, Euro-Par 2010, pages 305–313, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21877-4. URL http://dl.acm.org/citation.cfm?id=2031978.2032020.

[123] Berkin Ozisikyilmaz, Gokhan Memik, and Alok Choudhary. Machine learning models to predict performance of computer system design alternatives. In *Parallel Processing, 2008. ICPP'08. 37th International Conference on*, pages 495–502. IEEE, 2008.

[124] Petra Perner. *Machine Learning and Data Mining in Pattern Recognition: 9th International Conference, MLDM 2013, New York, NY, USA, July 19-25, 2013, Proceedings*, volume 7988. Springer, 2013.

[125] B. Pittl, W. Mach, and E. Schikuta. Cloudtax: A cloudsim-extension for simulating tax systems on cloud markets. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 35–42, Dec 2016. doi: 10.1109/CloudCom.2016.0021.

[126] Benedikt Pittl, Werner Mach, and Erich Schikuta. Bazaar-extension: A cloudsim extension for simulating negotiation based resource allocations. In *Services Computing (SCC), 2016 IEEE International Conference on*, pages 427–434. IEEE, 2016.

[127] John J Prevost, KranthiManoj Nagothu, Brian Kelley, and Mo Jamshidi. Prediction of cloud data center networks loads using stochastic and neural models. In *System of Systems Engineering (SoSE), 2011 6th International Conference on*, pages 276–281. IEEE, 2011.

[128] Kyle Rankin. Hack and /: Manage multiple servers efficiently. *Linux J.*, 2009 (177), January 2009. ISSN 1075-3583. URL http://dl.acm.org/citation.cfm?id=1502508.1502521.

[129] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[130] Robert G Sargent. Validation and verification of simulation models. In *Proceedings of the 36th conference on Winter simulation*, pages 17–28. Winter Simulation Conference, 2004.

[131] Robert G Sargent. Verification and validation of simulation models. In *Proceedings of the 37th conference on Winter simulation*, pages 130–143. winter simulation conference, 2005.

[132] Prince Kwame Senyo, Erasmus Addae, and Richard Boateng. Cloud computing research: A review of research themes, frameworks, methods and future research directions. *International Journal of Information Management*, 38(1):128–139, 2018.

[133] S. Shaoling, Z. Jing, C. Moliang, R. Hui, C. Yi, and F. Xiaodong. Network energy consumption analysis and dormancy mechanism based on ant colony algorithm in cloud computing environment for iot service and real-time embedded industrial control system. In *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pages 5568–5572, May 2015. doi: 10.1109/CCDC.2015.7161791.

[134] Manoel C. Silva Filho and Joel José P. C. Rodrigues. *Internet of Vehicles – Technologies and Services: First International Conference, IOV, Beijing, China, September 1-3, 2014. Proceedings*, chapter Human Readable Scenario Specification for Automated Creation of Simulations on CloudSim, pages 345–356. Springer International Publishing, Cham, 2014. ISBN 978-3-319-11167-4. doi: 10.1007/978-3-319-11167-4_34. URL `http://dx.doi.org/10.1007/978-3-319-11167-4_34`.

[135] Manoel C Silva Filho, Raysa L Oliveira, Claudio C Monteiro, Pedro RM Inácio, and Mário M Freire. Cloudsim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pages 400–406. IEEE, 2017.

[136] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya. Cloudsimsdn: Modeling and simulation of software-defined cloud data centers. In *Cluster, Cloud*

and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on, pages 475–484, May 2015. doi: 10.1109/CCGrid.2015.87.

[137] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. In *Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on*, pages 39–44. IEEE, 2017.

[138] S. Sotiriadis, N. Bessis, N. Antonopoulos, and A. Anjum. Simic: Designing a new inter-cloud simulation platform for integrating large-scale resource management. In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pages 90–97, March 2013. doi: 10.1109/AINA.2013.123.

[139] Alireza Souri, Nima Jafari Navimipour, and Amir Masoud Rahmani. Formal verification approaches and standards in the cloud computing: A comprehensive and systematic review. *Computer Standards & Interfaces*, 2017.

[140] Pengfei Sun, Qingni Shen, Ying Chen, Zhonghai Wu, Cong Zhang, Anbang Ruan, and Liang Gu. Poster: Lbms: Load balancing based on multilateral security in cloud. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 861–864, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0948-6. doi: 10.1145/2046707.2093512. URL `http://doi.acm.org/10.1145/2046707.2093512`.

