Suitability of microprocessor development boards for hosting small-scale database management systems

Adriaan Cornelis Fokker

22887547

Dissertation submitted in fulfilment of the requirements for the degree Magister Scientiae in Computer Science at the Vaal Triangle Campus of the North-West University

Supervisor: A.R. Botes

Co-supervisor: I. Smit

Graduation: 2018

http://www.nwu.ac.za/

(PAGE INTENTIONALLY LEFT BLANK)

DECLARATION

I, Adriaan Cornelis Fokker declare that

Suitability of microprocessor development boards for hosting small-scale database management systems

is my own work and that all the sources I have used or quoted have been indicated and acknowledged by means of complete references.

Signature: 

Date: 2 November 2017

ACKNOWLEDGEMENTS

I would like to thank everyone involved in the completion of this dissertation. This study broadened and challenged my knowledge in the research and computer science field and would not have been possible without the guidance and services of the following individuals:

Firstly, I would like to express my sincere gratitude to my supervisor, Romeo Botes for the support throughout this study. His immense knowledge in this field enabled me to not only find an interesting and meaningful topic but also guided me through the research, empirical work and other aspects of this dissertation.

Secondly, special and sincere thanks to Imelda Smit my co-supervisor for her assistance throughout the study. Her experience and technical knowledge as well as thorough and timely feedback greatly contributed to the success and quality of the dissertation. In addition, I appreciate the motivation and personal commitment.

Thirdly, thank you to Aldine Oosthuyzen for her expertise and input, specifically relating to the statistical analysis.

Fourthly, I would like to thank Hettie Sieberhagen for the timely language editing of the dissertation.

Fifthly, a special thank you to Marinda Vos for support with the graphical elements throughout this dissertation.

Finally, special thanks to my parents, brother, grandparents and close friends for their unconditional love and support. I express my sincere gratitude to my parents for the robust foundation they provided throughout the years.

ABSTRACT

A microprocessor development board (MPDB) is a less expensive alternative to commodity personal computers (PCs) and can be used for the same purposes – to a certain extent. The primary objective of this study is to investigate the possibility of using a MPDB instead of a commodity PC to host a small-scale database management system (DBMS).

Extensive research is conducted on literature relating to the study in terms of research methodologies, databases, DBMSs and processors. This study is positioned in the positivist research paradigm and makes use of a quantitative research design with hypothesis testing. By assessing the database and DBMS literature, a specific DBMS is chosen based on performance and compatibility with all devices and used with all devices in the study.

An experiment is designed to load test the MPDBs and commodity PC chosen for this study. Load is applied according to the load test design on each device by executing DBMS queries from multiple DBMS clients simultaneously. The DBMS clients are simulated from a separate personal computer with an application developed by the researcher namely, Multi-Client Simulator (MCS). Predefined metrics are captured through MCS during the experiment and stored as raw data in log files. The log file data are imported into a data warehouse to enable data drilldown and scaling for data analysis.

The data analysis is performed by extracting structured experiment data from the data warehouse and the use of statistical analysis software. The statistical analysis includes analysis of variance and allows for accurate comparisons between the performance of MPDBs and that of a commodity PC. The descriptive statistics and analysis of variance results are used to perform statistical analysis and hypothesis testing in order to address the primary objective of the study. The results show that MPDBs are capable of hosting a DBMS similar to a commodity PC to a certain extent.

Finally, the study is communicated by describing the research findings, summarising the experiment results and exploring possible future research. Recommendations are provided by considering the results of the study and the price difference between the tested MPDBs and a commodity PC.

Keywords: microprocessors, database management systems, small-scale database, micro development boards, scientific method, quantitative research

(PAGE INTENTIONALLY LEFT BLANK)

TABLE OF CONTENTS

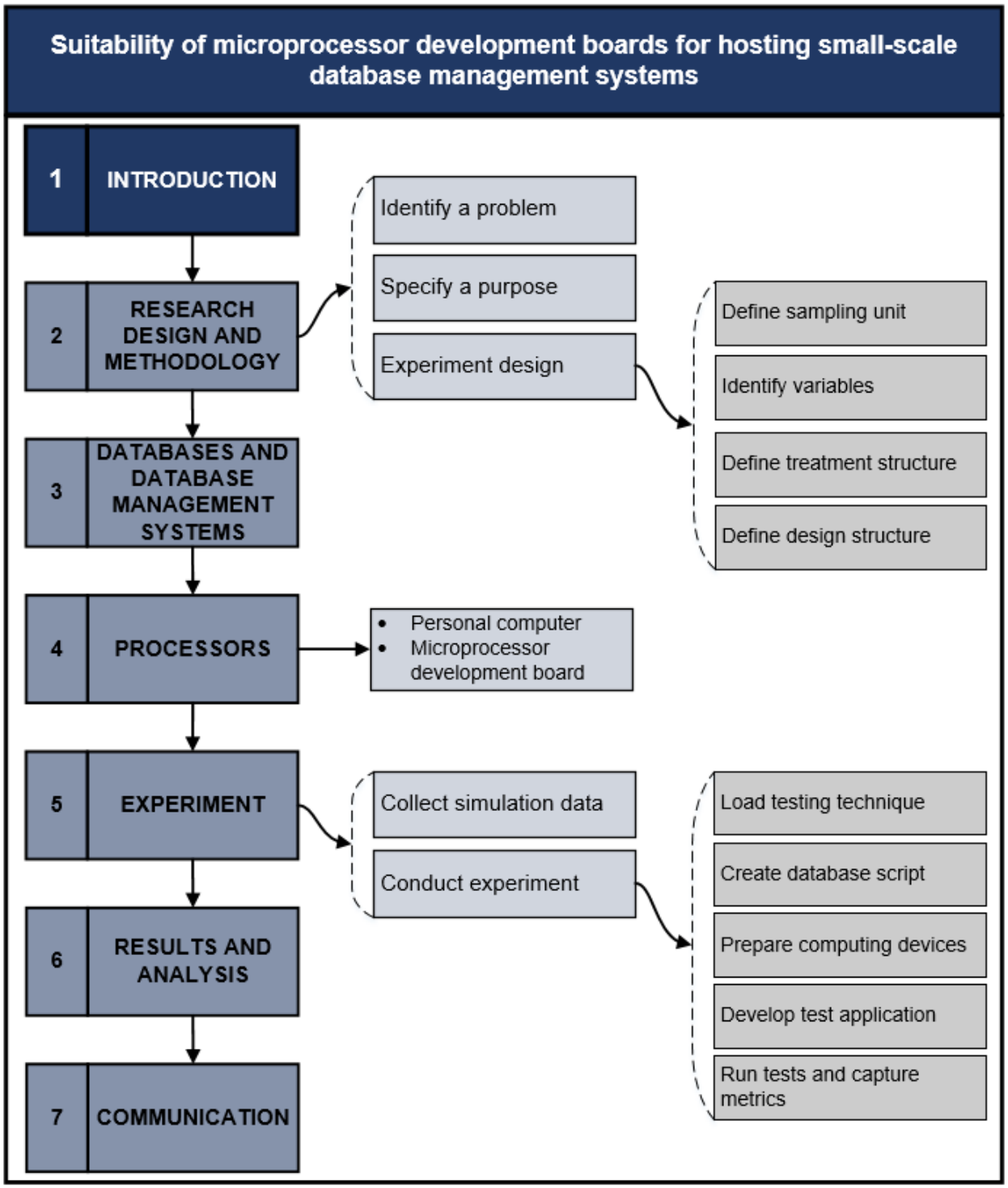
LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

Abbreviation	Meaning
3D	Three-dimensional
ACID	Atomicity, consistency, isolation, and durability
ANOVA	Analysis of variance
ARM	Advanced RISC machine
BBB	BeagleBone Black
CAD	Computer aided design
CLI	Command line interface
CPU	Central processing unit
CRUD	Create, read, update and delete
CSV	Comma separated values
DBMS	Database management system
DD	Digital divide
DSR	Design science research
eMMC	Embedded multi-media controller
ERD	Entity relationship diagram
GB	Gigabyte
GHz	Gigahertz
GPIO	General-purpose input/output
GUI	Graphical user interface
HDMI	High definition multimedia interface
IEA	Intel Edison Arduino
IoT	Internet of things
IP	Internet protocol
IS	Information systems
IT	Information technology
kHz	Kilohertz
MB	Megabyte
Mbps	Megabits per second
MCS	Multi-Client Simulator
MHz	Megahertz
MicroSD	Micro secure digital
MPDB	Microprocessor development board
NoSQL	Not only SQL
OLAP	Online analytical processing
OLTP	Online transaction processing
OS	Operating system
PC	Personal computer
RAM	Random access memory
RDBMS	Relational DBMS
RISC	Reduced instruction set computer
RPi3	Raspberry Pi Model 3
SoC	System on a chip
SP	Stored procedure
SPSS	Statistical package for the social sciences
SQL	Structured Query Language
SSH	Secure shell
TCP	Transmission control protocol
T-SQL	Transact-SQL
TV	Television
USB	Universal serial bus
VAT	Value added tax


Suitability of microprocessor development boards for hosting small-scale database management systems
1
INTRODUCTION
2
RESEARCH DESIGN AND METHODOLOGY
Identify a problem
Specify a purpose
Experiment design
Define sampling unit
Identify variables
Define treatment structure
Define design structure
3
DATABASES AND DATABASE MANAGEMENT SYSTEMS
4
PROCESSORS
Personal computer
Microprocessor development board
5
EXPERIMENT
Collect simulation data
Conduct experiment
Load testing technique
Create database script
Prepare computing devices
Develop test application
Run tests and capture metrics
6
RESULTS AND ANALYSIS
7
COMMUNICATION

CHAPTER 1: INTRODUCTION

The technological environment as we know it constantly introduces new technology that is better and faster than its predecessors. This is also true in the area of computer technology. Unfortunately, hardware performance is usually directly correlated with cost – the better the performance of the hardware, the higher the purchase price (Koomey *et al.*, 2009). In South Africa, the access to information and communication technologies contains gaps between certain groups of the public, showing that a large group of the population does not have access to such technologies (Bornman, 2016:264). Reasons for this lack of access may relate to the cost of commodity personal computers. Microprocessor development boards (MPDBs) introduced the world to an alternative that offers lower but reliable performance at a fraction of the price of a normal computer. MPDBs are similar to personal computers but are very compact, not much larger than a credit card, and are not sold like standard personal computers (PCs) with casing for the equipment. The Raspberry Pi 3 is an example of a popular MPDB and consists of a 1.2 GHz quad-core ARM Cortex-A53 CPU, 1GB of RAM, and it can run Linux, as well as Microsoft Windows 10 Internet of Things Core operating systems (Raspberrypi.org, 2015c; Raspberrypi.org, 2015e).

A series of experiments will be conducted through the study to determine whether MPDBs are able to successfully host small-scale database management systems (DBMSs). The extent to which the workload will be accommodated in terms of data processing, will be compared to that of a commodity PC. The hypothesis is that MPDBs support the data processing of small-scale DBMSs to a certain extent when compared to commodity PCs. If the hypothesis is not rejected, it would conclude that users, such as business owners, hobbyists and students alike, could significantly reduce technology infrastructure costs relating to small-scale DBMS implementations. This reduced infrastructure cost may be of great benefit to developing countries.

Note that the abbreviation list on the previous page may be used throughout the dissertation when the meaning of an abbreviation is unclear. The meaning of abbreviations will however be given with the first use in each chapter.

This chapter introduces the key concepts of the study (§1.1), which include databases, DBMSs and their performance, load testing, and computing technologies. Formulation of the problem statement (§1.2), the objectives of the study (§1.3), the research design and methodology (§1.4) that will guide the study, a chapter classification (§1.5) of the chapters to follow, and finally a chapter conclusion (§1.6).

1.1 Concepts key to the study

The key concepts are identified as database (§1.1.1), DBMS (§1.1.2), DBMS performance evaluation (§1.1.3), load testing (§1.1.4), and computing technologies (§1.1.5).

1.1.1 Database

A database can be defined as a shared, integrated computer structure that stores end user data, data about data, and procedures that handle data changes and retrieval operations (Elmasri & Navathe, 2011:4; Morris *et al.*, 2013:7). Coronel *et al.* (2012:9) state that databases can be categorised into different types. The database can be classified depending on how the data[1] is intended to be used and how the stored data is structured. Different types include flat file, hierarchical, network, relational and NoSQL databases (Srivastava, 2014).

Database scale is a subjective matter and therefore different kinds of database scale classifications exist as noted by Stackoverflow.com (2009). The scale of a database can be determined by considering one, or a combination of the following aspects relating to the specific database (Stackoverflow.com, 2009):

- The total number of records hosted by the database;
- the characteristics of the database;
- the duration of queries and optimisation; and
- the storage type into which the entire database fits.

1.1.2 Database management system

A DBMS is defined by Chapple (2016) and Ramakrishnan and Gehrke (2003:4) as software designed to assist in the maintenance and utilisation of large collections of data. DBMSs can either be accessed directly with the use of programming languages or the programming language of the DBMS (i.e. Transact-SQL) if it is integrated in the DBMS solution. Most DBMSs provide

[1] In context of this study, the concept of data relays two meanings. When data refers to information it is treated as singular; while data representing a number of facts are seen as plural. In the implementation of this principle, data created to run the pilot and experiment is seen as general information, which is as a single concept. Data generated to inform the results of the research, are seen as a collection of individual facts, being plural.

reporting and query tools that allow users to view data in the database (Altaviser, 2008; Chapple, 2016).

1.1.3 Database management system performance evaluation

In order to evaluate the performance of a DBMS on different technologies, standard performance metrics must be chosen. Hardware resources relate directly to DBMS performance (Mullins, 2010) – if resources are limited, the performance of the DBMS will be affected. With fewer resources available, such as insufficient random access memory (RAM), data loss may be a risk for transactions that did not commit (or rolled back) when the system crashes. Mullins (2010) defines five factors that influence database performance: workload, throughput, resources, optimisation, and contention.

The focus of this study, in terms of DBMS performance, is whether the hardware resources of different technologies are sufficient to handle small-scale DBMS operations and to what extent the operation can be executed. A DBMS's performance is directly, but not exclusively, determined by the capability of its server to host the DBMS efficiently; hence, a section on load testing will follow.

1.1.4 Load testing

Load testing assists in identifying the maximum operating capacity of a DBMS hosted by a computing platform and any bottlenecks that might be degrading performance (Meier *et al.*, 2007). Load testing will allow the researcher to determine to what extent database load can be dealt with by the computing platform in question and whether the test results indicate sufficient MPDB load capability to host a small-scale DBMS.

1.1.5 Computing technologies

This section describes two computing technologies: MPDBs and commodity PCs.

1.1.5.1 Microprocessor development boards

Intel (2017) and Kant (2007:17) define a microprocessor as an integrated circuit that contains all the functions of a CPU used in a PC. A MPDB is a printed circuit board that contains a microprocessor and the minimal support logic needed to use the microprocessor (Kant, 2007:17). Intel released the first commercial single-chip microprocessor in 1971. Figure 1-1 shows the Intel 4004, which was a 4-bit CPU designed for a calculator. Its operating speed was 740 kHz and it could execute approximately 60 000 instructions per second (Intel, 2015).

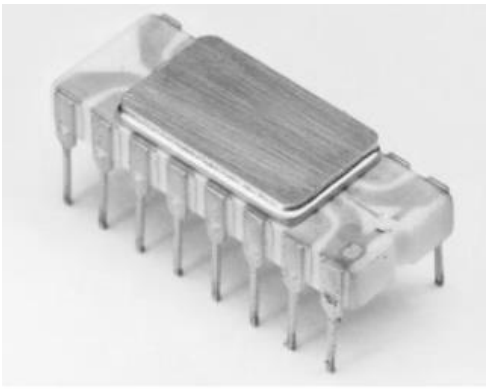


Figure 1-1: Intel 4004 (Intel, 2015)

There is a large list of uses for MPDBs, for example, the Raspberry Pi website has a project section, which is divided into categories. Each category has hundreds of different projects that the community is working on. The categories include (Raspberrypi.org, 2015d):

- *Automation, sensing, and robotics* – artefacts that are controlled by a MPDB.
- *Three-dimensional (3D) printing* – using MPDBs to operate 3D printers.
- *Gaming* – playing or designing games on and for a MPDB.
- *Graphics, sound, and multimedia* – such as using a MPDB as a media centre.
- *Media centres* – using MPDBs to manage multimedia (i.e. music and movies).
- *Networking and servers* – using a MPDB as a database server; this study will fall into this category.
- *Other projects* – projects not allocated to a specific category.

Some of the popular MPDBs in South Africa include BeagleBone Black, Intel Edison, PCDuino, and Raspberry Pi, all of which are similar in design and performance. The Raspberry Pi MPDB is illustrated in Figure 1-2.

Figure 1-2: Raspberry Pi board (Raspberrypi.org, 2015b)

1.1.5.2 Commodity personal computer

The Linux Information Project (2006) broadly defines a computer as any class of human-made devices or systems that is able to modify data in some meaningful way. A more specific definition of computer is given by Sipral (2007:2): a PC is an electronic device that stores, retrieves, and processes data. A computer can be programmed with instructions and it comprises hardware and software. Hardware refers to the physical components of the computer and software to the instructions that make the computer act in a certain way, depending on user input (Sipral, 2007:2).

It is important to note throughout the study, that the effectiveness of MPDBs for hosting small-scale DBMS solutions is compared to commodity PCs only because a commodity PC is considered as suitable for hosting a DBMS (Cdata.com, 2016; Microsoft, 2017; MySQL, 2017d).

1.2 Problem statement

The digital divide (DD) refers to a social inequality concerning the access or use of information and communication technologies (Soltan, 2016). The DD affects a number of countries, including South Africa, where the extent of access to information and communication technologies shows gaps between certain groups of the public. In South Africa, the DD exists between gender, population groups and in the levels of education (Bornman, 2016:264). In developing countries, especially with regards to certain areas in South Africa, the access to information and communication technologies, falls second to basic human needs, such as food and clothes (Dalvit & Gunzo, 2014:166).

The excessive cost of implementing a new small-scale DBMS with the use of traditional servers or commodity PCs is problematic owing to the high price of hardware for these servers and commodity PCs. The lower price of MPDBs could address the problem that the financially challenged may not have the financial freedom to purchase a commodity PC. The motivation for and purpose of this study is to prove that small-scale DBMSs can efficiently be hosted by a MPDB instead of a commodity PC. In other words, users of small-scale DBMS solutions can use MPDBs instead of commodity PCs to host their DBMS. Users can significantly reduce costs related to their IT infrastructure; however, the public may not be aware of the capabilities or even the existence of MPDBs, which might be used for this purpose.

The problem addressed by this study is the high price of commodity PCs that may be unaffordable to certain individuals. Business owners, hobbyists, students and other users could use MPDBs that are less expensive than commodity PCs, for small-scale DBMS solutions.

The research question of this study is

To what extent is it possible to host a small-scale DBMS on MPDBs?

1.3 Objectives of the study

The following objectives are formulated for the study:

1.3.1 Primary objective

The objective of this study is to evaluate the effectiveness of MPDBs compared to commodity PCs for hosting small-scale DBMS implementations.

The primary objective is supported by the following secondary objectives:

1.3.1.1 Hypothesis test

This secondary objective of the study is a hypothesis test. The hypothesis tests whether small-scale DBMS solutions can make use of MPDBs instead of the traditional commodity PCs or server computers. The scientific formulation for the hypothesis is shown in Table 1-1.

Table 1-1: Scientific hypothesis formulation for this study

Formulation	Description
μ1	Microprocessor development board
μ2	Commodity personal computer
H_o: μ1 = μ2	Microprocessor development boards support the data processing of small-scale database management system solutions similar to commodity personal computers
H_a: μ1 ≠ μ2	Microprocessor development boards do not support the data processing of small-scale database management system solutions similar to commodity personal computers

1.3.1.2 Divergence point determination

This secondary objective of this study is to determine a divergence point from stability between MPDBs and PCs for hosting small-scale DBMS solutions. The purpose of the divergence point is to indicate the point where each MPDB diverges from stability when load is applied on the device.

1.3.2 Theoretical objectives

In order to achieve the primary and supportive objectives, the following theoretical objectives are formulated for the study:

- Gain an understanding of the quantitative research process.
- Gain an understanding of DBMSs.
- Gain an understanding of commodity PCs.
- Gain an understanding of MPDBs and different types thereof.
- Gain an understanding of device load testing.

1.3.3 Empirical objectives

The following empirical objectives are formulated in accordance with the primary objective of the study:

- Design an experiment;
- Set up a testing environment by installing a DBMS on each MPDB to be tested;
- Run simulations on each type of technology while recording performance metrics relating to the process;
- Evaluate and compare the simulation results; and

- Propose recommendations based on the simulation results.

1.4 Research design and methodology

There are four research paradigms, which include positivism, interpretivism, critical social theory, and design science research. Positivists adhere to the view that only factual knowledge is gained from observations; observations should be quantifiable and lead to statistical analysis (Gray, 2014:20). The goal of interpretivism is to understand a theory by analysing the meaning instead of raw facts (Goldkuhl, 2012:4). Critical social theory, is used to emancipate the oppressed by deconstructing oppressing structures and reconstructing these without the oppressing structures (Gray, 2014:20). Design science research can be defined as the creation of new knowledge through the design of novel artefacts and the analysis of the use and/or the performance of such artefacts along with reflection and abstractions (Vaishnavi & Kuechler, 2004; Gregor & Hevner, 2013:341).

This study is based on factual metrics and observations followed by statistical analysis and it therefore clearly relates to the positivist paradigm. The study consists of a literature review and conducting a series of experiments according to the positivist paradigm.

Researchers in the positivist paradigm employ the scientific method of enquiry in an effort to derive conclusions (Edmonds & Kennedy, 2017:2). The presentation and interpretation of the scientific method varies between various disciplines and methods (for example qualitative or quantitative) with the premise remaining the same (Edmonds & Kennedy, 2017:2). The importance of quantitative research in the positivist paradigm (Adebesin *et al.*, 2011:310), which originated in the physical sciences and involves the use of mathematical models as the method for data analysis (Williams, 2007:66), provides the foundation of this study. Figure 1-3 shows the scientific method as suggested steps for a quantitative research design. Each step shown in Figure 1-3 is described as follows:

Step 1 – Identify a need. A need realises as a problem to be specified and justified.

Step 2 – Establish a theoretical foundation. Resources should be located; these should include books, journals, and electronic resources. Once the resources are identified, the relevant resources must be chosen and organised. The resources should then be summarised in a literature review.

Step 3 –Formulate the research question. Specific, narrow, and measurable or observable objectives must be declared for the study.

Step 4 – Design the study. Design a structure that guides the set up of the experiment, the data collection from participants or devices and statistical data analysis. Select the participants for the study, determine the data collection method, select or design data-collection instruments, and outline data-collection procedures.

Step 5 – Collect the data. Obtain permissions and gather data.

Step 6 – Analyse the data. Conduct statistical analysis on the collected data; make use of trends, comparisons, and predictions.

Step 7 – Report the results. Draw objective, unbiased conclusions about the analysed data and evaluate outcomes of the study.

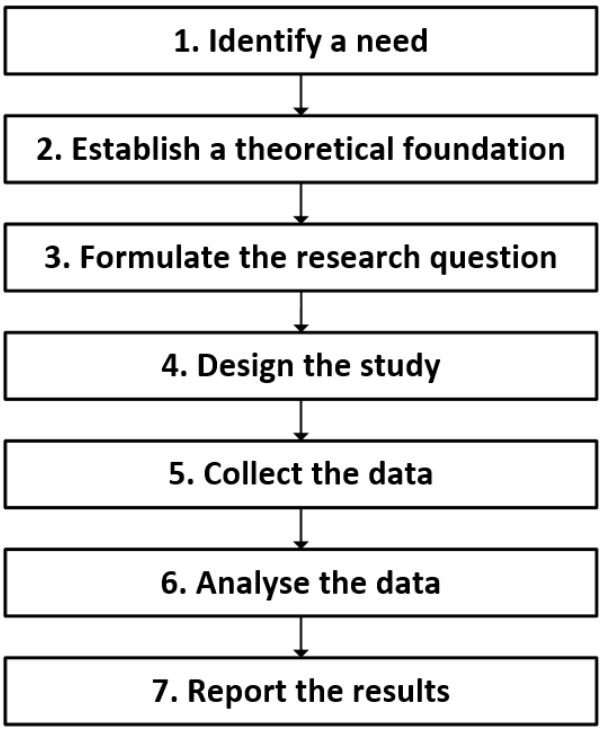


Figure 1-3: The scientific method (Edmonds & Kennedy, 2017:3)

The scientific method provided by Edmonds and Kennedy (2017:3) and adapted by Abraham S. Fischler School of Education (2012?)[2] informs the research structure of this study. The steps for

[2] The source of the citation refers to an educational institution and not an individual.

the scientific method as shown in Figure 1-3 are adapted to the study by integrating it with a load testing technique provided by Meier *et al.* (2007). The load testing technique is discussed in Chapter 4 Section 4.5. This study's adapted research design culminates in the following steps:

Step 1 – Identify a problem. The problem is described in the problem statement in Section 1.2. To summarise the problem statement: traditional costs relating to DBMS hardware infrastructure are high, and therefore a MPDB may be used as an alternative.

Step 2 – Specify a purpose. Section 1.3 describes the objectives of this study using primary, theoretical, and empirical objectives. A summary of the primary objective: evaluate the effectiveness of MPDBs compared to commodity PCs for hosting small-scale DBMS implementations.

Step 3 – Experiment design. The experiment will be designed to load-test the MPDBs and commodity PC.

Step 4 – Collect simulation data. In this study, testing data will be gathered from the Internet to populate the database on each computing platform.

Step 5 – Conduct experiment. A DBMS will be installed and simulations will be executed on each computing technology, from which performance metrics will be recorded.

Step 6 – Results and analysis. The data derived from the experiment will be statistically analysed in this step. The analysis will show the extent of the performance gap between MPDBs and PCs.

Step 7 – Discussion and conclusion. The evaluation will determine whether it is possible and effective for small-scaled DBMSs to be hosted by MPDBs.

1.5 Chapter classification

This study comprises the following chapters:

Chapter 1 Introduction. Introduces the reader to the study by giving an overview of the dissertation in terms of the background and motivation for the study, scope, and key concepts that guide the study.

Chapter 2 Research design and methodology. In-depth research on the positivist paradigm, the quantitative research process, and the experimental research design.

Chapter 3 Databases and database management systems. Literature review on databases in terms of database and DBMS types.

Chapter 4 Processors. Literature review on processors in terms of PCs and MPDBs.

Chapter 5 Experiment. This chapter explains how the experiment is conducted to ensure the repeatability of the experiment.

Chapter 6 Results and analysis. This chapter presents the results from the study in a structured arrangement in order to draw informed conclusions.

Chapter 7 Communication. This chapter provides a conclusion and final thoughts on the study. It also outlines a summary of the knowledge gained throughout the study.

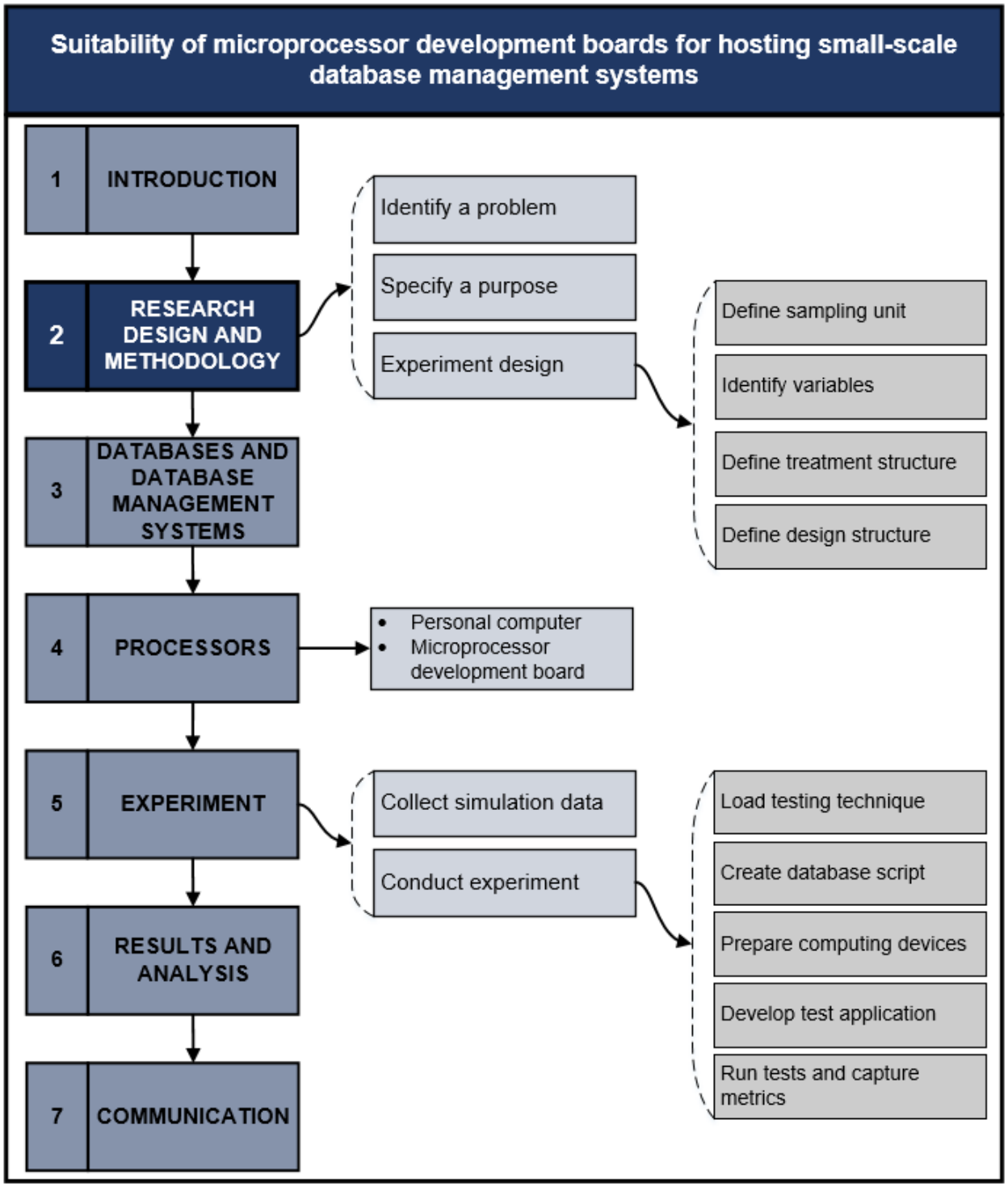
1.6 Conclusion

The objective of this chapter was to introduce the study and outline key aspects relating to it. This objective was accomplished by describing the concepts key to the study, introducing the purpose of the study in the problem statement, listing study objectives, describing the research design and methodology relating to the study, and finally describing the subsequent chapters.

The motivation for this study is to address the high cost of hardware for commodity PCs or servers for hosting small-scale DBMSs. This problem is addressed by using MPDBs as an alternative to commodity PCs for hosting small-scale DBMSs. The primary objective of this study is to evaluate the effectiveness of MPDBs compared to commodity PCs for hosting small-scale DBMS implementations.

To guide the research process of the study, the primary objective is supported by hypothesis testing and determining a point of divergence from stability. In addition, theoretical and empirical objectives have been formulated to ensure the success of the study.

The following chapter describes the literature relating to research methodology and progresses to positioning the study and design of its research structure.


Suitability of microprocessor development boards for hosting small-scale database management systems
1
INTRODUCTION
2
RESEARCH DESIGN AND METHODOLOGY
Identify a problem
Specify a purpose
Experiment design
Define sampling unit
Identify variables
Define treatment structure
Define design structure
3
DATABASES AND DATABASE MANAGEMENT SYSTEMS
4
PROCESSORS
Personal computer
Microprocessor development board
5
EXPERIMENT
Collect simulation data
Conduct experiment
Load testing technique
Create database script
Prepare computing devices
Develop test application
Run tests and capture metrics
6
RESULTS AND ANALYSIS
7
COMMUNICATION

CHAPTER 2: RESEARCH DESIGN AND METHODOLOGY

2.1 Introduction

The aim of this study is to evaluate the effectiveness of microprocessor development boards (MPDBs) for hosting small-scale database management systems (DBMSs), compared to the effectiveness of commodity personal computers (PCs).

This chapter analyses the nature of the study to identify the research paradigm into which it may be categorised. The definition of research, obtained from OECD (2002), states that it is creative work undertaken to increase knowledge via a well-ordered approach. This is followed by the selection of a suitable research methodology for the study. Methodology is defined as the systemic and theoretical analysis of methods in a field of study, which is comprised of theoretical analysis of the methods and principles pertaining to a field of study (Irny & Rose, 2005). Once the study has been positioned, the chapter proceeds to introduce the specifics of the research methodology. The research methodology provides the researcher with a foundation on which the study's research design is based.

This chapter covers the following sections: research philosophies (§2.2), research paradigms (§2.3), positioning of the study (§2.4), the positivist research paradigm which utilises the quantitative research process and methods (§2.5), the research design of this study (§2.6), and finally, the conclusion (§2.7).

2.2 Research philosophy

A research philosophy, as stated by Saunders *et al.* (2009:107), "relates to the development of knowledge as well as the nature of that knowledge". The research philosophy adopted by a researcher contains important assumptions on how the researcher views the world (Saunders *et al.*, 2009:107; Dudovskiy, 2016b). The assumptions guide the researcher in choosing a research strategy, as well as in the methods to be utilised. The philosophy adopted by the researcher is partly influenced by practical considerations, but the main influence is usually the researcher's view of the association between knowledge and the manner by which it is developed (Saunders *et al.*, 2009:108; Dudovskiy, 2016b). A research paradigm is selected in the next section (§2.3) based on the philosophical assumptions of the researcher. The research paradigm guides the selected research methods and the compiled research strategy during the study.

Hirschheim (2010:13) summarises four types of philosophical assumptions that ground a study. They are:

- *ontological assumptions* – which relate to the beliefs about the nature of the world, as people perceive it;
- *epistemological assumptions* – which are the views on how knowledge is acquired;
- *methodological assumptions* – which are the beliefs about which mechanisms are appropriate for acquiring knowledge; and
- *axiological assumptions* – which relate to the beliefs about the role of an individual's values in research.

Table 2-1 provides a summary of the four types of philosophical assumptions described above in the context of the paradigms, namely that of positivism, interpretivism, critical social theory, and design science described in this section.

Table 2-1: Philosophical assumptions in the context of the four research paradigms (Vaishnavi & Kuechler, 2004; Blanche *et al.*, 2006; Adebesin *et al.*, 2011:310)

Research Paradigms	Philosophical Assumptions			
	Ontology	**Epistemology**	**Methodology**	**Axiology**
Positivism	• Single, stable reality • Law-like	• Objective • Detached observer	• Experimental • Quantitative • Hypothesis testing	• Truth • Prediction
Interpretivism	• Multiple realities • Socially constructed	• Subjective • Empathetic observer	• Interactional • Qualitative • Interpretation	• Contextual understanding
Critical social theory	• Socially constructed reality • Discourse • Power	• Suspicious • Political • Constructing observer • Versions	• Deconstruction • Textual analysis • Discourse analysis	• Value-bound inquiry • Contextual understanding • Values of researcher affects study
Design science	• Multiple, contextually situated realities	• Acquire knowledge through design • Context-based construction	• Developmental • Impact analysis of artefact on composite system	• Control • Creation • Understanding

The four research paradigms are discussed in the following section to assist the researcher in positioning the study.

2.3 Research paradigms

Three classical research paradigms are identified by Blanche *et al.* (2006:6) Adebesin *et al.* (2011:309), namely; positivist, interpretivist, and critical social research. Design science research is a fourth research paradigm, which is popular in information systems (IS) research (Gregor & Hevner, 2013:337). In some research fields, these research paradigms may be referred to in different terms, for example, the critical social research paradigm is occasionally referred to as constructionism (Adebesin *et al.*, 2011:311).

Positivism is based on the assumption that there is an orderly arrangement to the world (Adebesin *et al.*, 2011:310). The positivist's core argument is that the social world exists externally to the researcher and its properties can be measured directly through observation (Noor, 2008:1602; Gray, 2014:21). Positivists conduct research with the following beliefs (Gray, 2014:21): reality consists of what is available to human senses, inquiry should be based on scientific observation, and ideas can only be incorporated into knowledge if they can be tested through empirical experience. Research is undertaken in a value-free way, therefore focusing on factual results and statements (Saunders *et al.*, 2009:114). This paradigm is adopted by the natural scientist and the general research methodology employed is quantitative in nature (Saunders *et al.*, 2009:113). Quantitative research is performed with the use of factual data based on predetermined research methods, which lead to statistical analysis (Creswell, 2003:17).

Interpretivism is based on the assumption that people's knowledge of reality is constructed in their minds as a result of individual experiences or perceptions. Individual experiences and perceptions can include language, shared meanings, and societal norms (Noor, 2008:1602). The interpretivist research paradigm therefore assumes that there is no single reality (Adebesin *et al.*, 2011:311; Myers, 2013). Interpretivism requires the researcher to understand differences between humans – it emphasises the difference between conducting research among people rather than objects. The researcher needs to adopt an empathetic stance by understanding the subjects' world from their point of view (Saunders *et al.*, 2009:116). Interpretive research is mostly adopted by social science researchers using qualitative data. Qualitative research makes use of emerging research methods in which data are obtained from open-ended questions and the resulting analysis is subjective rather than statistical (Creswell, 2003:17).

Critical social theory is similar to interpretivism, since it is also based on the assumption that reality is socially constructed (Myers, 1997:241). This research paradigm goes further than the interpretivist paradigm – it supports the notion that a particular social construction of reality is influenced by various power relations that exist among people, such as economic, political and cultural (Adebesin *et al.*, 2011:311). According to Saunders *et al.* (2009:111), this paradigm

supports the belief that the perceptions and subsequent actions of social actors are responsible for creating social phenomena. Moreover, these social phenomena are in a constant state of revision through the process of social interaction. Constructionism stresses the necessity to explore the subjective meanings that motivates the social actions in order for the researcher to understand these actions (Saunders *et al.*, 2009:111).

The three paradigms discussed above are referred to as classical research paradigms. They can briefly be summarised as follows; social sciences (interpretivism and critical social theory) often deal with actions of the individual while natural sciences (positivism) analyse data for consistencies (Gray, 2014:23).

A fourth paradigm relevant to IS research, **design science research** (DSR), is a modern research paradigm (Adebesin *et al.*, 2011:309). Gregor and Hevner (2013:337) state that DSR is regarded as a legitimate IS research paradigm although certain gaps exist in the understanding and application of the concepts and methods relating to DSR. It seeks to solve defined problems (Gregor & Hevner, 2013:341). By definition, DSR changes the state of the world through the introduction of artefacts created by people (Adebesin *et al.*, 2011:311). The purpose of DSR is to create innovations that provide the definition of ideas, practices, technical capabilities, and products in order to effectively and efficiently accomplish the analysis, design, implementation, management, and use of ISs (Hevner *et al.*, 2004:76; Gregor & Hevner, 2013:341). According to Gregor and Hevner (2013:338), DSR is traditionally adopted in engineering fields and in science of the artificial. Myers (2013) maintains that DSR usually involves the design of an artefact, which does not normally form part of classical paradigms. It is not uncommon to adopt some of the qualitative and quantitative research methods from the three classical research paradigms in DSR (Myers, 2013).

2.4 Positioning the study

This study is based on objects and not humans, thereby eliminating the multiple reality ontology. From an epistemological perspective, knowledge is gained through objective observations and the recording of numeric results. The method to gain knowledge will include the conducting of an experiment to obtain quantitative output data, statistically analysed, to test the hypothesis of the study. Experimental, quantitative and hypothesis testing methods form part of the research methodology in this study. Finally, axiology provides truth and predictions of the study by using the results of the research topic.

In terms of its ontology, epistemology, methodology, and axiology as listed in Table 2-1, this study clearly forms part of the positivist research paradigm:

- This study focuses on a law-like, single and stable reality proving conformance with the positivism *ontology*. Experiment metrics in this study are measured and not affected by external perceptions or experiences.
- The *epistemology* of the study is objective and the researcher or his views do not affect the results of the study. In this study, knowledge is gained through scientific observation and can be tested and proved through repeatable experiments.
- Regarding its *methodology*, experimental, quantitative and hypothesis testing techniques form the research methods that are used in this study.
- *Axiologically* the study provides truth to the research topic with an element of statistical prediction.

The following section subsequently discusses the positivist research paradigm extensively.

2.5 Positivist research paradigm

The French philosopher, August Comte (1798-1857), introduced the concept of positivism. Comte emphasised observation and reason as means of understanding human behaviour (Dash, 2005). He held that true knowledge is based on experience of senses and that knowledge is gained through observation and experiment. Positivist researchers adopt Comte's scientific method in order to generate knowledge (Dash, 2005; Bonet *et al.*, 2007:12). According to McGregor and Murnane (2010:3), positivism gained popularity in the early 1800s and was the dominant paradigm used to conduct research until the mid-1900s.

Positivists hold that only observable phenomena will lead to obtaining credible data where the research is based on facts and events that interact in a determined and an observable manner (Collins, 2010:38). The researcher acts as an independent observer and there are no provisions for human interests in the study (Dudovskiy, 2016a). The researcher generally generates hypotheses from existing theories. A hypothesis is tested to confirm it, in whole or in part, or refute it. Such a result may lead to further development of a theory, which again may be tested by additional research (Saunders *et al.*, 2009:113).

Positivism should be understood within the framework of the principles and assumptions of science. These assumptions include determinism, empiricism, parsimony, and generality (Dash, 2005):

- *Determinism* is the assumption that events are caused by certain circumstances which means that it is necessary to understand such casual links for prediction and control.
- *Empiricism* refers to the collection of verifiable empirical evidences that support hypotheses or theories.
- *Parsimony* relates to the explanation of the phenomena in the most economical way possible.
- *Generality* is the process of generalising the observation of the particular phenomenon to the world.

The scientific method, shown in Figure 2-1, used by positivists utilises qualitative or quantitative methods of enquiry (Edmonds & Kennedy, 2017:2). The scientific method used by this positivistic study is quantitative, which encapsulates a number of research methods, such as surveys, correlational research and experiments. The following section explains the quantitative research process, followed by a section on quantitative research methods.

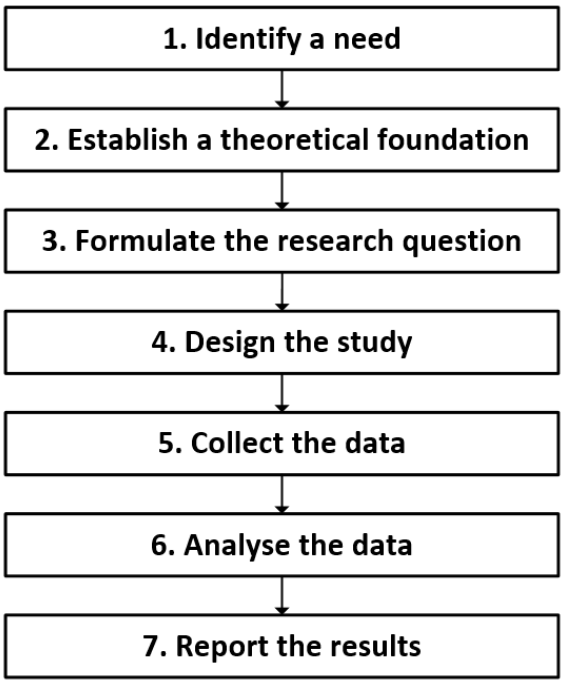


Figure 2-1: The scientific method (Edmonds & Kennedy, 2017:3)

2.5.1 Quantitative research process

Quantitative research is, according to Johnson and Harris (2002:102), characterised by its analytical approach to data that are generated. There are three broad types of quantitative research, namely: descriptive, comparative, and prescriptive. Descriptive research is an approach to simplify a description of some phenomenon facilitated by the use of data. Comparative research relates to the statistical comparison of data between two or more groups. Prescriptive research aims to predict results or situations by formulating models of cause and effect (Johnson & Harris, 2002:102).

Due to the fact that MPDBs are compared with one another and with the commodity PC, this study forms part of comparative research. Statistical analysis techniques can be used to accurately compare data from different groups. One example of such a technique is analysis of variance, where differences among group means are analysed. Analysis of variance (ANOVA) is used to test the null hypothesis, which states that there is no significant difference among predefined groups of data. The alternative hypothesis states that there is at least one significant difference among the groups of data (Statistics Solutions, 2013).

The quantitative research process, shown in Figure 2-2, is the adaption by Abraham S. Fischler School of Education (2012?) of the scientific method provided by Edmonds and Kennedy (2017:3) shown in Figure 2-1. Figure 2-2 represents a general quantitative research process that can be adopted by various types of quantitative research studies (Abraham S. Fischler School of Education, 2012?). With each research design, the general quantitative research process should be adjusted to suit the purpose of the specific research topic. The quantitative research design provided by Abraham S. Fischler School of Education (2012?) is discussed next:

Step 1 – Identify a problem. A problem must be specified and justified and a need should be suggested to study the problem.

Step 2 – Review the literature. Resources must be located, for example books, journals, and electronic resources. Once the resources are located, the relevant resources must be chosen and organised. The resources should then be summarised in a literature review.

Step 3 – Specify a purpose. Specific, narrow, and measurable or observable objectives must be declared for the study.

Step 4 – Collect data. Select the participants in the study, determine the data collection method, select or design data-collection instruments and outline data-collection procedures. Finally, obtain permissions and gather data.

Step 5 – Analyse and interpret data. Conduct a statistical analysis on the collected data with the use of trends, comparisons, and predictions.

Step 6 – Report and evaluate. Draw objective, unbiased conclusions about the analysed data and evaluate outcomes of the study.