[141] Thiago Teixeira Sá, N. Rodrigo Calheiros, and G. Danielo Gomes. *Cloud Computing: Challenges, Limitations and R&D Solutions*, chapter CloudReports: An Extensible Simulation Tool for Energy-Aware Cloud Computing Environments, pages 127–142. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10530-7. doi: 10.1007/978-3-319-10530-7_6. URL `http://dx.doi.org/10.1007/978-3-319-10530-7_6`.

[142] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya. Dcsim: A data centre simulation tool for evaluating dynamic virtualized resource management. In *Network and ser-*

vice management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm), pages 385–392, Oct 2012.

[143] Adel Nadjaran Toosi, Jungmin Son, and Rajkumar Buyya. Clouds-pi: A low-cost raspberry-pi based testbed for software-defined-networking in clouds. 2017.

[144] Fung Po Tso, D.R. White, S. Jouet, J. Singer, and D.P. Pezaros. The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures. In *Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on*, pages 108–112, July 2013. doi: 10.1109/ICDCSW.2013.25.

[145] P Turton and TF Turton. Pibrain-a cost-effective supercomputer for educational use. 2014.

[146] Janet M Twomey and Alice E Smith. Validation and verification, 1997.

[147] Blesson Varghese and Rajkumar Buyya. Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*, 79:849–861, 2018.

[148] Pedro Velho, Lucas Mello Schnorr, Henri Casanova, and Arnaud Legrand. On the validity of flow-level tcp network models for grid and cloud simulations. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 23(4):23, 2013.

[149] T Veni and S Mary Saira Bhanu. Prediction model for virtual machine power consumption in cloud environments. *Procedia Computer Science*, 87:122–127, 2016.

[150] Lizhe Wang, Rajiv Ranjan, Jinjun Chen, and Boualem Benatallah. *Cloud computing: methodology, systems, and applications*. CRC Press, 2017.

[151] N. Whittington, L. Liu, B. Yuan, and M. Trovati. Investigation of energy efficiency on cloud computing. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*, pages 2080–2087, Oct 2015. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.309.

[152] B. Wickremasinghe, R.N. Calheiros, and R. Buyya. Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 446–452, April 2010. doi: 10.1109/AINA.2010.32.

[153] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[154] He-Sheng Wu, Chong-Jun Wang, and Jun-Yuan Xie. Terascaler elb-an algorithm of prediction-based elastic load balancing resource management in cloud computing. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 649–654. IEEE, 2013.

[155] Tatu Ylonen and Chris Lonvick. The secure shell (ssh) protocol architecture. 2006.

[156] M. A. Zardari, L. T. Jung, and N. Zakaria. K-nn classifier for data confidentiality in cloud computing. In *Computer and Information Sciences (ICCOINS), 2014 International Conference on*, pages 1–6, June 2014. doi: 10.1109/ICCOINS.2014.6868432.

[157] Jidong Zhai, Wenguang Chen, and Weimin Zheng. Performance prediction of large scale parallel programs with sim-mpi.

[158] Jidong Zhai, Wenguang Chen, and Weimin Zheng. Phantom: predicting performance of parallel applications on large-scale parallel machines using a single node. In *ACM Sigplan Notices*, volume 45, pages 305–314. ACM, 2010.

[159] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[160] Ziming Zhang, Qiang Guan, and Song Fu. An adaptive power management framework for autonomic resource configuration in cloud computing infrastructures. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pages 51–60, Dec 2012. doi: 10.1109/PCCC.2012.6407738.

[161] Wei Zhao, Yong Peng, Feng Xie, and Zhonghua Dai. Modeling and simulation of cloud computing: A review. In *Cloud Computing Congress (APCloudCC), 2012 IEEE Asia Pacific*, pages 20–24. IEEE, 2012.

[162] Yiran Zhao, Shen Li, Shaohan Hu, Hongwei Wang, Shuochao Yao, Huajie Shao, and Tarek Abdelzaher. An experimental evaluation of datacenter workloads on low-power embedded micro servers. *Proceedings of the VLDB Endowment*, 9(9):696–707, 2016.

[163] Jiang Zhihua. Greencloud for simulating qos-based naas in cloud computing. In *Computational Intelligence and Security (CIS), 2013 9th International Conference on*, pages 766–770. IEEE, 2013.