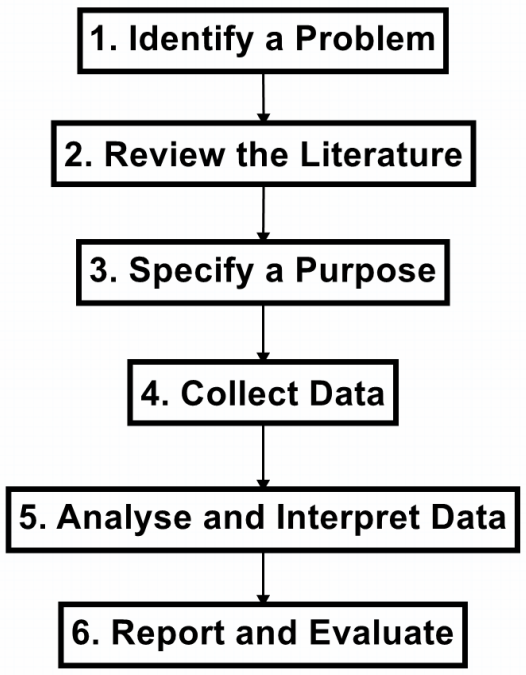


Figure 2-2: Quantitative research process adopted by Abraham S. Fischler School of Education (2012?)

The quantitative research design provided by Abraham S. Fischler School of Education (2012?) informs this study.

2.5.2 Quantitative research methods

This section provides a description of the most common quantitative research methods. They include surveys, correlational research and experiments.

2.5.2.1 Surveys

This research method enables the researcher to collect quantitative data from a sample of individuals, which can be analysed statistically. Surveys are most commonly used to answer

questions the researcher has, such as: who, what, where, and how many. The survey method allows the collection of data from a large population in a highly economical way. Data are often collected through a standard set of questionnaires administered to a sample population. Data collection techniques that often form part of the survey research method include questionnaires, structured observation, and structured interviews (Saunders *et al.*, 2009:144; Check & Schutt, 2012:160). Data collection through surveys is unlikely to be as comprehensive as other research methods. Another significant drawback is the fact that people might not do the survey truthfully or leave out important information (Saunders *et al.*, 2009:144).

Surveys do not form part of this study because the testing data are programmatically generated and not collected from individuals.

2.5.2.2 Correlational research

Privitera (2014:240) defines correlational research as:

> *"The measurement of two or more factors to determine or estimate the extent to which the values for the factors are related or change in an identifiable pattern."*

In other words, correlational research aims to identify a pattern (correlation) between multiple variables. It is important to note that correlational research determines to which extent variables are related and not to the extent changes in one variable affects the values of other variables (Privitera, 2014:240; Siegle, 2015).

Correlational research may be combined with another research method (i.e. surveys) during the data collection phase. Once data have been collected, they are statistically analysed, usually with the use of scatter diagrams, regression lines, and other relevant statistical analysis tools or models (Privitera, 2014:241).

Correlational research does not form part of this study because the devices are tested individually without the intent to identify correlation; the study aims to compare individual device performance.

2.5.2.3 Experiments

Yount (2006:1) defines an experiment as a prescribed set of conditions that permits measurement of the effects of a particular treatment. The goal of an experiment is to study causal links to determine if a change in one independent variable induces a change in another dependent

variable (Saunders *et al.*, 2009:142). The independent variable is controlled or set by the researcher and the dependent variable is measured (Yount, 2006:1).

In the experimental context, hindrances to good research design are called sources of experimental invalidity. Experiments can be validated by internal and external validity. Internal invalidity exists when other influences, such as extraneous sources of variation, have not been controlled by the researcher. External validity holds when the experimental findings can be confidently generalised to the world (Yount, 2006:2; Saunders *et al.*, 2009:143).

An article written by members of the Statistical Analysis System Institute (SASI), an analytical software institute, titled "*Concepts of experimental design*", provides a clear overview of the process of an experiment (SAS, 2005). Experimental design is the process of planning a study to meet specified objectives. It is important to plan an experiment well, to ensure that the right type of data representing a sufficient sample size are available to answer the research questions of interest as clearly and efficiently as possible (SAS, 2005:1).

The article referred to above, is used to guide the experimental design of this study due to its information technology and statistical analysis approach to experiments. An experiment can be designed by following the steps (SAS, 2005:2):

1. **Define the problem and questions to be addressed.** Clearly define the questions to be answered through the experiment, and identify the sources of variability in the experimental conditions.
2. **Define the population of interest.** Define and describe the population from which information are to be gathered. The population is the collective whole of subjects from which data are collected. The experiment should designate the population for which the problem is to be examined.
3. **Determine the need for sampling.** The researcher may select a sample from the population if the population of interest is too large or not available to study in its entirety. A sample is a sub-set of units that are selected from the population.
4. **Define the experimental design.** Clearly define the details of the experiment. This will ensure that the desired statistical analysis is possible and that the usefulness of the results is improved. Defining the experimental design consists of four activities:
    - *Define the sampling unit.* Clearly define the sampling unit. This is the smallest unit of analysis from which data will be collected in the experiment.
    - *Identify the types of variables.* Four categories of variables are important to the success of the experiment: background, constant, uncontrollable, and primary variables. Inconclusive results usually stem from a lack of defining these classifications.

Background variables can be identified and measured but cannot be controlled – these variables influence the outcome of the experiment and are co-variates. Constant variables can be controlled or measured and will be held constant throughout the study. Uncontrollable variables are variables that are prevented by the conditions in the study to be manipulated or are very difficult to measure. The primary variables are independent variables that are of interest to the researcher.

- *Define the treatment structure.* The treatment structure is concerned with the primary variables the researcher wants to study with the objective to derive inferences. The primary variables are controlled by the researcher and are expected to show the effects of interest on the dependent variables. The treatment structure should relate to the objectives of the experiment and the type of data that are available.
- *Define the design structure.* Experimental designs usually involve the allocation of sampling units to a range of different treatments; either randomly, or randomly with constraints. The latter refers to blocked designs. Blocks are groups of sampling units that share some characteristics; the blocks are formed to be as homogeneous as possible in terms of the characteristics of each block. Two commonly used design structures include completely randomised design and randomised complete block design. The completely randomised design structure involves assigning subjects to treatments at complete randomness. The randomised complete block design structure comprises dividing subjects into blocks according to demographic characteristics. Subjects in each block are then chosen at random and assigned to treatments so that all treatments appear in each block. The advantage of the block design over the completely randomised design is that it allows the researcher to make comparisons among treatments.

5. **Collecting data.** Document the data collection protocol and confirm the instruments to be valid, reliable, and calibrated prior to data collection. Follow the protocol strictly when the data collection process commences. The researcher must explain the data collection procedures to the person that will be doing the actual data collection to ensure that the data collector does not re-organise the data collection processes in an effort to be more efficient when in fact such an action may compromise the integrity of the data.
6. **Analysing data.** Experiment data may be analysed by using ANOVA that is designed for the particular experimental design. The analysis tools that are used are greatly dependent on the experimental design. These analysis tools include pivot tables, ANOVA and general statistical analysis techniques. Data analysis will enable the researcher to determine the differences in the responses across the range of treatments or any interaction between the treatment levels.

The process of experimental design described by SAS (2005) is hereby concluded. This study makes use of experiments to meet the relevant objectives and will now proceed to the research design followed in this study.

2.6 Research design of the study

The research structure of the study integrates a quantitative research process (Abraham S. Fischler School of Education, 2012?), as discussed earlier in this chapter in Section 2.5.1; the load testing technique adapted from Meier *et al.* (2007), introduced in Chapter 1 and described in Chapter 4 (§4.5); and the experimental design process supplied by SAS (2005), delineated in Section 2.5.2.3. This section describes how the aforementioned processes are integrated into the study. Figure 2-3 illustrates the research design of this study and will be illustrated before the start of each chapter while highlighting the chapter in dark blue. This section covers the following activities from Figure 2-3; experiment; results and analysis; and communication.

The experiment performed numerous steps, which is described in Chapter 5 and forms a significant part of the study. The purpose of the experiment is to allow the researcher to generate accurate, relevant data relating to each sampling unit. Each of the steps combined in the experiment, explains how the researcher plans to capture the required data to be analysed.

Note that a pilot is performed before the experiment to determine the feasibility and serve as a guide for the experiment. The pilot is a shortened, simplified and incomplete version of the main experiment.

2.6.1 Identify a problem

The problem was clearly identified in Chapter 1, Section 1.2; the focus of the problem statement is the high financial cost of the infrastructure of commodity PCs for the use of hosting small-scale DBMSs. With the use of a MPDB as an alternative to the commodity PC or traditional server, the high cost of implementing a DBMS can be reduced. It is important to note that a server is similar to a PC; it is only termed a server because of the type of tasks it performs. For example, if a computer (server) is dedicated to tasks that other computers (clients) depend on, the computer hosting the tasks is termed the server and the other computers that depend on those tasks are the clients. Servers therefore usually require hardware capable of higher performance than client computers.

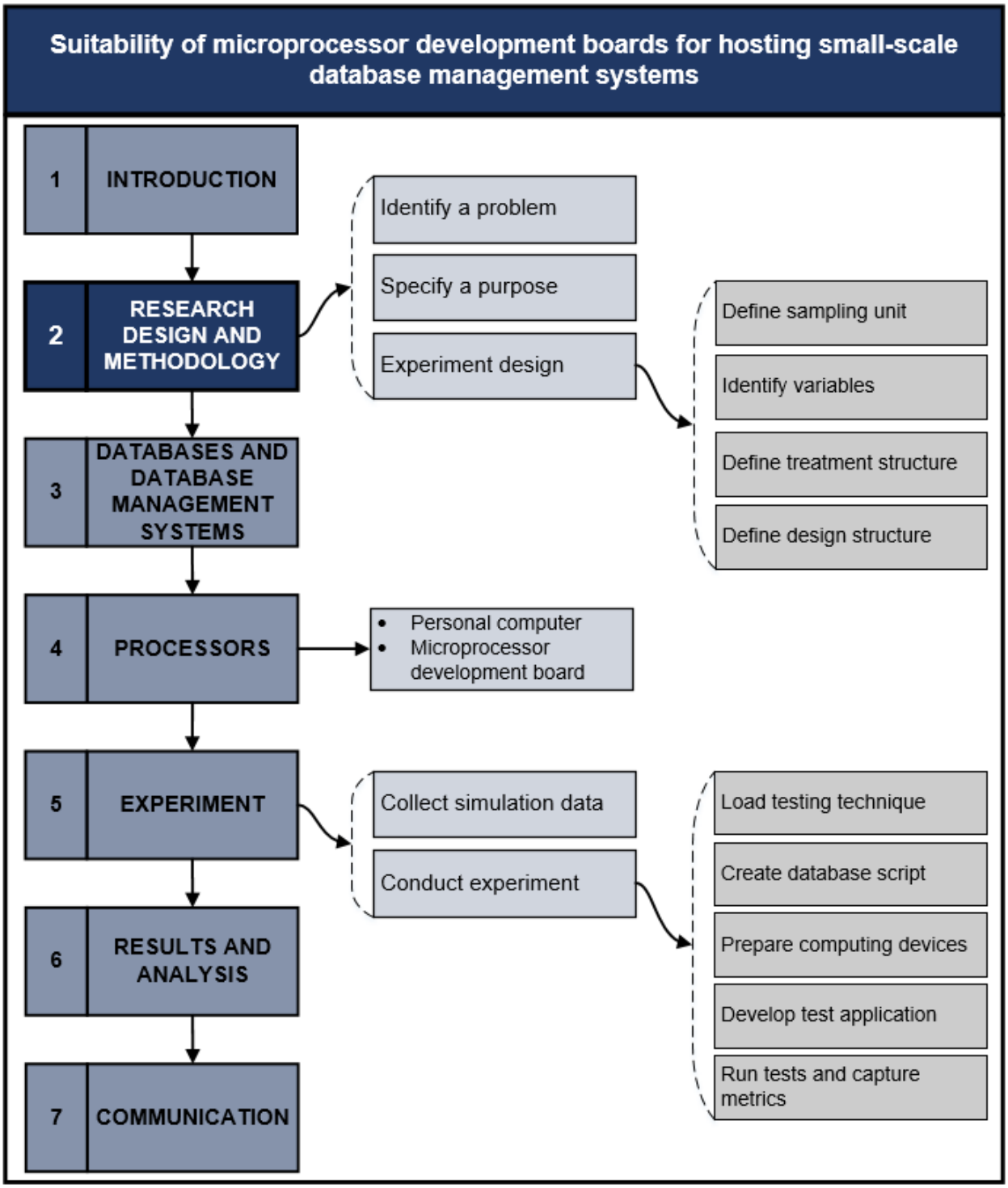


Figure 2-3: Research design of the study

The problem statement provides the motivation for the study, which is to test the hypothesis that small-scale DBMSs can efficiently be hosted by a MPDB instead of a commodity PC or server. Therefore, the problem addressed by this study is that commodity PCs may be unaffordable to certain individuals and that MPDBs may be used as an alternative to commodity PCs for hosting small-scale DBMSs.

The research question derived from the problem statement is:

To what extent is it possible to host a small-scale DBMS on MPDBs?

2.6.2 Specify a purpose

In Section 1.3 of Chapter 1, the purpose of the study through the specification of the primary, theoretical and empirical objectives is described. The primary objective of the study is reiterated in this section. The primary objective is to evaluate the effectiveness of MPDBs for hosting small-scale DBMS solutions compared to the effectiveness of commodity PCs. The primary objective is supported by two objectives, namely a hypothesis test and the determination of a divergence point from stability for each MPDB.

The hypothesis tests the statement that small-scale DBMS solutions can make use of MPDBs instead of traditional commodity PCs or server computers. The hypothesis statement is proven true if MPDBs are capable of hosting small-scale DBMS solutions in a practical way, similar to commodity PCs to a certain extent. The second supportive objective is the determination of the divergence point relating to performance instability when the MPDBs are tested to capacity. These are referred to as stress tests. The stress test refers to the load testing technique described in Section 2.6.5 Step 1. The graph in Figure 2-4 shows a point of divergence between the stability of MPDBs and PCs. The divergence point represents a situation in which it is no longer viable to make use of the particular computing technology owing to low performance. The load-stability divergence point of a MPDB is indicated on Figure 2-4, and transaction loading beyond this point will require a computing technology capable of higher performance. Any point below the line of stability is unstable and therefore not viable. It is important to note that the point of divergence indicated on Figure 2-4 is an example used to illustrate that MPDBs' performance is lower than that of PCs and that different types of MPDBs will not necessarily be plotted on the exact same point. Note that the visual representation of a point of divergence shown in Figure 2-4 was compiled by the researcher to facilitate the reader's understanding in this matter.

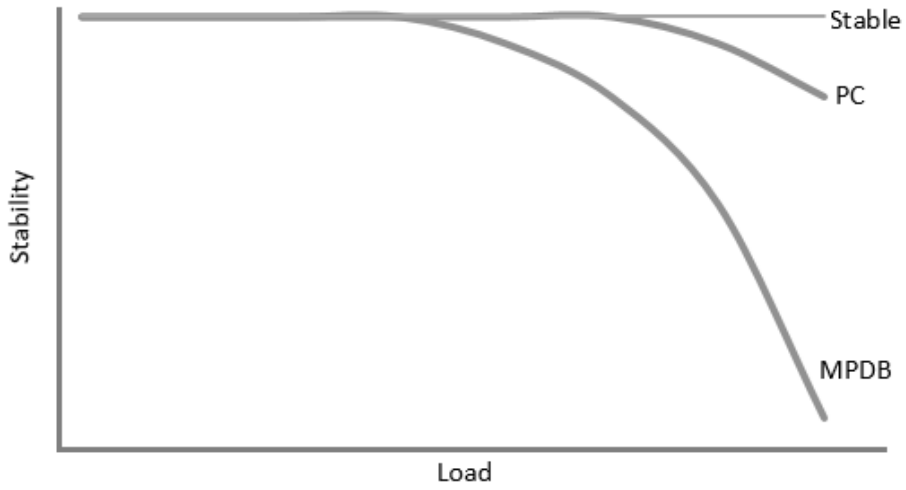


Figure 2-4: Load-stability divergence point

2.6.3 Design of the experiment

The design of the experiment is divided into four steps that will guide the researcher in designing a rigid experiment strategy to be followed throughout the experiment. The experimental design process is adapted from SAS (2005) and was described in Section 2.5.2.3. The purpose of this section is to inform the reader how each step relates to the specifics of this study.

1. **Define sampling unit.** The study will consist of two main categories of sampling units namely, MPDBs and PC. Both categories of sampling units are clearly defined in Chapter 1 (§1.1.5). The MPDB category consists of different models (or brands) of sampling units, in other words, one sampling unit relates to one computing device. Chapter 4 describes the sampling units relating to this study in detail.
2. **Identify variables.** The variables relevant to the study are allocated to the four variable-categories described by SAS (2005) and depicted in Table 2-2.

Table 2-2: Variables in the study

Variable	Type	Description
Primary	Load	The amount of load (work) to be executed by a computing device. The load amount can be altered by changing the number of consecutive DBMS queries and/or individual query complexity.
Background	Throughput	Refers to the data processing capability of the device, measured through the metrics identified in Section 2.6.5.
	Point of divergence (stability)	The point where the computing platform becomes unstable due to insufficient performance. The computing platform is regarded as unstable when it hangs, crashes, or unable to successfully complete DBMS queries, i.e. queries that time out. Instability are only considered they are caused by the primary variables or something internal to the design of the computing platform (and not for example faulty hardware or software components).
Constant	Season	The time of year the experiment is conducted is held constant throughout the study.
	Location	The same physical location is used for each computing platform throughout the experiment.
Uncontrollable	Room temperature	Owing to limitations to the study, room temperature cannot be controlled. Fluctuations in room temperature are however minimized to a certain extent by holding to the predefined constant variables.
	Temperature of device components	Some, but not all, of the computing platforms are capable of measuring hardware component temperatures. Device temperature does however relate directly to stability because hardware components' performance gradually decreases as temperature increases and will eventually come to a halt once a component overheats. If a device overheats, it is considered unstable, which can easily be monitored across all platforms.

3. **Define treatment structure.** The treatment structure lays out how the primary variable is to be manipulated throughout the experiment. The researcher is concerned with the behaviour of the background variables that are influenced by the primary variable. One primary variable was identified in Table 2-2, namely load. The load variable is composed of the load testing technique, described in Section 2.6.5 Step 1. Load will be manipulated to determine the point of divergence for each MPDB and throughput capability (background variables) for each computing platform.
4. **Define design structure.** The selected models of sampling units (Step 1) will each be allocated to the treatment structure described above. The design structure is therefore structured and not random, since identical treatments are applied to each sampling unit – represented by a computing platform.

2.6.4 Collect data

The content of the data required by the study is not of great importance since the study is focused on query performance and not on the data content itself. Variety in data type is more important than the knowledge that can be gained from analysing the data. The source of the data therefore

carries no weight in this study, as long as there is enough data for the researcher to be able to perform load tests and determine the point of divergence from stability on all MPDBs.

The type of data can either be computer-generated data or human-generated data which, for the purpose of this study, will not influence the outcome. The source of the data can range from simulation data that is shared on the Internet to actual data that was gathered in a business. This study makes use of simulation data generated from a script file.

2.6.5 Conduct experiment

Five steps are followed to complete the experiment; (1) load testing technique (with five sub-steps), (2) create database scripts, (3) prepare computing devices, (4) develop test application, and (5) run tests and capture metrics. Steps 3 and 5 are to be repeated for each computing platform relating to the experiment.

Subsequently, the steps are discussed:

1. **Load testing technique.** The load testing technique is adapted from Meier *et al.* (2007); Meier's original model is described in Chapter 4 Section 4.5. The adapted model excludes Step 3 (create a workload model), which incorporates user delay time. This step is excluded due to the fact that a large number of users that simultaneously perform actions on the system are simulated in the experiment. In other words, user actions constantly overlap and therefore no user delay time is considered.

   The first five steps of the adapted technique are described in this section, while Steps 6 and 7 relate to Section 2.6.5 Step 5 (run tests and capture metrics) and Section 2.6.6 (analysis and results) respectively:

   - ***Step 1 – Identify performance acceptance criteria.*** Performance acceptance criteria are based on the hypothesis test of the study; the acceptance criteria evaluate whether MPDBs' performance is sufficient to host a small-scale DBMS. Acceptable performance realises when a MPDB is able to process data in a DBMS similar to commodity PCs.
   - ***Step 2 – Identify key scenarios.*** General scenarios are identified in this step. A scenario relates to one database stored procedure (SP). Five key scenarios are defined and include extremely low, low, medium, high and extremely high demand. The demand refers to the nature of the SP in terms of data processing. High demand is for example a complex SP that needs to be executed via the DBMS and requires a high amount of data processing from the computing device. Low demand refers to simple SPs sent to the DBMS and does not require a great deal of data processing. These SPs are written

by the researcher and each SP contains one or more database query that read or alter data in the database.

- ***Step 3 – Identify target load levels.*** The target load levels are planned to include the key scenarios described in the previous step. Each load target consists of one or more key scenario (or SP). Four target load levels are identified, namely; low, medium, high, and extremely high load. Computing platforms are load-tested by starting with low load and progressively increasing load.
- ***Step 4 – Identify metrics.*** The metrics will be captured during the experiment and include; task duration, random access memory (RAM) utilisation, central processing unit (CPU) utilisation, whether each task is completed successfully, overall device load average over time from a combination of device resources. Metrics are described in more detail in Section 2.6.6.
- ***Step 5 – Design specific tests.*** The tests are designed with different combinations of database SPs; each test forms part of a load level category (Step 3 above). This test design strategy allows the researcher to thoroughly test each computing platform with a uniform load test design. The load test design with the relating database SPs is discussed in Chapter 5 Section 5.4.

2. **Create database script.** A database script file is generated in order to set up the databases on the different computing technologies. The script file contains the database structure specifics as well as the testing data.
3. **Prepare computing devices.** The chosen operating system (OS) and DBMS is installed on the computing platform. The choice of OS and DBMS depends on the compatibility of this software on each computing platform utilised by the experiment. Once the required software is installed, the database script created in the previous step is executed on each computing device to create the database schema (i.e. tables and SPs) and insert the testing data into the created tables.
4. **Develop test application.** An application, namely Multi-Client Simulator, is developed by the researcher to control the experiment and capture metrics; the details of the application are discussed in Chapter 5 Section 5.7.
5. **Run tests and capture metrics.** Tests are conducted and guided by the load test design through the use of the Multi-Client Simulator to capture the metrics identified in Step 4 of the load testing technique. These metrics are captured during the experiment by executing database SPs designed according to the key scenarios described in Step 2 of the load testing technique.

2.6.6 Analysis and results

Analysis and results refer to Chapter 6, where the results from the pilot and experiment are analysed and presented in an organised way. This allows the researcher to draw accurate and proper conclusions about the study.

The metrics described below are captured during the experiment and then imported into a data warehouse to enable data drilldown and aggregation for analysis purposes. Each metric provides information relating to device performance:

- *Duration of each task allocated to the computing device* – recorded in seconds; a task with a longer duration on a certain device compared to other devices indicates lower performance for the specific device.
- *Whether each task is completed successfully* – recorded as true or false; tasks that fail or time out on a specific device indicate a lack of performance.
- *Device RAM utilisation* – indicated as a percentage of total RAM; when RAM usage is close to 100 percent, it is an indication that the device is working close to full capacity in terms of RAM and that increased load may cause the device to fail or require a longer duration for certain tasks. A device showing lower RAM usage for a specific task compared to another device, shows that the specific device may have more efficient memory usage and therefore higher performance.
- *Device CPU utilisation* – indicated as a percentage of full CPU capacity; when CPU usage is close to 100 percent, it is an indication that the device is working close to full capacity in terms of processing power and that increased load may cause the device to fail or require a longer duration for certain tasks. A device showing lower CPU usage for a specific task compared to another device, shows that the specific device may have a faster and more efficient CPU and therefore higher performance.
- *Overall device load average over time in terms of all resources, including RAM, CPU and disk utilisation* – recorded as a decimal value; this value must be divided by the device's number of CPU threads to indicate actual device load. For example, a value of 1.0 for a single thread CPU is equal to full device load or 100 percent load. A value of 2.0 for the same CPU shows that the device is 100 percent overloaded or on 200 percent load. A load value of 2.0 for a device with a CPU containing four threads, shows that the device is running at 50 percent load (Hoffman, 2014). This load metric is recorded over a duration of one, five and 15 minutes. Only the one-minute load average is analysed in this study because some of the tasks that are monitored do not reach five minutes in duration. This metric does not form part of the pilot.

2.6.7 Communication

This step of the research process relates to the final chapter (Chapter 7) of the study in which a discussion will provide a summary of the research and practical work performed, as well as the results of the study. This chapter also draws the conclusion in order to close the study.

2.7 Conclusion

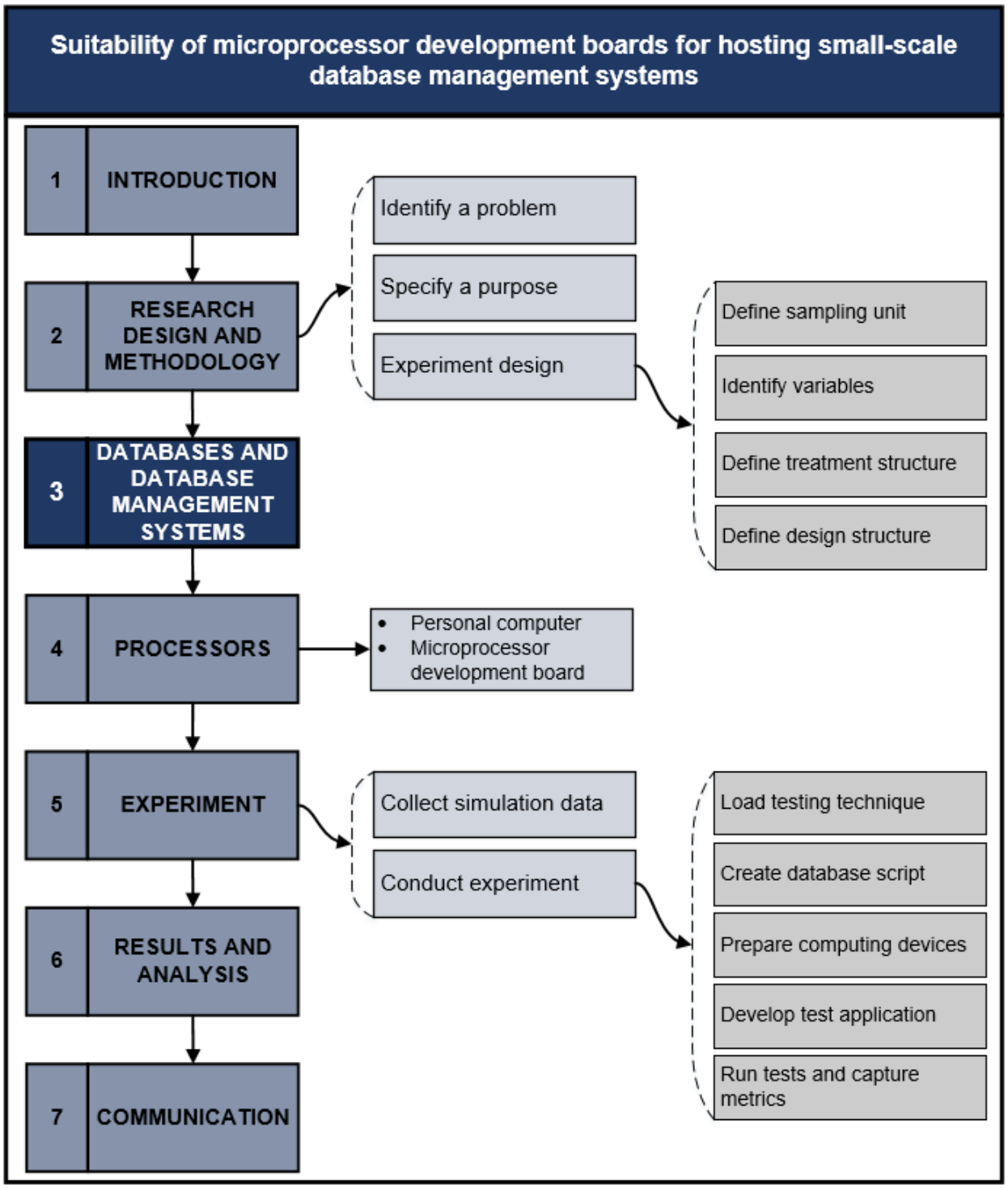
This chapter started with the research philosophies and different research paradigms utilised in IS, and showed how these two concepts are related. The research paradigms each consists of a set of philosophical assumptions which include ontology, epistemology, methodology and axiology. Four research paradigms were described, namely: positivism, interpretivism, critical social theory, and DSR, where DSR does not form part of the three classical research paradigms.

The chapter continued to position the study into the positivist paradigm along with its relevant philosophical assumptions. A detailed discussion on the positivist paradigm in terms of research processes and methods followed to further motivate the positioning of the study. The positivist research methods include surveys, correlational research, and experiments. The experiment research method was chosen to allow the study to meet the set objectives.

After positioning the study, the chapter progressed to the research design of this study. In this section, the researcher explained that a pilot, similar to the main experiment, is conducted to guide the researcher through the experiment. The section detailed how the experiment is to be performed in terms of problem identification, defining a purpose, experiment design, collection of simulation data, conducting the experiments, analysis and results, and communication after the completion of the experiment.

A strong understanding of the positivist paradigm and the quantitative research process were developed in this chapter to guide the research in the remainder of the study. In addition, the following theoretical objective was achieved in this chapter; develop an understanding of the quantitative research process. The research design and methodology chapter is hereby concluded and the database literature is discussed in the next chapter.

(P*AGE INTENTIONALLY LEFT BLANK)*


Suitability of microprocessor development boards for hosting small-scale database management systems
1
INTRODUCTION
2
RESEARCH DESIGN AND METHODOLOGY
Identify a problem
Specify a purpose
Experiment design
Define sampling unit
Identify variables
Define treatment structure
Define design structure
3
DATABASES AND DATABASE MANAGEMENT SYSTEMS
4
PROCESSORS
Personal computer
Microprocessor development board
5
EXPERIMENT
Collect simulation data
Conduct experiment
Load testing technique
Create database script
Prepare computing devices
Develop test application
Run tests and capture metrics
6
RESULTS AND ANALYSIS
7
COMMUNICATION

CHAPTER 3: DATABASES AND DATABASE MANAGEMENT SYSTEMS

3.1 Introduction

The primary objective of this study is to determine the suitability of microprocessor development boards (MPDBs) for hosting small-scale database management systems in relation to commodity personal computers (PCs). In order to achieve this, a discussion of the existing literature on databases and database management systems (DBMSs) is required.

This chapter discusses the terms and concepts relating to databases and DBMSs that will be used throughout the study. A broad overview of databases and DBMSs is that databases store data while users and computer programs can use DBMSs to interact with the database (Morris *et al.*, 2013:7; Chapple, 2016). In this study, databases and DBMSs are used during the experiments and analysis, and therefore form an important part of the study.

In this literature review chapter, databases (§3.2) in terms of flat file, hierarchical, network, relational and NoSQL are discussed first. Secondly, a discussion on DBMSs (§3.3), and thirdly, an analysis on DBMS performance evaluation (§3.4) are presented. Fourthly, literature on the MySQL DBMS is discussed (§3.5), and then this chapter is concluded (§3.6).

3.2 Database

A database can be defined as a shared, integrated computer structure that stores end user data, data about data, and procedures that deal with data changes and retrieval operations (Elmasri & Navathe, 2011:4; Morris *et al.*, 2013:7).

Stored procedures (SPs) form an important part of databases that allow developers to store blocks of instructions (or queries) that can modify and return data based on input parameters (Byham & Guyer, 2017). A SP is defined as a block of code written with input and/or output parameters in the database's programming language that can be stored directly in the database with a specified name, and executed from the database or external applications (Gabry, 2016).

There are different contexts in which databases can be classified; these include how the data will be used and how the stored data is structured. Online analytical processing (OLAP) and online transaction processing (OLTP) are two categories of how database data is used. It also considers the time sensitivity of the data; this refers to how accurate and timeous the data needs to be (Coronel *et al.*, 2012:9).

According to Coronel *et al.* (2012:9) and Kimball *et al.* (2008:608), OLTP refers to operational databases, meaning that such a database captures day-to-day operations in the form of data. This business data should be recorded immediately and accurately to ensure that when the data is referenced to at a later stage, the data can be trusted to be accurate. Online analytical processing is a database design strategy that focuses on storing historical data and business metrics such as aggregates. Databases that are used exclusively for decision-making are categorised as OLAP databases, whether the nature of the decision is tactical or strategic (Kimball *et al.*, 2008:608).

A data warehouse is classified as an OLAP database and is defined by Golfarelli and Rizzi (2009:5) as "a collection of data that supports decision-making processes". It is further described as a relational database designed for query and analysis rather than processing of transactions. Data warehouses normally contain historical data that is derived from transaction data, but can include data from other sources as well. The transaction data can be derived from multiple sources, for example two separate and completely different OLTP databases (Golfarelli & Rizzi, 2009:5).

The experiment conducted as part of this study makes use of an OLTP database, since it is designed to simulate day-to-day database transactions of a business. The data analysis in this study entails analysing metrics recorded from the experiment. The experiment results will be imported into a data warehouse to simplify the analysis process.

As described in Chapter 1 Section 1.1.1, the database scale classification process is of a subjective nature. A number of different methods of determining database scale exist and the classification process can make use of more than one method (Stackoverflow.com, 2009). These methods are described in Table 3-1.

Table 3-1: Database scale classification methods (Stackoverflow.com, 2009)

Classification Method	Database Scale
Total number of records hosted by the database	**Small** – 10^5 or fewer records. **Medium** – 10^5 to 10^7 records. **Large** – 10^7 to 10^9 records. **Very large** – 10^9 or greater number of records.
Characteristics of the database	**Small** – Performance is not a concern and only marginal increases in performance are seen through optimisation such as indexing. **Medium** – Only one or two staff members are responsible for the maintenance of the database in a part-time fashion. The primary role of these staff members is only to ensure minimal downtime and resolve intolerable performance complications. **Large** – Dedicated expert staff members are responsible for database maintenance and optimisation. The health and status of the database are monitored closely. **Very large** – The database hosts vast volumes of data that must be readily available. Performance optimisation is a non-negotiable requirement and with its absence, the database would be, or nearly, unusable.

Classification Method	Database Scale
Duration of queries and optimisation	**Small** – Queries complete in less than one second and do not require optimisation. **Medium** – Queries run for more than one second when no indexes are in place. **Large** – A query optimisation task require more than an hour to complete.
Storage type into which the database fits	**Small** – The entire database fits into the random access memory (RAM) of the database server. **Medium** – The entire database fits into the hard disk drives of the database server. **Large** – The database files are distributed across multiple hard disk drives of multiple servers.

This study classifies database scale in accordance with the total number of records method (Table 3-1) since it is easily comparable across devices and databases.

Different types of databases exist, classified by how the database structures the stored data as illustrated in Figure 3-1 below, followed by a discussion of each type. Figure 3-1 depicts a timeline ranging from the 1980s to 2014, which reveals the sequence and approximate date of the most popular database types that were used in the particular time frames.

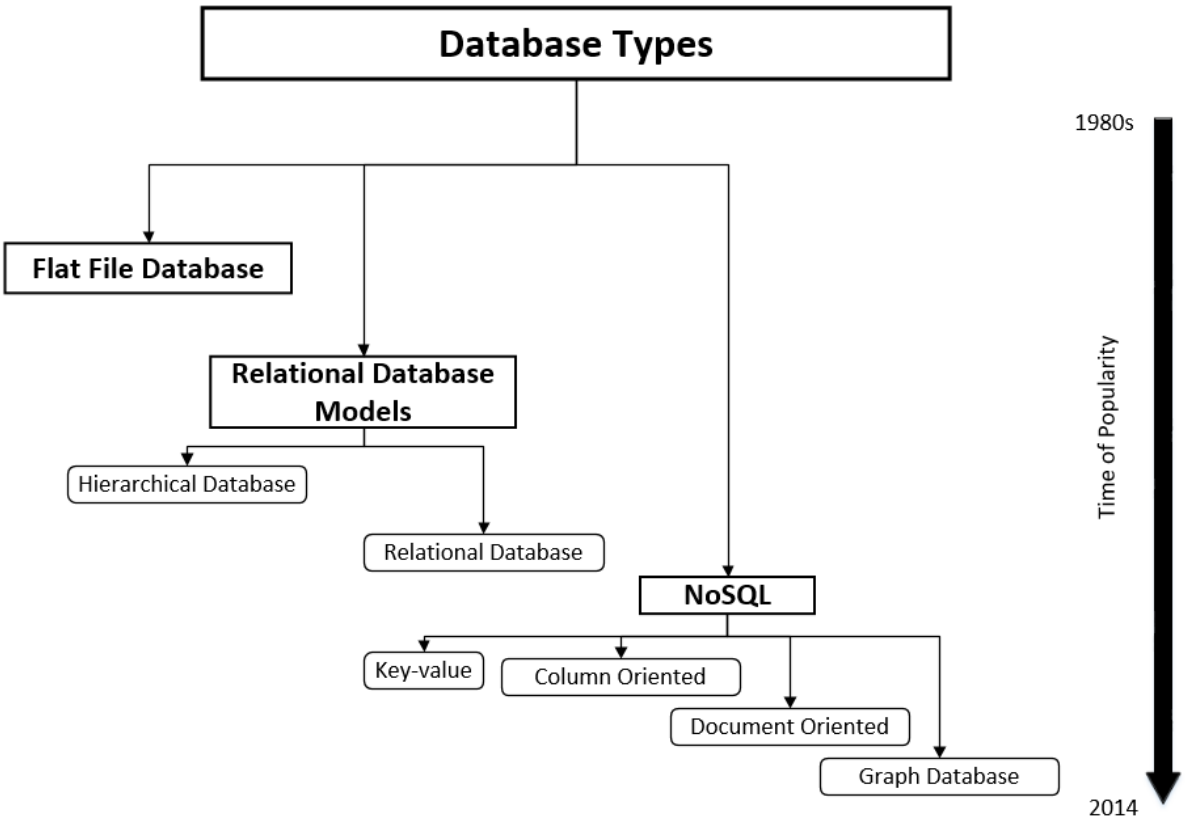


Figure 3-1: Database types (Srivastava, 2014)

3.2.1 Flat file database

The earliest databases were characterised by flat file database structures and only very primitive data analytics were possible on these databases. A flat file database can be thought of as a single large table that holds all the data relating to the particular database; these type of databases are rarely used in the modern day (Srivastava, 2014).

Flat file databases do not have the option to establish relationships between different sets of data and all logic relating to data handling and maintenance is the user's responsibility (Raparthi, 2014). A flat file database is in essence a text file where the lines of text refer to each row in a table (Raparthi, 2014).

The most popular databases today include relational and NoSQL databases (Srivastava, 2014). These databases are described in Sections 3.2.3 and 3.2.4. By referring to Figure 3-1, it can be seen that hierarchical databases, which includes network databases due to the similarities between the structures, were popular after the flat file database era and will be discussed in the following section.

3.2.2 Hierarchical and network database

According to Srivastava (2014), hierarchical databases are similar to the folder structures of computers – each folder can contain sub-folders and each sub-folder can hold more folders. Every node (sub-folder) will therefore have a single parent (folder). Figure 3-2 illustrates a diagram of a typical hierarchical database model.

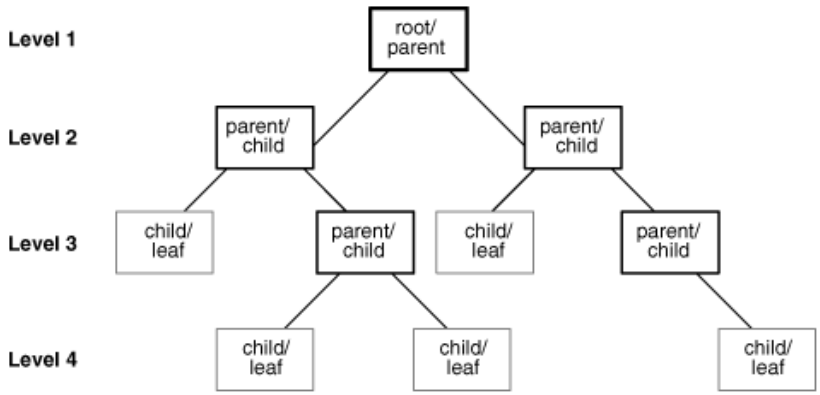


Figure 3-2: Hierarchical model example (Srivastava, 2014)

The limitation of hierarchical databases is that child nodes can only relate to one parent node with a one-to-one relationship, which is problematic in many real-life situations (Srivastava, 2014). If an employee, for example, has two managers, the hierarchical design falls short in fulfilling the design requirement. The limitations of hierarchical databases necessitated the need for relational databases.

This shortcoming of hierarchical databases was addressed by the introduction of network databases. Network databases are similar to hierarchical databases except that a single node can relate to many different nodes. The database includes sets of relationships where a set represents a one-to-many relationship between the owner and the member (Comphist.org, 2004). According to Samiksha (2016), the main limitation of network databases is that they are complex in terms of design and maintenance, which can be seen in Figure 3-3. This figure shows that each employee relates to each department and vice versa. This complexity stems from the lack of structural independence.

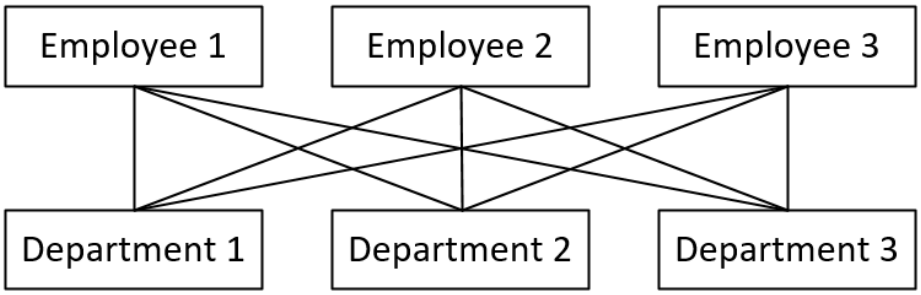


Figure 3-3: Network model example (Samiksha, 2016)

By referring to Figure 3-1, it can be seen that relational databases were popular after the hierarchical (including network) database era and will be discussed in the following section.

3.2.3 Relational database

The relational model was developed by Dr E.F. Codd (1970) to address the structural dependence and other limitations of network databases (Comphist.org, 2004; Samiksha, 2016). According to Kimball *et al.* (2008:611), a relational database is based on relationships between tables, which supports commands to manipulate and retrieve database data via Transact Structured Query Language. The relational database uses a series of tables with rows and columns to organise and store data. Relational databases allow users to join tables together based on a user-selected column, which is useful in data analysis and decisions on how to manipulate data. This study will

make use of a relational database as it can undoubtedly fulfil the database requirements of the experiment.

Figure 3-4 shows an example of a relational database model, which makes use of different relationships between multiple tables (also called nodes). Relationships that can be used in relational databases include: one-to-one, one-to-many, and many-to-many. An example of a one-to-one relationship can be seen in Figure 3-4 where Bus/Driver relationship is one-to-one; meaning one driver can only drive one bus and a bus can only have one driver. The Bus/Schedule relationship in Figure 3-4, is an example of a one-to-many relationship; this relationship indicates that one bus can have more than one schedule while one schedule relates to only one bus. Many-to-many relationships are usually divided into two one-to-many relationships with an associative table that can hold additional data about the relationship (Bentley & Whitten, 2007:276). This is shown in Figure 3-4 where the Route/Stop many-to-many relationship is resolved with the introduction of the associative table, Route_Stop. The Route_Stop table contains one record for each Route/Stop combination and can therefore contain additional information about this combination, such as the time of the stop in the route.

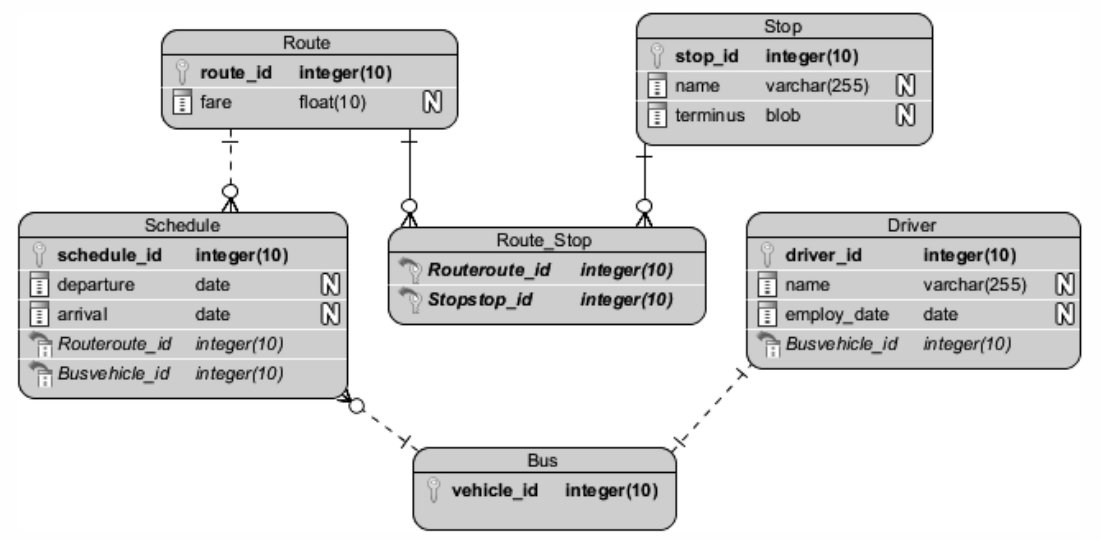


Figure 3-4: Entity relationship diagram (Visual Paradigm, 2011)

3.2.4 NoSQL databases

NoSQL, which means “Not only SQL”, is the most recent type of database and can be divided into four different subcategories, namely: key-value, column orientated, document orientated and

graph data stores (Noller, 2014). According to Cattell (2011:1), NoSQL usually is comprised of six main features:

1. Ability to horizontally scale simple operation throughput over many servers; meaning multiple servers can share query loads.
2. Can replicate and distribute data over many servers.
3. The call interface is simple, in other words, setting up queries to be executed from applications is an effortless task.
4. Shows weaker concurrency models than Atomicity, Consistency, Isolation, and Durability (ACID) transactions of relational databases. The weakest concurrency model will allow updates from different sources on the same segment of data.
5. More efficient use of RAM and storage than relational databases, which allows for improved query performance.
6. Can dynamically add new attributes (columns) to records, where attributes in relational databases are created and maintained by developers. With NoSQL databases, users can add attributes from an application level.

Different types of NoSQL databases exist, classified by how the data is stored and retrieved. The four NoSQL data store subcategories are summarised as follows:

1. **Key-value stores.** Values are stored in the system and each value is accompanied by an index (key) defined by a programmer which is used to find or refer to the value (Cattell, 2011:3).
2. **Column-oriented stores.** The system stores extensible records that can be partitioned vertically and horizontally across nodes (Cattell, 2011:3).
3. **Document-oriented stores.** These systems store documents that are indexed and make use of simple query mechanisms to retrieve information (Cattell, 2011:3).
4. **Graph database.** Graph structures are used for queries with nodes, edges and properties to represent and store data (Srivastava, 2014).

NoSQL databases are not within the scope of this study, therefore they are not discussed in greater detail; relational databases fulfil the database requirements of the experiment and analysis. The following section describes the role of DBMSs and how they relate to databases.

3.3 Database management system

A DBMS is defined by Chapple (2016) and Ramakrishnan and Gehrke (2003:4) as software designed to assist in the maintenance and utilisation of large collections of data which are stored in databases. A DBMS simplifies the process of organising, storing, retrieving and interacting with the data contained in a database (Seymour, 2010).

Seymour (2010) describes common functions performed by DBMSs:

- Provide developers with a simple user interface to create databases and ways to query and change the data in each database.
- Handle the technical tasks relating to data storage and retrieval.
- Authenticate users attempting to connect to the database while protecting sensitive data.
- Schedule database backups and redundancy.
- Create database log files that can be monitored by database administrators.
- Enforce rules, such as ensuring the correct data format is stored in fields.
- Calculate averages and aggregate data sets by using analysis and mathematical tools.
- Monitor and optimise performance by using tools that allow database administrators to improve DBMS performance.

There are many different packages of DBMS products available in the market. Organisations that develop DBMSs include Fujitsu, Hewlett-Packard, Hitachi, IBM, Microsoft, NCR Teradata, Oracle, Progress, SAS Institute and Sybase (Mohammed, 2016). Well known relational DBMS (RDBMS) packages include (Mohammed, 2016):

- *Microsoft SQL Server* – Well known for its native integration in the Microsoft Windows operating system (OS) environment.
- *Oracle* – Integrates well with Microsoft Windows and Linux OSs.
- *IBM DB2* – Compatible with Microsoft Windows, Linux and UNIX.
- *MySQL* – Compatible with various versions of Linux and UNIX as well as Microsoft Windows.
- *Microsoft Access* – Only compatible with Microsoft Windows.

In order to host a DBMS successfully via a certain platform, a number of DBMS-specific requirements need to be considered (Mullins, 2013:75). Platform, in this context, refers to a certain OS and hardware combination. These requirements include:

- **Software requirements.** The OS must be compatible with the chosen DBMS as well as any software that is intended to be used in conjunction with the DBMS. The OS and related software versions should also be considered to ensure that it is supported by the DBMS (Mullins, 2013:75; Microsoft, 2017).
- **Processing requirements.** The central processing unit (CPU) version and minimum clock speed need to align with the DBMS's minimum requirements (Mullins, 2013:76; Microsoft, 2017). In other words, the version of the CPU should be supported by the DBMS and the CPU's clock speed need to be greater than the minimum required clock speed of the particular DBMS.
- **Memory requirements.** These requirements relate to RAM, which will not only be used to process data, but also to store basic DBMS tasks and functionalities. Insufficient memory will drastically affect query performance (Microsoft, 2017). The reason for this is that cached data, which resides in RAM, is faster to access than data that resides in local storage such as hard drives. The loss in performance is caused by the automatic use of local storage as a supplement for insufficient RAM (Mullins, 2013:78).
- **Storage requirements.** A DBMS requires a certain amount of local storage for the installation itself. This does not include database data files. Each type, brand or version of a DBMS demands a certain amount of hard disk space and should be considered to ensure sufficient storage is available for the DBMS installation as well as the database files that will be created by the user through the DBMS's interface (Mullins, 2013:76; Microsoft, 2017).

In addition to these requirements, the purpose of the information system should also be considered in order to choose a DBMS that will fulfil the requirements of the information system (Mullins, 2013:76).

In this study, a DBMS needs to be selected that will be used in the experiments and fulfils the requirements described above. The following section covers factors that influence database performance and the performance of popular DBMS packages, which will enable the researcher to make an informed decision on the DBMS package to be used in this study.

3.4 Database management system performance evaluation

Database performance is an integral part of this study since it influences the outcome of the experiment conducted in the study. Mullins (2010) defines five factors that influence database performance namely; workload, throughput, resources, optimisation, and contention. These factors are described below (Charvet & Pande, 2003:5; Mullins, 2010).

The *workload* on a DBMS is the demand for database actions at any given moment, which consists of a combination of different types of database transactions. Workloads can fluctuate significantly over periods of time; these fluctuations can occur over intervals such as seconds, minutes, hours and days. In terms of this study, identical incremental workloads will be applied to all devices tested in the experiments.

Throughput refers to the computer's capability to process data. It is influenced by all factors relating to the computing system that affect data processing, including hardware and software. To ensure that throughput is not affected by software related variables in this study, identical software installations will be performed on the devices used in the experiment.

The hardware and software *resources* available to the system include all types of memory and how these are managed through the operating system. In this study, the software managing the primary memory, also known as RAM, will not be altered so that it is entirely handled and optimised by the operating system on each device.

All types of systems are *optimisable* in terms of factors such as system parameters, application design and databases. An advantage of DBMSs is that optimisation occurs internal to the DBMS. In this study, all optimisation parameters are kept constant across all devices relating to the experiment.

An example of *contention* is when two updates from different sources are applied to the same data segment. In the context of this study, contention may occur when two of the simulated users attempt to update the same field of a particular record. When set up to do so, RDBMSs ensure concurrency by locking pieces of data until the first transaction is complete. This means that as contention increases, locking follows, and results in a decreased throughput. As the extent of concurrency increases, the response time also increases. There is therefore a trade-off between concurrency and performance as depicted in Figure 3-5. Through this figure, it is shown that the number of concurrent users (x-axis) can increase, but at a certain number of concurrent users (decision point), the query response time (y-axis) will also increase. Figure 3-5 further shows that if concurrency controls (such as request queuing), are implemented at the decision point, the negative effect on response time is mitigated.

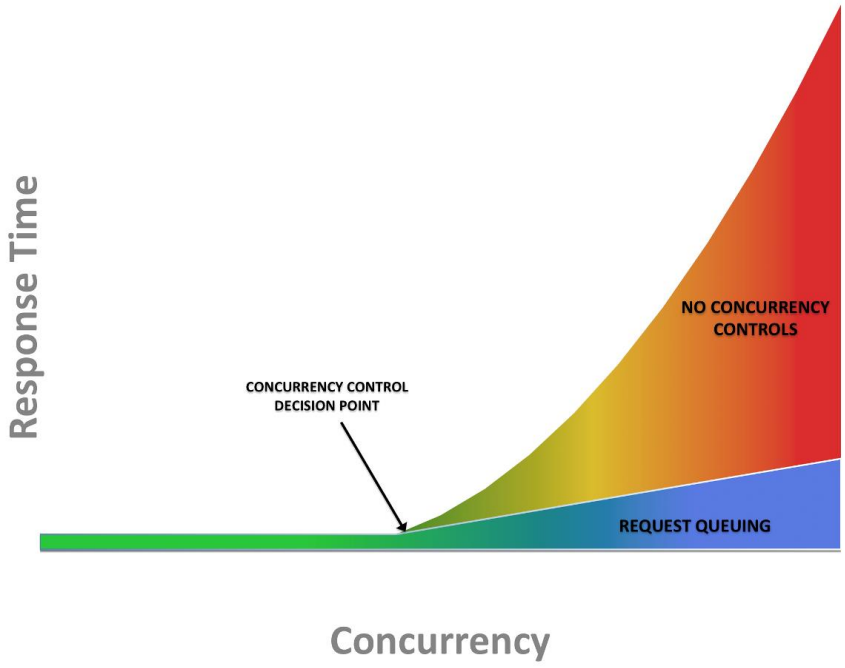


Figure 3-5: Concurrency and query response time (Soni, 2010)

A summary of the factors described above that influence database performance is shown in Table 3-2.

Table 3-2: Five factors that influence database performance (Charvet & Pande, 2003:5; Mullins, 2010)

Workload	Throughput	Resources	Optimisation	Contention
The demand on the DBMS such as: • Online transactions, • Batch jobs, • Ad hoc queries, • Business, intelligence queries & analysis, • Utilities, and • System commands.	The overall data processing capability. Factors that influence throughput: • Input/output speed, • Central processing unit speed, • Parallel capabilities, and • Efficiency of software.	The tools and hardware available to the system i.e.: • Primary memory, • Secondary memory, and • Cache controllers.	Primarily accomplished internal to the DBMS. Other factors to be optimised: • Database & system parameters, • Script coding, and • Application design.	Condition where two or more components are attempting to use a single resource in a conflicting way.

Bassil (2011:27) conducted a study on the performance of the most popular DBMSs and provided performance metrics on the average resource usage of the different DBMSs by running a standard set of simulations on each. The five DBMSs include: SQL Server, Oracle 10g, IBM DB2, MySQL 5.0 and Microsoft Access. This paper allows the researcher to make an informed decision in choosing a DBMS. All MPDBs will host the same DBMS to eliminate differences in resource usage between different types of DBMSs.

The performance metrics obtained from Bassil (2011:27) guide the researcher in terms of the expected resource requirements of different DBMSs. Figure 3-6 shows the average CPU utilisation and Figure 3-7 shows the average memory usage of each DBMS as results from the study of the journal article. Microsoft Access used the least resources in both figures, while MySQL utilised the second least resources. The Microsoft SQL Server had an average of 27% CPU utilisation and 13.2MB average memory usage. IBM's DB2 average CPU utilisation was 29% and 22.6MB average memory usage. Oracle 10g was the most resource intensive RDBMS in the study (Bassil, 2011:27).

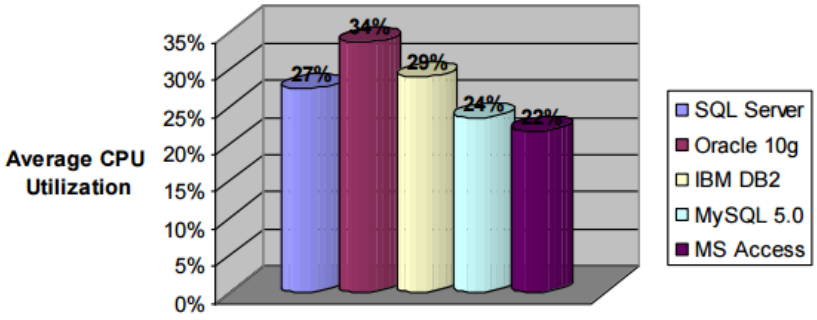


Figure 3-6: Average CPU utilisation (Bassil, 2011:27)

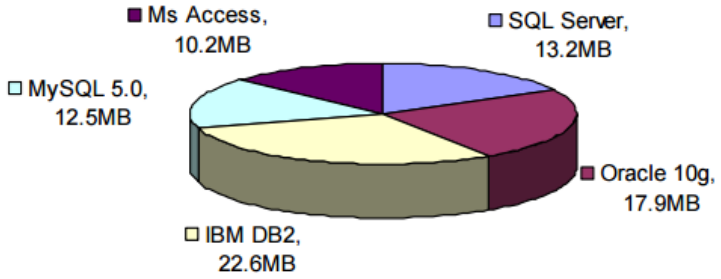


Figure 3-7: Average memory usage (Bassil, 2011:27)

A requirement of this study is that the chosen DBMS must be used on all devices and therefore has to be compatible. As stated in Chapter 4 Section 4.4, Linux is the chosen operating system in this study; the DBMS must therefore be compatible with Linux. The reason for considering the performance of DBMSs is because of the lower hardware capabilities of MPDBs when compared to commodity PCs. From Figure 3-6 and Figure 3-7, it can be seen that Microsoft Access has the lowest resource usage and would be an ideal candidate for this study. The unfortunate

shortcoming is that Microsoft Access is only compatible on Microsoft operating systems (Algonquin College, 2012). The DBMS with the second lowest resource usage is MySQL which is compatible on various versions of Linux (MySQL, 2017a). MySQL is therefore the DBMS used in this study.

3.5 MySQL database management system

MySQL is a widely used high performance, easy to use, and reliable open source RDBMS, used to store and retrieve data from databases (Heng, 2017; Oracle.com, 2017). It also runs on many platforms, such as Windows and many different distributions of Linux (Suehring, 2002:11).

Furthermore, MySQL provides high quality support for the products it offers and has presentations, case studies, forums and webinars available on the official website (MySQL, 2017b). Many large organisations, such as Facebook and Google, make use of MySQL due to the benefits listed above as well as the savings in total cost of ownership (TCO) compared to competitors (MySQL, 2017b). The TCO for leading DBMS providers over three years is illustrated in Figure 3-8. The TCO is calculated based on the system configuration and products defined in Table 3-3. From Figure 3-8, it can be seen that the TCO for MySQL over three years is significantly lower than Microsoft SQL Server and Sybase ASE.

Table 3-3: Product and system configuration for TCO calculation (MySQL, 2017c)

Products	
MySQL	Enterprise Edition
Microsoft SQL Server	Enterprise Edition
Sybase ASE	Enterprise Edition
System Configuration	
Number of users	Unlimited
Application types	No restrictions
Database use	No restrictions
Database size	No restrictions
Total servers	4
Sockets per server	4
Cores per socket	8
Total sockets	16
Total cores	128

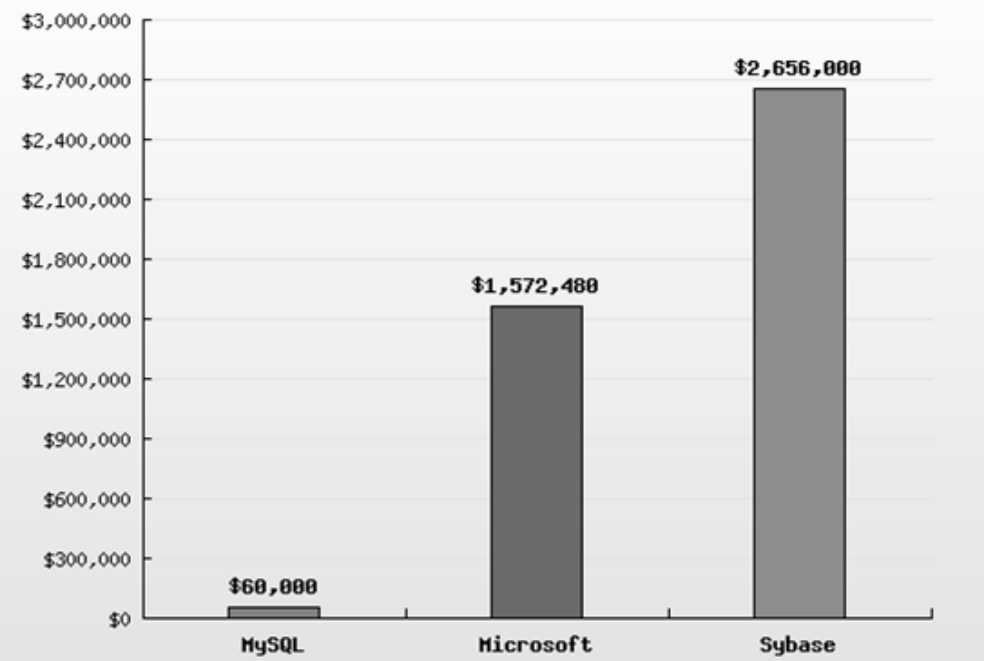


Figure 3-8: Three year DBMS TCO (MySQL, 2017c)

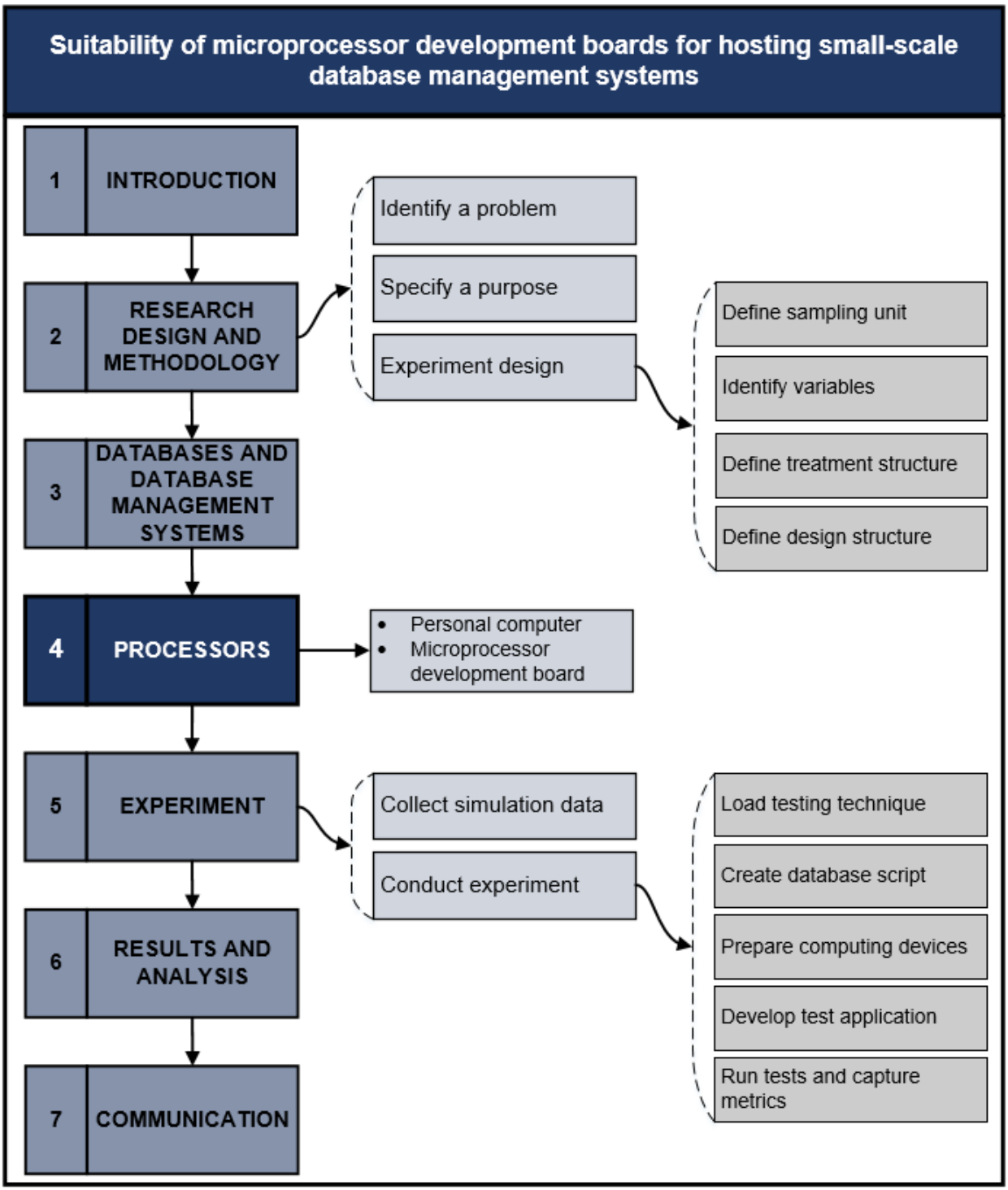
3.6 Conclusion

The purpose of this chapter was to gain an understanding of the different terms and concepts that relate to databases and DBMSs. These concepts included database, DBMS and DBMS performance evaluation. This objective was met through the literature provided in this chapter.

An understanding of databases, DBMSs and different types thereof were developed to guide the researcher through the empirical work. The performance evaluation of DBMSs made use of an article to compare resource usage of five different DBMSs, namely, Microsoft Access, MSSQL Server, Oracle 10g, IBM DB2 and MySQL 5.0. Based on the information from the performance evaluation, it was concluded that MySQL is the ideal DBMS to be used in this study.

The theoretical objective that was achieved in this chapter was to gain an understanding of DBMSs. The literature review on databases and DBMSs, Chapter 3, is hereby concluded and is followed by Chapter 4 in which concepts relating to the literature of MPDBs and commodity PCs are reviewed.

(P*AGE INTENTIONALLY LEFT BLANK)*


Suitability of microprocessor development boards for hosting small-scale database management systems
1
INTRODUCTION
2
RESEARCH DESIGN AND METHODOLOGY
Identify a problem
Specify a purpose
Experiment design
Define sampling unit
Identify variables
Define treatment structure
Define design structure
3
DATABASES AND DATABASE MANAGEMENT SYSTEMS
4
PROCESSORS
Personal computer
Microprocessor development board
5
EXPERIMENT
Collect simulation data
Conduct experiment
Load testing technique
Create database script
Prepare computing devices
Develop test application
Run tests and capture metrics
6
RESULTS AND ANALYSIS
7
COMMUNICATION

CHAPTER 4: PROCESSORS

4.1 Introduction

In order to determine whether microprocessor development boards (MPDBs) are a viable alternative to commodity personal computers (PCs) for hosting small-scale database management systems (DBMSs), a literature review of processors is required. Each processor's accompanying hardware and resources, such as the central processing unit (CPU) and random access memory (RAM), form an important part of the literature review in this chapter.

This chapter is comprised of discussions on the terms and concepts relating to processors. In this study, processors and the accompanying hardware, will be tested to determine the load that each device is capable of dealing with, relative to other devices in the study. This comparison will allow the researcher to address the objectives of this study.

In this chapter on processors, MPDBs are discussed in terms of four different MPDB brands (§4.2). Commodity PCs are then discussed (§4.3), followed by computing technologies comparison and selection (§4.4). In the comparison, the described MPDB brands and a low-range PC are compared and MPDBs are selected for the experiment. A load testing technique (§4.5) suggested by Meier *et al.* (2007) is explained. Finally, this chapter is concluded (§4.6).

4.2 Microprocessor development board

A MPDB is a printed circuit board that contains a microprocessor and only the minimal support logic needed to use the microprocessor (Kant, 2007:17). As mentioned in Chapter 1, there are many different uses for MPDBs, such as building and automation, three-dimensional (3D) printing, media centre hosts, and networking and servers (Raspberrypi.org, 2015d). This study attempts to expand the uses of MPDBs by configuring the devices to act as DBMS hosts. This section will describe some of the popular MPDB brands in South Africa and the specifics of each brand; they include Raspberry Pi, BeagleBone Black, Intel Edison Arduino and PCDuino.

4.2.1 Raspberry Pi

The Raspberry Pi Foundation is an educational charity organisation based in the United Kingdom. Their goal is to use MPDBs to advance the education of adults and children in the field of

computers and computer science. In 2011, Raspberry Pi initiated its first mass production of MPDBs for the consumer market (Raspberrypi.org, 2015b).

A Raspberry Pi is a MPDB and is defined as a low cost, credit card sized computer (Raspberrypi.org, 2015a). The Raspberry Pi can plug into a computer monitor or a TV and it uses a standard universal serial bus (USB) keyboard and USB mouse. It can be used as a normal computer and allows activities such as Internet browsing, playing high-definition videos, creating spreadsheets, performing word processing, and playing games. Figure 4-1 shows a typical Raspberry Pi MPDB.

Raspberry Pi's latest model is the Raspberry Pi 3 Model B, as seen in Figure 4-1. The Raspberry Pi 3 (RPi3) has four USB ports, 40 general-purpose input/output pins (used to connect peripherals for example), full high definition multimedia interface (HDMI) port, 3.5mm audio jack, Ethernet port, display interface, camera interface, micro secure digital (SD) card slot, and VideoCore IV 3D graphics core (Raspberrypi.org, 2015c).

Figure 4-1: Raspberry Pi 3 Model B (Raspberrypi.org, 2015c)

The first Raspberry Pi in production and available to the public was the Raspberry Pi 1 Model B which was released in June 2012 (eLinux.org, 2017). Table 4-1 show the evolution of the Raspberry Pi models since the Raspberry Pi 1 Model B. The release dates and hardware specifications for each model are recorded in this table. From Table 4-1, it can be seen that Raspberry Pi 1 has a 700MHz single core CPU, Raspberry Pi 2 has a 900MHz quad core CPU and Raspberry Pi 3 has a 1.2GHz quad core CPU. The Raspberry Pi 1 models with a plus (+) sign, feature nine more GPIO pins to which peripherals can be connected. Note that Model A and Model A+ do not have adapters that allow them to connect to a network, while all other models contain an Ethernet port with data transfer speeds of up to 100Mbps.

Table 4-1: Raspberry Pi model evolution (Lyons, 2015; eLinux.org, 2017)

	Raspberry Pi 1 Model B	Raspberry Pi 1 Model A	Raspberry Pi 1 Model B+	Raspberry Pi 1 Model A+	Raspberry Pi 2 Model B	Raspberry Pi 3 Model B
Release date	June 2012	February 2013	July 2014	November 2014	Feb 2015	Feb 2016
CPU	ARMv6 700MHz single core	ARMv6 700MHz single-core	ARMv6 700MHz single-core	ARMv6 700MHz single-core	ARM Cortex-A53 900MHz quad core	ARM Cortex-A53 1.2GHz quad core
RAM	512MB	256MB	512MB	256MB	1GB	1GB
Secondary memory	SD card slot	SD card slot	SD card slot	SD card slot	MicroSD card slot	MicroSD card slot
Network interface	Ethernet up to 100Mbps	None	Ethernet up to 100Mbps	None	Ethernet up to 100Mbps	Ethernet up to 100Mbps
USB ports	2	1	4	1	4	4
GPIO pins	8	8	17	17	40	40

4.2.2 BeagleBone Black

The BeagleBoard.org Foundation is a non-profit organisation based in the United States of America. The purpose of the company is to provide education and promotion in the design and use of open-source software and hardware in embedded computing (Beagleboard.org, 2014b). The initial purpose of the development of BeagleBoards was to improve support for ARM (Advanced RISC [Reduced Instruction Set Computer] Machine) devices through the use of Linux distributions. The focus has since shifted to enabling simplified physical computing on advanced graphical user interface enabled and/or networked-enabled devices. These devices are characterised by simple learning experiences and support development environments that are familiar to developers. The development environments include Ubuntu, QNX, Windows Embedded, Android, as well as web tools (Beagleboard.org, 2014b). The BeagleBone Black (BBB) runs a Linux operating system (OS) and is a community-supported development platform. Like the RPi3 and PCDuino, it has an HDMI port, Ethernet port and USB ports (Beagleboard.org, 2014a). Figure 4-2 shows the BeagleBone Black MPDB.

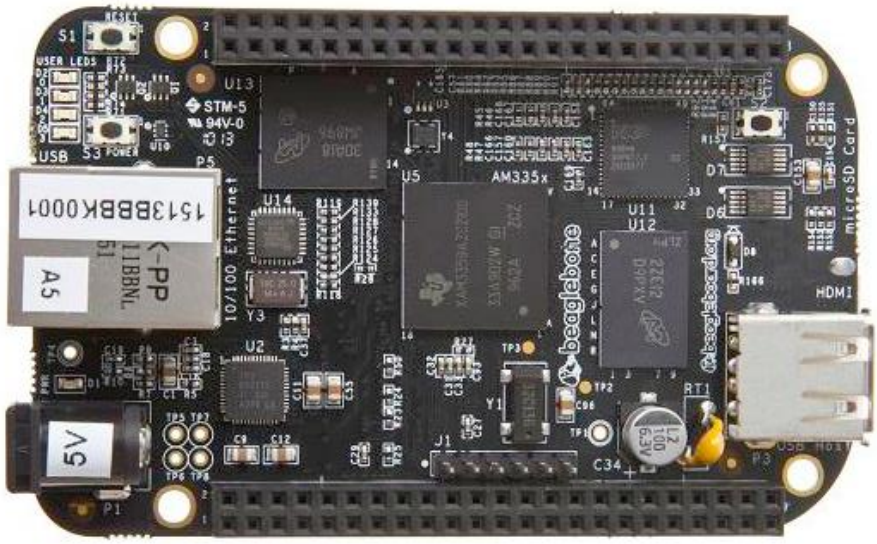


Figure 4-2: BeagleBone Black board (Beagleboard.org, 2014a)

4.2.3 Intel Edison Arduino

The Intel Edison compute module is a system on a chip (SoC) that hosts an Intel Atom 500Mhz dual core CPU as well as a microcontroller (Intel Corporation, 2015:9). The SoC module was designed by Intel Corporation for the use with Internet of Things and wearable technology products. Figure 4-3 shows the Intel Edison SoC module, which is 35.5 × 25 × 3.9 mm in size.

Figure 4-3: Intel Edison compute module (Intel Corporation, 2017)

The compute module requires a breakout board, for the purpose of this study, which allows the developer to access to the module via a PC in order to install an OS and supply electricity to the module (Arduino, 2015; Intel Corporation, 2015). Note that the Intel Edison is the only device in

this study that requires a breakout board. The Intel Edison compute module can be attached to an Arduino breakout board to allow for the aforementioned communication with the module from a PC. This study names the assembled combination of the compute module and the breakout board, Intel Edison Arduino (IEA), which can be seen in Figure 4-4. The IEA has on-board WiFi and does not have a HDMI port, an audio jack or an Ethernet port.

Figure 4-4: Intel Edison Arduino (Arduino, 2015)

4.2.4 PCDuino

PCDuino is an open source MPDB designed, manufactured, and sold by a company called LinkSprite. The PCDuino board is LinkSprite's flagship product (Linksprite.com, 2015).

Its official website defines the PCDuino board as a mini PC platform that hosts an OS very similar to Ubuntu and Android (Pcduino.com, 2015). As can be seen from Figure 4-5, the PCDuino supports network access through an Ethernet port, has two USB ports for a keyboard and mouse, and HDMI output for a computer monitor. The PCDuino MPDB is very similar to the RPi3 in terms of looks and capabilities.

Figure 4-5: PCDuino board (Pcduino.com, 2015)

The discussion on the MPDBs is hereby concluded; the next section continues to commodity PCs, followed by a comparison in terms of price and specification between a commodity PC and the described MPDB brands.

4.3 Commodity personal computer

A computer can broadly be defined as any class of human-made device or system that is able to modify data in some meaningful way (Linux Information Project, 2006). According to Sipral (2007:18), a commodity PC has the following typical components: a CPU, which is responsible for the computer's arithmetic, logic, and control circuitry on an integrated circuit; primary memory, which includes RAM and secondary memory (hard disks). A computer also has various typical input and output devices, such as a display screen, a mouse, a keyboard, and a printer; and an Ethernet port which is important for network connections to and from the computer, allowing computers to share data (on a network) between different computers (Sipral, 2007:24).

Synonyms of the word "computer" include (this list is not exhaustive): personal computer, laptop, microprocessor development board, mac, mainframe, and microcomputer. Each synonym typically refers to a characteristic of the computer, i.e. size, portability or performance.

According to Allan (2001:2/7) a prototype of the first personal computer was developed at the Massachusetts Institute of Technology in 1962 and 16 units were produced by 1963 at an approximate cost of $32 000 per unit. Since then the technology has developed rapidly and today computers are mainly used for (Sipral, 2007:27):

- Home use, for purposes that include playing games, Internet banking and browsing the Internet.
- Educational software packages, such as statistical analysis software.
- Automation of production systems, i.e. computer-controlled robots assembling cars.
- Computer aided design (CAD), to produce exact specifications and detailed drawings.
- Stock control systems and pay point terminals.
- Writing computer applications using a programming interface.

The extent to which personal computers are used may hold the following advantages (Consortini, 2013:3; Ramey, 2013):

- *High speed* – computers can perform calculations for high volumes of data in a short time.
- *Accuracy* – computers will perform programmed tasks without any error.
- *Storage capability* – computers can store large amounts of data in different formats.
- *Diligence* – computers execute tasks thoroughly and continuously.
- *Versatility* – computers are not limited to perform only in certain areas or fields.
- *Reliability* – modern electronic equipment is designed for reliability and is durable.
- *Automation* – computers can perform programmed tasks automatically without any human interaction.
- *Reduction in paper work* – computers in an organisation lead to reduced paper work, which also improves process time.
- *Reduction in cost* – although computers require a larger initial investment cost, they reduce individual transaction cost.

Personal computers also have certain disadvantages as described by Consortini (2013:9) and Ramey (2013):

- *Require training* – not everyone is competent to use a computer and companies might need to spend additional money on training or incur additional costs to hire professionals.
- *Computer crime* – many groups with malicious intents target companies to access business or confidential information stored in a computer network.
- *Require additional infrastructure* – infrastructure costs can be high when integrating different computers in a business.

- *Replaces human labour* – computers automate many tasks or transactions in a business that previously required human labour, which leads to job losses and unemployment.

4.4 Computing technologies comparison and selection

This section will provide a price, hardware and software specification comparison between the computing technologies described in Section 4.2 and 4.3. The comparison, as illustrated in Table 4-2, is in terms of price, CPU, RAM, secondary memory, network interface and OS. The prices (inclusive of VAT) are illustrative of prices prevalent in 2017 and are only indicated in order to show the proportional difference in price between the categories of computing technologies. It is also important to note that the prices do not include accessories such as screens and keyboards. This is true for the MPDBs, as well as the PC. The OSs listed in Table 4-2 are not exhaustive, and there are some more uncommon compatible OSs available on the market.

Table 4-2: Computing technologies comparison

	Low-Range Commodity PC	Raspberry Pi 3 Model B	BeagleBone Black	Intel Edison Arduino	PCDuino 3
Price	R 4 299.00	R 646.35	R 1 173.32	R 2 419.14	R 1 777.50
Price Ratio	100%	15.03%	27.29%	56.27%	41.35%
CPU	Intel Pentium 3.50GHz dual core, 3M cache	ARM Cortex-A53 1.2GHz quad core	ARM Cortex A8 1GHz single core	Intel Atom 500MHz dual core	ARM Cortex A7 1GHz dual core
CPU Threads	4	4	1	2	2
RAM	8GB	1GB	512MB	1GB	1GB
Secondary Memory	1TB hard drive SATA3	MicroSD card slot	4GB flash memory	4GB flash memory, MicroSD card slot	4GB flash memory, MicroSD card slot
Network Interface	Ethernet up to 1Gbps	Ethernet up to 100Mbps	Ethernet up to 100Mbps	WiFi only up to 150Mbps	Ethernet up to 1Gbps
OS	Windows Linux (Debian) Chrome OS Mac OS	Linux (Debian) Raspbian OpenELEC Snappy Ubuntu Core	Linux (Debian) Ubuntu Android	Linux (Debian)	Linux (Debian) Ubuntu Android
Price Source	Evetech (2017)	RS Components (2017a)	RS Components (2017b)	RS Components (2017c)	Riecktron (2017)
Specification Source	Evetech (2017)	Raspberrypi.org (2015c)	Beagleboard.org (2014a)	Intel Corporation (2015); Intel Corporation (2017)	Pcduino.com (2015)

Table 4-2 reveals that MPDBs can be acquired at a fraction of the price of a PC while still being able to perform most of the same functions PCs can perform. Most MPDBs are not much larger

than a credit card, whereas PCs are much larger. The performance of MPDBs is the only downside when compared to PCs. However, if the price comparison is brought into the equation, the decision of which one to purchase depends on the computer's intended use. In terms of prices from Table 4-2, it can be seen that the RPi3's price amounts to 15.03 percent, the BBB 27.29 percent, the IEA 56.27 percent, and the PCDuino 41.35 percent of the price of the low-range commodity PC.

The common denominator in terms of OS between these devices is Linux Debian and is therefore the chosen OS to be used with all sampling units during the experiments in this study. The devices in Table 4-2 are capable of running Linux and connecting to a network; these devices are therefore viable options to be used in this study.

The low-range commodity PC servers as a benchmark and the MPDB sample for this study includes:

- RPi3;
- BBB; and
- IEA

The PCDuino does not form part of the sample due to financial restrictions in the study; the three MPDBs that were chosen do however provide sufficient diversification in MPDB brands and the validity of the experiments is not threatened by excluding the PCDuino from this study. Also note that the IEA has been discontinued, announced by Intel in 2017; the IEA is however still available until the end of 2017 (List, 2017).

4.5 Load testing

Load testing assists in identifying the maximum operating capacity of the device hosting the DBMS and any bottlenecks that might have a negative impact on performance (Meier *et al.*, 2007). Load testing allows the researcher to determine to what extent database load can be dealt with by the computing platform in question and, in the case of this study, whether the test results indicate sufficient MPDB load capability to host a small-scale DBMS. There are many tools available for load testing on the market, some are open source, and others are more sophisticated load testing applications that can be purchased. According to Meier *et al.* (2007), some DBMSs have load test result logs integrated in the DBMS that can be used to analyse the extent of the load after the tests have been run. In this study, the load testing is controlled and results are captured using the Multi-Client Simulator application developed by the researcher.

Figure 4-6 shows a load testing technique developed by Meier *et al.* (2007), the eight steps are described below. The technique is adapted to the study in Chapter 2 (§2.6.5 – Step 1).

Step 1 – Identify performance acceptance criteria. This criterion can include factors such as response time, throughput, resource utilisation, maximum user load, and business related metrics such as business volume at normal and peak values.

Step 2 – Identify key scenarios. These scenarios refer to expected requests or operations that will be sent to the database, including those that are considered high risk, the most common requests, and those with a significant performance hit.

Step 3 – Create a workload model. The workload model refers to the way users apply load to the system. The model takes into account user "think" time or user delay. This model can be used to create artificial workloads at any time, with slight alterations that are well controlled. The goal is to generate workloads that can be used to evaluate performance, and the artificial workload should be similar to those that occur in practice on real systems.

Step 4 – Identify target load levels. Target (or minimum required) load levels to be applied to each workload identified in the workload model. Load levels are identified in order to predict or compare a variety of production load conditions through tests.

Step 5 – Identify metrics. Identify the metrics that are most relevant to the specific performance objectives of the DBMS. If too many metrics are collected, it can result in an unwieldy analysis, as well as negatively affect actual performance of the DBMS.

Step 6 – Design specific tests. With the use of the key scenarios, key metrics, and the workload model, tests can now be designed. Each test typically has a different goal, collects unique data, includes various scenarios, and has diverse target load levels. The tests should be designed while keeping the purpose of the load testing as a whole in mind – to evaluate database load with different scenarios.

Step 7 – Run tests. Ensure that the load simulation meets the test design and reflects the production environment as closely as possible while noting all differences between the test and production environments.

Step 8 – Analyse the results. Analyse the measured metrics to identify potential bottlenecks that may indicate a trend toward or away from the performance objectives.

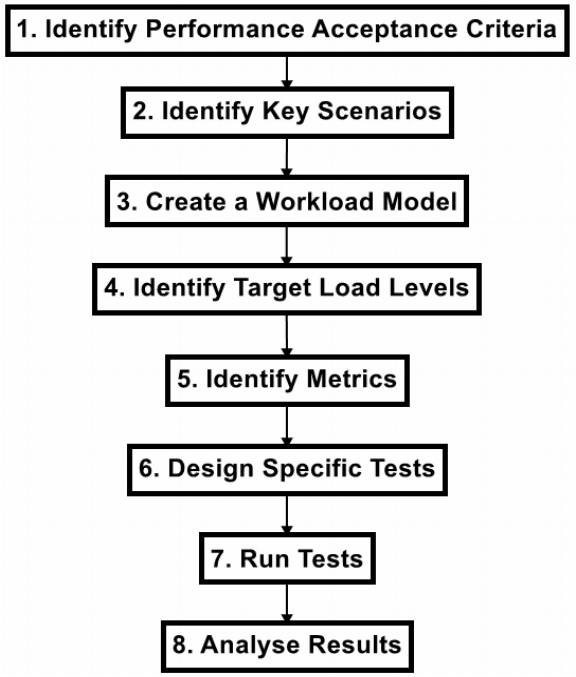


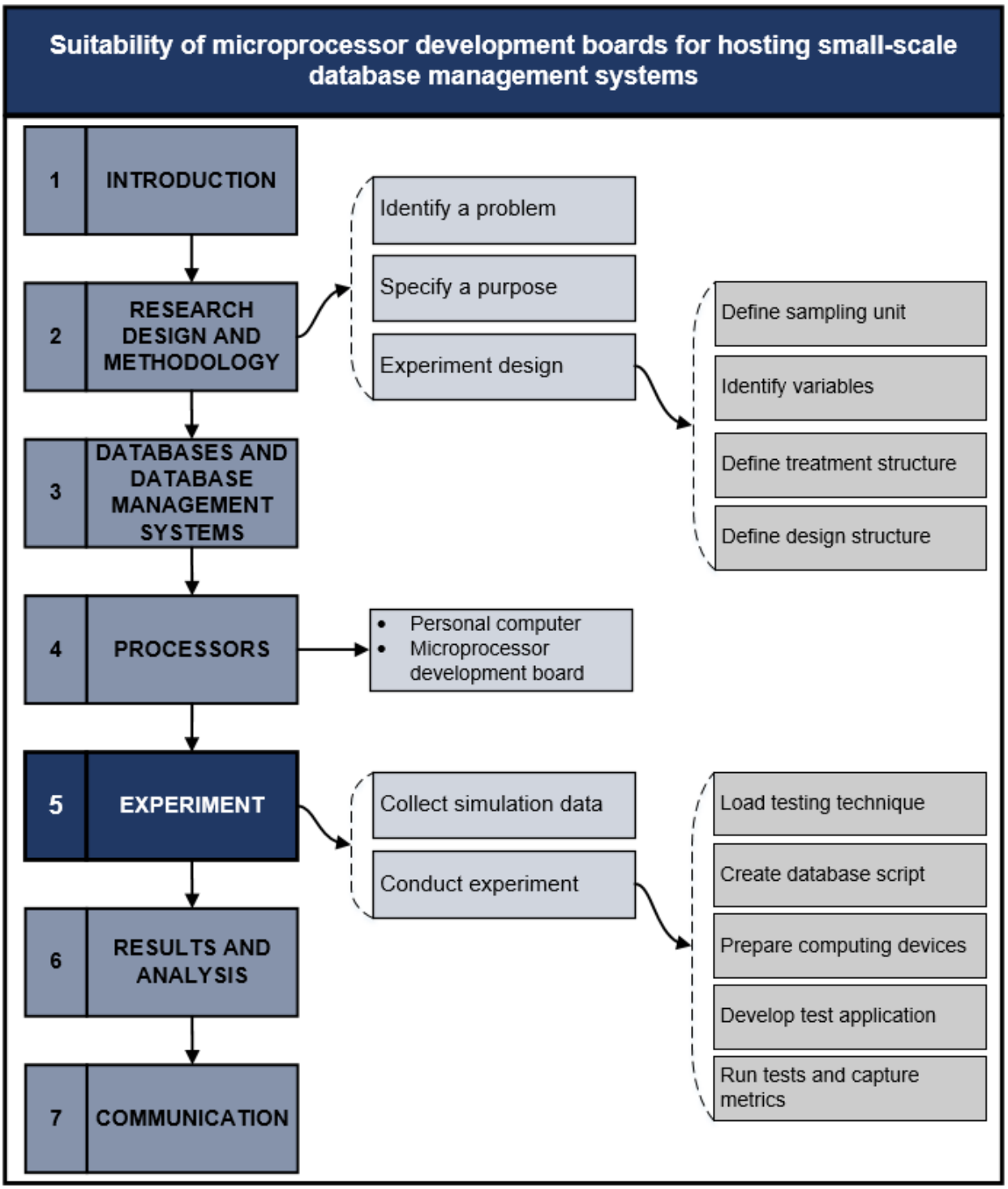
Figure 4-6: Load testing technique (Meier *et al.*, 2007)

4.6 Conclusion

The purpose of this chapter was to gain an understanding of commodity PCs and MPDBs, and to compare them to one another. This objective was achieved through the theoretical analysis of MPDBs, commodity PCs and the load testing technique of computing devices.

Commodity PCs, MPDBs and some of the different available MPDB brands were discussed in terms of hardware, software, prices and uses. The MPDBs that were discussed in this chapter include: RPi3, BBB, IEA, and PCDuino. The comparison between the low-range commodity PC and the MPDBs revealed that MPDBs have an anticipated lower performance when compared to commodity PCs, based on hardware specifications. MPDBs are however sold at lower prices. The MPDB sample for this study includes the RPi3, BBB and IEA. The load testing technique created by Meier *et al.* (2007) provides eight steps for a load testing technique, which will guide the researcher in the determination of the suitability of MPDBs for hosting small-scale DBMSs.

The literature review on processors, Chapter 4, is hereby concluded and is followed by the experiment chapter in which the researcher describes the specifics of the experiment.


Suitability of microprocessor development boards for hosting small-scale database management systems
1
INTRODUCTION
2
RESEARCH DESIGN AND METHODOLOGY
Identify a problem
Specify a purpose
Experiment design
Define sampling unit
Identify variables
Define treatment structure
Define design structure
3
DATABASES AND DATABASE MANAGEMENT SYSTEMS
4
PROCESSORS
Personal computer
Microprocessor development board
5
EXPERIMENT
Collect simulation data
Conduct experiment
Load testing technique
Create database script
Prepare computing devices
Develop test application
Run tests and capture metrics
6
RESULTS AND ANALYSIS
7
COMMUNICATION

CHAPTER 5: EXPERIMENT

5.1 Introduction

In order to determine the extent to which microprocessor development boards (MPDBs) are able to host small-scale database management systems (DBMSs) compared to commodity personal computers (PCs), an experiment is performed. This chapter covers the device preparation, setup and experiment details followed to test hypothesis of the study.

Two empirical objectives formulated in Chapter 1 are addressed in this chapter. The first objective is to set up a testing environment by installing a DBMS on each MPDB. The second objective is run simulations on each type of technology while recording performance metrics relating to the process.

An application, the Multi-Client Simulator (MCS), was developed by the researcher and is used to successfully conduct the experiments and record performance metrics. A pilot is firstly performed to evaluate the feasibility of the experiment. The pilot and experiment each requires a separate version of the MCS application. This chapter also provides empirical detail relating to the load testing technique, designed in Chapter 2 (§2.6.5), followed by both the pilot and the experiment. Finally, pilot and experiment metrics are imported into a data warehouse designed by the researcher in order to prepare the data for analysis.

The following sections are covered in this chapter: experiment approach (§5.2), generating simulation data (§5.3), the load testing technique (§5.4), the creation of the database script (§5.5), computing device preparation in terms of hardware and software (§5.6), test application development (§5.7), running tests and capturing metrics (§5.8), data preparation for analysis (§5.9), and finally, this chapter is concluded (§5.10).

5.2 Experiment approach

To perform the pilot and the experiment, a high end PC is used to generate a database script file and develop the MCS application. This PC is referred to as the Windows development PC or Windows PC.

The purpose of the MCS is to simulate multiple clients (or threads) connecting simultaneously to a device and applying load through the execution of multiple database queries per thread according to the load test design. The MCS captures metrics that relate to each device and its

performance. Device resources are captured throughout the experiment by means of an application named Dstat. This application is a Linux based resource statistics tool that allows the user to view or log system resources in real time in order to use these at a later stage (Wieërs, 2016).

The pilot is conducted with MCS Version 1, while the experiment is performed with the use of MCS Version 2. A clear distinction between the pilot and the experiment is made throughout the chapter. The purpose of the pilot is to enable the researcher to evaluate the feasibility of the experiment.

The devices on which the experiment is conducted are referred to as devices or computing platforms/technologies. These devices, introduced in Chapter 1 and discussed in Chapter 4, include the following:

- Raspberry Pi 3 (RPi3);
- BeagleBone Black (BBB);
- Intel Edison Arduino (IEA); and
- the commodity PC (also referred to as PC).

A clear understanding of the term “client” used in the current and subsequent chapters is necessary to ensure that the experiment, results and their analysis are interpreted correctly. Simulated clients are referred to as “threads” or “clients” interchangeably. This has to be distinguished from records in the CLIENT table which are referred to as “CLIENT table records”.

The complete code for the pilot and experiment applications, as well as the data generator script can be viewed in Appendix D.

5.3 Generation of simulation data

The simulation data is generated with a Microsoft SQL (MS SQL) script acquired from Ganaye (2012). The script creates a database stored procedure (SP) and executes it with two parameters, namely number of clients and number of orders. Upon the execution of the SP, five related tables are created and filled with data generated from the SP.

The amount of data that the script generates depends on the values provided in the number of clients and number of orders parameters. The parameter values used for the pilot and experiment are 20 000 for clients and 5 000 000 for orders. This simulated number of orders represents an

information system that creates 1984 orders per day, over 252 workdays per year, over a 10-year period. For the purpose of the study, the number of orders is exaggerated when compared to a realistic information system used in a small-scale business. The database is classified as a medium-scale database according to the “number of records” database scale classification method described in Chapter 3 Table 3-1. The table names with the number of records in each table are listed below:

- CLIENT – 20 000 records;
- OCCUPATION – 330 records;
- ORDER – 5 000 000 records;
- ORDERLINE – 32 611 992 records; and
- PRODUCT – 1 438 records.

The Entity Relationship Diagram (ERD) design of the simulation data is shown in Figure 5-1 below. Each CLIENT table record relates to one OCCUPATION record, and zero to many ORDER records. Each ORDER table record relates to one to multiple ORDERLINE records, while each PRODUCT table record can relate to zero to many ORDERLINE records.

5.4 Load test design

This section details the empirical aspects of the load testing technique described in Chapter 2 Section 2.6.5 Step 1, which is followed by the researcher. This technique is the foundation of the experiment and therefore serves as a guide throughout the pilot and the experiment.

5.4.1 Identify performance acceptance criteria

The performance acceptance criteria identified in Chapter 2 (§2.6.5 – Step 1) requires each MPDB to be capable of processing data in a DBMS similar to commodity PCs. The outcome will be evaluated by analysing the results from the experiment through the hypothesis test in Chapter 6 (§6.5).

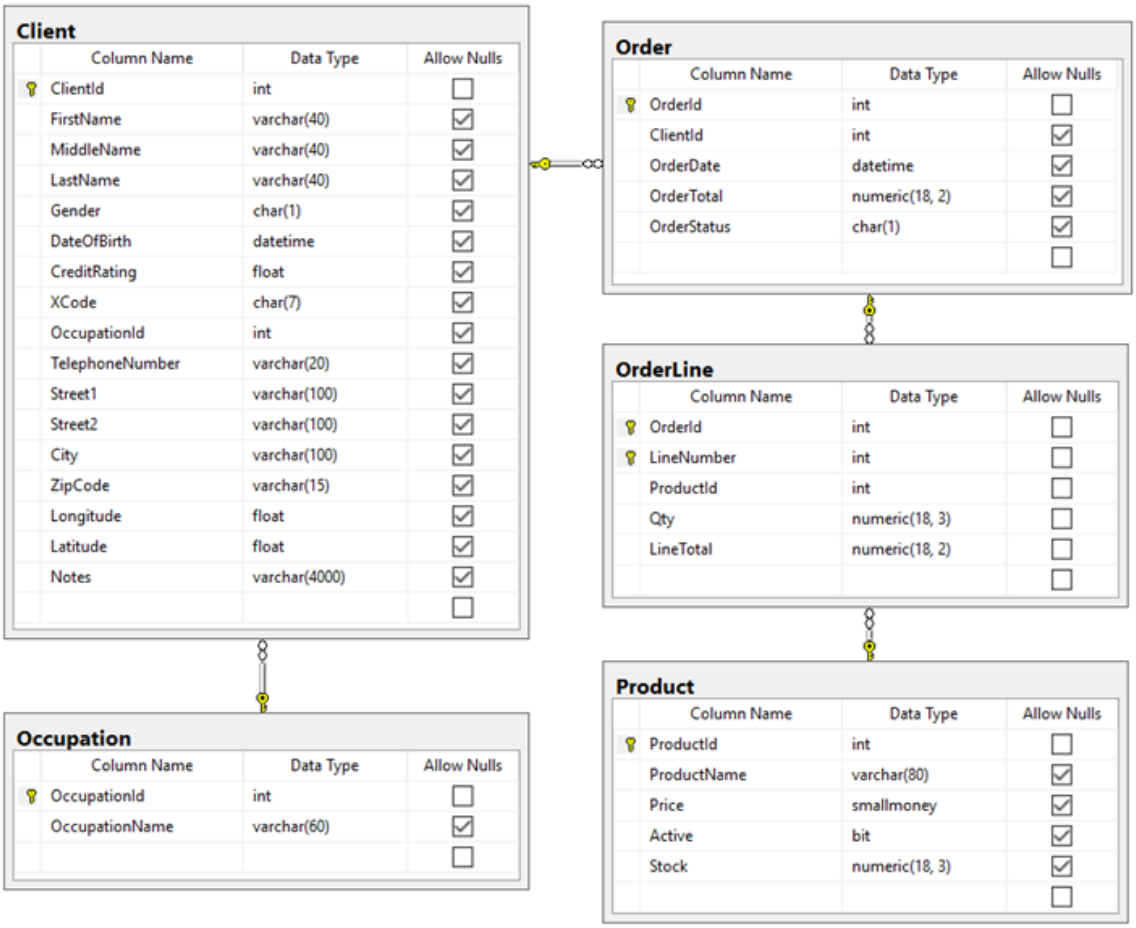


Figure 5-1: ERD design for the simulation data

5.4.2 Key scenarios

In Table 5-1 below, the five key scenarios identified in Chapter 2 (§2.6.5 – Step 1), namely; extremely low, low, medium, high, and extremely high demand, are allocated to SP execution duration. In other words, Table 5-1 shows how single SPs are classified per demand group, where demand refers to the extent of data processing required to complete the query. For example, a SP that requires 11 seconds to complete, is classified as a high demand SP with demand code 3. This table is used to guide the load test design in Section 5.4.5.

Table 5-1: SP demand classification

Demand	Description	Total Duration (s)
0	Extremely low demand	< 0.5
1	Low demand	0.5 - 1.5
2	Medium demand	1.6 - 5.9
3	High demand	6 - 15.9
4	Extremely high demand	> 15.9

5.4.3 Target load levels

The load is applied by executing combinations of different database SPs on each device, increasing progressively from low to extremely high load. The target load levels defined in Chapter 2 (§2.6.5), include low, medium, high and extremely high. To reach each target load level, SPs classified by the five key scenarios are executed on each device.

A target load level can be reached by combining SPs with different demand classifications. For example, a load level of medium, may contain one SP classified as medium demand as well as two SPs classified as extremely low demand. Table 5-3 shows with which SPs the load levels are reached.

5.4.4 Identify metrics

The identified metrics are used to determine the throughput of each device and the load-stability point of divergence of each MPDB in the result analysis phase of the study. These metrics are captured for each device and enable the researcher to compare the performances of the tested devices with one another.

The thread task that is referred to below, relates to a SP or in other words a database query sent to the device. Each phase consists of numerous threads and each thread executes one or more SPs in accordance with the load test design (Table 5-3). The status of these threads are captured to determine whether each thread's SPs were executed successfully. Thread failures play a key role in determining the point of divergence and occur in the pilot and experiment when:

- The MySQL query times out because no result is returned from the computing device.
- The computing device becomes indefinitely unresponsive.
- Any failure occurs; which is related to the computing device that prevented the result to be returned to MCS.

During the pilot, MCS is used to query the device for metrics and include:

- Duration of each thread to complete its task(s).
- Whether the task of each thread is completed successfully.
- Device random access memory (RAM) utilisation.
- Device central processing unit (CPU) utilisation.

During the experiment, Dstat is used to record device resource metrics locally on each device while MCS records metrics relating to the tasks sent to each device. These metrics include:

- Duration of each thread to complete its task(s); recorded by MCS.
- Whether the task of each thread is completed successfully; recorded by MCS.
- Device RAM utilisation; recorded by Dstat.
- Device CPU utilisation; recorded by Dstat.
- Overall device load average over time in terms of all resources, including RAM, CPU and disk utilisation; recorded by Dstat.

The metrics are stored in log files generated throughout the experiment with the use of MCS and the Dstat application installed on each device. During the pilot, only MCS log files are generated, because Dstat is not used in the pilot. These metrics are analysed throughout Chapter 6 and is important in order to meet the objective of the study. By statistically analysing the metrics, it enables the researcher to perform hypothesis tests and draw accurate conclusions relating to the study.

5.4.5 Design specific tests

The load tests are conducted through the execution of specific SPs, while recording metrics of each computing device. The generated simulation data only consists of the tables and data. Therefore, the researcher is responsible to design and write the SPs necessary to create, read, update and delete (CRUD) the data on each device.

5.4.5.1 Stored procedures

The SPs that manipulate and retrieve data from the database upon execution are shown in Table 5-2. The average execution and data fetch duration of each SP is recorded next to each SP. When the data fetch duration is zero, it suggests that a small collection of data are returned from the SP. A high execution duration implies that the query is of a more complex nature and therefore has a high demand on the computing device.

The RPi3 was used as a baseline to determine the average query execution duration by executing each SP three times. The RPi3 was chosen as the baseline because its CPU specification is higher than the BBB and IEA; it is therefore expected that the RPi3 can provide a more consistent

baseline for query times. Note that the duration baseline is used to estimate the relative demand between SPs and that the actual duration in seconds is not of great importance. The SP times from Table 5-2 are used to classify SPs according to the key scenarios detailed in Table 5-1; the "Demand" column in Table 5-2 refers to the SP classification in Table 5-1.

Table 5-2: Stored procedures created for the experiment

Queries					
Demand	ID	Stored Procedures	Average Duration (s)		
			Execution Duration	Data Fetch Duration	Total
0		*Create*			
	1	0_C_Product	0.110	0.000	0.110
	2	0_C_Client	0.140	0.000	0.140
	3	0_C_Order	0.230	0.000	0.230
		Retrieve			
	4	0_R_ClientDetails	0.030	0.000	0.030
	5	0_R_OrderDetails	0.045	0.000	0.045
	6	0_R_ClientOccupation	0.220	0.000	0.220
		Update			
	7	0_U_Product	0.078	0.000	0.078
	8	0_U_Order	0.094	0.000	0.094
	9	0_U_Client	0.125	0.000	0.125
	10	0_U_OrderLine	0.172	0.000	0.172
		Delete			
	11	0_D_Client	0.125	0.000	0.125
1		*Retrieve*			
	12	1_R_ClientOrders	1.090	0.000	1.090
	13	1_R_ClientOrdersPaid	0.170	0.950	1.120
	14	1_R_ClientOrdersCancelled	0.600	0.700	1.300
2		*Retrieve*			
	15	2_R_ClientNotes	0.020	2.200	2.220
	16	2_R_ClientNotesCity	0.020	2.300	2.320
3		*Retrieve*			
	17	3_R_CreditCheck	8.100	0.000	8.100
	18	3_R_ClientNotesOrders	8.500	0.000	8.500
	19	3_R_AmountOfOrdersDate	13.200	0.000	13.200
4		*Retrieve*			
	20	4_R_SoldPerProduct	23.000	0.000	23.000
	21	4_R_ClientOrdersFullDetails	0.200	36.000	36.200

The SPs do not require any parameters and they are deliberately designed to eliminate the need to specify conditions (i.e. the customer's name on which the query is run) at application level. The conditions for queries are randomly generated, within the bounds of the data contained in the tables, through functions written by the researcher. Functions are used in each SP to generate random query conditions that are relevant to table data. This design allows each SP to return variable data upon execution. It ensures that the database server returns fresh and not cached data, because when cached data is returned, no query load is applied to the DBMS.

5.4.5.2 Query execution plan

The query execution plan dictates how the load is applied to each device throughout the experiment. The plan is examined in Table 5-3 and makes use of the SPs tabled in Table 5-2 above. The column “SP ID” refers to the “ID” column in Table 5-2, meaning each thread must execute each SP listed in the particular phase (see the example described below Table 5-3). The target load levels, described in Section 5.4.3, are reflected in the “Load Level” column, each consisting of a number of phases.

Table 5-3: Load test design

Load Level	Phase	No. of Threads	SP IDs	Sum of Average Total Duration per Thread	Relative Load (%)
Low	1	5	4	0.030	0.0002
	2	25	5	0.045	0.0013
	3	30	12	1.090	0.0366
	4	25	4;5	0.075	0.0419
Medium	5	40	14	1.300	0.0582
	6	10	4;5;6	0.295	0.0990
	7	7	1;2;3;11	0.605	0.1895
	8	12	7;8;9;10	0.469	0.2518
	9	30	2;9;11	0.390	0.3926
	10	10	13;14	2.420	0.5414
High	11	60	17	8.100	0.5436
	12	100	18	8.500	0.9508
	13	150	4;5;6	0.295	1.4849
	14	200	2;4;9;11	0.420	3.7584
Extremely high	15	150	20	23.000	3.8591
	16	45	17;18	16.600	16.7114
	17	150	12;13;14	3.510	17.6678
	18	60	18;19	21.700	29.1275
	19	50	15;20	25.220	28.2103
	20	100	17;18;19	29.800	100.0000

Each phase is tested on each computing device; an example to explain the meaning of the table values and how the values are calculated for each phase follows.

Example of Phase 4:

Phase 4 (P4) requires 25 threads executed simultaneously. Each of the 25 threads executes SP IDs 4 and 5, that is 0_R_ClientDetails and 0_R_OrderDetails, respectively (as reflected in Table 5-2) by using the “ID” column as reference.

The SPs 0_R_ClientDetails (0.03s) and 0_R_OrderDetails (0.045s) have a total execution duration of 0.075s. This is seen in the “Sum of average total duration per thread” column, calculated from the “Total” column under “Average duration (s)” in Table 5-2.

Relative load is calculated as a percentage of the phase with the highest product value of “No. of Threads” and “Sum of average total duration per thread” (P20), as shown in Equation 5-1 below.

$$Phase\ 4\ Relative\ Load = \left(\frac{\text{No. of Threads} \times \text{Sum of average total duration per thread}}{\text{P20 No. of Threads} \times \text{P20 Sum of average total duration per thread} \times \text{P20 No. of SP IDs} \times 10} \right) \text{No. of SP IDs} \times 10$$

$$= \left(\frac{25 \times 0.075}{100 \times 29.8 \times 3 \times 10} \right) 2 \times 10$$

$$= 0.0419\%$$

Equation 5-1: Phase 4 relative load equation

Phases with only one SP ID are calculated slightly differently, see Equation 5-2 below. The difference is that the relative load value is not multiplied by the number of SP IDs and 10. The reason for this difference is that the more SPs each thread executes, the higher the load.

$$Phase\ 1\ Relative\ Load = \left(\frac{\text{No. of Clients} \times \text{Sum of average total duration per client}}{\text{P20 No. of Clients} \times \text{P20 Sum of average total duration per client} \times \text{P20 No. of SP IDs} \times 10} \right)$$

$$= \left(\frac{5 \times 0.03}{100 \times 29.8 \times 3 \times 10} \right)$$

$$= 0.0002\%$$

Equation 5-2: Phase 1 relative load equation

The relative load formula designed by the researcher acts as a means of predicting the load of each phase to determine the sequential order of increasing load through the phases. This relative load does not accurately predict load; it only serves as a guide to order the phases from low to high load and is not critical to the study.

5.5 Creation of database script

The process of creating a database script is performed in a Windows environment on the development PC. Before the script can be created, data migration from MS SQL to MySQL is required and described below.

5.5.1 Data migration

The DBMS that is utilised by the study and installed on all devices is MySQL, as discussed in Chapter 3, while the data generator script (described in Section 5.3) is in MS SQL format. Therefore, the format is converted from MS SQL to MySQL.

MySQL Workbench 6.3 has a built-in function, Migration Wizard, which is used in an attempt to convert the MS SQL data generator script to MySQL format. The migration failed because some of the functions used in the script are unknown to MySQL. The migration approach is therefore altered to perform the data generation in MS SQL and migrate the tables and data to MySQL. The data migration process is completed successfully and the MySQL DBMS installed on the Windows development PC now contains the generated tables and data.

5.5.2 Database script creation

At this point, the SPs described in Section 5.4.5.1 are designed and written with the use of MySQL Workbench 6.3 in Windows.

After the creation of the SPs, the MySQL database is ready to be scripted and distributed to all computing devices used in the study. A full database script of the database is therefore generated with the use of the Data Export function in MySQL Workbench 6.3. The database script is then executed on all the devices' installations of MySQL server to create a uniform database structure (tables and SPs) and populate the tables on each device with the exact same data.

5.6 Preparation of computing devices

All devices require an OS and an installation of a DBMS to be configured. The chosen OS and DBMS is Linux Debian 8 and MySQL 5, respectively. Hardware preparation (§A.1.1), OS installation (§A.1.2) and DBMS installation (§A.1.3) are detailed in Appendix A. This section details the final preparation steps followed on all devices and the database setup.

5.6.1 Final preparation steps followed on all devices

The four devices are set up in the previous steps (Appendix A) and require the user to log in at this stage. All devices are connected to the local network, which includes an Internet connection

and enabled to accept Secure Shell connections. Secure Shell (SSH) provides a secure protocol for clients and servers to communicate over encrypted network connections (SSH Communications Security, 2017). This allows communication through a SSH tunnel from a Windows environment.

The IEA is the only device lacking an Ethernet port and is therefore set up to use its on-board WiFi to connect to the network; the other devices are connected via Ethernet cables and no network setup is therefore required. The network setup for the IEA and the SSH authentication is set up via the command line interface of each device.

All devices are now accessible through the network with the use of a SSH tunnel. The SSH tunnel can be established with a SSH client like PuTTY which is used throughout the setup process to interact with the devices.

Once each device is connected to the network and accessible via SSH, Dstat is installed to allow for device resource monitoring. The Dstat application is set up to collect device resource information in a log file (csv format) stored on the device's local storage. A script file created on each device is used to execute Dstat and provide an appropriate log file name. Each Dstat log file contains the following data for every second since the start of the log file:

- log entry timestamp;
- CPU usage divided between user, system, idle, wait, hardware interrupts, software interrupts;
- memory usage divided between used, buffer, cache, free;
- disk read and write cycles; and
- overall load average over one, five and 15 minutes.

Note that Dstat is not utilised in the pilot. The OS installations and setup is hereby concluded and the devices are ready for the DBMS installation.

5.6.2 Database setup

With all the devices relating to the experiment connected to the local network, each device's database server instance is managed through MySQL Workbench 6.3 installed on the Windows development PC that is connected to the same network. In order to access the database server instance, a connection must be established from MySQL Workbench on the Windows PC to each

device's database server instance (Johnson, 2014). The protocol used for this connection is called "TCP/IP over SSH".

As soon as the TCP/IP over SSH connection is opened, MySQL Workbench is used to execute the database script generated in Section 5.5.2. The script creates the database and fills it with the tables, SPs and data contained in the script. The script is executed on each device's instance of MySQL Server. MySQL Server variables, attached in Appendix A (§A.2), are verified to ensure all variables are uniform across the devices.

All devices are set up similarly in terms of the OS, software and settings. This similarity is important for the success of the experiment since it minimises potential differences in variables across devices that relate to database service and device performance.

5.7 Test application

A test application is required to perform the load tests on the devices and gather statistics during the tests' execution. The application simulates multiple clients, each running in a separate thread, connecting to a database server instance to execute the SPs residing on the database. The application user specifies the number of clients simulated for each phase and the SPs executed by each client in the particular phase. This application is MCS, introduced in Chapter 2 Section 2.6.5. The MCS is operated through a basic front end with configurable settings in a configuration (.ini) file.

Two versions of MCS forms part of the study:

- MCS Version 1 used with the pilot; and
- MCS Version 2 used with the experiment.

The MCS class diagram is shown in Figure 5-2; each class in the diagram is described below while highlighting the differences between the two versions of MCS in each section. The main difference between the pilot and the experiment is that the pilot requires a great deal of manual intervention between phases and does not make use of Dstat. Device resource logging during the pilot is accomplished through MCS querying the device for resource metrics, while the experiment makes use of Dstat to acquire these metrics. Log files containing the aforementioned data for the pilot and experiment can be viewed in Appendix E.

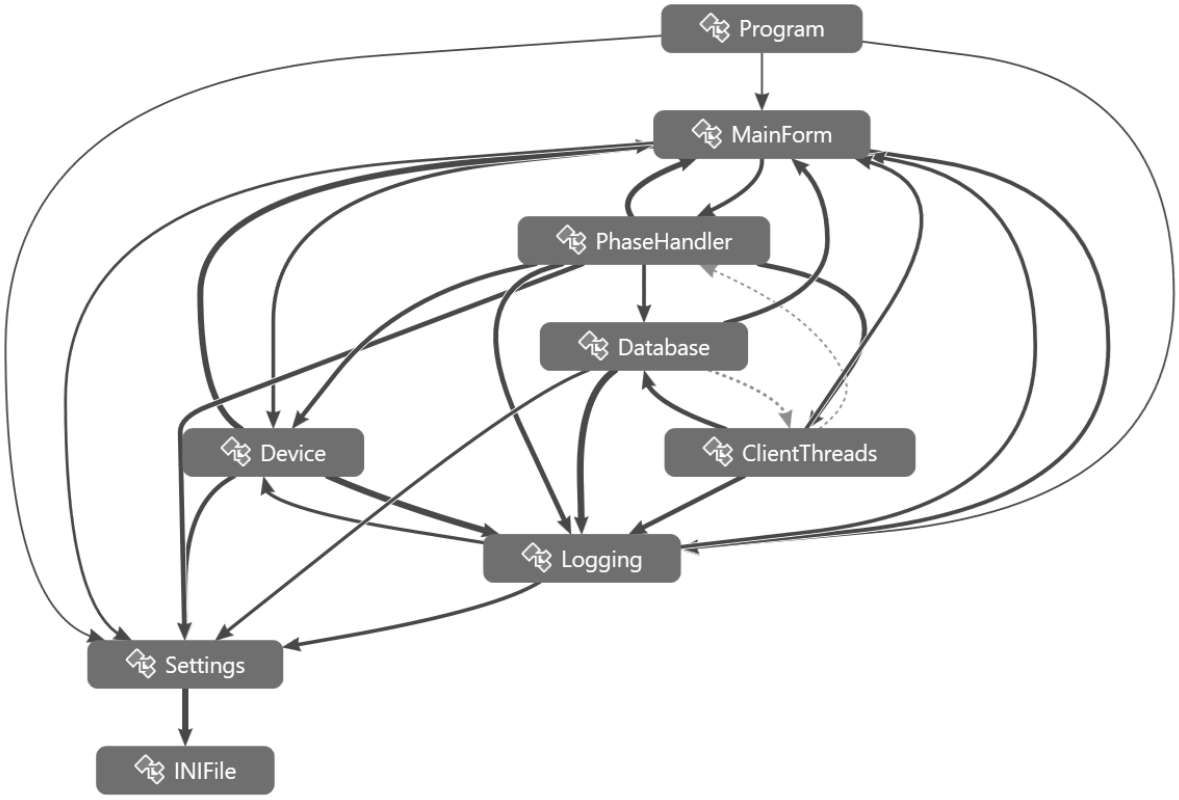


Figure 5-2: Multi-Client Simulator class diagram

The *Program class* consist of two functions, starting the application and closing it. The main function initialises the Settings class and MainForm that displays the graphical user interface (GUI). The close function closes the log file and the connection to the device hosting the database service and is called when the user exits the application.

MainForm initialises the GUI, which allows the user to interact with the connected device and observe crucial information about the experiment. The GUI of the pilot, shown in Figure 5-3, consists of a label that shows the name of the device that the application is connecting to, a textbox into which the user can enter the number of clients to be simulated, and the device actions.

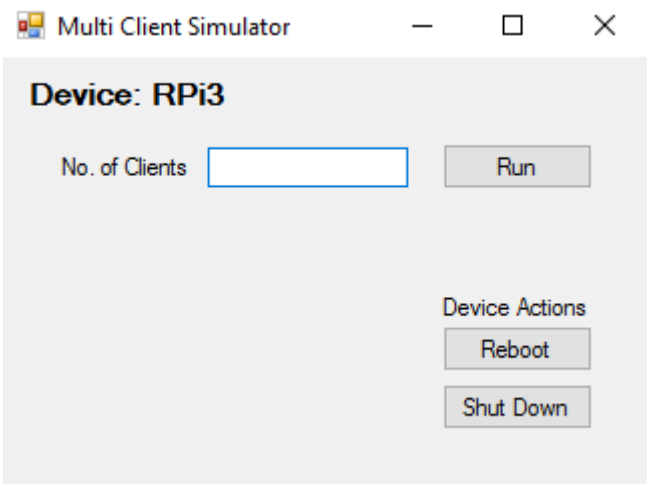


Figure 5-3: Pilot MCS GUI

The experiment GUI shows real-time information during the execution of the experiment to inform the user of the current actions and device status. The GUI is shown in Figure 5-4 and reflects the following:

- name of the device that the application is connecting to;
- a dropdown list for phase selection;
- a textbox into which the user can enter the starting iteration;
- a “Run” button to start the process;
- device actions;
- the estimated completion time of the running process; and
- a window that shows output similar to the log file.

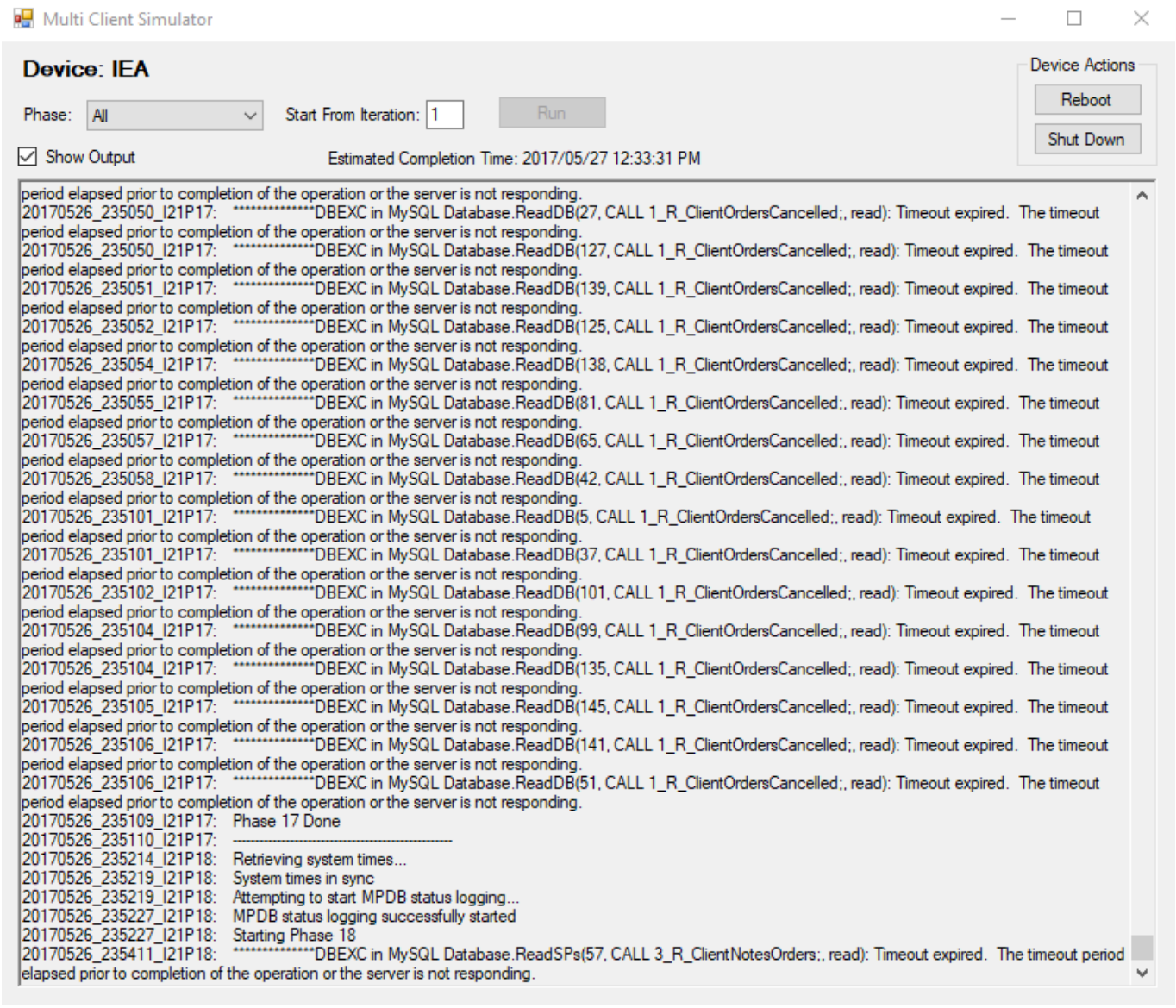


Figure 5-4: MCS GUI

The *Device class* interacts directly with the connected device and is responsible for shutting down and rebooting the device, as well as for capturing device resource metrics. Resource monitoring during the pilot is accomplished through direct monitoring by MCS. The device status monitoring is run from a separate thread responsible for retrieving the status of the connected device and logging the data to a text file. The frequency of the status retrieval is determined by a setting in the configuration file; the frequency for all devices is set to 0.5 Hz throughout the pilot. The status metrics retrieved from each device include CPU usage, CPU temperature and RAM. Four RAM metrics are retrieved: free, cached, used and total RAM.

The following functions are added to the Device class for the experiment:

- Windows PC and device time synchronisation.
- Initialising and stopping the device resource monitoring (Dstat).
- Retrieving device status log files from the connected device.

In the experiment, the direct device resource monitoring through MCS is replaced with Dstat. Time synchronisation between the device and the Windows PC ensures the alignment of time stamps in MCS and Dstat log files. The time synchronisation takes place before the start of each new Dstat log file; in other words, before the start of each phase, device and Windows PC times are synchronised. After each iteration, the Dstat log files are programmatically moved from the device to the Windows PC for storage and backup purposes.

The *PhaseHandler class* is responsible for executing the selected phases in order and repeating these phases by the number of iterations the user selects. This class is introduced in the experiment after the implementation of the pilot, where each phase is executed manually. The number of iterations, phases and SPs of each device are defined in the configuration file described in the Settings class. Before the start of each phase, its SPs are passed to the ClientThreads class. The PhaseHandler class also reboots the device after each iteration, prepares for a new iteration by running each read (select) type SP three times to improve query times, and estimates the completion time displayed on the GUI.

The *ClientThreads class* initiates a new thread for each client specified by the user. The ClientThreads class in the pilot is similar to that of the experiment, except that in the pilot the number of clients depends on the number the user specifies in the “No. of Clients” textbox, and the SPs executed are retrieved directly from the configuration file. Each thread (or simulated client) creates a new device and database connection and executes the SPs received from the configuration file (pilot) or PhaseHandler class (experiment). The PhaseHandler class retrieves the number of clients from the configuration file. This threaded approach accurately simulates multiple clients (i.e. PCs) connecting concurrently to the database service and performing database actions. Once a client’s database actions are completed, the connection to the device and the thread itself is terminated.

The *Database class* creates new connections to the device hosting the database service, and dictates how different types of SPs are dealt with by the application. The study makes use of four types of SPs: Create, Read, Update, and Delete (CRUD). This class also provides the application with the results returned from the database. The MCS writes the acquired results to the log file.

Both the *Device and Database classes* make separate connections to the connected device. The Database class connection is made one layer below that of the Device class connection, because

it connects to the device and then to the database service hosted by the device, while the Device class does not connect to the database service. Figure 5-5 illustrates the described connection architecture between the server and MCS.

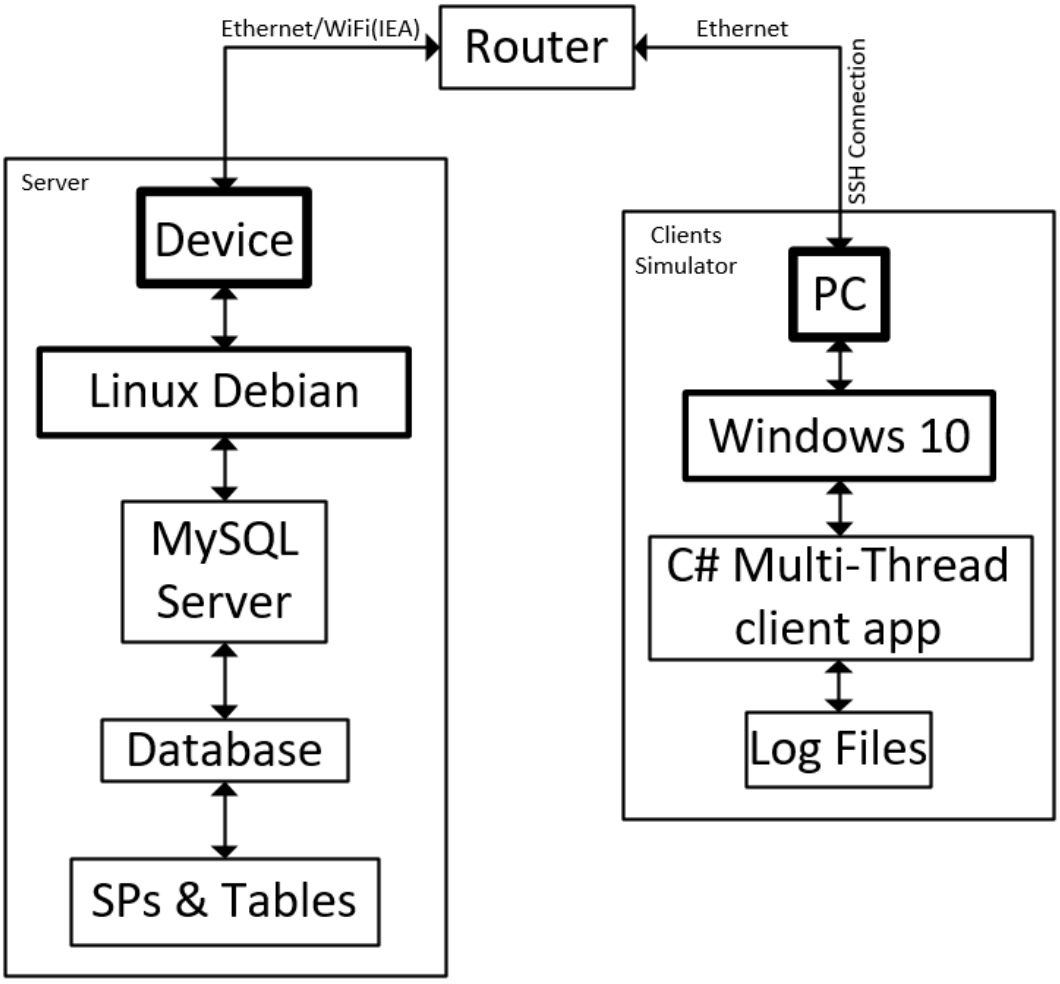


Figure 5-5: Server-Client connection architecture

The *Settings class* makes use of the INIFile class that contains a function to read from the configuration (.ini) file. The configuration file is stored in the application's working directory. The Settings class is used by other classes to obtain settings for specific tasks such as establishing a device or database connection. A setting is recorded in the configuration file by defining a section, setting name (key name), and a value. The configuration files of the experiment differ significantly from those of the pilot, due to automation of processes in the experiment. The pilot requires a unique configuration file for every device and every phase, while the experiment only requires one configuration file per device. Examples and more information on the configuration files are reflected in Appendix A (§A.3).

The *Logging class* is initialised with the start of MCS. The initialisation entails creating a new text file (log file) in the directory where the application files reside, titled with the current date and time. This class includes a Log function that requires two parameters: the client number and the text logged. The Log function automatically adds a timestamp to each line written to the log file. This function is used throughout the application to record information, database results, errors and metrics. The Logging class also includes a function to close the log file when the application is terminated.

5.8 Tests and metrics

The MCS application used to run the load test of each phase from the Windows development PC, forms a crucial part of the experiment. The experiment data captured using MCS allows the researcher to perform statistical data analysis in order to draw accurate conclusions about the study in the subsequent chapters. The pilot is firstly discussed, followed by the experiment.

The load test design (Table 5-3) consists of 20 phases; each phase is tested once on each device during the pilot. Figure 5-6 illustrates the load execution plan for the pilot.

Both decision nodes shown in Figure 5-6 require human intervention; each phase on each device requires a configuration file. In other words, 80 unique configuration files were needed for the pilot.

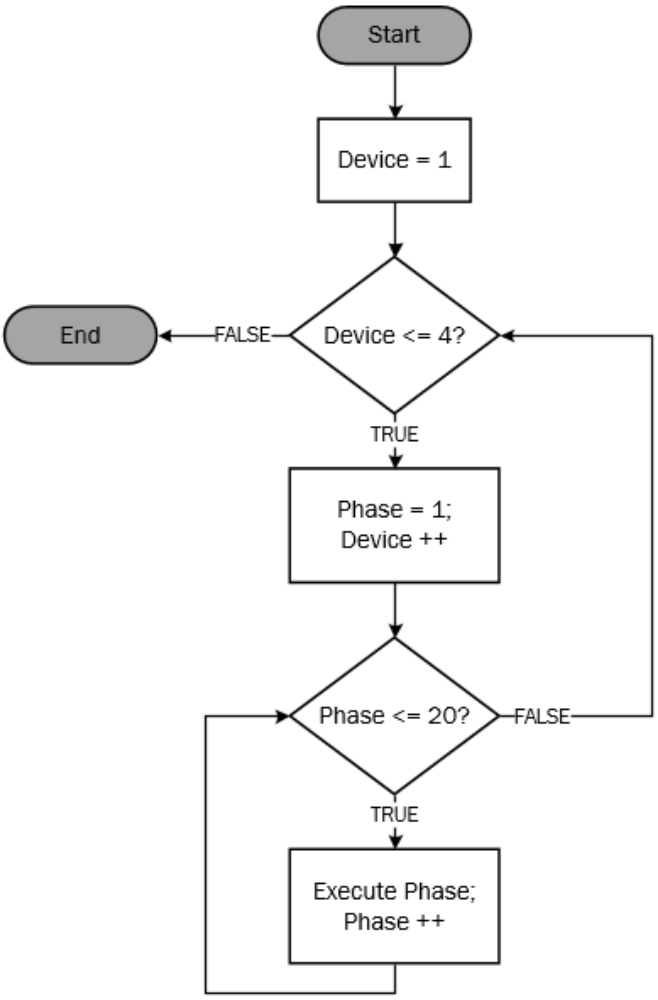


Figure 5-6: Pilot load execution flow diagram

The settings shown in Figure 5-7 are duplicated in all configuration files (general settings):

```
[LOG]
DiagnosticLogging=false
LogDbLines=false

[MPDBSTATUS]
TimerFrequency=2
RamUsage=true
CpuLoad=true
CpuTemperature=true
```

Figure 5-7: Duplicated configuration file settings

The steps below detail how the application is used after the default settings are recorded in the configuration file.

1. Set up the configuration file (refer to Appendix A – Section A.3.2 for an example):
    1.1. for a device in the MPDBCONNECTION section.
    1.2. for the device's database service in the DBCONNECTION section.
    1.3. for a phase according to the load test design in Table 5-3 in configuration file sections SP_CREATE, SP_READ, SP_UPDATE and SP_DELETE. Unused values are left blank.
2. Start the application and enter the number of clients to be simulated on the GUI (Figure 5-3), which is found in Table 5-3 in column "No. of Threads".
3. Click the "Run" button on the GUI and wait until the following message is displayed:
4. "DONE - All threads completed in (x) seconds".
5. Click "Ok" to dismiss the message box and close MCS.
6. Repeat Steps 1.3 to 3 for all phases tabled in Table 5-3.
7. Repeat Steps 1 to 4 when switching to another device.

A log file for each phase on each device is generated by MCS in which experiment and device resource metrics are captured. The log files are created and reside on the Windows PC. Figure 5-8 shows an extract from a log file. Note that the device resource metrics are included in the pilot (MCS) log file. Thread 0 in the pilot is not only used for additional application information, but also for device resource metrics.

```
Log file opened for RPi3 at C:\Users\Riaan Fokker\Google
Drive\Masters\Report\Sections\Chapter 4\Empirical
Work\MultiClientApp\MultiClientApp\bin\Release\LOG\20160731_132811.txt
13:28:15.275:  0:  CPU Usage=2,36%
13:28:15.808:  0:  CPU temperature: 39,0'C
13:28:15.840:  0:  RAM: free=12MB; cached=676MB; used=912MB; total=925MB
13:28:17.261:  0:  CPU Usage=3,38%
13:28:17.599:  0:  ---------------Threads starting---------------
13:28:17.794:  0:  CPU temperature: 38,5'C
13:28:17.822:  0:  RAM: free=11MB; cached=676MB; used=913MB; total=925MB
13:28:18.241:  2:  CALL 0_R_ClientDetails();
13:28:18.245:  1:  CALL 0_R_ClientDetails();
13:28:18.247:  3:  CALL 0_R_ClientDetails();
13:28:18.354:  5:  CALL 0_R_ClientDetails();
13:28:18.382:  1:  Client 1 done in 0,7781978s
13:28:18.390:  4:  CALL 0_R_ClientDetails();
13:28:18.397:  2:  Client 2 done in 0,7792s
13:28:18.404:  3:  Client 3 done in 0,7912183s
13:28:18.450:  5:  Client 5 done in 0,8463024s
13:28:18.477:  4:  Client 4 done in 0,8728437s
13:28:18.483:  0:  ---------------DONE - All threads completed in 0,878 seconds---------------
13:28:19.264:  0:  CPU Usage=25,25%
13:28:19.796:  0:  CPU temperature: 39,5'C
13:28:19.833:  0:  RAM: free=13MB; cached=674MB; used=912MB; total=925MB
13:28:21.264:  0:  CPU Usage=1,43%
13:28:21.796:  0:  CPU temperature: 40,1'C
13:28:21.830:  0:  RAM: free=13MB; cached=674MB; used=911MB; total=925MB
13:28:23.264:  0:  CPU Usage=0,48%
13:28:23:  Log file closed for RPi3 at C:\Users\Riaan Fokker\Google
Drive\Masters\Report\Sections\Chapter 4\Empirical
Work\MultiClientApp\MultiClientApp\bin\Release\LOG\20160731_132811.txt
```

Figure 5-8: Pilot MCS log file example

Successful completion of the pilot, including its statistical analysis, shows that the experiment is feasible; refer to Chapter 6 Section 6.3 for the pilot's statistical analysis. The experiment, like the pilot, makes use of the load test design (Table 5-3), which consists of 20 phases; each phase is tested on each device 30 times to ensure the captured metrics are statistically significant. The flow diagram in Figure 5-9 illustrates the load execution plan on each device.

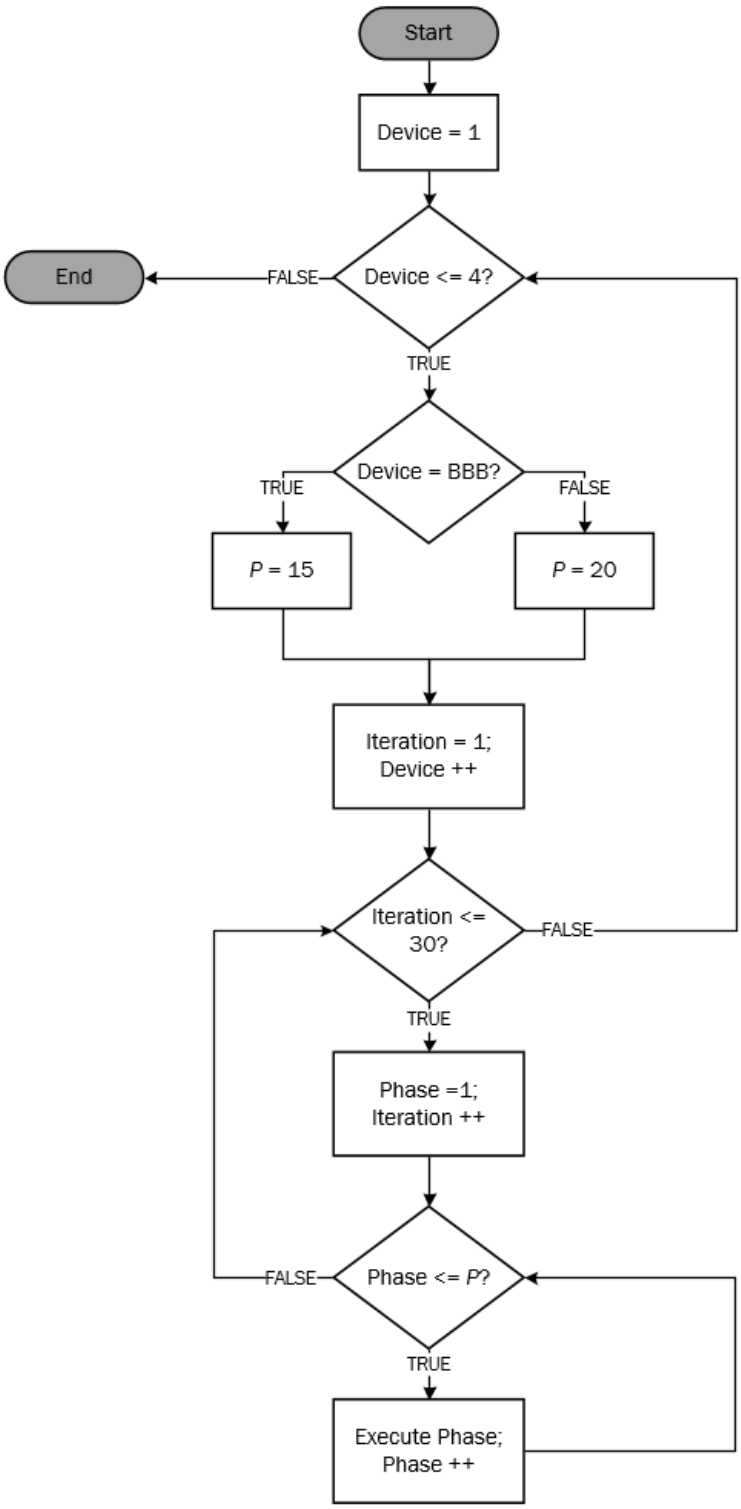


Figure 5-9: Load execution flow diagram

During the experiment, it is found that all devices except the BBB are capable of performing phases 1 to 20. The MPDBs capable of performing phases 1 to 20 have unsuccessful database queries in certain phases owing to high load, but are able to continue to the subsequent phases

thereafter. The BBB however, becomes unreachable after Phase 15 owing to system instability as a result of the incremental load. While in this state, the user has to perform a physical reset of the device by unplugging the device's power. During the pilot, it was possible to force the device through Phases 16 to 20 by manually resetting the device after each of these phases. Based on this knowledge, a decision was made to adjust the load test of the BBB to perform only Phases 1 to 15 in the experiment. From the results of the pilot in Chapter 6 (§6.3), it is clear that the BBB had a 100% failure rate in Phases 16 through 20; subsequently removing these phases from the load test of the BBB does not degrade the experiment in any way.

Only the device (first decision node in Figure 5-9) requires human intervention, the rest of the flow diagram is managed by the PhaseHandler class. The aforementioned intervention requires the researcher to set up the configuration file of the chosen device on the Windows PC; each device's configuration file is detailed in Appendix A (§A.3.1). The MCS application makes use of the configuration file throughout all 30 iterations. Once the configuration file is set up, MCS is started and the user is presented with the GUI (Figure 5-4). The user then selects "All" in the "Phase" dropdown list and click "Run". Note that the "Start From Iteration" textbox defaults to "1" and may be used when an unexpected condition occurs, such as an interruption in electricity.

This concludes the process for conducting the experiment. The applications, Dstat and MCS, each generate a log file for each phase on each device to capture experiment and device resource metrics. The Dstat and MCS log files reside on the Windows PC. Note that the Dstat log files are created locally on each device and are moved to the Windows PC at the end of each iteration for storage and backup purposes.

Figure 5-10 shows the MCS log file for the RPi3, Iteration 1, Phase 10; for each line logged; the first field is the current time, the second field is the client (or thread) number, and the third field is the log line information. Note that Thread 0 relates to additional application information and not to a thread that interacts with the database.

```
Log file opened for RPi3 at D:\Google Drive\Masters\Report\Sections\Chapter 4\Empirical Work
\MultiClientApp\MultiClientApp\bin\Release\LOG\RPi3\20170523_180417_I1P10.txt
18:04:17.634:  0:  Attempting to start MPDB status logging...
18:04:24.274:  0:  MPDB status logging successfully started
18:04:24.274:  0:  ---------------Threads starting---------------
18:04:24.743:  3:  CALL 1_R_ClientOrdersPaid;
18:04:25.509:  2:  CALL 1_R_ClientOrdersPaid;
18:04:26.353:  1:  CALL 1_R_ClientOrdersPaid;
18:04:26.978:  4:  CALL 1_R_ClientOrdersPaid;
18:04:27.024:  6:  CALL 1_R_ClientOrdersPaid;
18:04:28.056:  8:  CALL 1_R_ClientOrdersPaid;
18:04:28.103:  7:  CALL 1_R_ClientOrdersPaid;
18:04:28.103:  10:  CALL 1_R_ClientOrdersPaid;
18:04:28.462:  1:  CALL 1_R_ClientOrdersCancelled;
18:04:28.493:  9:  CALL 1_R_ClientOrdersPaid;
18:04:28.493:  5:  CALL 1_R_ClientOrdersPaid;
18:04:28.868:  2:  CALL 1_R_ClientOrdersCancelled;
18:04:28.868:  3:  CALL 1_R_ClientOrdersCancelled;
18:04:30.712:  4:  CALL 1_R_ClientOrdersCancelled;
18:04:31.149:  3:  Client 3 done in 6,8593862s
18:04:31.290:  6:  CALL 1_R_ClientOrdersCancelled;
18:04:31.509:  7:  CALL 1_R_ClientOrdersCancelled;
18:04:31.931:  10:  CALL 1_R_ClientOrdersCancelled;
18:04:32.040:  9:  CALL 1_R_ClientOrdersCancelled;
18:04:33.493:  1:  Client 1 done in 9,2031412s
18:04:33.509:  6:  Client 6 done in 9,2187662s
18:04:34.149:  4:  Client 4 done in 9,8593923s
18:04:34.196:  9:  Client 9 done in 9,9062671s
18:04:34.446:  10:  Client 10 done in 10,1562673s
18:04:34.556:  8:  CALL 1_R_ClientOrdersCancelled;
18:04:34.821:  5:  CALL 1_R_ClientOrdersCancelled;
18:04:34.978:  2:  Client 2 done in 10,6875188s
18:04:35.446:  7:  Client 7 done in 11,1562693s
18:04:35.946:  5:  Client 5 done in 11,656269s
18:04:40.353:  8:  Client 8 done in 16,0625288s
18:04:40.368:  0:  ---------------DONE - All threads completed in 16,085 seconds---------------
18:04:43:  Log file closed for RPi3 at D:\Google Drive\Masters\Report\Sections\Chapter
4\Empirical Work\MultiClientApp\MultiClientApp\bin\Release\LOG\RPi3\20170523_180417_I1P10.txt
```

Figure 5-10: MCS log file example

Figure 5-11 shows an Excel formatted extract from a Dstat log file. This extract shows 20 lines logged over 20 seconds; each line shows metrics relating to CPU, RAM and disk usage as well as the load average for one, five and 15 minutes. These results are further explained in Chapter 6, where the pilot and experiment results are analysed.

system	total cpu usage							memory usage				dsk/total		load avg		
time	usr	sys	idl	wai	hiq	siq	used	buff	cach	free	read	writ	1m	5m	15m
2017/05/23 18:04:22.422	5.085	1.249	90.846	2.756	0.0	0.064	281157632.0	3964928.0	665079808.0	20606976.0	2979837.441	43034.657	0.360	0.690	0.580
2017/05/23 18:04:23.423	2.015	0.252	97.733	0.0	0.0	0.0	278872064.0	3964928.0	665083904.0	22888448.0	0.0	0.0	0.360	0.690	0.580
2017/05/23 18:04:24.424	13.400	2.481	84.119	0.0	0.0	0.0	282140672.0	3964928.0	665112576.0	19591168.0	0.0	0.0	0.360	0.690	0.580
2017/05/23 18:04:25.425	19.202	2.993	77.057	0.748	0.0	0.0	284229632.0	3964928.0	665358336.0	17256448.0	491520.0	0.0	0.410	0.700	0.580
2017/05/23 18:04:26.426	38.636	5.808	53.030	2.525	0.0	0.0	285192192.0	3973120.0	665972736.0	15671296.0	1146880.0	90112.0	0.410	0.700	0.580
2017/05/23 18:04:27.427	51.500	8.250	36.500	3.500	0.0	0.250	286388224.0	3973120.0	666574848.0	13873152.0	1507328.0	0.0	0.410	0.700	0.580
2017/05/23 18:04:28.428	74.623	10.553	11.307	2.513	0.0	1.005	289841152.0	3973120.0	664125440.0	12869632.0	1933312.0	0.0	0.410	0.700	0.580
2017/05/23 18:04:29.429	82.908	13.520	2.296	0.255	0.0	1.020	291037184.0	3964928.0	661721088.0	14086144.0	2883584.0	0.0	0.410	0.700	0.580
2017/05/23 18:04:30.430	87.593	8.685	2.730	0.0	0.0	0.993	290852864.0	3964928.0	663273472.0	12718080.0	3506176.0	0.0	1.260	0.870	0.640
2017/05/23 18:04:31.431	79.695	14.975	3.299	0.761	0.0	1.269	291033088.0	3973120.0	660783104.0	15020032.0	2760704.0	57344.0	1.260	0.870	0.640
2017/05/23 18:04:32.432	85.422	9.463	4.092	0.256	0.0	0.767	290652160.0	3973120.0	662024192.0	14159872.0	2621440.0	0.0	1.260	0.870	0.640
2017/05/23 18:04:33.433	85.823	10.633	2.532	0.506	0.0	0.506	289251328.0	3973120.0	663490560.0	14094336.0	3047424.0	0.0	1.260	0.870	0.640
2017/05/23 18:04:34.434	80.307	12.532	5.627	1.535	0.0	0.0	287191040.0	3973120.0	664805376.0	14839808.0	2850816.0	0.0	1.260	0.870	0.640
2017/05/23 18:04:35.435	72.475	6.566	15.909	4.040	0.0	1.010	285048832.0	3997696.0	666066944.0	15695872.0	2359296.0	172032.0	1.400	0.900	0.650
2017/05/23 18:04:36.436	38.958	3.970	53.846	2.978	0.0	0.248	283340800.0	3997696.0	666746880.0	16723968.0	1507328.0	0.0	1.400	0.900	0.650
2017/05/23 18:04:37.437	21.411	2.015	74.055	2.519	0.0	0.0	282681344.0	3997696.0	667324416.0	16805888.0	1048576.0	0.0	1.400	0.900	0.650
2017/05/23 18:04:38.438	20.907	2.267	74.055	2.771	0.0	0.0	282460160.0	3997696.0	667832320.0	16519168.0	1212416.0	0.0	1.400	0.900	0.650
2017/05/23 18:04:39.439	20.551	2.757	73.935	2.757	0.0	0.0	281812992.0	3997696.0	668606464.0	16392192.0	1277952.0	0.0	1.400	0.900	0.650
2017/05/23 18:04:40.440	19.799	4.511	71.930	3.759	0.0	0.0	283041792.0	3997696.0	669253632.0	14516224.0	1507328.0	0.0	1.450	0.920	0.660
2017/05/23 18:04:41.441	12.219	1.995	85.786	0.0	0.0	0.0	284463104.0	4005888.0	669310976.0	13029376.0	0.0	40960.0	1.450	0.920	0.660

Figure 5-11: Dstat log file example

5.9 Data preparation for analysis

The pilot and the experiment generate raw data as explained in Section 5.8, which must be prepared to allow for data analysis. The data required for analysis need to be correctly structured and aggregated, which is accomplished with the use of a data warehouse created in MS SQL and pivot tables in Microsoft Excel. The pivot table data are accessed and displayed by creating a connection between the data warehouse and the Power Pivot function in Microsoft Excel.

The data preparation process for the experiment is similar to that of the pilot; differences are highlighted throughout this section. Note that the main difference between the pilot and experiment is that the experiment has 30 iterations while the pilot has one iteration.

The first step of the data preparation process is to design a data warehouse structure. The second step is to import the raw data into the data warehouse. A separate data warehouse is created for the pilot and the experiment. The data warehouse design is explained in Section 5.9.1 and the data import process is explained in Section 5.9.2.

5.9.1 Data warehouse design

The data warehouse is designed by the researcher and consists of two fact tables and four dimension tables. Figure 5-12 illustrates the data warehouse design that holds the data from the pilot. The tables and relationships were created in MS SQL. The fact table containing the log lines also consists of a foreign key from the device, time and thread dimension. The fact table that houses the load metrics consists of load data based on foreign keys from the device, time and phase dimension.

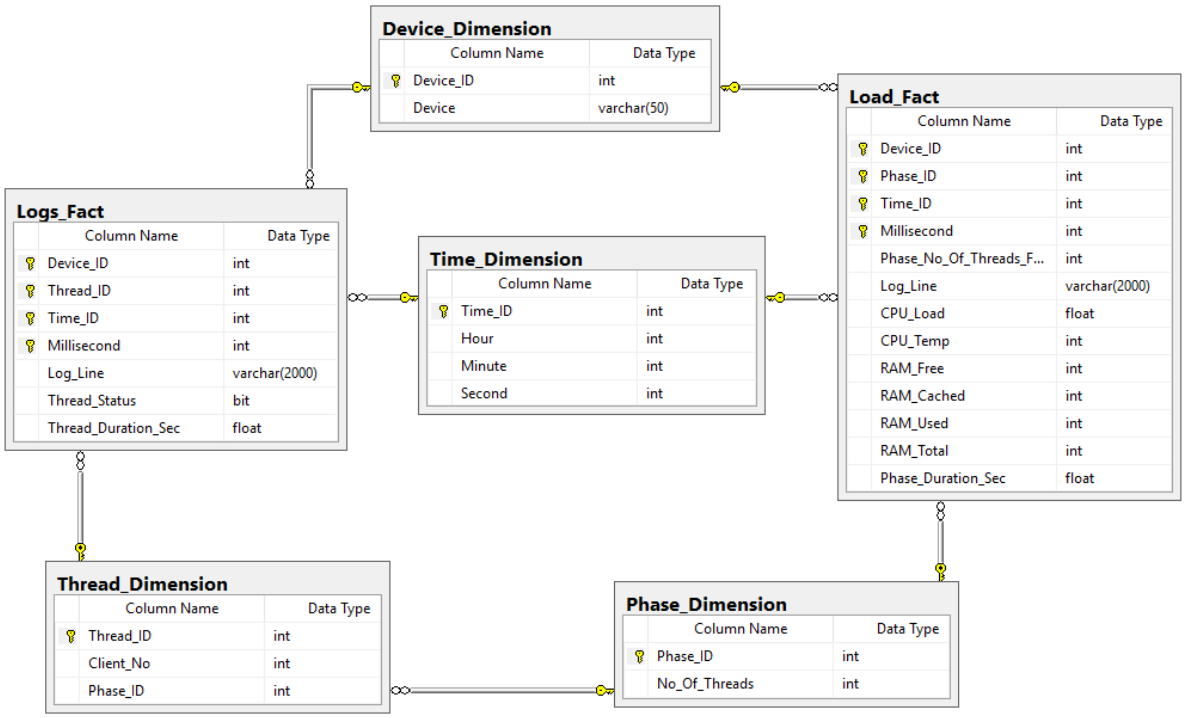


Figure 5-12: Pilot data warehouse design

Figure 5-13 shows the data warehouse design for the experiment, very similar to that of the pilot. As seen in Figure 5-12 and Figure 5-13, the significant differences between the experiment and pilot data warehouse designs are:

- The experiment consists of 30 iterations, which are accounted for in the PHASE_DIMENSION table.
- The experiment metrics captured in the LOAD_FACT table, relating to the load of each device, are more comprehensive than those of the pilot.

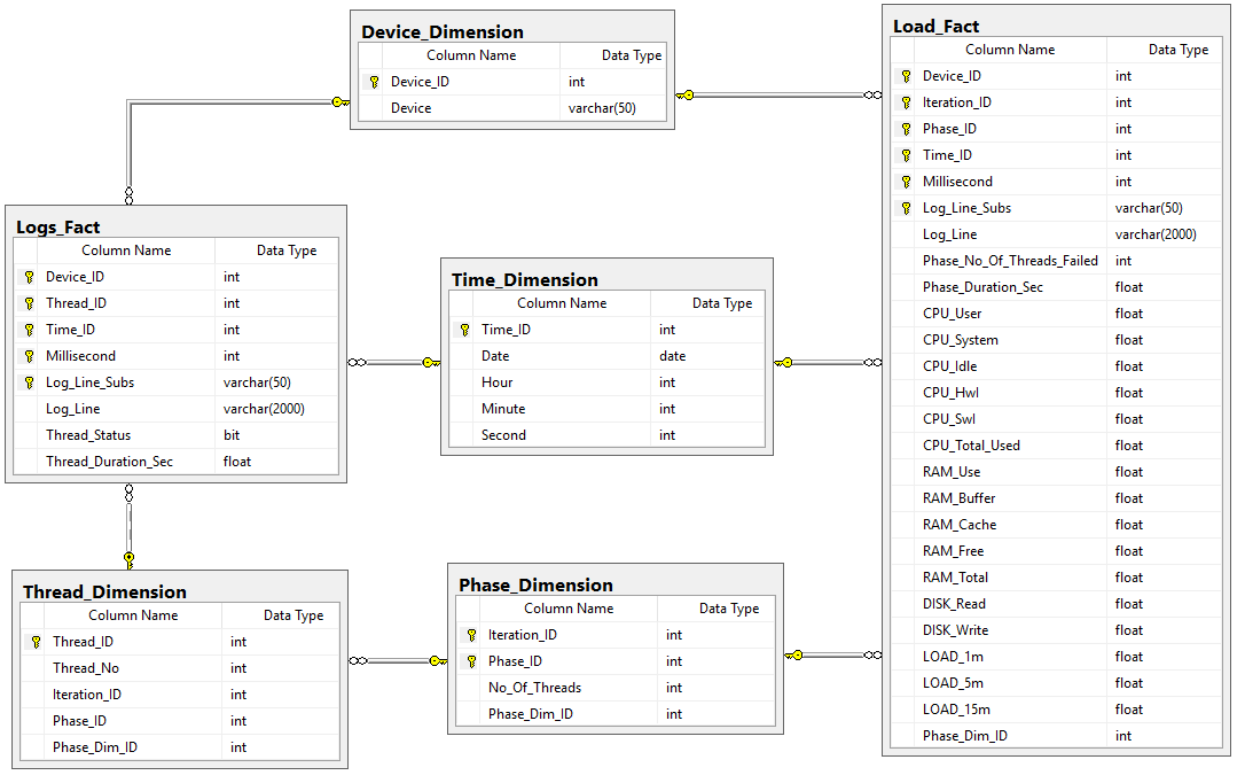


Figure 5-13: Data warehouse design

This data warehouse design enables the researcher to easily aggregate the data. It also includes the ability to drill down to data that are more detailed. The following section describes how the log files generated by the pilot and the experiment are imported into the tables. The database structure scripts and data import scripts for the pilot and experiment can be viewed in Appendix F.

5.9.2 Import data

This section describes the process of importing the log file data into the data warehouse. Log files have to be modified (§5.9.2.1) to adhere to a certain format in order to import the data (§5.9.2.2) into the data warehouse.

5.9.2.1 Log file cleaning

To ensure the success of the data import process, all log files have to be modified. Figure 5-14 shows an original MCS log file and Figure 5-15 shows the modified log file. All log files are modified using a function in Notepad++ called “Replace in Files” – by using this function, Notepad++ automatically replaces the given characters with new specified characters in all files stored in a specific directory. Note that the same process is followed to modify the log files generated during the pilot.

The modification process of the MCS log files in Notepad++ include (search mode in brackets):

- Replace ";" with "." (Normal).
- Replace ": " with ";" (Normal).
- Replace ".*log file opened.*\r?\n" with "" (Regular expression).
- Replace ".*Log file closed.*\r?\n" with "" (Regular expression).

```
Log file opened for RPi3 at D:\Google Drive\Masters\Report\Sections\Chapter 4\Empirical Work
\MultiClientApp\MultiClientApp\bin\Release\LOG\RPi3\20170523_180417_I1P10.txt
18:04:17.634:  0:  Attempting to start MPDB status logging...
18:04:24.274:  0:  MPDB status logging successfully started
18:04:24.274:  0:  ---------------Threads starting---------------
18:04:24.743:  3:  CALL 1_R_ClientOrdersPaid;
18:04:25.509:  2:  CALL 1_R_ClientOrdersPaid;
18:04:26.353:  1:  CALL 1_R_ClientOrdersPaid;
18:04:26.978:  4:  CALL 1_R_ClientOrdersPaid;
18:04:27.024:  6:  CALL 1_R_ClientOrdersPaid;
18:04:28.056:  8:  CALL 1_R_ClientOrdersPaid;
18:04:28.103:  7:  CALL 1_R_ClientOrdersPaid;
18:04:28.103:  10:  CALL 1_R_ClientOrdersPaid;
18:04:28.462:  1:  CALL 1_R_ClientOrdersCancelled;
18:04:28.493:  9:  CALL 1_R_ClientOrdersPaid;
18:04:28.493:  5:  CALL 1_R_ClientOrdersPaid;
18:04:28.868:  2:  CALL 1_R_ClientOrdersCancelled;
18:04:28.868:  3:  CALL 1_R_ClientOrdersCancelled;
18:04:30.712:  4:  CALL 1_R_ClientOrdersCancelled;
18:04:31.149:  3:  Client 3 done in 6,8593862s
18:04:31.290:  6:  CALL 1_R_ClientOrdersCancelled;
18:04:31.509:  7:  CALL 1_R_ClientOrdersCancelled;
18:04:31.931:  10:  CALL 1_R_ClientOrdersCancelled;
18:04:32.040:  9:  CALL 1_R_ClientOrdersCancelled;
18:04:33.493:  1:  Client 1 done in 9,2031412s
18:04:33.509:  6:  Client 6 done in 9,2187662s
18:04:34.149:  4:  Client 4 done in 9,8593923s
18:04:34.196:  9:  Client 9 done in 9,9062671s
18:04:34.446:  10:  Client 10 done in 10,1562673s
18:04:34.556:  8:  CALL 1_R_ClientOrdersCancelled;
18:04:34.821:  5:  CALL 1_R_ClientOrdersCancelled;
18:04:34.978:  2:  Client 2 done in 10,6875188s
18:04:35.446:  7:  Client 7 done in 11,1562693s
18:04:35.946:  5:  Client 5 done in 11,656269s
18:04:40.353:  8:  Client 8 done in 16,0625288s
18:04:40.368:  0:  ---------------DONE - All threads completed in 16,085 seconds---------------
18:04:43:  Log file closed for RPi3 at D:\Google Drive\Masters\Report\Sections\Chapter 4\Empirical Work
\MultiClientApp\MultiClientApp\bin\Release\LOG\RPi3\20170523_180417_I1P10.txt
```

Figure 5-14: Original MCS log file example

```
18:04:17.634;0;Attempting to start MPDB status logging...
18:04:24.274;0;MPDB status logging successfully started
18:04:24.274;0;---------------Threads starting---------------
18:04:24.743;3;CALL 1_R_ClientOrdersPaid.
18:04:25.509;2;CALL 1_R_ClientOrdersPaid.
18:04:26.353;1;CALL 1_R_ClientOrdersPaid.
18:04:26.978;4;CALL 1_R_ClientOrdersPaid.
18:04:27.024;6;CALL 1_R_ClientOrdersPaid.
18:04:28.056;8;CALL 1_R_ClientOrdersPaid.
18:04:28.103;7;CALL 1_R_ClientOrdersPaid.
18:04:28.103;10;CALL 1_R_ClientOrdersPaid.
18:04:28.462;1;CALL 1_R_ClientOrdersCancelled.
18:04:28.493;9;CALL 1_R_ClientOrdersPaid.
18:04:28.493;5;CALL 1_R_ClientOrdersPaid.
18:04:28.868;2;CALL 1_R_ClientOrdersCancelled.
18:04:28.868;3;CALL 1_R_ClientOrdersCancelled.
18:04:30.712;4;CALL 1_R_ClientOrdersCancelled.
18:04:31.149;3;Client 3 done in 6,8593862s
18:04:31.290;6;CALL 1_R_ClientOrdersCancelled.
18:04:31.509;7;CALL 1_R_ClientOrdersCancelled.
18:04:31.931;10;CALL 1_R_ClientOrdersCancelled.
18:04:32.040;9;CALL 1_R_ClientOrdersCancelled.
18:04:33.493;1;Client 1 done in 9,2031412s
18:04:33.509;6;Client 6 done in 9,2187662s
18:04:34.149;4;Client 4 done in 9,8593923s
18:04:34.196;9;Client 9 done in 9,9062671s
18:04:34.446;10;Client 10 done in 10,1562673s
18:04:34.556;8;CALL 1_R_ClientOrdersCancelled.
18:04:34.821;5;CALL 1_R_ClientOrdersCancelled.
18:04:34.978;2;Client 2 done in 10,6875188s
18:04:35.446;7;Client 7 done in 11,1562693s
18:04:35.946;5;Client 5 done in 11,656269s
18:04:40.353;8;Client 8 done in 16,0625288s
18:04:40.368;0;---------------DONE - All threads completed in 16,085 seconds---------------
```

Figure 5-15: Cleaned MCS log file example

Log files were generated by Dstat only during the experiment, and not during the pilot, which requires modification as well. These comma separated value (.csv) log files are modified with the use of Notepad++ (search mode in brackets):

- Replace ".*"Dstat 0.7.2 CSV output".*\r?\n" with "" (Regular expression).
- Replace ".*"Author:","Dag.*\r?\n" with "" (Regular expression).
- Replace ".*"Host:",".*\r?\n" with "" (Regular expression).
- Replace ".*\n"system",".*\r?\n" with "" (Regular expression).
- Replace ".*"time",.*\r?\n" with "" (Regular expression).
- Replace "," with ";" (Normal).
- Replace "\n" with "\r\n" (Extended).
- Replace "\r\n\r\n" with "\r\n" (Extended).

Figure 5-16 shows the original Dstat log file with additional information at the start of the log file.

```
"Dstat 0.7.2 CSV output""Author:","Dag Wieers <dag@wieers.com>",,,,"URL:","http://dag.wieers.com/home-
made/dstat/""Host:","pc",,,,"User:","root""Cmdline:","dstat --time --cpu --mem --disk --load --output
/home/pc/dstat/csv/20170522_182819_I1P2.csv",,,,"Date:","22 May 2017 18:28:23 SAST""system","total cpu
usage",,,,,,"memory usage",,,,"dsk/total",,"load avg",,
"time","usr","sys","idl","wai","hiq","siq","used","buff","cach","free","read","writ","1m","5m","15m"22-05
18:28:23,0.782,6.292,81.224,11.591,0.0,0.111,232812544.0,10424320.0,1806454784.0,6455898112.0,3271004.541,1
7735.990,0.130,0.330,0.19022-05
18:28:24,0.500,3.0,96.500,0.0,0.0,0.0,231165952.0,10424320.0,1806458880.0,6457540608.0,0.0,61440.0,0.130,0.
330,0.19022-05
18:28:25,4.455,11.881,82.673,0.0,0.0,0.990,234364928.0,10432512.0,1806462976.0,6454329344.0,0.0,49152.0,0.1
20,0.320,0.19022-05
18:28:26,17.949,29.231,52.308,0.513,0.0,0.0,237240320.0,10432512.0,1806491648.0,6451425280.0,114688.0,0.0,0
.120,0.320,0.19022-05
18:28:27,15.152,24.747,58.081,1.010,0.0,1.010,234434560.0,10432512.0,1806663680.0,6454059008.0,81920.0,0.0,
0.120,0.320,0.190
```

Figure 5-16: Original Dstat log file example

The additional information is removed, as shown in Figure 5-17, to allow the researcher to import the log files into the data warehouse.

```
22-05 18:28:23;0.782;6.292;81.224;11.591;0.0;0.111;232812544.0;10424320.0;1806454784.0;6455898112.0;3271004.54
22-05 18:28:24;0.500;3.0;96.500;0.0;0.0;0.0;231165952.0;10424320.0;1806458880.0;6457540608.0;0.0;61440.0;0.130
22-05 18:28:25;4.455;11.881;82.673;0.0;0.0;0.990;234364928.0;10432512.0;1806462976.0;6454329344.0;0.0;49152.0;
22-05 18:28:26;17.949;29.231;52.308;0.513;0.0;0.0;237240320.0;10432512.0;1806491648.0;6451425280.0;114688.0;0.
22-05 18:28:27;15.152;24.747;58.081;1.010;0.0;1.010;234434560.0;10432512.0;1806663680.0;6454059008.0;81920.0;0
```

Figure 5-17: Cleaned Dstat log file example

5.9.2.2 Import table data

The import process of log file data is performed through MS SQL scripts written by the researcher. The script reads through each log file and captures all log file data into the relevant tables. The script identifies semicolons (“;”) as the delimiter and converts the text extracts to the right data type before it is stored in the tables. The import script for the pilot is shown in Appendix B (§B.1.1) and that of the experiment in Appendix B (§B.1.2).

Imported data are verified by performing random spot checks and comparing the number of data contained in the LOGS_FACT and LOAD_FACT tables to a log line count in the MCS and Dstat log files. The total number of log lines in the log files are obtained with the use of “Find in Files” and regular expressions in Notepad++ which match the number of records in the LOGS_FACT and LOAD_FACT table. At this stage, all data contained in the log files have successfully and accurately been imported into the MS SQL data warehouse.

This concludes the process of converting the raw data in the log files to organised data contained in the data warehouse. The data required for analysis can now be extracted from the data warehouse with the use of the Power Pivot function in Microsoft Excel.

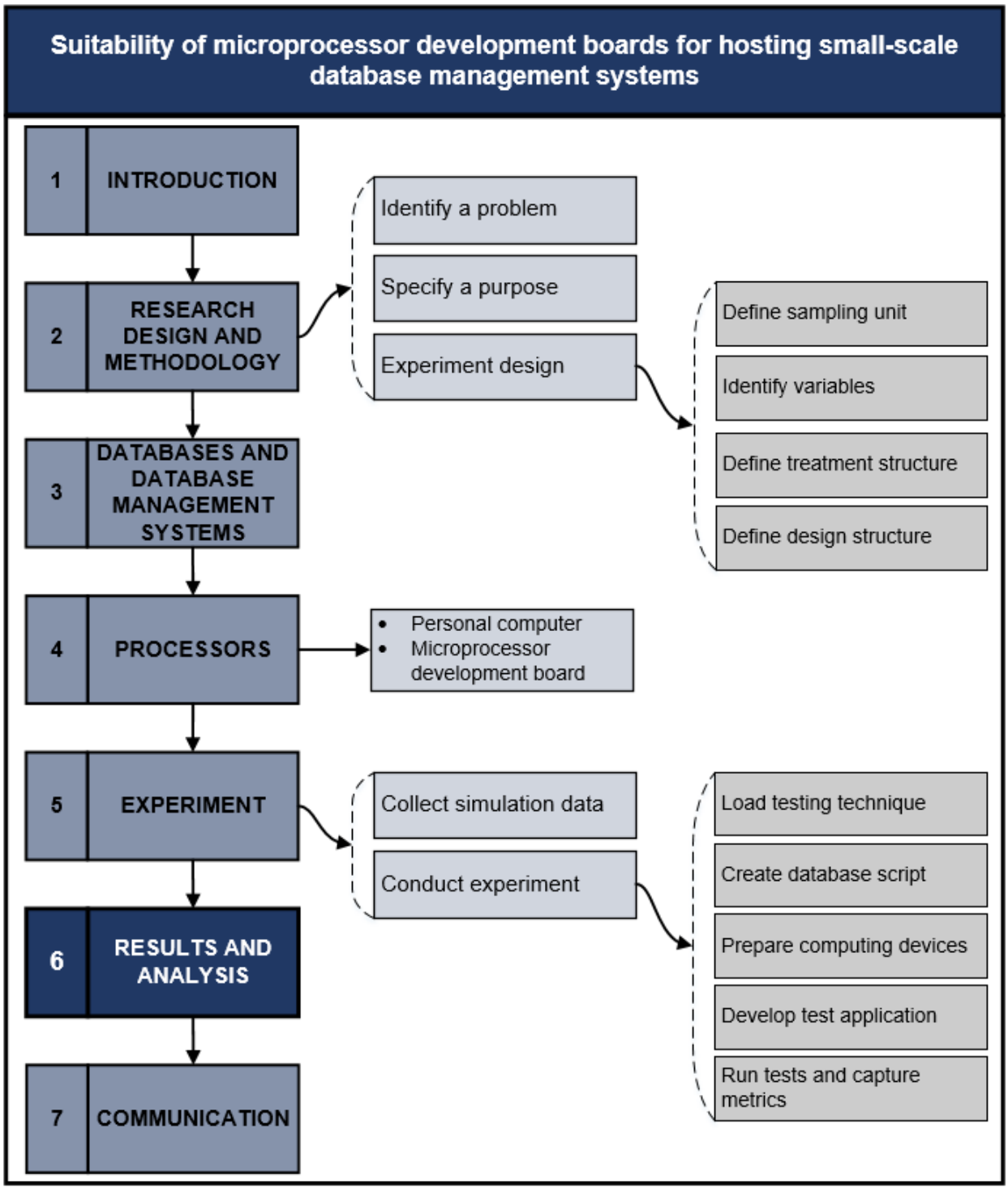
5.10 Conclusion

This chapter accurately explained how the experiment was set up and conducted to ensure repeatability. It showed how the load tests were designed and described key metrics captured from the experiment. The simulation data was generated with a script and the researcher created the SPs in the MySQL database and distributed the same database across devices. Linux Debian 8 is the OS installed on all devices. All devices were set up similarly to eliminate unnecessary and unknown variables, which allowed the experiment to extract accurate performance metrics for each device.

A stable application called MCS was developed by the researcher to simulate the act of multiple clients connecting to a device simultaneously. Two versions of the application were used, the first to conduct the pilot and the final to perform the experiment. The pilot was performed first, and through this, it was determined that the experiment is feasible. The application was responsible for capturing metrics and device resource statistics in log files. Log file data were imported into a data warehouse designed by the researcher in order to view and analyse the results in Chapter 6.

The experiment was performed successfully and advanced as expected with all devices being able to host a DBMS and allow multiple clients to query the database, except for the BBB that became unresponsive beyond Phase 15. In Chapter 6, the results from the pilot and experiment are utilised to derive meaningful statistics and draw conclusions regarding each individual device.

(P*AGE INTENTIONALLY LEFT BLANK)*


Suitability of microprocessor development boards for hosting small-scale database management systems
1
INTRODUCTION
2
RESEARCH DESIGN AND METHODOLOGY
Identify a problem
Specify a purpose
Experiment design
Define sampling unit
Identify variables
Define treatment structure
Define design structure
3
DATABASES AND DATABASE MANAGEMENT SYSTEMS
4
PROCESSORS
Personal computer
Microprocessor development board
5
EXPERIMENT
Collect simulation data
Conduct experiment
Load testing technique
Create database script
Prepare computing devices
Develop test application
Run tests and capture metrics
6
RESULTS AND ANALYSIS
7
COMMUNICATION

CHAPTER 6: RESULTS AND ANALYSIS

6.1 Introduction

The suitability of microprocessor development boards (MPDBs) for hosting small-scale database management systems (DBMSs) is determined in this chapter by analysing the data captured during the experiment. The primary objective of this study is to evaluate the effectiveness of MPDBs compared to commodity PCs for hosting small-scale DBMS implementations. This objective is addressed in this chapter by means of hypothesis testing and the determination of the point of divergence from stability.

This chapter describes how the results obtained from the pilot and the experiment performed in Chapter 5 are analysed. Pivot tables, graphs and statistical analysis software are used to analyse the metrics from the pilot and experiment. The results that were generated are used to analyse the outcome of the experiment and draw conclusions on the four devices that were tested:

- BeagleBone Black (BBB);
- Intel Edison Arduino (IEA);
- Raspberry Pi 3 (RPi3); and
- commodity personal computer, also referred to as personal computer (PC).

A clear distinction of the “client” term used in the chapter is necessary to ensure that the results and their analysis are interpreted correctly; simulated clients are referred to as threads or clients interchangeably, while records in the CLIENT table are referred to as CLIENT table records.

This results and analysis chapter is presented through the following sections: statistical approach (§6.2), pilot statistical analysis in terms of thread, load and duration metrics (§6.3), experiment statistical analysis in terms of load and duration metrics (§6.4), hypothesis testing (§6.5), the point of divergence from stability determined through analysis of the thread metrics (§6.6), experiment analysis result summary (§6.7), and finally, this chapter is concluded (§6.8).

6.2 Statistical approach

A pilot was performed and described in Chapter 5 to evaluate the feasibility of the experiment. The experiment is feasible if each device is capable of hosting a DBMS and allows multiple clients to query the DBMS simultaneously. The pilot is a scaled down and incomplete version of the

experiment. Statistical analysis in terms of the pilot is performed with the use of aggregated and summarised data obtained from pivot tables created in Microsoft Excel. The data preparation and import process that provides access to the data through pivot tables are described in Chapter 5 (§5.9). The pilot results are analysed in terms of the total number of threads that failed during each phase per device (§6.3.1); each device's load in terms of the average central processing unit (CPU) and random access memory (RAM) usage per phase (§6.3.2); and the total duration each device required per phase (§6.3.3). The statistical analysis of the pilot is concluded by summarising the results analysed and evaluating the feasibility of the experiment (§6.3.4).

Statistical analysis of the experiment results is performed with the use of the statistical analysis software, namely statistical package for the social sciences (SPSS). Organised data are required for SPSS, which is obtained and correctly structured with the use of Microsoft Excel pivot tables described in Chapter 5 Section 5.9. The load and duration metrics, obtained from the 30 iterations in the experiment, are analysed in terms of descriptive statistics and one-way analysis of variance (ANOVA) with the use of SPSS in Section 6.4. Descriptive statistics include the mean, standard deviation, standard error, minimum and maximum values, and the 95 percent confidence interval for the mean. In descriptive statistics tables generated by SPSS, the Total rows are interpreted in conjunction with the column headings. The ANOVA results provide comprehensive comparisons between devices for each phase and enable the researcher to perform hypothesis testing based on a five percent significance level.

The primary objective of this study is divided into two supporting objectives, namely hypothesis testing and divergence point determination. The hypothesis is tested on each MPDB and the commodity PC for each phase in Section 6.5. To determine the point of divergence from stability for each MPDB in Section 6.6, the thread metrics from the experiment are statistically analysed through pivot tables and graphs.

Throughout this chapter and the subsequent chapter, graphs are depicted in both two-dimensional (2D) and three-dimensional (3D) format. The only exception is Figure 6-8, where a 2D graph is sufficient. By showing both types of graphs, the following graph design problems relating to 2D and 3D are overcome:

- *2D* – When data points overlap, some data points become invisible.
- *3D* – Difficult to visualise what the actual value of a data point is.

This chapter now progresses to the statistical analysis of the results from the pilot after which the experiment results are statistically analysed.

6.3 Pilot statistical analysis

This section covers descriptive statistics relating to the pilot in Sections 6.3.1, 6.3.2 and 6.3.3. Section 6.3.4 provides a summary of the results and the feasibility evaluation for conducting the experiment.

6.3.1 Thread metrics

The number of threads that failed due to the device during each phase is shown in Table 6-1. The total threads that were started in each phase are shown in the “Total Threads Executed” column. The following is an example to facilitate the interpretation of Table 6-1: Phase 5 required each device to run 40 threads (“Total Threads Executed”) and none of the BBB threads completed successfully, the IEA had two thread failures, and the PC and RPi3 had no thread failures.

Table 6-1: Pilot thread failures per phase for each device

Phase	Total Threads Executed	Device			
		BBB	IEA	RPi3	PC
1	5	0	0	0	0
2	25	0	0	0	0
3	30	0	0	0	0
4	25	0	0	0	0
5	40	40	2	0	0
6	10	0	0	0	0
7	7	0	0	0	0
8	12	0	0	0	0
9	30	0	0	0	0
10	10	10	0	0	0
11	60	60	60	27	0
12	100	100	100	90	0
13	150	10	0	0	0
14	200	0	0	0	0
15	150	150	150	150	0
16	45	45	45	44	0
17	150	150	135	112	0
18	60	60	60	60	0
19	50	50	50	50	0
20	100	100	100	100	0
Total	**1259**	**775**	**702**	**633**	**0**

Figure 6-1 represents Table 6-1 in terms of the percentage fail rate per phase calculated from the number of threads that failed on each device divided by the total threads executed.

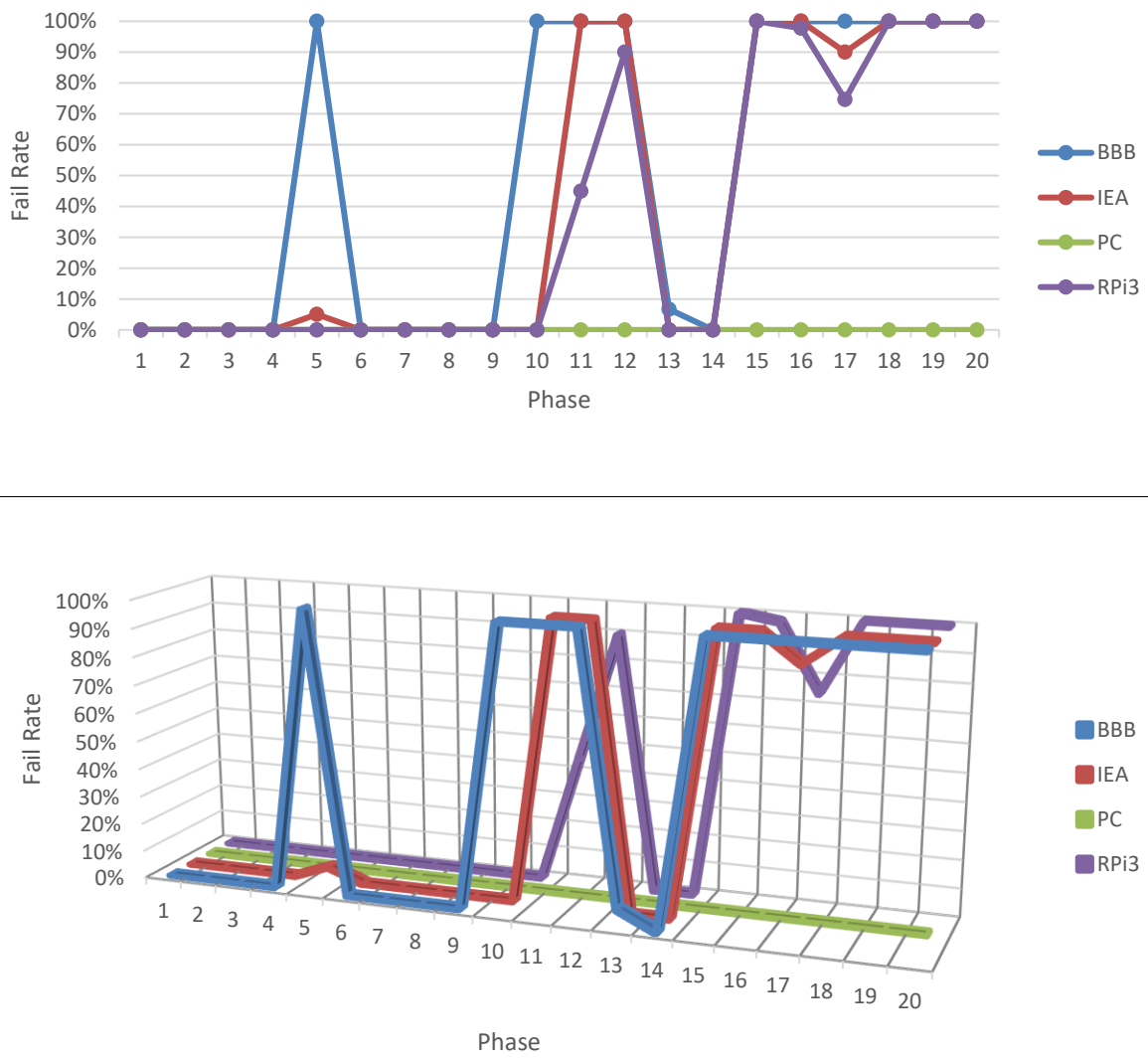


Figure 6-1: Pilot phase fail rate per device

When analysing the totals shown in Table 6-1, it is clear that, while each device had to perform 1 259 thread tasks, the PC did not have a single thread failure throughout the entirety of the pilot. The RPi3 had 633, the IEA 702, and the BBB 775 thread failures. In terms of thread execution, the PC had the highest rate of success, RPi3 second, IEA third and the BBB fourth.

6.3.2 Load metrics

This section shows the load metrics in terms of average CPU and RAM usage in each phase during the pilot. Table 6-2 shows the average CPU and RAM usage per device for each phase; Figure 6-2 illustrates the CPU usage from Table 6-2 visually.

Table 6-2: Pilot average CPU and RAM usage per phase for each device

Phase	Device							
	BBB		IEA		RPi3		PC	
	CPU (%)	RAM (%)	CPU (%)	RAM (%)	CPU (%)	RAM (%)	CPU (%)	RAM (%)
1	8.00	97.15	21.00	98.18	6.20	98.68	0.60	28.18
2	8.42	96.70	50.60	98.19	2.88	98.14	11.50	28.19
3	18.00	97.14	55.75	98.40	7.86	98.53	17.00	28.21
4	43.90	96.59	44.79	98.54	6.78	98.57	9.29	28.47
5	21.20	95.67	78.02	97.43	34.67	97.17	28.50	28.30
6	45.50	94.69	30.19	96.81	5.00	96.70	7.46	28.44
7	35.08	92.78	38.09	96.75	5.88	97.86	4.09	28.42
8	42.65	93.97	35.59	97.98	3.55	98.05	7.16	28.57
9	8.14	88.05	59.00	95.66	32.50	94.77	13.67	28.37
10	7.60	94.78	77.56	96.53	41.89	96.05	29.38	28.48
11	90.32	93.04	95.52	97.36	79.92	97.05	70.94	28.54
12	92.41	90.56	93.05	95.40	80.29	95.59	65.19	28.82
13	58.21	90.30	49.08	97.20	8.31	96.81	12.76	30.75
14	17.88	76.04	93.08	71.57	59.15	90.61	10.79	29.11
15	11.53	90.33	83.57	96.89	27.86	95.88	35.15	30.79
16	93.59	89.72	96.21	98.15	90.07	97.93	60.88	31.25
17	42.00	91.70	89.98	98.39	17.66	98.17	22.99	33.07
18	95.24	96.94	97.29	75.59	95.37	86.57	72.62	31.75
19	81.90	92.97	70.32	86.76	20.44	96.77	44.00	31.61
20	94.05	95.47	97.29	95.75	86.27	97.45	75.23	33.03
Average	**45.78**	**92.72**	**67.80**	**94.38**	**35.63**	**96.37**	**30.00**	**29.62**

When analysing the average CPU usage, it can be deduced that the IEA had a higher CPU load throughout the phases when compared to the other devices; the PC had the lowest total average CPU usage during the phases. See Phase 5 for example; the IEA CPU usage was 78.02 percent, while the CPU usage of all other devices was less than 40 percent.

The RAM usage of the MPDBs was mostly above 90 percent, while the average RAM usage of the PC was 30.00 percent. This shows that the MPDBs were using near to full capacity in terms of RAM resources and the PC had RAM resources to spare.

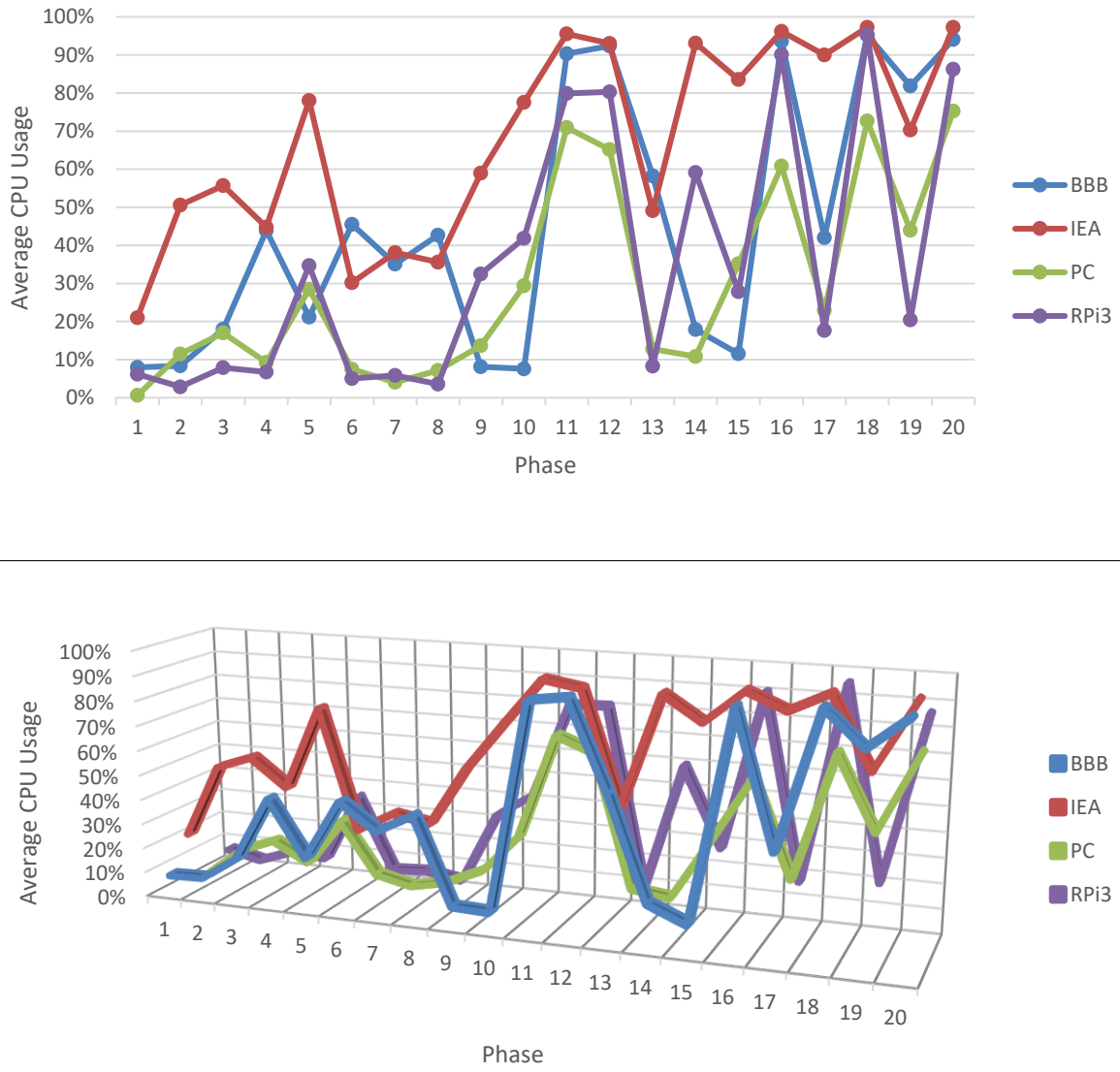


Figure 6-2: Pilot average CPU usage per phase for each device

6.3.3 Duration metrics

The duration to complete each phase on each device is tabled in Table 6-3 and illustrated in Figure 6-3. The number of threads that failed during each phase is displayed in Table 6-3, because thread failures affects phase duration. This can be seen in Phase 5, where the duration of this phase for the BBB was substantially longer than the other devices because of the BBB's high thread fail rate. The reason for this phenomenon is that execution time is increased when there are thread failures and MySQL timeouts since MCS waits for database results to be returned from the computing device.

Table 6-3: Pilot phase duration and thread fail rate per phase for each device

Phase	Device							
	BBB		IEA		RPi3		PC	
	Duration (s)	Fail Rate (%)	Duration (s)	Fail Rate (%)	Duration (s)	Fail Rate (%)	Duration (s)	Fail Rate (%)
1	2.61	0.00	1.96	0.00	0.88	0.00	0.53	0.00
2	12.28	0.00	9.82	0.00	2.93	0.00	1.84	0.00
3	37.55	0.00	12.62	0.00	31.00	0.00	2.30	0.00
4	35.80	0.00	32.99	0.00	26.98	0.00	25.96	0.00
5	166.72	100.00	85.69	5.00	49.20	0.00	10.10	0.00
6	24.17	0.00	24.23	0.00	19.46	0.00	17.85	0.00
7	20.38	0.00	17.47	0.00	14.41	0.00	14.35	0.00
8	36.48	0.00	39.49	0.00	37.44	0.00	31.59	0.00
9	42.80	0.00	33.88	0.00	9.85	0.00	6.00	0.00
10	154.83	100.00	29.62	0.00	13.49	0.00	10.07	0.00
11	178.14	100.00	175.48	100.00	104.08	45.00	26.02	0.00
12	214.22	100.00	219.93	100.00	198.71	90.00	50.25	0.00
13	380.87	6.67	340.34	0.00	369.40	0.00	303.47	0.00
14	376.29	0.00	290.05	0.00	85.60	0.00	58.57	0.00
15	231.30	100.00	253.88	100.00	214.64	100.00	87.18	0.00
16	289.36	100.00	290.22	100.00	176.97	97.78	56.93	0.00
17	711.12	100.00	524.94	90.00	565.52	74.67	318.71	0.00
18	311.14	100.00	311.99	100.00	295.63	100.00	108.59	0.00
19	250.99	100.00	257.50	100.00	232.95	100.00	70.91	0.00
20	378.92	100.00	479.02	100.00	469.32	100.00	258.52	0.00
Total	**3855.97**		**3431.11**		**2918.46**		**1459.74**	
Average		**50.33**		**39.75**		**35.37**		**0.00**

The total duration per device displayed in Table 6-3 show that the PC is the highest performer in terms of phase completion time, RPi3 second, IEA third, and the BBB the weakest performer. The RPi3 has the shortest duration amongst the MPDBs.

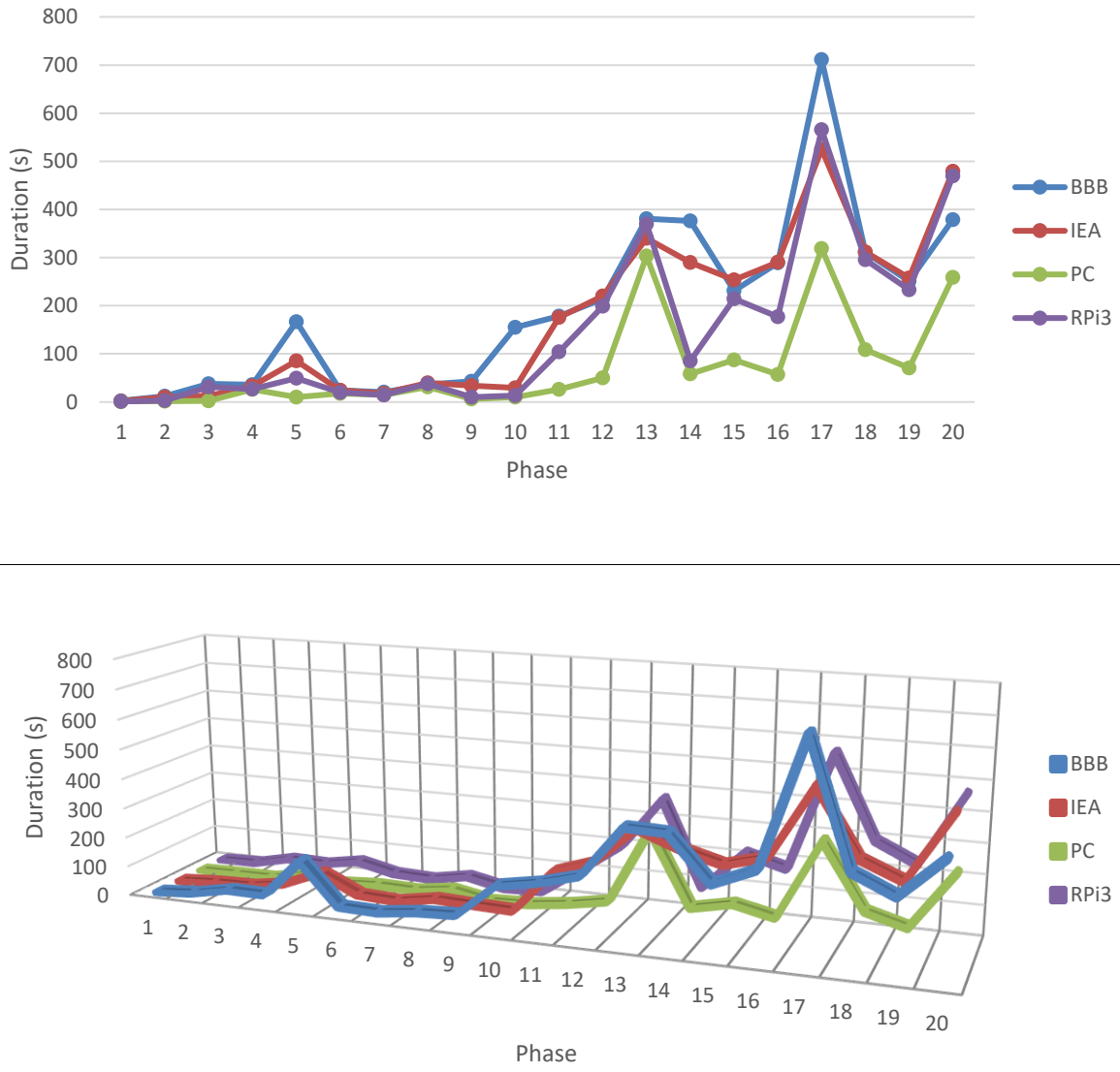


Figure 6-3: Pilot phase duration per phase for each device

6.3.4 Summary

The pilot was performed to evaluate the feasibility of the experiment. The experiment is only feasible if each device is capable of hosting a DBMS and allows multiple clients query the DBMS simultaneously.

The data analysed from the pilot show that the PC is the best performer, RPi3 second, IEA third, and BBB the weakest. Table 6-4 shows a summary of the pilot results in terms of the totals and averages from the thread (Table 6-1), load (Table 6-2) and duration (Table 6-3) metrics.

Table 6-4: Pilot result summary

Rank	Device	Total Thread Failures	Average Load		Total Duration (s)
			CPU (%)	RAM (%)	
1	PC	0	30	29.62	1459.74
2	RPi3	633	35.63	96.37	2918.46
3	IEA	702	67.8	94.38	3431.11
4	BBB	775	45.78	92.72	3855.97

The results from the pilot show that each device is able to host a DBMS and accept queries from multiple clients simultaneously, thereby indicating that the experiment is feasible. These metrics provide some insight into the performance of the tested devices, but in order to statistically analyse the devices and to conduct hypothesis testing, additional experiment data is required. Each phase was only run once on each device during the pilot; in order to thoroughly test each device, phases should be repeated numerous times to obtain each device's average performance. Therefore, the experiment includes 30 iterations on each phase for each device. The subsequent section provides the statistics from the experiment.

6.4 Experiment statistical analysis

This section covers the statistical analysis relating to the experiment in terms of load (§6.4.1) and duration metrics (§6.4.2). Each of these sections provides descriptive statistics and graphs in which the metrics are plotted. In addition, ANOVA is performed with the use of SPSS to determine if there are statistically significant differences between the load means of the devices as well as duration means between devices. With the ANOVA results, a significance probability value of greater than five percent indicates statistical significance.

As explained in Chapter 5 (§5.8), the BBB could not perform Phase 16 through 20 due to system instability, and this should be noted when analysing the results in this section.

6.4.1 Load metrics

Each device's load metrics were acquired through the use of Dstat, which was set up to log metrics relating to device resources at a one hertz frequency during the experiment. Table 6-5 shows the average CPU utilisation, RAM usage and load metrics for each phase per device. The load average considers all resources such as RAM, CPU and disk. The system load is a measurement of computational tasks that the system is performing over a duration of one minute (Hoffman, 2014). The "Average" rows show the average for all phases and iterations.

Table 6-5: Average CPU, RAM and load per phase for each device

Phase	Device											
	BBB			IEA			RPi3			PC		
	CPU (%)	RAM (%)	Load	CPU (%)	RAM (%)	Load	CPU (%)	RAM (%)	Load	CPU (%)	RAM (%)	Load
1	73.89	98.26	1.12	54.60	98.28	1.23	28.83	98.44	1.13	9.08	22.70	0.98
2	80.04	98.17	0.54	70.03	98.31	0.70	33.75	98.41	0.51	12.70	22.85	0.39
3	90.55	97.84	0.74	76.17	98.32	0.72	50.41	98.52	0.60	12.06	22.86	0.20
4	83.91	97.12	0.87	80.01	98.39	0.75	38.87	98.35	0.52	12.31	23.10	0.21
5	89.71	97.80	10.73	88.72	97.68	5.97	80.63	98.03	3.41	35.90	22.98	0.26
6	82.53	97.43	3.75	70.46	95.69	2.41	29.64	96.23	1.93	10.80	23.11	0.26
7	80.53	96.49	1.28	71.02	96.08	1.05	35.50	96.79	0.84	10.81	23.17	0.15
8	84.40	91.77	0.84	77.67	96.83	0.90	37.52	97.75	0.99	12.26	23.30	0.18
9	86.49	84.09	1.11	85.41	97.43	1.10	49.57	97.96	1.60	14.24	23.07	0.25
10	96.97	97.43	4.74	87.77	98.07	2.29	63.90	98.07	1.45	26.62	23.15	0.26
11	99.03	96.42	28.99	97.18	97.71	29.03	96.00	97.70	22.51	73.74	23.37	1.14
12	99.10	93.97	39.67	97.67	96.19	46.12	91.00	96.72	37.31	48.29	23.28	1.30
13	88.44	92.29	5.20	93.95	90.24	6.75	56.61	95.13	9.14	17.22	23.33	0.86
14	88.58	90.05	3.10	95.14	91.18	3.00	64.14	95.74	4.40	19.60	23.58	0.97
15	25.80	97.46	10.19	97.86	98.05	26.03	95.21	98.32	29.82	90.77	28.94	7.37
Average	**83.33**	**95.11**	**7.52**	**82.91**	**96.56**	**8.54**	**56.77**	**97.48**	**7.74**	**27.09**	**23.52**	**0.99**
16				98.39	98.14	31.55	95.27	97.72	21.15	73.94	31.23	5.20
17				97.14	98.32	19.91	89.58	97.55	22.63	61.40	31.30	3.24
18				98.30	97.71	36.81	97.97	97.91	30.77	83.83	31.53	5.34
19				95.20	93.26	16.93	85.56	98.23	14.93	69.17	31.60	3.92
20				98.94	97.65	67.91	98.71	97.00	53.32	93.41	32.10	11.94
Average	**83.33**	**95.11**	**7.52**	**86.58**	**96.68**	**15.06**	**65.93**	**97.53**	**12.95**	**39.41**	**25.53**	**2.22**

When considering the total average CPU usage up to and including Phase 15 in Table 6-5, it can be seen that the BBB and IEA's average CPU usage was above 80 percent, while the average CPU usage of the RPi3 was significantly lower at 56.77 percent. The PC had the lowest average CPU usage at 27.09 percent. The total average CPU usage for the 20 phases and all devices except the BBB, shows that the PC had the lowest average CPU usage, the RPi3 second and the IEA third. Lower CPU utilisation of devices indicates that such device has a faster CPU compared to devices with higher CPU usage.

The total average RAM utilisation up to and including Phase 15 in Table 6-5 shows that the MPDBs had similar average RAM figures above 95 percent. The PC had significantly lower total average RAM usage of 23.52 percent. The total average RAM usage for the 20 phases and all devices except the BBB, show very similar values when compared to each device's average for 15 phases. The average RAM usage figures indicate that the MPDBs used close to the total RAM available to each, while the PC only used a quarter of the RAM available during the phases of the experiment.

The load values from Table 6-5 are used to continue the analysis in this section, because it is an overall load indicator that takes CPU and RAM usage described above into account. According

to Hoffman (2014), a load value of one indicates that the system is fully utilised, while a load value of two suggests that the system is overloaded by 100 percent. This statement is only true for CPUs with one thread because the load value does not take into account the number of available processor threads. The system load for systems with more than one CPU thread has to be interpreted differently. For example, a system with two CPU threads and a load value of two, shows that the system is fully utilised; in other words, the load should be divided by the number of CPU threads available in a system. The number of CPU threads for each device in this study is thus detailed:

- BBB: one thread;
- IEA: two threads;
- RPi3: four threads; and
- PC: four threads.

Table 6-6 compares the adjusted (actual) average load between all devices for the first 15 phases and without the BBB for Phases 16 through 20. These adjusted (or actual) load metrics are used for the remainder of the load analysis. The total average load for Phase 1-15 shows that the BBB was under the highest load with significant decreases in load when considering the IEA, RPi3 and PC. This trend is held in the final five phases between the IEA, RPi3 and PC. The total average load for phases 1 through 15 on each device is summarised from Table 6-6:

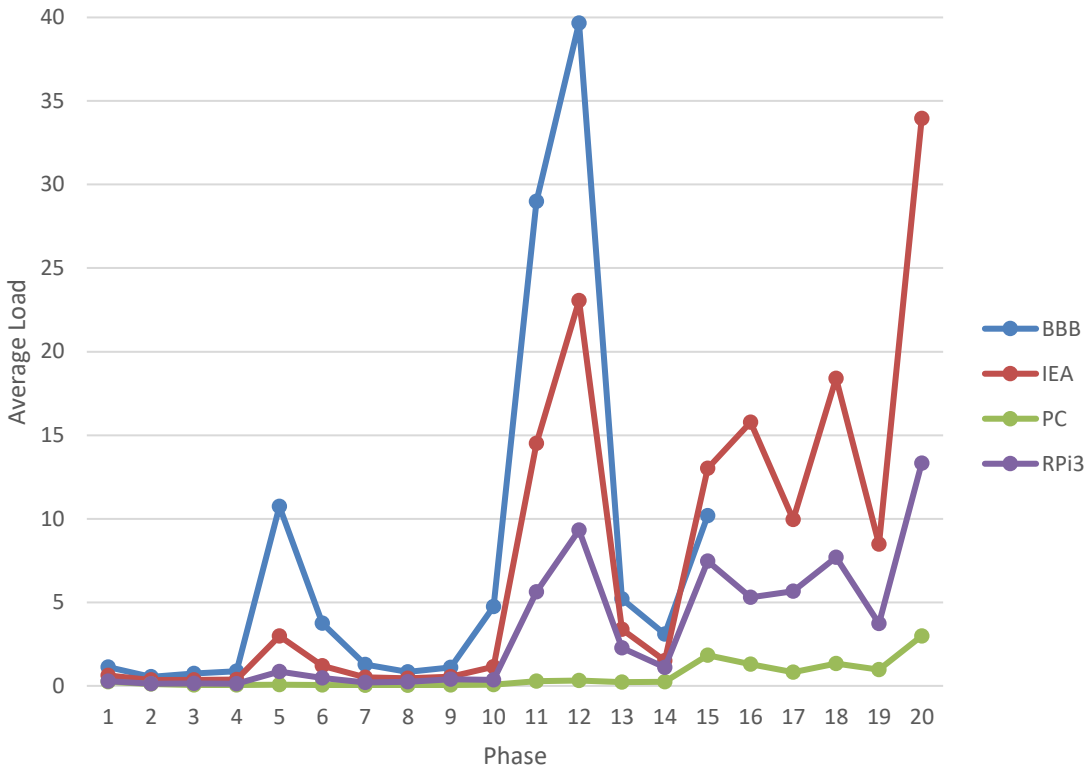
- BBB: 652 percent overloaded;
- IEA: 327 percent overloaded;
- RPi3: 94 percent overloaded; and
- PC: -75 percent overloaded (or 25 percent load).

Table 6-6: Load average per phase for each device

Phase	Device			
	BBB (%)	IEA (%)	RPi3 (%)	PC (%)
1	112	62	28	24
2	54	35	13	10
3	74	36	15	5
4	87	38	13	5
5	1073	298	85	7
6	375	120	48	6
7	128	52	21	4
8	84	45	25	4
9	111	55	40	6
10	474	115	36	7
11	2899	1452	563	29
12	3967	2306	933	32
13	520	338	228	22
14	310	150	110	24
15	1019	1301	746	184
Average	**752**	**427**	**194**	**25**
16		1577	529	130
17		996	566	81
18		1840	769	134
19		847	373	98
20		3396	1333	299
Average	**752**	**753**	**324**	**56**

The load average for each device is plotted in Figure 6-4. When the load unit on the y-axis is equal to one, the device is running at 100 percent capacity; a load value of two indicates that the device is 100 percent overloaded.

In Figure 6-4, it can be seen that the BBB had the highest load in most phases, where in Phase 11 the device was 27.99 times over capacity and in Phase 12, it was 38.67 times over capacity. The reason for this sudden load increase in Phases 11 and 12 may be due to the complexity of the stored procedure (SP) executed in Phase 11 and 12. From the load test design in Chapter 5 (Table 5-3), it can be seen that the duration baseline for each thread in Phase 11 and 12 is 8.1 and 8.5 seconds respectively, a significantly longer duration than in the previous phases as well as Phases 13 and 14. As explained in Chapter 5, the BBB is not able to perform Phases 16 through 20 owing to the device becoming unresponsive; this may be due to Phase 16 containing both SPs from Phase 11 and 12 (where load increased significantly). In other words, during Phase 16, each thread executes the SP from Phase 11 and 12 simultaneously. It is the researcher's opinion that the high amount of load applied in Phase 16 is more than the BBB is capable of handling, hence the reason for it becoming unresponsive in this phase.



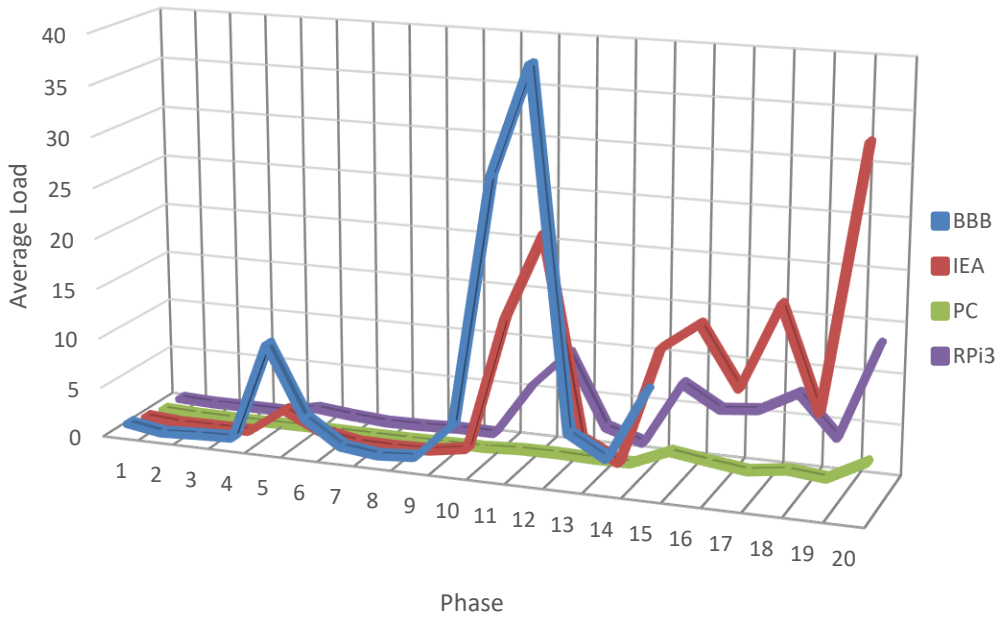


Figure 6-4: Load average per phase for each device

The descriptive statistics in terms of the load derived throughout the experiment per phase for each device are detailed in Table 6-7. This table reveals the standard deviation, standard error, the lower and upper bound 95 percent confidence interval and the minimum and maximum load metrics across all iterations. In Phase 1 the PC had 27 occurrences instead of 30. This is due to the fact that the PC completed these phase occurrences in under one second and Dstat logs load metrics at a frequency of one hertz which amounts to one second. Phases 15 through 20 do not include the BBB.

Table 6-7: Descriptive statistics for Phases 1-20 using the load metric

Phase	Device	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
1	BBB	30	1.115	0.089	0.016	1.082	1.148	0.977	1.285
	IEA	30	0.617	0.081	0.015	0.586	0.647	0.480	0.885
	PC	27	0.245	0.017	0.003	0.238	0.252	0.218	0.275
	RPi3	30	0.280	0.028	0.005	0.270	0.291	0.228	0.358
	Total	117	0.572	0.357	0.033	0.507	0.638	0.218	1.285
2	BBB	30	0.540	0.134	0.024	0.490	0.590	0.362	0.851
	IEA	30	0.348	0.065	0.012	0.324	0.373	0.236	0.476
	PC	30	0.096	0.022	0.004	0.087	0.104	0.064	0.147
	RPi3	30	0.126	0.021	0.004	0.118	0.134	0.083	0.174
	Total	120	0.277	0.196	0.018	0.242	0.313	0.064	0.851
3	BBB	30	0.741	0.229	0.042	0.655	0.826	0.374	1.274
	IEA	30	0.362	0.085	0.016	0.330	0.394	0.168	0.549
	PC	30	0.051	0.020	0.004	0.043	0.058	0.018	0.111
	RPi3	30	0.151	0.044	0.008	0.134	0.167	0.080	0.251
	Total	120	0.326	0.293	0.027	0.273	0.379	0.018	1.274
4	BBB	30	0.872	0.243	0.044	0.781	0.963	0.407	1.553
	IEA	30	0.378	0.070	0.013	0.351	0.404	0.249	0.549
	PC	30	0.051	0.025	0.005	0.041	0.060	0.010	0.119
	RPi3	30	0.132	0.043	0.008	0.115	0.148	0.070	0.276
	Total	120	0.358	0.346	0.032	0.295	0.420	0.010	1.553
5	BBB	30	10.735	0.883	0.161	10.405	11.065	8.660	12.295
	IEA	30	2.974	0.404	0.074	2.823	3.125	2.131	3.736
	PC	30	0.068	0.048	0.009	0.050	0.086	0.013	0.278
	RPi3	30	0.845	0.280	0.051	0.740	0.950	0.339	1.463
	Total	120	3.655	4.271	0.390	2.883	4.427	0.013	12.295
6	BBB	30	3.754	0.301	0.055	3.641	3.866	3.202	4.405
	IEA	30	1.206	0.179	0.033	1.139	1.273	0.906	1.531
	PC	30	0.063	0.037	0.007	0.049	0.076	0.016	0.168
	RPi3	30	0.503	0.138	0.025	0.451	0.554	0.250	0.731
	Total	120	1.381	1.447	0.132	1.120	1.643	0.016	4.405
7	BBB	30	1.277	0.146	0.027	1.222	1.331	0.932	1.495
	IEA	30	0.520	0.088	0.016	0.487	0.553	0.395	0.748
	PC	30	0.038	0.021	0.004	0.030	0.046	0.007	0.097
	RPi3	30	0.210	0.060	0.011	0.188	0.233	0.103	0.315
	Total	120	0.511	0.485	0.044	0.424	0.599	0.007	1.495
8	BBB	30	0.840	0.172	0.031	0.776	0.904	0.573	1.355
	IEA	30	0.441	0.110	0.020	0.400	0.482	0.267	0.724
	PC	30	0.041	0.028	0.005	0.031	0.052	0.008	0.127
	RPi3	30	0.226	0.083	0.015	0.195	0.257	0.050	0.401
	Total	120	0.387	0.318	0.029	0.329	0.444	0.008	1.355

Phase	Device	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
9	BBB	30	1.109	0.221	0.040	1.026	1.191	0.653	1.750
	IEA	30	0.551	0.139	0.025	0.499	0.603	0.323	0.909
	PC	30	0.058	0.023	0.004	0.049	0.066	0.016	0.109
	RPi3	30	0.365	0.141	0.026	0.313	0.418	0.106	0.691
	Total	120	0.521	0.411	0.038	0.446	0.595	0.016	1.750
10	BBB	30	4.728	0.202	0.037	4.652	4.803	4.325	5.143
	IEA	30	1.144	0.113	0.021	1.102	1.186	0.939	1.406
	PC	30	0.067	0.031	0.006	0.055	0.078	0.024	0.141
	RPi3	30	0.364	0.103	0.019	0.326	0.402	0.125	0.602
	Total	120	1.576	1.874	0.171	1.237	1.914	0.024	5.143
11	BBB	30	29.001	0.930	0.170	28.654	29.349	27.584	30.854
	IEA	30	14.525	0.583	0.107	14.307	14.742	13.252	15.815
	PC	30	0.291	0.156	0.029	0.233	0.350	0.102	0.699
	RPi3	30	5.624	0.505	0.092	5.436	5.813	4.294	6.283
	Total	120	12.360	10.932	0.998	10.384	14.336	0.102	30.854
12	BBB	30	39.658	2.384	0.435	38.768	40.548	34.447	43.903
	IEA	30	23.084	1.023	0.187	22.702	23.466	21.201	24.948
	PC	30	0.324	0.159	0.029	0.264	0.383	0.080	0.881
	RPi3	30	9.216	1.117	0.204	8.799	9.633	7.299	11.249
	Total	120	18.070	14.998	1.369	15.359	20.781	0.080	43.903
13	BBB	30	5.207	0.679	0.124	4.953	5.460	4.020	6.828
	IEA	30	3.378	0.284	0.052	3.272	3.485	2.689	4.005
	PC	30	0.228	0.071	0.013	0.201	0.254	0.110	0.472
	RPi3	30	2.295	0.223	0.041	2.211	2.378	1.816	2.731
	Total	120	2.777	1.850	0.169	2.442	3.111	0.110	6.828
14	BBB	30	3.107	0.297	0.054	2.996	3.218	2.372	3.652
	IEA	30	1.498	0.209	0.038	1.420	1.576	1.264	2.217
	PC	30	0.239	0.042	0.008	0.223	0.254	0.169	0.329
	RPi3	30	1.108	0.190	0.035	1.037	1.179	0.601	1.396
	Total	120	1.488	1.064	0.097	1.295	1.680	0.169	3.652
15	BBB	30	40.077	9.129	1.667	36.668	43.486	0.714	63.050
	IEA	30	13.017	1.196	0.218	12.570	13.464	11.220	16.395
	PC	30	1.854	0.242	0.044	1.763	1.944	1.216	2.278
	RPi3	30	7.452	0.458	0.084	7.281	7.623	6.660	8.480
	Total	120	15.600	15.421	1.408	12.812	18.387	0.714	63.050
16	BBB	0	.	.	.	.	.	.	.
	IEA	30	15.777	0.452	0.083	15.609	15.946	14.839	16.539
	PC	30	1.309	0.222	0.040	1.226	1.392	0.912	2.030
	RPi3	30	5.287	0.445	0.081	5.121	5.453	4.471	6.056
	Total	90	7.458	6.149	0.648	6.170	8.746	0.912	16.539
17	BBB	0	.	.	.	.	.	.	.
	IEA	30	9.990	1.046	0.191	9.599	10.380	8.368	13.214
	PC	30	0.840	0.252	0.046	0.746	0.934	0.422	1.482
	RPi3	30	5.663	0.715	0.130	5.396	5.930	4.259	7.327
	Total	90	5.497	3.830	0.404	4.695	6.300	0.422	13.214
18	BBB	0	.	.	.	.	.	.	.
	IEA	30	18.403	1.099	0.201	17.992	18.813	15.741	19.899
	PC	30	1.354	0.480	0.088	1.175	1.533	0.537	2.097
	RPi3	30	7.675	0.528	0.096	7.478	7.872	6.404	8.811
	Total	90	9.144	7.116	0.750	7.654	10.634	0.537	19.899
19	BBB	0	.	.	.	.	.	.	.
	IEA	30	8.457	0.643	0.117	8.217	8.698	7.202	9.805
	PC	30	1.012	0.355	0.065	0.879	1.144	0.422	1.600
	RPi3	30	3.725	0.354	0.065	3.593	3.857	3.108	4.510

Phase	Device	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
	Total	90	4.398	3.129	0.330	3.743	5.053	0.422	9.805
20	BBB	0	.	.	.	.	.	.	.
	IEA	30	33.958	0.681	0.124	33.704	34.213	32.752	35.288
	PC	30	2.997	0.916	0.167	2.655	3.339	1.629	4.781
	RPi3	30	13.338	0.657	0.120	13.092	13.583	11.342	14.446
	Total	90	16.764	12.964	1.367	14.049	19.480	1.629	35.288

The total standard deviation in terms of load across devices per phase from Table 6-7 shows that Phases 5, 6, 10, 13, 14, 16, 17, 18 and 19 have a high total standard deviation between 1.064 and 7.116. The high total standard deviation for Phases 5, 6 and 10 is due to the high mean load of the BBB. The high total standard deviation for Phases 13, 14, 16, 17, 18 and 19 is due to the combined high mean load of the MPDBs compared to the low mean load of the PC. Phases 11, 12, 15 and 20 have a very high total standard deviation between 10.932 and 15.421 due to the combined high mean load of the MPDBs compared to the low mean load of the PC.

High spreads of the minimum and maximum load in phases are seen in Phases 5, 11, 12, 15, 16, 17, 18, 19 and 20. The spread in load for these phases is due to the MPDBs having significantly higher load when compared to the load of the PC. This is true, except for Phase 15, where the BBB showed a minimum load of 0.714, a maximum of 63.050 and a mean load of 40.077. The minimum load shown for the BBB is due to an unknown error internal to the device, which did not respond to client requests in Iteration 8 Phase 15. While the BBB was in this state, it did not execute any of the queries sent to it from MCS, explaining why it was not under load.

The one-way ANOVA results for the load metric are shown in Table 6-8 for each phase. The groups for Phases 1-15 include all devices in the study while the final five phases only show the results for the RPi3, IEA and PC. It can be seen that through Phases 1 to 20, the significance probability (“p” or “Sig.”) is smaller than five percent (0.05), thus p (0.00) < 0.05. With the significance probability below five percent, the null hypothesis can be rejected. It can therefore be stated that the MPDBs have significantly higher load than the PC throughout all phases. In other words, the MPDBs need a larger portion of total available resources, when compared to the PC, to perform the same task. However, Phases 16 to 20 were not performed on the BBB and Phase 15 is statistically out of control because of the exceptionally high standard deviation caused by the BBB in this phase. Phases 15 through 20 can therefore not be considered as valid results in Table 6-8.

Table 6-8: ANOVA for Phases 1-20 using the load metric

Phase	Group	Sum of Squares	df	Mean Square	F	Sig. (p)
1	Between Groups	14.342	3	4.781	1200.311	0.000
	Within Groups	0.450	113	0.004		
	Total	14.792	116			
2	Between Groups	3.893	3	1.298	224.761	0.000
	Within Groups	0.670	116	0.006		
	Total	4.563	119			
3	Between Groups	8.394	3	2.798	179.647	0.000
	Within Groups	1.807	116	0.016		
	Total	10.201	119			
4	Between Groups	12.311	3	4.104	246.870	0.000
	Within Groups	1.928	116	0.017		
	Total	14.239	119			
5	Between Groups	2140.635	3	713.545	2785.058	0.000
	Within Groups	29.720	116	0.256		
	Total	2170.355	119			
6	Between Groups	245.108	3	81.703	2286.248	0.000
	Within Groups	4.145	116	0.036		
	Total	249.254	119			
7	Between Groups	26.996	3	8.999	1088.964	0.000
	Within Groups	0.959	116	0.008		
	Total	27.955	119			
8	Between Groups	10.606	3	3.535	286.516	0.000
	Within Groups	1.431	116	0.012		
	Total	12.038	119			
9	Between Groups	17.556	3	5.852	264.145	0.000
	Within Groups	2.570	116	0.022		
	Total	20.126	119			
10	Between Groups	416.008	3	138.669	8511.015	0.000
	Within Groups	1.890	116	0.016		
	Total	417.898	119			
11	Between Groups	14179.317	3	4726.439	12732.562	0.000
	Within Groups	43.060	116	0.371		
	Total	14222.377	119			
12	Between Groups	26535.457	3	8845.152	4419.608	0.000
	Within Groups	232.156	116	2.001		
	Total	26767.613	119			
13	Between Groups	389.848	3	129.949	870.768	0.000
	Within Groups	17.311	116	0.149		
	Total	407.159	119			
14	Between Groups	129.840	3	43.280	1017.811	0.000
	Within Groups	4.933	116	0.043		
	Total	134.773	119			
15	Between Groups	25834.099	3	8611.366	405.035	0.000
	Within Groups	2466.250	116	21.261		
	Total	28300.349	119			
16	Between Groups	3351.996	2	1675.998	11136.535	0.000
	Within Groups	13.093	87	0.150		
	Total	3365.089	89			
17	Between Groups	1256.989	2	628.495	1130.630	0.000
	Within Groups	48.362	87	0.556		
	Total	1305.351	89			
18	Between Groups	4456.862	2	2228.431	3894.494	0.000
	Within Groups	49.781	87	0.572		
	Total	4506.643	89			

Phase	Group	Sum of Squares	df	Mean Square	F	Sig. (p)
19	Between Groups	851.956	2	425.978	1922.431	0.000
	Within Groups	19.278	87	0.222		
	Total	871.233	89			
20	Between Groups	14907.327	2	7453.664	12880.328	0.000
	Within Groups	50.346	87	0.579		
	Total	14957.673	89			

Phase 15 in Table 6-8 has a significantly higher mean square within groups than in other phases, which is due to the BBB Iteration 8 anomaly. The anomaly also causes a Type I statistical error in the duration metric analysis. This anomaly, and the fact that the BBB was not capable of performing the final five phases, are addressed by repeating the one-way ANOVA for Phases 15 to 20 without the BBB, shown in Table 6-9. With the significance probability below five percent for all phases in this table, the null hypothesis can be rejected. It can therefore be stated that the RPi3 and IEA have significantly higher loads than the PC throughout Phases 15 to 20.

Table 6-9: ANOVA for Phases 15-20 without the BBB using the load metric

Phase	Group	Sum of Squares	df	Mean Square	F	Sig. (p)
15	Between Groups	1869.346	2	934.673	1650.168	0.000
	Within Groups	49.278	87	0.566		
	Total	1918.624	89			
16	Between Groups	3351.996	2	1675.998	11136.535	0.000
	Within Groups	13.093	87	0.150		
	Total	3365.089	89			
17	Between Groups	1256.989	2	628.495	1130.630	0.000
	Within Groups	48.362	87	0.556		
	Total	1305.351	89			
18	Between Groups	4456.862	2	2228.431	3894.494	0.000
	Within Groups	49.781	87	0.572		
	Total	4506.643	89			
19	Between Groups	851.956	2	425.978	1922.431	0.000
	Within Groups	19.278	87	0.222		
	Total	871.233	89			
20	Between Groups	14907.327	2	7453.664	12880.328	0.000
	Within Groups	50.346	87	0.579		
	Total	14957.673	89			

Table 6-10 provides load results for each device compared directly to other devices in each phase where the significance probability is greater than five percent. When considering the significance probability, it can be seen that the mean of the load for the RPi3 and PC are similar in Phases 1, 2 and 4. The RPi3 is the only device with similar device load compared to the PC in these phases; in no phase were the loads of the IEA or BBB similar to that of the PC.

Table 6-10: ANOVA post hoc multiple-device comparisons using the load metric

Dependent Variable (Phase)	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
1	BBB	IEA	0.498	0.016	0.000	0.456	0.541
		PC	0.870	0.017	0.000	0.826	0.914
		RPi3	0.835	0.016	0.000	0.792	0.877
	IEA	BBB	-0.498	0.016	0.000	-0.541	-0.456
		PC	0.372	0.017	0.000	0.328	0.415
		RPi3	0.336	0.016	0.000	0.294	0.379
	PC	BBB	-0.870	0.017	0.000	-0.914	-0.826
		IEA	-0.372	0.017	0.000	-0.415	-0.328
		RPi3	**-0.035**	**0.017**	**0.155**	**-0.079**	**0.008**
	RPi3	BBB	-0.835	0.016	0.000	-0.877	-0.792
		IEA	-0.336	0.016	0.000	-0.379	-0.294
		PC	**0.035**	**0.017**	**0.155**	**-0.008**	**0.079**
2	BBB	IEA	0.191	0.020	0.000	0.140	0.242
		PC	0.444	0.020	0.000	0.393	0.495
		RPi3	0.414	0.020	0.000	0.362	0.465
	IEA	BBB	-0.191	0.020	0.000	-0.242	-0.140
		PC	0.253	0.020	0.000	0.202	0.304
		RPi3	0.222	0.020	0.000	0.171	0.274
	PC	BBB	-0.444	0.020	0.000	-0.495	-0.393
		IEA	-0.253	0.020	0.000	-0.304	-0.202
		RPi3	**-0.030**	**0.020**	**0.411**	**-0.082**	**0.021**
	RPi3	BBB	-0.414	0.020	0.000	-0.465	-0.362
		IEA	-0.222	0.020	0.000	-0.274	-0.171
		PC	**0.030**	**0.020**	**0.411**	**-0.021**	**0.082**
4	BBB	IEA	0.494	0.033	0.000	0.408	0.581
		PC	0.821	0.033	0.000	0.735	0.908
		RPi3	0.740	0.033	0.000	0.654	0.827
	IEA	BBB	-0.494	0.033	0.000	-0.581	-0.408
		PC	0.327	0.033	0.000	0.240	0.414
		RPi3	0.246	0.033	0.000	0.159	0.333
	PC	BBB	-0.821	0.033	0.000	-0.908	-0.735
		IEA	-0.327	0.033	0.000	-0.414	-0.240
		RPi3	**-0.081**	**0.033**	**0.076**	**-0.168**	**0.006**
	RPi3	BBB	-0.740	0.033	0.000	-0.827	-0.654
		IEA	-0.246	0.033	0.000	-0.333	-0.159
		PC	**0.081**	**0.033**	**0.076**	**-0.006**	**0.168**

The complete table relating to Table 6-10 for Phase 1 to 15 can be seen in Appendix C (§C.1.1) and the table containing Phases 15 to 20 without the BBB in Appendix C (§C.1.2). The statistical analysis of the load metrics is hereby concluded; the following section covers the analysis of the duration metrics.

6.4.2 Duration metrics

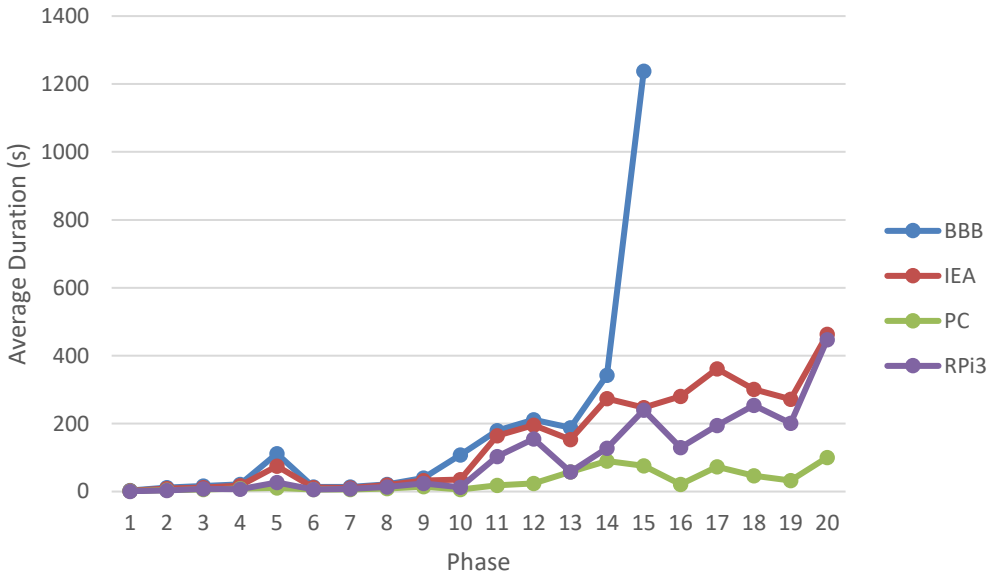
The average phase duration in seconds and the average phase fail rate for each device is shown in Table 6-11. The average phase fail rate in terms of the number of threads that failed is included for reference because phase duration is increased when there are thread failures and MySQL timeouts. This is because MCS waits for database results to be returned from the computing device until the timeout period of 30 seconds (for each thread) is reached. See the increased duration in BBB-Phase 5 (111.34s) where most threads failed compared to BBB-Phase 4 (21.28s) and BBB-Phase 6 (13.46s), where all threads succeeded.

The duration from Table 6-11 provides a total average for Phases 1 through 15 as well as for all phases; where the "Duration" total average is calculated as the sum of average duration and the "Fail Rate" total average, calculated as an average fail rate across phases. By comparing the total average duration for Phases 1-15, it is clear that the PC was the fastest and the BBB the slowest. Among the MPDBs, the RPi3 was the fastest and the IEA ranked second. Note that the RPi3 average duration for Phases 1, 2, 4 and 6 is faster than the PC's average duration in these phases. It can be seen that from Phase 7 the RPi3 did not match the PC's performance. The reason for this is not clear – the PC was expected to perform the best across all phases.

Table 6-11: Average phase duration per device

Phase	Device							
	BBB		IEA		RPi3		PC	
	Duration (s)	Fail Rate (%)	Duration (s)	Fail Rate (%)	Duration (s)	Fail Rate (%)	Duration (s)	Fail Rate (%)
1	2.96	0.00	2.03	0.00	0.84	0.00	1.23	0.00
2	11.48	0.00	8.87	0.00	3.33	0.00	4.15	0.00
3	16.26	0.00	12.00	0.00	7.48	0.00	6.23	0.00
4	21.28	0.00	17.28	0.00	7.14	0.00	8.89	0.00
5	111.34	99.58	74.61	5.03	26.81	0.00	10.94	0.00
6	13.46	0.00	11.33	0.00	5.67	0.00	6.12	0.00
7	13.37	1.05	11.01	0.00	7.74	0.00	5.60	0.00
8	21.53	1.89	18.93	0.00	13.41	0.00	8.56	0.00
9	39.69	4.06	32.03	0.00	24.12	0.00	14.11	0.00
10	108.16	30.33	35.48	0.00	13.08	0.00	6.26	0.00
11	179.63	100.00	164.44	100.00	102.77	94.75	18.64	0.00
12	211.12	92.27	195.79	99.42	155.21	83.07	23.78	0.00
13	187.91	0.00	152.69	0.00	57.94	0.00	57.96	0.00
14	342.07	0.25	273.75	0.00	127.24	0.00	89.77	0.00
15	1238.35	99.99	247.59	100.00	239.73	100.00	75.85	0.00
Average	**2518.61**	**28.63**	**1257.83**	**20.30**	**792.51**	**18.52**	**338.09**	**0.00**
16		100.00	280.69	100.00	129.71	70.43	21.34	0.00
17		100.00	361.03	99.57	194.44	28.99	73.18	0.00
18		100.00	300.73	100.00	253.63	100.00	46.33	0.00
19		100.00	271.96	99.67	201.19	100.00	32.77	0.00
20		100.00	462.81	100.00	447.58	99.53	100.56	0.00
Average	**2518.59**	**46.47**	**2935.05**	**40.18**	**2019.06**	**33.84**	**612.29**	**0.00**

The results from Table 6-11 are also shown graphically in Figure 6-5. The BBB shows an increasing duration trend from Phase 10 through 15; in Phase 15 the BBB duration was significantly higher than the other devices. This is partly due to the unknown error internal to the device in Iteration 8, described in the previous section, where duration was significantly higher than the mean duration for this phase of the BBB. This high duration in Phase 15 of the BBB also gives some insight into the fact that it was not capable of performing the final five phases.



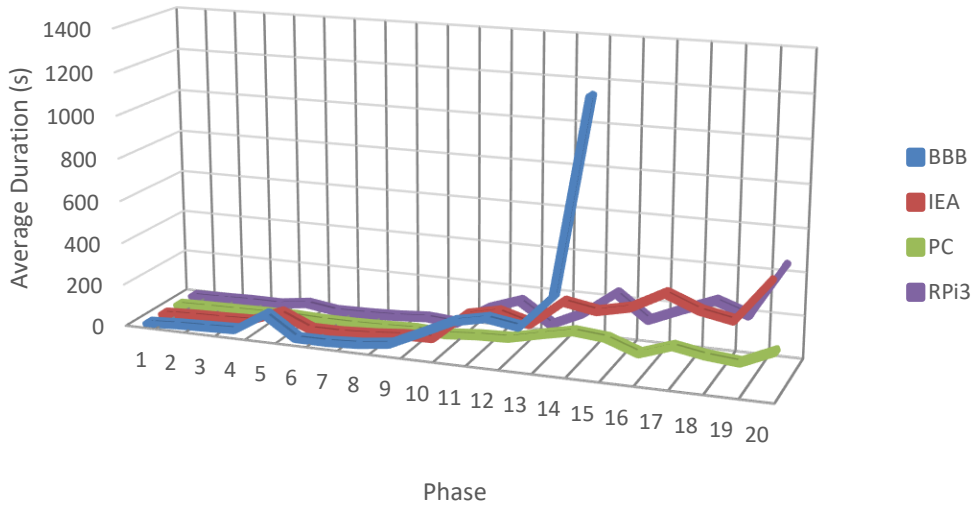


Figure 6-5: Average phase duration per device

Table 6-12 describes the experiment descriptive statistics in terms of the duration metric in seconds. This table reveals the standard deviation, standard error, the lower and upper bound 95 percent confidence interval and the minimum and maximum duration metrics across all iterations.

Table 6-12: Descriptive statistics for Phases 1-20 using the duration metric

Phase	Device	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
1	BBB	30	2.964	0.542	0.099	2.761	3.166	2.133	3.957
	IEA	30	2.030	0.076	0.014	2.002	2.059	1.931	2.329
	PC	30	1.232	0.297	0.054	1.121	1.343	0.457	1.983
	RPi3	30	0.840	0.148	0.027	0.785	0.895	0.724	1.476
	Total	120	1.766	0.876	0.080	1.608	1.925	0.457	3.957
2	BBB	30	11.477	1.153	0.210	11.046	11.907	9.851	14.108
	IEA	30	8.867	0.245	0.045	8.776	8.959	8.589	9.925
	PC	30	4.147	1.953	0.356	3.418	4.876	1.606	6.703
	RPi3	30	3.333	0.461	0.084	3.161	3.505	3.027	4.998
	Total	120	6.956	3.563	0.325	6.312	7.600	1.606	14.108
3	BBB	30	16.261	0.591	0.108	16.040	16.481	15.283	17.897
	IEA	30	12.003	0.197	0.036	11.930	12.077	11.623	12.364
	PC	30	6.230	2.584	0.472	5.265	7.194	1.963	12.278
	RPi3	30	7.478	1.055	0.193	7.083	7.872	5.961	9.561
	Total	120	10.493	4.222	0.385	9.730	11.256	1.963	17.897
4	BBB	30	21.278	1.907	0.348	20.566	21.990	18.772	28.148
	IEA	30	17.281	0.215	0.039	17.201	17.362	17.005	18.121
	PC	30	8.890	3.813	0.696	7.466	10.314	3.038	14.674
	RPi3	30	7.141	1.203	0.220	6.691	7.590	5.736	9.981
	Total	120	13.647	6.260	0.571	12.516	14.779	3.038	28.148
5	BBB	30	111.342	4.039	0.737	109.834	112.850	100.574	116.185
	IEA	30	74.610	5.937	1.084	72.393	76.826	64.415	85.612
	PC	30	10.941	2.500	0.456	10.007	11.874	6.588	16.197
	RPi3	30	26.813	2.580	0.471	25.849	27.776	22.139	31.759
	Total	120	55.926	40.022	3.653	48.692	63.160	6.588	116.185
6	BBB	30	13.457	0.936	0.171	13.108	13.807	12.519	15.640
	IEA	30	11.325	0.696	0.127	11.065	11.585	10.581	13.212
	PC	30	6.125	2.129	0.389	5.330	6.920	1.975	9.124
	RPi3	30	5.665	2.204	0.402	4.842	6.488	3.736	12.429
	Total	120	9.143	3.723	0.340	8.470	9.816	1.975	15.640
7	BBB	30	13.371	1.345	0.246	12.869	13.874	11.660	16.023
	IEA	30	11.013	1.700	0.310	10.379	11.648	9.758	16.804
	PC	30	5.601	2.334	0.426	4.730	6.473	1.774	8.384
	RPi3	30	7.741	2.971	0.543	6.631	8.850	3.534	19.070
	Total	120	9.432	3.686	0.336	8.765	10.098	1.774	19.070
8	BBB	30	21.529	1.626	0.297	20.922	22.137	19.205	26.483
	IEA	30	18.925	2.937	0.536	17.829	20.022	16.486	26.642
	PC	30	8.562	3.801	0.694	7.143	9.982	3.051	13.858
	RPi3	30	13.408	8.402	1.534	10.271	16.545	6.065	33.475
	Total	120	15.606	6.987	0.638	14.343	16.869	3.051	33.475
9	BBB	30	39.685	2.933	0.535	38.590	40.780	34.216	47.381
	IEA	30	32.029	1.374	0.251	31.516	32.542	30.545	35.175
	PC	30	14.111	6.458	1.179	11.700	16.522	5.204	24.625
	RPi3	30	24.119	9.546	1.743	20.555	27.684	11.039	38.459
	Total	120	27.486	11.207	1.023	25.460	29.512	5.204	47.381
10	BBB	30	108.156	12.757	2.329	103.392	112.920	89.450	134.384

Phase	Device	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
	IEA	30	35.484	3.496	0.638	34.178	36.789	28.950	43.937
	PC	30	6.265	1.910	0.349	5.551	6.978	3.512	11.568
	RPi3	30	13.077	2.271	0.415	12.229	13.926	9.467	18.669
	Total	120	40.745	41.111	3.753	33.314	48.176	3.512	134.384
11	BBB	30	179.629	4.525	0.826	177.939	181.319	167.747	183.007
	IEA	30	164.438	6.039	1.103	162.183	166.693	154.688	175.240
	PC	30	18.638	1.671	0.305	18.014	19.262	15.417	23.257
	RPi3	30	102.774	2.346	0.428	101.898	103.649	97.764	107.220
	Total	120	116.370	63.732	5.818	104.850	127.890	15.417	183.007
12	BBB	30	211.118	5.600	1.022	209.027	213.208	200.856	219.887
	IEA	30	195.791	8.207	1.498	192.727	198.856	179.863	209.963
	PC	30	23.778	14.767	2.696	18.264	29.292	11.377	98.995
	RPi3	30	155.206	19.530	3.566	147.913	162.499	119.907	185.147
	Total	120	146.473	75.174	6.862	132.885	160.062	11.377	219.887
13	BBB	30	187.908	10.394	1.898	184.026	191.789	174.284	219.259
	IEA	30	152.691	1.331	0.243	152.194	153.188	151.005	155.961
	PC	30	57.961	21.332	3.895	49.996	65.927	24.591	90.469
	RPi3	30	57.939	5.320	0.971	55.952	59.925	48.451	75.947
	Total	120	114.125	59.017	5.387	103.457	124.792	24.591	219.259
14	BBB	30	342.073	20.317	3.709	334.486	349.659	321.109	396.585
	IEA	30	273.746	2.805	0.512	272.699	274.793	270.385	282.106
	PC	30	89.773	35.105	6.409	76.665	102.882	48.338	146.011
	RPi3	30	127.244	12.775	2.332	122.474	132.014	101.522	163.366
	Total	120	208.209	105.980	9.675	189.052	227.366	48.338	396.585
15	BBB	30	1238.345	5236.249	956.004	-716.902	3193.593	258.619	28961.844
	IEA	30	247.589	6.759	1.234	245.065	250.112	230.242	258.462
	PC	30	75.853	5.699	1.040	73.725	77.981	62.886	84.985
	RPi3	30	239.731	7.475	1.365	236.939	242.522	223.809	250.047
	Total	120	450.379	2625.880	239.709	-24.268	925.027	62.886	28961.844
16	BBB	0	.	.	.	.	.	.	.
	IEA	30	280.688	5.620	1.026	278.589	282.786	269.746	289.587
	PC	30	21.335	2.641	0.482	20.349	22.321	17.626	27.723
	RPi3	30	129.707	7.473	1.364	126.916	132.497	111.739	143.277
	Total	90	143.910	107.095	11.289	121.479	166.341	17.626	289.587
17	BBB	0	.	.	.	.	.	.	.
	IEA	30	361.031	18.040	3.294	354.295	367.768	330.544	396.622
	PC	30	73.184	12.352	2.255	68.572	77.797	52.828	92.776
	RPi3	30	194.442	5.023	0.917	192.566	196.318	186.780	205.242
	Total	90	209.553	119.348	12.580	184.556	234.550	52.828	396.622
18	BBB	0	.	.	.	.	.	.	.
	IEA	30	300.732	4.802	0.877	298.939	302.525	289.596	307.325
	PC	30	46.331	4.058	0.741	44.816	47.847	40.443	58.914
	RPi3	30	253.631	19.542	3.568	246.333	260.928	199.895	292.609
	Total	90	200.231	111.745	11.779	176.827	223.636	40.443	307.325
19	BBB	0	.	.	.	.	.	.	.
	IEA	30	271.965	11.409	2.083	267.704	276.225	233.751	285.981
	PC	30	32.771	4.558	0.832	31.069	34.473	24.482	40.185
	RPi3	30	201.187	21.921	4.002	193.001	209.372	150.237	239.477
	Total	90	168.641	101.902	10.741	147.298	189.984	24.482	285.981
20	BBB	0	.	.	.	.	.	.	.
	IEA	30	462.813	6.363	1.162	460.437	465.190	447.534	471.515
	PC	30	100.556	4.748	0.867	98.783	102.329	93.429	115.393
	RPi3	30	447.582	9.722	1.775	443.952	451.212	426.761	471.142
	Total	90	336.984	168.385	17.749	301.716	372.251	93.429	471.515

The total standard deviation in terms of duration across devices per phase from Table 6-12 shows that Phases 5 and 10 through 20 have a high total standard deviation due to the MPDBs requiring a longer amount of time to complete these phases. Phase 15 shows an exceptionally high total standard deviation of 2625.880; this is due to the high mean duration of the BBB in this phase.

High spreads of the minimum and maximum duration that are greater than 100 seconds are seen in Phases 5 and 10 through 20. This spread in duration is due to the MPDBs having significantly higher duration when compared to the duration of the PC. The maximum duration in Phase 15 is 28961.844 seconds while the total mean is 450.379 seconds; this high duration is due to the BBB's unknown internal error that occurred in Iteration 8 Phase 15, as described in the previous section.

The one-way ANOVA results for the duration metric are shown in Table 6-13 for each phase. The groups for Phases 1-15 include all devices in the study while the final five phases only show the results for the RPi3, IEA and PC. It can be seen that through Phases 1 to 20, the significance probability is smaller than five percent (0.05), except for Phase 15. This is classified as a Type I statistical error or false positive and can therefore not be used in the analysis. This statistical error occurred due the BBB's Iteration 8 Phase 15 anomaly, where the phase duration was exceptionally high.

With the significance probability below five percent for Phases 1 to 14, the null hypothesis can be rejected for these phases. It can therefore be stated that the MPDBs have a significantly longer phase duration than the PC throughout all phases. In other words, the MPDBs require a greater amount of time, when compared to the PC, to perform the same task.

Table 6-13: ANOVA for Phases 1-20 using the duration metric

Phase	Group	Sum of Squares	df	Mean Square	F	Sig. (p)
1	Between Groups	79.393	3	26.464	258.733	0.000
	Within Groups	11.865	116	0.102		
	Total	91.258	119			
2	Between Groups	1353.270	3	451.090	333.266	0.000
	Within Groups	157.011	116	1.354		
	Total	1510.281	119			
3	Between Groups	1884.510	3	628.170	307.319	0.000
	Within Groups	237.108	116	2.044		
	Total	2121.617	119			
4	Between Groups	4092.257	3	1364.086	277.419	0.000
	Within Groups	570.379	116	4.917		
	Total	4662.636	119			
5	Between Groups	188736.643	3	62912.214	3903.903	0.000
	Within Groups	1869.364	116	16.115		

Phase	Group	Sum of Squares	df	Mean Square	F	Sig. (p)
	Total	190606.008	119			
6	Between Groups	1337.403	3	445.801	165.894	0.000
	Within Groups	311.723	116	2.687		
	Total	1649.126	119			
7	Between Groups	1066.604	3	355.535	74.950	0.000
	Within Groups	550.261	116	4.744		
	Total	1616.865	119			
8	Between Groups	3016.454	3	1005.485	41.764	0.000
	Within Groups	2792.733	116	24.075		
	Total	5809.187	119			
9	Between Groups	10790.431	3	3596.810	100.389	0.000
	Within Groups	4156.126	116	35.829		
	Total	14946.557	119			
10	Between Groups	195789.283	3	65263.094	1420.481	0.000
	Within Groups	5329.545	116	45.944		
	Total	201118.828	119			
11	Between Groups	481457.112	3	160485.704	9839.593	0.000
	Within Groups	1891.983	116	16.310		
	Total	483349.095	119			
12	Between Groups	652245.114	3	217415.038	1245.578	0.000
	Within Groups	20247.742	116	174.550		
	Total	672492.857	119			
13	Between Groups	397272.884	3	132424.295	893.025	0.000
	Within Groups	17201.339	116	148.287		
	Total	414474.223	119			
14	Between Groups	1283911.583	3	427970.528	942.545	0.000
	Within Groups	52670.793	116	454.059		
	Total	1336582.376	119			
15	Between Groups	25399702.928	3	8466567.643	1.235	0.300
	Within Groups	795134591.299	116	6854608.546		
	Total	820534294.226	119			
16	Between Groups	1018033.636	2	509016.818	16175.877	0.000
	Within Groups	2737.685	87	31.468		
	Total	1020771.322	89			
17	Between Groups	1253112.868	2	626556.434	3735.076	0.000
	Within Groups	14594.191	87	167.749		
	Total	1267707.059	89			
18	Between Groups	1099113.062	2	549556.531	3912.153	0.000
	Within Groups	12221.254	87	140.474		
	Total	1111334.316	89			
19	Between Groups	905871.743	2	452935.872	2151.741	0.000
	Within Groups	18313.275	87	210.497		
	Total	924185.018	89			
20	Between Groups	2518893.916	2	1259446.958	23981.578	0.000
	Within Groups	4569.002	87	52.517		
	Total	2523462.919	89			

Because of the Type I statistical error in Phase 15 and the fact that the BBB was not capable of performing the final five phases, the one-way ANOVA using the duration metric is repeated for Phases 15 to 20 without the BBB in Table 6-14. With the significance probability below five percent for all phases in this table ($p < 0.05$), the null hypothesis can be rejected. It can therefore

be stated that the RPi3 and IEA require a significantly longer duration than the PC to complete Phase 15 to 20.

Table 6-14: ANOVA for Phases 15-20 without the BBB using the duration metric

Phase	Group	Sum of Squares	df	Mean Square	F	Sig. (p)
15	Between Groups	564104.102	2	282052.051	6313.437	0.000
	Within Groups	3886.715	87	44.675		
	Total	567990.817	89			
16	Between Groups	1018033.636	2	509016.818	16175.877	0.000
	Within Groups	2737.685	87	31.468		
	Total	1020771.322	89			
17	Between Groups	1253112.868	2	626556.434	3735.076	0.000
	Within Groups	14594.191	87	167.749		
	Total	1267707.059	89			
18	Between Groups	1099113.062	2	549556.531	3912.153	0.000
	Within Groups	12221.254	87	140.474		
	Total	1111334.316	89			
19	Between Groups	905871.743	2	452935.872	2151.741	0.000
	Within Groups	18313.275	87	210.497		
	Total	924185.018	89			
20	Between Groups	2518893.916	2	1259446.958	23981.578	0.000
	Within Groups	4569.002	87	52.517		
	Total	2523462.919	89			

Table 6-15 provides duration results for each device compared directly to other devices in each phase where the significance probability is greater than five percent. Phase 15 contains a Type I statistical error (false positive) and the result of the phase is therefore ignored in this table. When considering the significance probability, it can be seen that mean duration of the RPi3 and PC are similar in Phases 6 and 13 while the BBB and IEA show similar mean durations in Phase 8. In addition, from Table 6-12, it is evident that the RPi3 not only completed Phases 6 and 13 in a shorter mean duration than the PC, but also Phases 1, 2 and 4.

Table 6-15: ANOVA post hoc multiple-device comparisons using the duration metric

Dependent Variable (Phase)	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
6	BBB	IEA	2.132	0.423	0.000	1.029	3.235
		PC	7.333	0.423	0.000	6.229	8.436
		RPi3	7.792	0.423	0.000	6.689	8.895
	IEA	BBB	-2.132	0.423	0.000	-3.235	-1.029
		PC	5.201	0.423	0.000	4.098	6.304
		RPi3	5.660	0.423	0.000	4.557	6.763
	PC	BBB	-7.333	0.423	0.000	-8.436	-6.229
		IEA	-5.201	0.423	0.000	-6.304	-4.098
		RPi3	**0.459**	**0.423**	**0.699**	**-0.644**	**1.563**
	RPi3	BBB	-7.792	0.423	0.000	-8.895	-6.689
		IEA	-5.660	0.423	0.000	-6.763	-4.557
		PC	**-0.459**	**0.423**	**0.699**	**-1.563**	**0.644**
8	**BBB**	**IEA**	**2.604**	**1.267**	**0.174**	**-0.698**	**5.907**
		PC	12.967	1.267	0.000	9.665	16.269
		RPi3	8.122	1.267	0.000	4.819	11.424
	IEA	**BBB**	**-2.604**	**1.267**	**0.174**	**-5.907**	**0.698**
		PC	10.363	1.267	0.000	7.060	13.665
		RPi3	5.517	1.267	0.000	2.215	8.820
	PC	BBB	-12.967	1.267	0.000	-16.269	-9.665
		IEA	-10.363	1.267	0.000	-13.665	-7.060
		RPi3	-4.845	1.267	0.001	-8.148	-1.543
	RPi3	BBB	-8.122	1.267	0.000	-11.424	-4.819
		IEA	-5.517	1.267	0.000	-8.820	-2.215
		PC	4.845	1.267	0.001	1.543	8.148
13	BBB	IEA	35.217	3.144	0.000	27.021	43.413
		PC	129.946	3.144	0.000	121.750	138.142
		RPi3	129.969	3.144	0.000	121.773	138.165
	IEA	BBB	-35.217	3.144	0.000	-43.413	-27.021
		PC	94.729	3.144	0.000	86.534	102.925
		RPi3	94.752	3.144	0.000	86.556	102.948
	PC	BBB	-129.946	3.144	0.000	-138.142	-121.750
		IEA	-94.729	3.144	0.000	-102.925	-86.534
		RPi3	**0.023**	**3.144**	**1.000**	**-8.173**	**8.218**
	RPi3	BBB	-129.969	3.144	0.000	-138.165	-121.773
		IEA	-94.752	3.144	0.000	-102.948	-86.556
		PC	**-0.023**	**3.144**	**1.000**	**-8.218**	**8.173**
15	BBB	IEA	990.757	675.998	0.462	-771.345	2752.858
		PC	1162.492	675.998	0.318	-599.610	2924.593
		RPi3	998.615	675.998	0.455	-763.487	2760.716
	IEA	BBB	-990.757	675.998	0.462	-2752.858	771.345
		PC	171.735	675.998	0.994	-1590.366	1933.837
		RPi3	7.858	675.998	1.000	-1754.243	1769.960
	PC	BBB	-1162.492	675.998	0.318	-2924.593	599.610
		IEA	-171.735	675.998	0.994	-1933.837	1590.366
		RPi3	-163.877	675.998	0.995	-1925.979	1598.224
	RPi3	BBB	-998.615	675.998	0.455	-2760.716	763.487
		IEA	-7.858	675.998	1.000	-1769.960	1754.243
		PC	163.877	675.998	0.995	-1598.224	1925.979

The complete table relating to Table 6-15 for Phases 1 to 15 can be seen in Appendix C (§C.2.1) and the table containing Phases 15 to 20 without the BBB in Appendix C (§C.2.2). The statistical analysis of the duration metrics is hereby concluded. The ANOVA results from the load and duration metrics are used and summarised in the following hypothesis testing section.

6.5 Hypothesis testing

The ANOVA results revealed that in most phases the mean load and duration show a statistically significant difference between devices. In other words, phase differences that are statistically significant indicate that the null hypothesis is rejected for the specific phases. In this section, the results from Section 6.4 are used to test the hypothesis of the study.

The primary objective of the study is to evaluate the effectiveness of MPDBs compared to commodity PCs for hosting small-scale DBMS implementations. This objective is supported by a hypothesis test and the determination of divergence points; the point of divergence is analysed in Section 6.6.

The hypothesis formulation for this study is shown in Table 6-16.

Table 6-16: Hypothesis for the study

Formulation	Description
$\mu 1$	Microprocessor development board
$\mu 2$	Commodity personal computer
H_o: $\mu 1 = \mu 2$	Microprocessor development boards support the data processing of small-scale database management system solutions similar to commodity personal computers
H_a: $\mu 1 \neq \mu 2$	Microprocessor development boards do not support the data processing of small-scale database management system solutions similar to commodity personal computers

The hypothesis is tested for each phase on each device in terms of duration and load. Results for the hypothesis were obtained from the significance probability values in the ANOVA post hoc multiple-device comparisons, detailed in Table 6-10 and Table 6-15. The hypothesis testing results are shown for Phases 1 to 15 in Table 6-17 and Phases 15 to 20 in Table 6-18. It can be seen from this table that the RPi3 is the only device that failed to reject the null hypothesis in some phases; with Phases 6 and 13 only partially on duration and not load.

Table 6-17: Hypothesis testing for Phases 1-20 in terms of the load and duration metrics

Device	Phase	Null Hypothesis		Exceptions
		Load Reject?	Duration Reject?	
BBB, IEA, PC, RPi3	1	Yes	Yes	RPi3 vs. PC load metric: fail to reject the null hypothesis with p (0.155) > 0.05. Therefore, in Phase 1 the RPi3's load is significantly similar to a PC. RPi3 vs. PC duration metric: fail to reject the null hypothesis with descriptive statistics. The mean duration of the RPi3 in Phase 1 is 0.84s, which is lower than that of the PC (1.23s).
	2	Yes	Yes	RPi3 vs. PC load metric: fail to reject the null hypothesis with p (0.411) > 0.05. Therefore, in Phase 2 the RPi3's load is significantly similar to a PC. RPi3 vs. PC duration metric: fail to reject the null hypothesis with descriptive statistics. The mean duration of the RPi3 in Phase 2 is 3.33s, which is lower than that of the PC (4.15s).
	3	Yes	Yes	None
	4	Yes	Yes	RPi3 vs. PC load metric: fail to reject the null hypothesis with p (0.076) > 0.05. Therefore, in Phase 4 the RPi3's load is significantly similar to a PC. RPi3 vs. PC duration metric: fail to reject the null hypothesis with descriptive statistics. The mean duration of the RPi3 in Phase 4 is 7.14s, which is lower than that of the PC (8.89s).
	5	Yes	Yes	None
	6	Yes	Yes	RPi3 vs. PC duration metric: fail to reject the null hypothesis with p (0.699) > 0.05. Therefore, in Phase 6 the RPi3's duration is significantly similar to a PC.
	7	Yes	Yes	None
	8	Yes	Yes	BBB vs. IEA duration metric: significantly similar with p (0.174) > 0.05. Therefore, in Phase 8 the BBB's duration is significantly similar to the IEA, but no significance is shown when compared to a PC.
	9	Yes	Yes	None
	10	Yes	Yes	None
	11	Yes	Yes	None
	12	Yes	Yes	None
	13	Yes	Yes	RPi3 vs. PC duration metric: fail to reject the null hypothesis with p (1.00) > 0.05. Therefore, in Phase 13 the RPi3's duration is significantly similar to a PC.

Device	Phase	Null Hypothesis		Exceptions
		Load Reject?	Duration Reject?	
	14	Yes	Yes	None
	15	N/A	N/A	Type I statistical error
BBB, IEA, PC, RPi3	16	N/A	N/A	Refer to Table 6-18 below
	17	N/A	N/A	
	18	N/A	N/A	
	19	N/A	N/A	
	20	N/A	N/A	

When considering the hypothesis test relating to the load metric in Table 6-17, it can be seen that only the RPi3 had significantly similar load compared to that of a PC in Phases 1, 2 and 4. The BBB and IEA did not show a mean load in any phase that is significantly similar to the mean load of a PC.

In terms of the duration metric, Phases 6 and 13 show significantly similar duration between the RPi3 and PC. By analysing the duration descriptive statistics in Table 6-12, it can be seen that mean duration of the RPi3 is shorter than that of a PC in not only Phases 6 and 13, but also Phases 1, 2, and 4. The BBB and IEA did not show a mean duration that is significantly similar to the mean duration of a PC, but did however illustrate a significantly similar duration in Phase 8.

The Type I statistical error in Phase 15 is caused by the high standard deviation as explained in Sections 6.4.1 and 6.4.2. The BBB was not able to perform Phases 16 through 20 due to it becoming unresponsive and thereby unreachable through the network. The hypothesis tests for Phases 15 to 20 is therefore performed separately without the BBB detailed in Table 6-18.

Table 6-18 – Hypothesis testing: Phases 15-20

Device	Phase	Null Hypothesis		Exceptions
		Load Reject?	Duration Reject?	
IEA, PC, RPi3	15	Yes	Yes	None
	16	Yes	Yes	
	17	Yes	Yes	
	18	Yes	Yes	
	19	Yes	Yes	
	20	Yes	Yes	

After analysing Table 6-17 and Table 6-18, it can be seen that the null hypothesis is rejected in all phases, indicating that MPDBs do not support the data processing of small-scale DBMS solutions similar to commodity PCs. However, the RPi3 was the only MPDB that did not reject the null hypothesis in certain phases in terms of the load and duration metric. This is evident in Table 6-17 with Phases 1, 2 and 4 where the RPi3's load and duration was not rejected in terms of the

null hypothesis. Also in Phases 6 and 13, the null hypothesis was partially rejected, only in terms of the RPi3 and the duration metric.

The testing data used for the experiment was exaggerated when compared to a small-scale database to allow the researcher to determine the points of failure in the MPDBs. The testing data consisted of five million orders spread over ten years or 1 984 orders per work day for ten years. A small to medium-sized business in a real world situation would usually archive older data regularly to reduce the number of records stored in tables that are queried on a daily basis. By testing the hypothesis, it is proved that MPDBs do not support the data processing capability similar to commodity PCs, but MPDBs are able to host a DBMS capable of data processing up to a certain extent. This extent, or otherwise referred to as the divergence point is determined in the next section to show the degree to which the MPDBs tested in the experiment are able to process data.

6.6 Point of divergence

The point of divergence relates to the point where the MPDB starts to move away from the constant of stability and enables the researcher to accurately rank devices in terms of performance. The determination of the point of divergence relates to the second supporting objective of this study's primary objective.

To identify this point where MPDBs start to deviate from stability, thread metrics are analysed and from this analysis, the divergence point for each MPDB is plotted (§6.6.1). In addition, a follow-up experiment is performed to provide more insight into each MPDB's performance relative to the other MPDBs in this study (§6.6.2).

6.6.1 Determining the point of divergence through thread metrics

The average number of failed threads per phase for each device across all iterations, is shown in Table 6-19. Note that because the BBB was incapable of performing Phases 16 through 20, it is assumed that these phases have a 100 percent failure rate and are therefore shown in accordance with this assumption. The "Total" calculates the sum (not the average) of the rows in the table, in which it can be seen that the BBB had the most thread failures on average, the IEA second most, and the RPi3 fared the best across the MPDBs; the commodity PC had, as expected, the least failures. This result is indicative of each device's performance, with the commodity PC performing the best while the BBB was the worst performer.

Table 6-19: Average thread failures per phase for each device

Phase	Total Threads Executed	Average per Device			
		BBB	IEA	RPi3	PC
1	5	0.00	0.00	0.00	0.00
2	25	0.00	0.00	0.00	0.00
3	30	0.00	0.00	0.00	0.00
4	25	0.00	0.00	0.00	0.00
5	40	39.83	2.01	0.00	0.00
6	10	0.00	0.00	0.00	0.00
7	7	0.07	0.00	0.00	0.00
8	12	0.23	0.00	0.00	0.00
9	30	1.22	0.00	0.00	0.00
10	10	3.03	0.00	0.00	0.00
11	60	60.00	60.00	56.85	0.00
12	100	92.27	99.42	83.07	0.00
13	150	0.00	0.00	0.00	0.00
14	200	0.49	0.00	0.00	0.00
15	150	149.98	150.00	150.00	0.00
Total	**854**	**347.12**	**311.43**	**289.92**	**0.00**
16	45	45.00	45.00	31.69	0.00
17	150	150.00	149.36	43.49	0.00
18	60	60.00	60.00	60.00	0.00
19	50	50.00	49.83	50.00	0.00
20	100	100.00	100.00	99.53	0.00
Total	**1259**	**752.12**	**715.62**	**574.63**	**0.00**

The thread failure analysis is continued in Figure 6-6 where the average thread failures per phase for each device is calculated as a percentage of the total threads executed. From this figure, it is evident that the BBB failed nearly all threads (99.58 percent) in Phase 5, while all other devices failed less than ten percent. Throughout all phases, the RPi3 had the least failures amongst the MPDBs; this is the most noticeable in Phases 12, 16 and 17. The results illustrated on the graph support the results and conclusion drawn from Table 6-19.

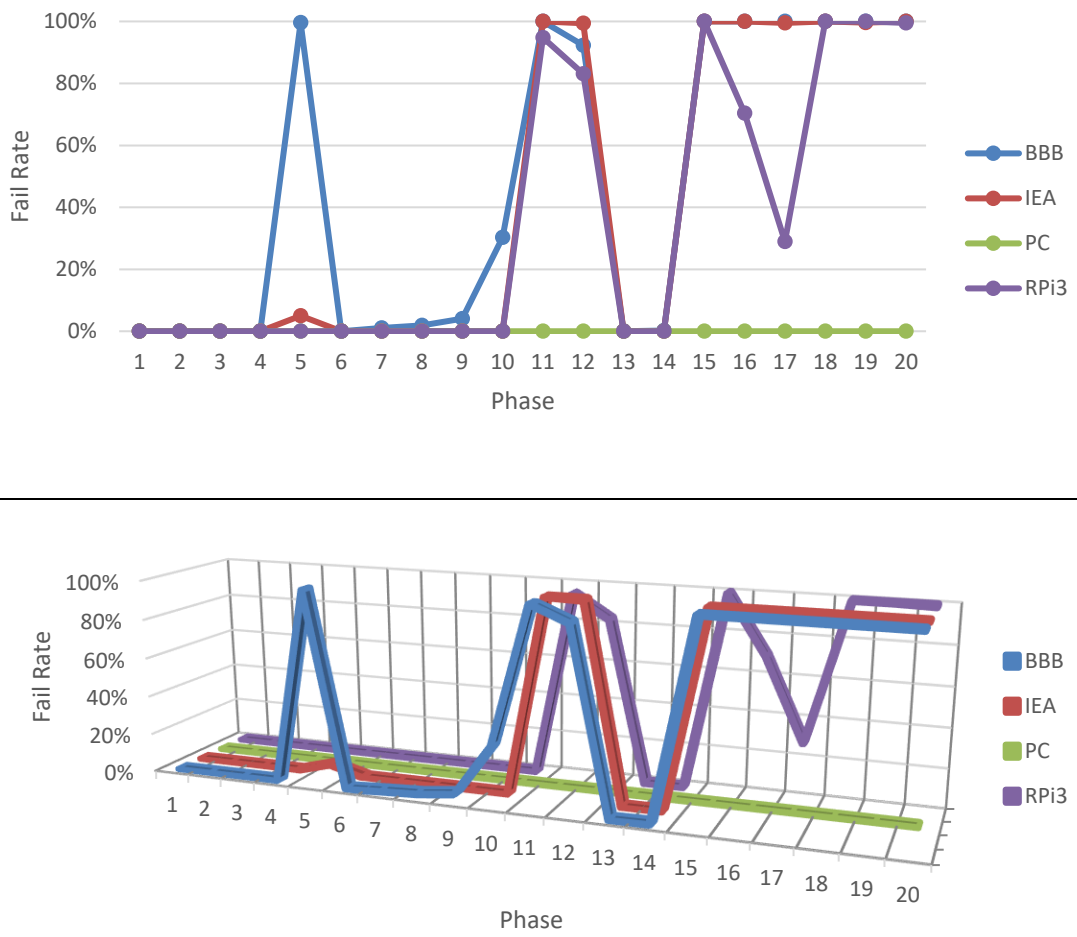


Figure 6-6: Average fail rate per phase for each device

The results from Table 6-19 and Figure 6-6 are based on the average device performance across the 30 iterations. The analysis is continued in Table 6-20 by detailing the minimum and maximum thread failures per phase for each device. The results from this table again show that the PC was the strongest performer, followed by the RPi3, the IEA, and the BBB. It is worthwhile to note that the RPi3 showed a high spread of minimum to maximum thread failures in Phases 11, 12, 16 and 17. In these phases, the BBB and IEA had high levels of failure.

Table 6-20: Minimum and maximum thread failures per phase

Phase	Total Threads Executed	Device							
		BBB		IEA		RPi3		PC	
		Min Failed	Max Failed	Min Failed	Max Failed	Min Failed	Max Failed	Min Failed	Max Failed
1	5	0	0	0	0	0	0	0	0
2	25	0	0	0	0	0	0	0	0
3	30	0	0	0	0	0	0	0	0
4	25	0	0	0	0	0	0	0	0
5	40	39	40	0	6	0	0	0	0
6	10	0	0	0	0	0	0	0	0
7	7	0	1	0	0	0	0	0	0
8	12	0	3	0	0	0	0	0	0
9	30	0	3	0	0	0	0	0	0
10	10	0	6	0	0	0	0	0	0
11	60	60	60	60	60	37	60	0	0
12	100	84	97	96	100	66	100	0	0
13	150	0	0	0	0	0	0	0	0
14	200	0	3	0	0	0	0	0	0
15	150	148	150	150	150	150	150	0	0
16	45	45	45	45	45	14	45	0	0
17	150	150	150	147	150	19	90	0	0
18	60	60	60	60	60	60	60	0	0
19	50	50	50	48	50	50	50	0	0
20	100	100	100	100	100	98	100	0	0
Total	**1259**	**736**	**768**	**706**	**721**	**494**	**655**	**0**	**0**

In order to examine the reason for the threads that failed, the SPs that each thread executed in different phases are summarised in Table 6-21. This table is composed of information from the load test design and different tables in Chapter 5: Table 5-1, Table 5-2 and Table 5-3. This table also shows the number of simultaneous threads in the phases, each executing all SPs relating to the specific phase.

Table 6-21: SP description for each phase

Phase	No. of Threads	SPs Executed per Thread	SPs Description
1	5	0_R_ClientDetails	One extremely low demand read
2	25	0_R_OrderDetails	One extremely low demand read
3	30	1_R_ClientOrders	One low demand read
4	25	0_R_ClientDetails, 0_R_OrderDetails	Two extremely low demand reads
5	40	1_R_ClientOrdersCancelled	Low demand read
6	10	0_R_ClientDetails, 0_R_OrderDetails, 0_R_ClientOccupation	Three extremely low demand reads
7	7	0_C_Product, 0_C_Client, 0_C_Order, 0_D_Client	Three extremely low demand updates and one extremely low demand delete
8	12	0_U_Product, 0_U_Order, 0_U_Client, 0_U_OrderLine	Four extremely low demand updates
9	30	0_C_Client, 0_U_Client, 0_D_Client	One extremely low demand create, one extremely low demand update and one extremely low demand delete

Phase	No. of Threads	SPs Executed per Thread	SPs Description
10	10	1_R_ClientOrdersPaid, 1_R_ClientOrdersCancelled	Two low demand reads
11	60	3_R_CreditCheck	One high demand read
12	100	3_R_ClientNotesOrders	One high demand read
13	150	0_R_ClientDetails, 0_R_OrderDetails, 0_R_ClientOccupation	Three extremely low demand reads
14	200	0_C_Client, 0_R_ClientDetails, 0_U_Client, 0_D_Client	One extremely low demand create, one extremely low demand read, one extremely low demand update and one extremely low demand delete
15	150	4_R_SoldPerProduct	One extremely high demand read
16	45	3_R_CreditCheck, 3_R_ClientNotesOrders	Two high demand reads
17	150	1_R_ClientOrders, 1_R_ClientOrdersPaid, 1_R_ClientOrdersCancelled	Three low demand reads
18	60	3_R_ClientNotesOrders, 3_R_AmountOfOrdersDate	Two high demand reads
19	50	2_R_ClientNotes, 4_R_SoldPerProduct	One medium demand read and one extremely high demand read
20	100	3_R_CreditCheck, 3_R_ClientNotesOrders, 3_R_AmountOfOrdersDate	Three high demand reads

The BBB showed a high rate of failure for Phase 5, the opposite when compared to the rate of failure for other devices. From Table 6-21 it can be seen that Phase 5 entails 40 simultaneous threads executing a low demand read on the DBMS. It is assumed that the load applied through this phase is higher than what the BBB is able to manage.

In Phases 16 and 17, the RPi3 performed significantly better than the BBB and IEA in terms of the thread metric. Phase 16 contained two high demand reads repeated by 45 simultaneous threads, while Phase 17 is run by 150 simultaneous clients, each executing three low demand reads. By considering the difference in the rate of failure, this is a clear indication that the RPi3's performance was better than that of the BBB and IEA.

Phase 15 was the first phase where all MPDBs failed nearly 100 percent of all SPs executed. This phase contains an extremely high demand read executed by 150 simultaneous threads. It is the researcher's opinion that the combination of the load that built up from this phase and Phase 16 caused the unresponsive state of the BBB and therefore it was not able to perform the final five phases. From the detail in Table 6-21 and this analysis, it is clear that the fail rate of each device is dependent on its performance and the amount of load applied on it, which is determined by the load test design.

In order to obtain the divergence points, the thread pass rate per phase for each device is ordered from high to low in Table 6-22. It is important to note that the number of phases on the graph in

Figure 6-7 is an indicator of quantity and not the actual phase numbers. The values shown in Table 6-22 are therefore the *pass rate* of threads in each phase and not the *fail rate* that were depicted in Figure 6-6 from Table 6-19.

Table 6-22: Thread pass rate in descending order for each device

Number of Phases	Device			
	BBB (%)	IEA (%)	RPi3 (%)	PC (%)
1	100.00	100.00	100.00	100.00
2	100.00	100.00	100.00	100.00
3	100.00	100.00	100.00	100.00
4	100.00	100.00	100.00	100.00
5	100.00	100.00	100.00	100.00
6	100.00	100.00	100.00	100.00
7	99.75	100.00	100.00	100.00
8	98.95	100.00	100.00	100.00
9	98.11	100.00	100.00	100.00
10	95.94	100.00	100.00	100.00
11	69.67	100.00	100.00	100.00
12	7.73	94.97	100.00	100.00
13	0.42	0.58	71.01	100.00
14	0.01	0.43	29.57	100.00
15	0.00	0.33	16.93	100.00
16	0.00	0.00	5.25	100.00
17	0.00	0.00	0.47	100.00
18	0.00	0.00	0.00	100.00
19	0.00	0.00	0.00	100.00
20	0.00	0.00	0.00	100.00

From Table 6-22, it can be seen that the BBB completed six of the 20 phases successfully and is regarded as unstable for 14 of the 20 phases. The BBB's divergence point is therefore at a quantity of six phases in this study. Figure 6-7 illustrates the results from Table 6-22 graphically, where it is depicted that the commodity PC passed all phases without any failures, it therefore also represents the stability constant on the graph. Any device that deviates from the 100 percent pass rate is regarded as unstable at that particular point.

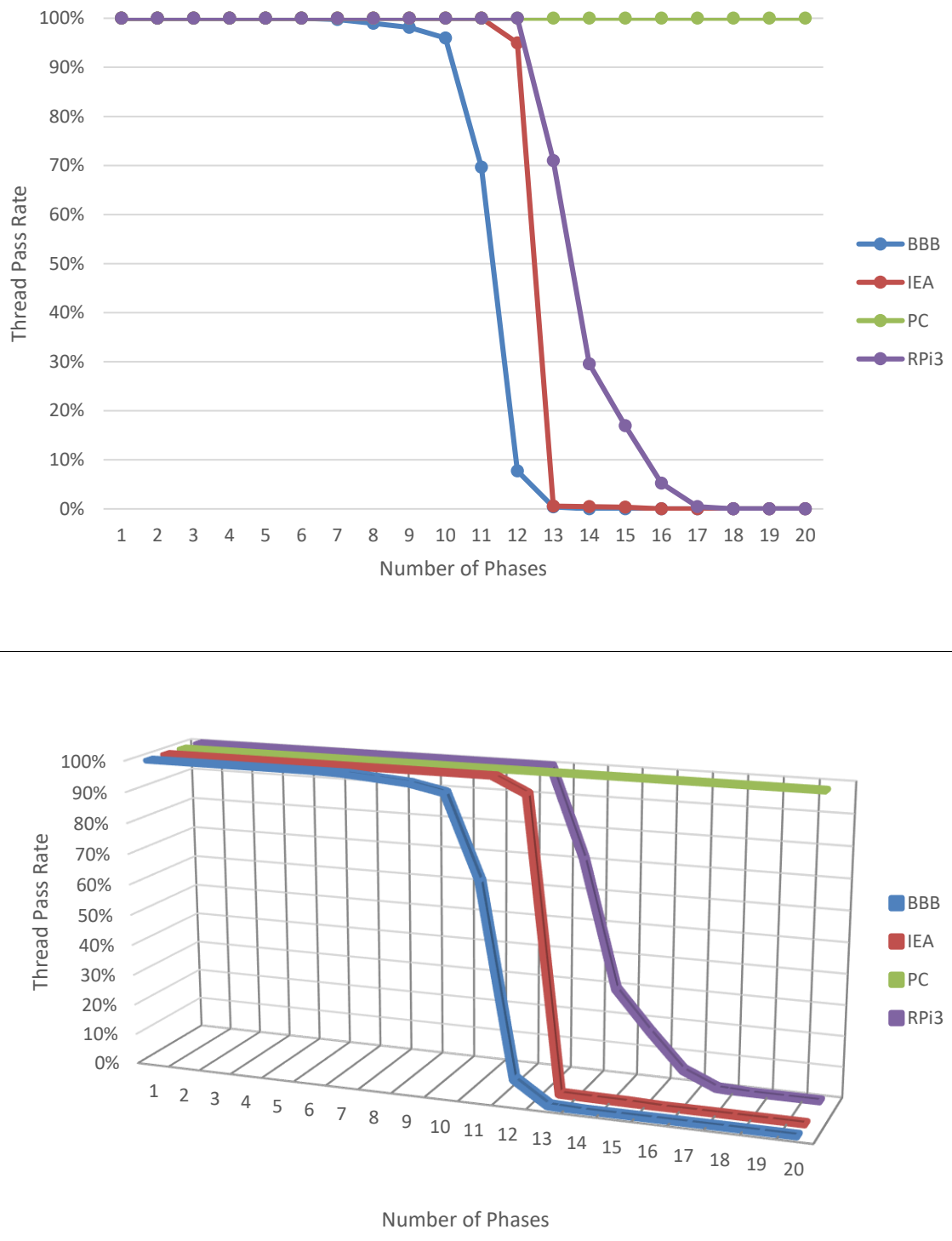


Figure 6-7: Divergence points

The divergence points reveal that the BBB is the lowest performing device by showing stability in six of the 20 phases. The IEA is ranked third with stability in 11 phases. The RPi3 is ranked second and as the best performing MPDB by showing stability through 12 of the 20 phases. The

PC was stable throughout the entirety of the experiment; it passed all phases with not a single thread failure in any of the phases.

The objective to determine the divergence points is hereby achieved and the subsequent section describes a follow-up experiment conducted to provide additional insight into the performance of the MPDBs.

6.6.2 Follow-up experiment

A follow-up experiment is performed to further investigate the performance of each MPDB relative to one another. This follow-up attempts to identify the highest possible number of data each MPDB can host without a single thread failure, in other words remaining in a stable state, while applying load in accordance with the load test design. This is termed the breakpoint of each MPDB for the purpose of this study. The database design including the SPs are identical to that of the experiment; the only difference in the follow-up is that the tables contain less data.

In technical terms, the purpose of this follow-up experiment is to determine the point where the MPDBs can successfully complete all phases with the maximum number of CLIENT and ORDER table records. The data is, like the experiment, generated by the data generator script described in Chapter 5 Section 5.3. The number of ORDERLINE table records is dependent on the number of records that are generated for the ORDER table; the number of records is therefore programmatically derived and not specified by the researcher. The number of records in the OCCUPATION and PRODUCT tables is the same as that of the experiment. In the follow-up, the number of records are treated as variables and are defined in terms of each table:

- CLIENT: primary variable;
- ORDER: primary variable;
- ORDERLINE: derived variable;
- OCCUPATION: constant variable; and
- PRODUCT: constant variable.

To determine the breakpoint, the researcher conducted numerous iterations according to the load test design with different combinations of number of CLIENT and ORDER table records. The researcher decreased the number of data records when any failures were encountered and increased the number of data records when there were no failures. This process was repeated

numerous times for each device, moving closer to the device's breakpoint with each iteration, until the maximum number of database data records for each device were identified.

The results from the follow-up are detailed in Table 6-23, showing the original number of records from the experiment as well as the number of records at the breakpoint of each MPDB in this study. By considering the CLIENT, ORDER and ORDERLINE table records in Table 6-23, it can be seen that the IEA performed better than the BBB while the RPi3 far outperformed the IEA and BBB. The results of the follow-up experiment rank the devices in the same order as with the experiment.

Table 6-23: Follow-up results

Table	Experiment Number of Records	Device		
		BBB	IEA	RPi3
CLIENT	20000	2000	4000	8000
ORDER	5000000	55125	126000	724500
ORDERLINE	32611992	357784	820585	4720365
OCCUPATION	330	330	330	330
PRODUCT	1438	1438	1438	1438

By examining Table 6-23, a breakpoint ratio is calculated for each device relative to the highest performing MPDB, the RPi3. The breakpoint ratio is calculated from the number of records in the ORDERLINE table, since it contains the most records amongst the tables; the breakpoint ratio is calculated and rounded to three decimals for the IEA and BBB:

- *RPi3* – 1.0;
- *IEA* – 0.174; and
- *BBB* – 0.076.

When considering the breakpoint ratio, the breakpoint of the RPi3 is 5.75 times higher than that of the IEA and 13.19 times higher than that of the BBB. The IEA's breakpoint is 2.29 times higher than the breakpoint of the BBB. This is illustrated in Figure 6-8.

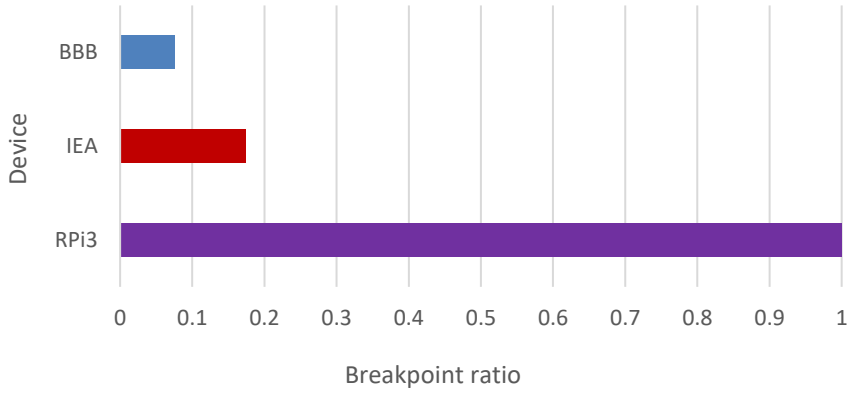


Figure 6-8: Breakpoint ratio

The load test design as discussed in Chapter 5 (Table 5-3) was adhered to in the follow-up and not altered in any way. By performing the follow-up, further insight was gained into the relative performance between the MPDBs in this study. In addition, the follow-up showed that MPDBs are able to remain in a stable state while under load, like the commodity PC in the experiment, up to a certain amount of database data. Follow-up log files can be viewed in Appendix G.

6.7 Experiment analysis result summary

The experiment results, analysed in terms of the load and duration metrics, highlighted the insufficient capacity of the BBB to perform Phases 16 through 20 as well as an unknown error internal to the device in Iteration 8 Phase 15. Each of the metrics analysed provided valuable information relating to each device.

Analysis of the load metric illustrated that, in general, the BBB had the highest load, the IEA second highest, the RPi3 third highest, while the PC had the lowest load. The ANOVA revealed that MPDBs require a larger portion of total available resources, when compared a PC, to perform the same task. For example, Phase 5 in Table 6-6 showed an average load for the PC at seven percent, while the RPi3 showed 85 percent load, the IEA 298 percent or 198 percent overloaded, and the BBB 1073 percent load or 973 percent overloaded. All devices in this example display different load levels for an identical task performed by each device. The load metric is therefore a valuable indicator of performance.

The duration metric analysis revealed the same overall device rank as from the load metric described above. From Figure 6-5, it can be seen that the mean duration of the MPDBs started

to deviate significantly from the shorter durations of the PC from Phase 10 onwards as more load is applied to the devices. This indicates that MPDBs have poorer performance than a PC in that they require longer amounts of time to complete the same tasks; this statement is also confirmed through the ANOVA results relating to the duration metrics. An unexpected result was seen between the mean durations of the PC compared to the RPi3, showing that for Phases 1, 2, 4, 6 and 13 the RPi3 had a shorter phase duration than the PC; in Phases 6 and 13 the RPi3's duration is significantly similar to that of the PC. However, the average phase duration of the RPi3 was significantly slower than that of the PC in the other phases.

The hypothesis test was performed for every phase in Section 6.5 in terms of the device load metrics as well as the duration metrics. This study rejects the null hypothesis and it is therefore stated that MPDBs do not support the data processing of small-scale DBMS solutions similar to commodity PCs. However, the following exceptions exist in terms of the hypothesis outcome stemming from the performance of the RPi3:

- *Phase 1* – fails to reject null hypothesis due to RPi3 load being significantly similar to a PC and duration better than a PC.
- *Phase 2* – fails to reject null hypothesis due to RPi3 load being significantly similar to a PC and duration better than a PC.
- *Phase 4* – fails to reject null hypothesis due to RPi3 load being significantly similar to a PC and duration better than a PC.
- *Phase 6* – partially fails to reject null hypothesis due to RPi3 duration being significantly similar to a PC.
- *Phase 13* – partially fails to reject null hypothesis due to RPi3 duration being significantly similar to a PC.

The study's null hypothesis is rejected regardless of these exceptions, because MPDBs as a population do not support the data processing of small-scale DBMS solutions similar to commodity PCs.

Each MPDB's point of divergence from stability was determined in Section 6.6.1 through the analysis of the thread metrics. The thread metrics analysis showed that the phase failure rate of each MPDB is dependent on the device's performance and the amount of load applied on the MPDB; the load applied is in accordance with the load test design, which can be viewed in Chapter 5 in Table 5-3. Each MPDB's divergence point shows the extent of its data processing capability compared to other devices in this study and is summarised:

1. The PC is the highest performing device and did not deviate from stability in any of the 20 phases.
2. The RPi3 is the MPDB with the best performance from the MPDBs in this study, showing stability in 12 of the 20 phases.
3. The IEA is ranked second in terms of the MPDBs, showing stability in 11 of the 20 phases.
4. The BBB is the lowest performing device and shows stability in six of the 20 phases.

The follow-up was performed in Section 6.6.2 to further investigate the MPDBs' performance relative to one another. Results from the follow-up showed that the breakpoint of the RPi3 is 5.74 times higher than the IEA's breakpoint and 13.15 times higher than that of the BBB. The breakpoint of the IEA is 2.29 times higher than the breakpoint of the BBB.

From the analysis results in this chapter, it is clear that MPDBs are capable of hosting small-scale DBMSs with data processing capabilities up to a certain extent. This extent depends greatly on the individual performance of the MPDB.

6.8 Conclusion

This chapter explained how the data generated by the pilot and the experiment were analysed to draw meaningful conclusions about the computing devices as subjects of the study. The pilot data were analysed with the use of pivot tables and graphs. Subsequently, the experiment data were analysed with a combination of pivot tables, graphs and SPSS.

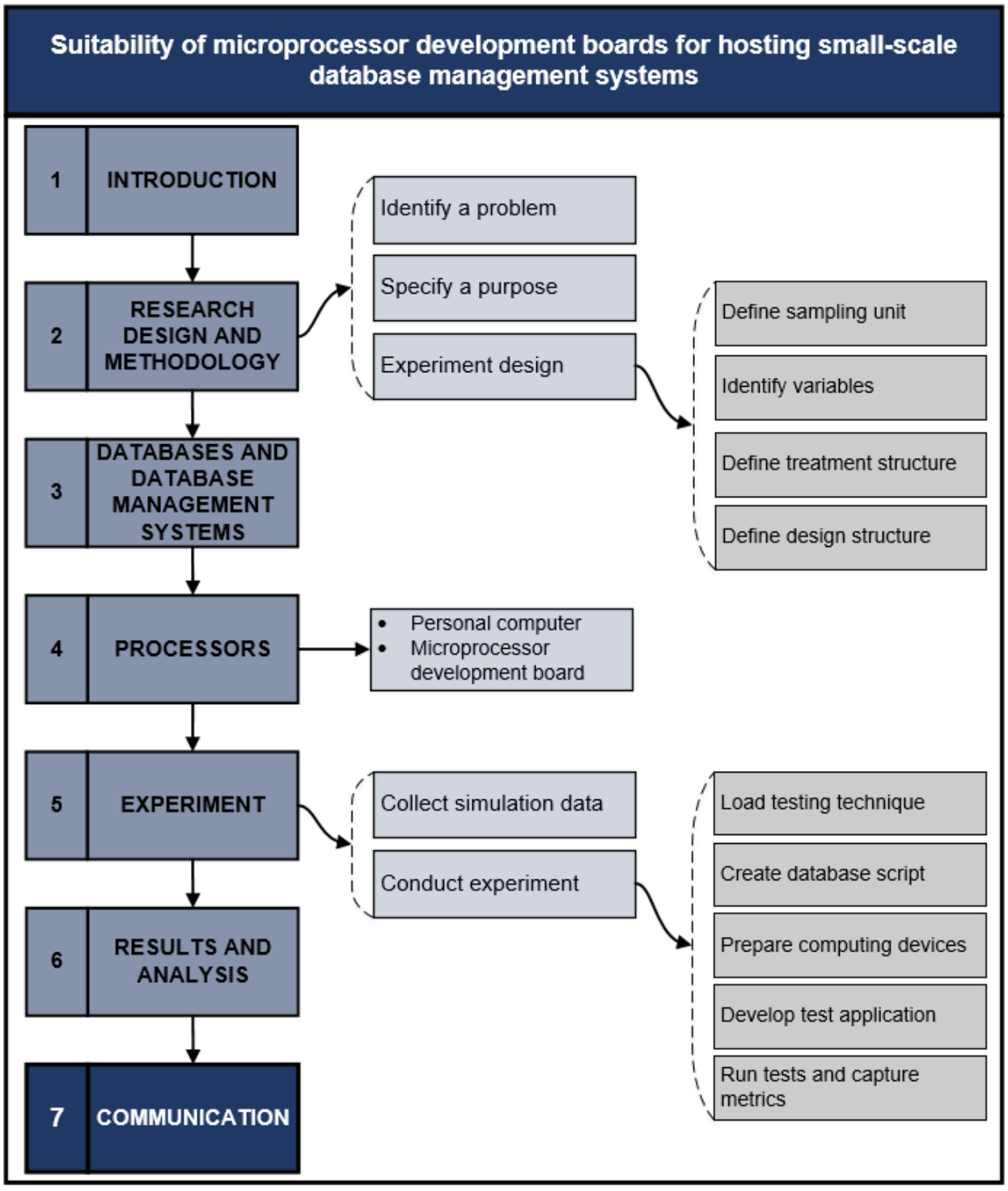
The data analysis of the experiment allowed for hypothesis testing which revealed that in most phases MPDBs could not match the performance of a commodity PC. The RPi3 is the only MPDB that is able to process data similar to the PC in some phases.

The researcher determined the point of divergence from the stability for each MPDB. The divergence point provided the researcher with insight into the proportional performance difference between each MPDB. This analysis revealed that the devices can be ranked in terms of performance in the following order:

1. The performance of the commodity PC is the highest (as expected with it being the benchmark of the study).
2. The RPi3 is the best performer amongst the MPDBs.
3. The IEA is situated between the RPi3 and BBB.
4. The BBB is the weakest performer.

The objective achieved through this chapter was to perform hypothesis testing and to determine the point of divergence from stability for the MPDBs. In addition to achieving the objective, a follow-up experiment was conducted, with the same load test design, but with a reduced number of data stored in the database. The follow-up was performed numerous times for each MPDB to identify the maximum number of data each MPDB can successfully host. This revealed that the RPi3's data processing capability is much higher than that of the IEA and BBB.

Chapter 6 is hereby concluded; the final chapter, Chapter 7, will provide conclusions and recommendations based on the knowledge gained throughout the study.


Suitability of microprocessor development boards for hosting small-scale database management systems
1
INTRODUCTION
2
RESEARCH DESIGN AND METHODOLOGY
Identify a problem
Specify a purpose
Experiment design
Define sampling unit
Identify variables
Define treatment structure
Define design structure
3
DATABASES AND DATABASE MANAGEMENT SYSTEMS
4
PROCESSORS
Personal computer
Microprocessor development board
5
EXPERIMENT
Collect simulation data
Conduct experiment
Load testing technique
Create database script
Prepare computing devices
Develop test application
Run tests and capture metrics
6
RESULTS AND ANALYSIS
7
COMMUNICATION

CHAPTER 7: COMMUNICATION

7.1 Introduction

The purpose of this study is to determine whether microprocessor development boards (MPDBs) can be used as an alternative to commodity personal computers (PCs) to host small-scale database management systems (DBMSs).

This chapter centres on the effective communication and reflection of the results found within this study. It focuses on drawing summaries from previous chapters, addressing the objectives, and answering the research question.

The purpose of this chapter is to conclude the study by communicating final feedback and recommendations for future research. This will be accomplished with the following sections: research findings of the study (§7.2), research findings of the experiment (§7.3), limitations relating to the study (§7.4), recommendations (§7.5), future research (§7.6), and finally, this study is closed (§7.7).

7.2 Research findings of the study

The research question was formulated for the study: to what extent is it possible to run a small-scale DBMS solution on MPDBs? The primary objective of this study is to evaluate the effectiveness of MPDBs compared to commodity PCs for hosting small-scale DBMS implementations; supported by hypothesis testing and the determination of the divergence point from stability.

Theoretical objectives for the study were achieved in the chapters listed; followed by a summary of each objective:

- *Chapter 2* – Gain an understanding of the quantitative research process.
- *Chapter 3* – Gain an understanding of database management systems (DBMSs).
- *Chapter 4* – Gain an understanding of commodity personal computers.
- *Chapter 4* – Gain an understanding of microprocessor development boards and different types thereof.
- *Chapter 4* – Gain an understanding of load testing.

The research and methodology chapter set the foundation for the study through analysis of research philosophies and paradigms in order to position the study in the positivist paradigm and defining the research design. The positivist paradigm encapsulates the scientific method with the quantitative research process adopted from Abraham S. Fischler School of Education (2012?) for this study.

The researcher was able to make an informed choice on the type of DBMS to be used in the experiment by investigating the literature relating to databases. In this chapter a brief history of databases was discussed as well as a performance evaluation between modern DBMSs. The DBMS chosen for the study was MySQL due to its compatibility across different types of operating systems and performance.

During the processors literature review, the commodity PC was selected as the benchmark device. Different types of MPDBs were researched and three MPDBs were identified for the experiment. The selected MPDBs were tested and they include: Raspberry Pi 3 (RPi3), Intel Edison Arduino (IEA), and BeagleBone Black (BBB). The hardware relating to each device was investigated, and the chapter included a comparison between the selected devices in terms of hardware specifications and prices. Finally, the load testing technique developed by Meier *et al.* (2007) was described.

7.3 Research findings of the experiment

The knowledge gained through researching and achieving the theoretical objectives enabled the researcher to address the empirical objectives. The empirical objectives that were formulated to achieve the primary objective of the study are listed together with the chapter in which it was addressed and followed by a summary of each:

- Chapter 2: Design an experiment.
- Chapter 5: Set up a testing environment by installing a DBMS on each MPDB to be tested.
- Chapter 5: Run simulations on each type of technology while recording performance metrics relating to the process.
- Chapter 6: Evaluate and compare the simulation results.
- Chapter 7: Propose recommendations based on the simulation results.

The experiment design was adapted from SAS (2005) in which sampling units, variables, treatment structure, including the load test design, and the design structure were defined. The purpose of this design was to serve as a detailed guide in order to perform a controlled, replicable experiment.

The primary objective of the study is supported by hypothesis testing and the determining of a point of divergence from stability. The hypothesis formulated for this study states that small-scale DBMS solutions can make use of MPDBs instead of the traditional commodity PC or server computers. Scientific formulation of the hypothesis, shown in Table 7-1, relates to determining whether MPDBs support the data processing of small-scale DBMS solutions similar to commodity PCs. The divergence point relates to the point where a MPDB starts to deviate from the constant of stability.

Table 7-1: Hypothesis for the study

Formulation	Description
$\mu 1$	Microprocessor development board
$\mu 2$	Commodity personal computer
H_o: $\mu 1 = \mu 2$	Microprocessor development boards support the data processing of small-scale database management system solutions similar to commodity personal computers
H_a: $\mu 1 \neq \mu 2$	Microprocessor development boards do not support the data processing of small-scale database management system solutions similar to commodity personal computers

In order to perform the experiment, hardware and software were prepared on all devices; the Linux Debian 8 operating system, the MySQL DBMS and the Dstat resource monitoring tool include the software installed on each device. Each device was connected to a local network and communication to and from the devices was established through the secure shell network protocol. This network connection was required in order to simulate multiple database clients connecting each device.

The load testing technique formed an important part of the experiment because it enabled the researcher to compare device performance based on each device's data processing capability and rank these devices based on the extent of load each device was able to support. An application, namely Multi-Client Simulator, was developed by the researcher that controlled the pilot and experiment in accordance with the load testing technique. This application was also responsible for simulating multiple database clients simultaneously and populating log files with pilot and experiment metrics.

The experiment was preceded by a pilot which is a scaled down and incomplete version of the experiment. After conducting the pilot, statistical analysis was performed on the captured metrics. This analysis revealed that the pilot had been successfully performed, thereby proving the

feasibility of the experiment. The captured metrics from the pilot and experiment were imported into a data warehouse designed by the researcher to allow for simple data navigation, queries and aggregation. With a combination of the organised data in the data warehouse and pivot tables, the researcher was able to extract the data required for statistical analysis.

The metrics captured from the experiment enabled statistical analysis of the data in terms of device load, duration and thread failures. One-way analysis of variance (ANOVA) was conducted on device load and duration that provided results from which hypothesis testing was possible. The hypothesis test results are repeated in Table 7-2. Hypothesis testing revealed that MPDBs do not support the data processing of small-scale DBMSs similar to commodity PCs; with the exception of the RPi3 in some phases. The BBB resulted in a Type I statistical error in Phase 15 due to its exceptionally high standard deviation in this phase that stems from an unknown error internal to the device in Iteration 8. In addition, the BBB was not capable of performing Phases 16 through 20 through all iterations owing to it becoming unresponsive after Phase 15.

Table 7-2: Hypothesis testing summary for Phases 1-20 in terms of the load and duration metrics

Device	Phase	Null Hypothesis		Exceptions
		Load Reject?	Duration Reject?	
BBB, IEA, PC, RPi3	1	Yes	Yes	RPi3 vs. PC load metric: fail to reject the null hypothesis with p (0.155) > 0.05. Therefore, in Phase 1 the RPi3's load is significantly similar to a PC. RPi3 vs. PC duration metric: fail to reject the null hypothesis with descriptive statistics. The mean duration of the RPi3 in Phase 1 is 0.84s, which is lower than that of the PC (1.23s).
	2	Yes	Yes	RPi3 vs. PC load metric: fail to reject the null hypothesis with p (0.411) > 0.05. Therefore, in Phase 2 the RPi3's load is significantly similar to a PC. RPi3 vs. PC duration metric: fail to reject the null hypothesis with descriptive statistics. The mean duration of the RPi3 in Phase 2 is 3.33s, which is lower than that of the PC (4.15s).
	3	Yes	Yes	None
	4	Yes	Yes	RPi3 vs. PC load metric: fail to reject the null hypothesis with p (0.076) > 0.05. Therefore, in Phase 4 the RPi3's load is significantly similar to a PC. RPi3 vs. PC duration metric: fail to reject the null hypothesis with descriptive statistics. The mean duration of the RPi3 in Phase 4 is 7.14s, which is lower than that of the PC (8.89s).
	5	Yes	Yes	None
	6	Yes	Yes	RPi3 vs. PC duration metric: fail to reject the null hypothesis with p (0.699) > 0.05. Therefore, in Phase 6 the RPi3's duration is significantly similar to a PC.
	7	Yes	Yes	None

Device	Phase	Null Hypothesis		Exceptions
		Load Reject?	Duration Reject?	
	8	Yes	Yes	BBB vs. IEA duration metric: significantly similar with p (0.174) > 0.05. Therefore, in Phase 8 the BBB's duration is significantly similar to the IEA, but no significance is shown when compared to a PC.
	9	Yes	Yes	None
	10	Yes	Yes	None
	11	Yes	Yes	None
	12	Yes	Yes	None
	13	Yes	Yes	RPi3 vs. PC duration metric: fail to reject the null hypothesis with p (1.00) > 0.05. Therefore, in Phase 13 the RPi3's duration is significantly similar to a PC.
	14	Yes	Yes	None
IEA, PC, RPi3	15	Yes	Yes	Type I statistical error when BBB was included due to its high standard deviation in Iteration 8.
	16	Yes	Yes	None
	17	Yes	Yes	None
	18	Yes	Yes	None
	19	Yes	Yes	None
	20	Yes	Yes	None

This study rejects the null hypothesis and it is therefore stated that MPDBs do not support the data processing of small-scale DBMS solutions similar to commodity PCs.

From the analysis of the thread metrics, a device can be ranked in terms of its point of divergence; this is indicative of the relative performance of each device to another. The following successfully completed phases are depicted in Figure 7-1:

1. *Commodity PC* – all phases completed successfully.
2. *RPi3* – 12 of 20 phases completed successfully.
3. *IEA* – 11 of 20 phases completed successfully.
4. *BBB* – six of 20 phases completed successfully.

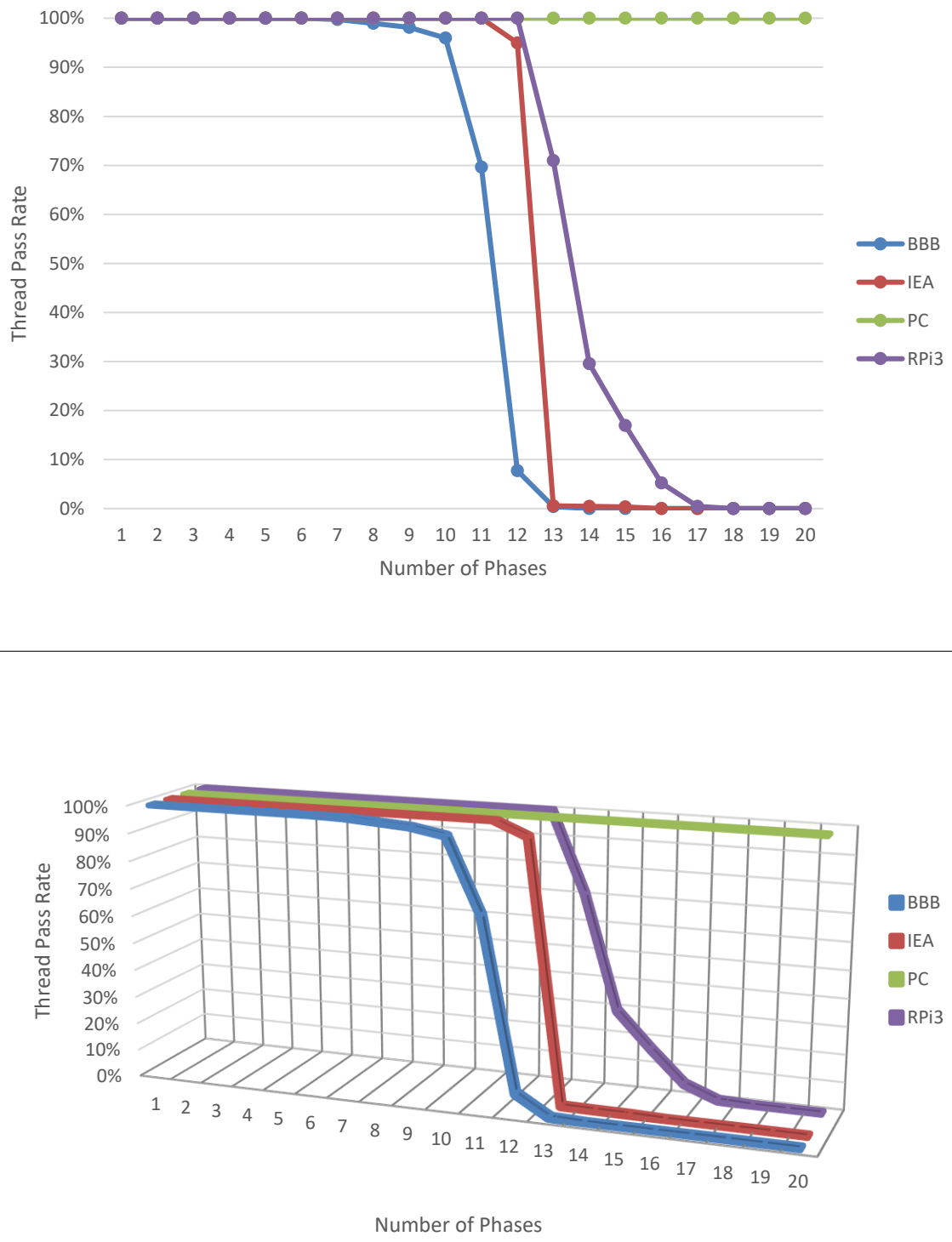


Figure 7-1: Divergence points

The divergence point of each device shows the BBB diverging from stability at the lowest load and the RPi3 at the highest load among the MPDBs. A follow-up was performed after the analysis of the experiment to further investigate the relative performance of the MPDBs in this study. The

purpose of the follow-up was to determine the maximum number of data each MPDB is able to successfully host (breakpoint) with the same database design and load testing technique from the experiment. The follow-up revealed that the breakpoint of the RPi3 is 5.74 times higher than the IEA's breakpoint and 13.15 times higher than that of the BBB. The breakpoint of the IEA is 2.29 times higher than the breakpoint of the BBB.

The commodity PC was the benchmark in the experiment and significantly outperformed the MPDBs as expected. In terms of the MPDBs, the RPi3 performed the best and the IEA's performance was significantly lower than that of the RPi3. The performance of the BeagleBone Black (BBB) was notably lower than the performance of the IEA and substantially lower than that of the RPi3.

The final outcome of the experiment results show that MPDBs are a viable alternative to commodity PCs for the hosting of DBMSs up to a certain extent of data processing. When considering a MPDB for a DBMS solution, the interested party should thoroughly analyse the expected load on the database server to ensure that a MPDB will be capable of handling the load demand. Performance of the specific MPDB and type of DBMS also play a role in the extent of load it will be able to support.

7.4 Limitations to the study

A limitation of the study is that the tested MPDBs are a sample of the population and there may be MPDBs available with increased or decreased performance. The experiment was conducted in a simulated environment and results from this type of implementation in a business or other form of practical purpose may differ. The experiment was however conducted to simulate a realistic business environment by generating simulation data and simulating database clients. MPDBs with similar (or better) performance than the MPDBs tested in the experiment should deliver similar (or improved) results.

The simulation data that was generated for use in the experiment consisted of only five tables and the table data may differ from real word data in terms of quantity and metadata. Businesses will typically have more than five tables in a DBMS. The researcher was aware of this limitation throughout the experiment and therefore neutralised the limitation by exaggerating the number of threads (database clients) and table records. In terms of this limitation, it is important to note that the number of table records and queries on these tables affect server performance, rather than the number of tables.

Another limitation to the study is that only one type of operating system-DBMS combination was used. In other words, the Linux Debian operating system was installed on all devices with MySQL as the DBMS. Other combinations of operating systems and DBMSs may deliver different results.

All limitations were known prior to the experiment and do not threaten the validity of the results.

7.5 Recommendations

In the context of the rapid progression of computer hardware and software technology, the researcher suggests research on other MPDBs, which may deliver MPDBs with higher performance than the MPDBs assessed in the study. When considering only the MPDBs evaluated by this study, the researcher would recommend the RPi3 above the IEA and BBB for the following reasons:

- The RPi3 performed significantly better than the IEA and BBB. In some phases with lower load, the RPi3 completed the simulation in a shorter duration than the PC.
- At the time of purchase, the RPi3's price amounted to only a fraction of the cost of the low-range commodity PC; a smaller fraction when compared to IEA and BBB. Prices are listed below relative to the price of the commodity PC:
  - *Low-range commodity PC* – 100 percent;
  - *RPi3* – 15.03 percent;
  - *IEA* – 56.27 percent; and
  - *BBB* – 27.29 percent.
- The researcher noticed that the RPi3 is more popular than the IEA and BBB and therefore has a larger online community for technical support.
- The RPi3 and BBB give users the option of using a MicroSD card as well as an external USB hard drive, which can be set up with ease.
- The Intel Edison module would be more appropriate for solutions such as robotics and wearable technology due to its stable rather than high performance nature; Intel however discontinued the production of this device in 2017 (List, 2017).
- The researcher would not recommend the BBB due to its high price and low performance; the RPi3 renders the BBB obsolete.

7.6 Future research

Small businesses such as cafés or restaurants may be able to make use of this technology and will allow for a relatively inexpensive hardware implementation of an information system. Unfortunately, the price of software and development thereof will not be affected and may be expensive, thereby discouraging small businesses without information systems from implementing an information system altogether. Users such as hobbyists and students that develop their own software might be more interested in implementing such solutions than owners of small businesses. Interested parties would be able to make use of MPDBs for hosting a small-scale DBMS; the data processing capability of the MPDB and the number of simultaneous database clients must however be considered.

Possibilities of future research identified during this study include:

- Database clusters are a research possibility in which the database load can be balanced across multiple devices; for example, comparing multiple RPi3 devices maintaining one database instance against multiple commodity PCs for the same purpose.
- Other types of databases and DBMS products can be explored and tested on MPDBs such as NoSQL databases.
- Possibilities of modifying MPDBs to introduce an element of hardware temperature control such as fans or water cooling that can allow devices to be overclocked and thereby improving device performance.
- The possibility that countries affected by the digital divide or with limited financial resources can extract benefit from the lower hardware costs of MPDBs compared to commodity PCs. The lower cost of MPDBs may allow users access to information technology without the expenses normally associated with it.
- Case studies of businesses implementing a MPDB to host a DBMS that is queried by multiple clients on a daily basis could add value to the information and communications technology field.

7.7 Closure of the study

The purpose of this study was to evaluate the suitability of MPDBs for hosting small-scale DBMSs. The study focused on the evaluation of MPDBs for hosting DBMSs in terms of feasibility and performance. A low-range commodity PC was used as a benchmark and compared to a selected

group of MPDBs. The researcher was able to conduct the comparison through statistical analysis after a series of experiments in which performance metrics relating to each device were captured.

This study found that, based on the results from Chapter 6, MPDBs support the hosting of a DBMS up to a certain amount of load from database queries, but not similar to that of commodity PCs. Therefore, real world implementation of such solutions need to be carefully considered in terms of MPDB performance and expected DBMS load. In the context of the digital divide (DD), the higher cost of implementing a small-scale DBMS on PCs is problematic, while the more conservative pricing of MPDBs may address the problem of the poor regarding accessing information and communication technology. Users may use MPDBs instead of PCs to implement small-scale DBMS solutions at a significantly reduced cost.

The study concludes with the statement:

> *Microprocessor development boards (MPDBs) are feasible alternatives to commodity personal computers (PCs) for the hosting of small-scale database management systems (DBMS) to a certain extent of data processing capability.*

The extent to which a MPDB is able to successfully host a DBMS depends on the performance capability of the specific MPDB.

REFERENCE LIST

Abraham S. Fischler School of Education. 2012? Quantitative research methods. http://education.nova.edu/Resources/uploads/app/35/files/arc_doc/quantitative_research_methods.pdf Date of access: 2 Nov. 2017.

Adebesin, F., Kotzé, P. & Gelderblom, H. 2011. Design research as a framework to evaluate the usability and accessibility of the digital doorway. Paper presented at the proceedings of the 2011 design, development and research conference, Cape Peninsula University of Technology, 26 September.

Algonquin College. 2012. What are the minimum requirements for Microsoft Access? http://help.algonquincollege.com/articles/FAQ/What-are-the-minimum-requirements-for-Microsoft-Access Date of access: 1 Aug. 2017.

Allan, R.A. 2001. A history of the personal computer: the people and the technology. Ontario: Allan Publishing.

Altaviser. 2008. Types of DBMS. http://www.dbms.ca/concepts/types.html Date of access: 13 Mar. 2015.

Arduino. 2015. Intel Edison. https://www.arduino.cc/en/ArduinoCertified/IntelEdison Date of access: 12 Feb. 2017.

Bassil, Y. 2011. A comparative study on the performance of the top DBMS systems. *Journal of Computer Science & Research*, 1(1):20-31.

Beagleboard.org. 2014a. BeagleBone Black. http://beagleboard.org/BLACK Date of access: 14 Mar. 2015.

Beagleboard.org. 2014b. About BeagleBoard.org and the BeagleBoard.org foundation. http://beagleboard.org/about Date of access: 26 Mar. 2015.

Bentley, L.D. & Whitten, J.L. 2007. Systems analysis & design for the global enterprise. 7th ed. New York: McGraw-Hill.

Blanche, M.J.T., Blanche, M.T., Durrheim, K. & Painter, D. 2006. Research in practice: applied methods for the social sciences. 3rd ed. Cape Town: University of Cape Town Press.

Bonet, E., Jensen, H.S., Iglesias, O. & Sauquet, A. 2007. Background of the debate on quantitative and qualitative methods. https://www.academia.edu/176086/Background_of_the_debate_on_qualitative_and_quantitative_methods Date of access: 25 Apr. 2015.

Bornman, E. 2016. Information society and digital divide in South Africa: results of longitudinal surveys. *Information, communication & society*, 19(2):264-278.

Byham, R. & Guyer, C. 2017. Stored procedures (database engine). https://docs.microsoft.com/en-us/sql/relational-databases/stored-procedures/stored-procedures-database-engine Date of access: 14 Oct. 2017.

Cattell, R. 2011. Scalable SQL and NoSQL data stores. http://cattell.net/datastores/Datastores.pdf Date of access: 24 Jan. 2016.

Cdata.com. 2016. System requirements: Microsoft Access 2016. http://cdn.cdata.com/help/DCB/cd/pg_startrequirementsmft.htm Date of access: 3 Mar. 2018.

Chapple, M. 2016. Database management system. https://www.thoughtco.com/database-management-system-1019609 Date of access: 7 Oct. 2017.

Charvet, F. & Pande, A. 2003. Database performance study. https://mis.umsl.edu/files/pdfs/TuningPaperV5.pdf Date of access: 24 Feb. 2018.

Check, J. & Schutt, R.K. 2012. Research methods in education. Boston: Sage.

Codd, E.F. 1970. A relational model of data for large shared data banks. *Communications of the association for computing machinery*, 13(6):377-387.

Collins, H. 2010. Creative research: the theory and practice of research for the creative industries. Lausanne: AVA Publications.

Comphist.org. 2004. A brief history of database systems. http://www.comphist.org/computing_history/new_page_9.htm Date of access: 25 Jul. 2017.

Consortini, A. 2013. Advantages and disadvantages of using computers in education and research. https://www.osapublishing.org/DirectPDFAccess/C56928C7-00B8-2B78-27676A2DA8726034_269194/ETOP-2013-EThB1.pdf Date of access: 24 Feb. 2018.

Coronel, C., Morris, S. & Rob, P. 2012. Database principles: fundamentals of design, implementation, and management. 10th ed. Boston: South Western Educational Publishing.

Creswell, J.W. 2003. Research design: qualitative, quantitative and mixed methods approaches. 2nd ed. Nebraska: Sage Publications.

Dalvit, L. & Gunzo, F. 2014. One year on: a longitudinal case study of computer and mobile phone use among rural South African youth. (*In* Steyn, J. & Van Greunen, D. Proceedings of the 8th International Development Informatics Association Conference organised by Nelson Mandela Metropolitan University, Port Elizabeth, South Africa. p. 164-173).

Dash, N.K. 2005. Selection of the research paradigm and methodology. http://www.celt.mmu.ac.uk/researchmethods/Modules/Selection_of_methodology/ Date of access: 25 Apr. 2015.

DevRandom. 2014. MySQL data on secondary hard disk. http://dev-random.net/mysql-data-on-secondary-hard-disk/ Date of access: 15 Oct. 2016.

Dudovskiy, J. 2016a. Positivism research philosophy. http://research-methodology.net/research-philosophy/positivism/#_ftn1 Date of access: 20 May. 2017.

Dudovskiy, J. 2016b. Pragmatism research philosophy. https://research-methodology.net/research-philosophy/pragmatism-research-philosophy/ Date of access: 9 Oct. 2017.

Edmonds, W.A. & Kennedy, T.D. 2017. An applied reference guide to research designs: quantitative, qualitative, and mixed methods. Thousand Oaks: Sage.

eLinux.org. 2015. Beagleboard: Debian on BeagleBone Black. http://elinux.org/Beagleboard:Debian_On_BeagleBone_Black Date of access: 9 Oct. 2016.

eLinux.org. 2017. RPi HardwareHistory. https://elinux.org/RPi_HardwareHistory Date of access: 11 Oct. 2017.

Elmasri, R. & Navathe, S.B. 2011. Fundamentals of database systems. 6th ed. Boston: Pearson Education.

Evetech. 2017. Intel 7th gen Pentium G4560 3.5GHz desktop PC. https://www.evetech.co.za/Computer-systems/intel-pentium-pc-203.aspx Date of access: 14 Sep. 2017.

Gabry, O. 2016. Database-indexing, transactions & stored procedures (part 9): optimization, working with sensitive data and re-usability. https://medium.com/omarelgabrys-blog/database-indexing-and-transactions-part-9-a24781d429f8 Date of access: 14 Oct. 2017.

Ganaye, P. 2012. Create a sample SQL database in less than 2 minutes. https://www.codeproject.com/Tips/326527/Create-a-Sample-SQL-Database-in-Less-Than-Minute Date of access: 16 Sep. 2017.

Goldkuhl, G. 2012. Pragmatism vs interpretivism in qualitative information systems research. *European Journal of Information Systems*, 21(2):135-146.

Golfarelli, M. & Rizzi, S. 2009. Data warehouse design: modern principles and methodologies. Bologna: McGraw-Hill.

Gray, D.E. 2014. Doing research in the real world. 3rd ed. London: Sage.

Gregor, S. & Hevner, A.R. 2013. Positioning and presenting design science research for maximum impact. *Management information systems quaterly*, 37(2):337-355.

Heng, C. 2017. What is MySQL? What is a database? What is SQL? https://www.thesitewizard.com/faqs/what-is-mysql-database.shtml Date of access: 1 Oct. 2017.

Hevner, A.R., March, S.T., Park, J. & Ram, S. 2004. Design science in information systems research. *Management information systems quaterly*, 28(1):75-105.

Hirschheim, R. 2010. Philosophy of science and research methods in information systems. http://www.darsis.dk/fileadmin/user_upload/Darsis_course_Oct_2010/Philosophy_of_Science_and_Research_Methods_ISDS7950_v2.pdf Date of access: 19 Apr. 2015.

Hoffman, C. 2014. Understanding the load average on Linux and other Unix-like systems. https://www.howtogeek.com/194642/understanding-the-load-average-on-linux-and-other-unix-like-systems/ Date of access: 7 Aug. 2017.

Hymel, S. 2015. Loading Debian on the Edison. https://learn.sparkfun.com/tutorials/loading-debian-ubilinux-on-the-edison Date of access: 9 Oct. 2016.

Intel. 2015. The story of the Intel 4004. http://www.intel.com/content/www/us/en/history/museum-story-of-intel-4004.html Date of access: 18 Mar. 2015.

Intel. 2017. What is a microprocessor? http://download.intel.com/newsroom/kits/40thanniversary/pdfs/What_is_a_Microprocessor.pdf Date of access: 7 Oct. 2017.

Intel Corporation. 2015. Intel Edison compute module. http://download.intel.com/support/edison/sb/edisonmodule_hg_331189004.pdf Date of access: 12 Feb. 2017.

Intel Corporation. 2017. Intel Edison compute module. https://software.intel.com/en-us/iot/hardware/edison Date of access: 9 Aug. 2017.

Irny, S.I. & Rose, A.A. 2005. Designing a strategic information systems planning methodology for Malaysian institutes of higher learning. *Issues in information systems*, 6(1):325-326.

Johnson, A. 2014. Raspberry Pi MySQL Workbench. http://andrewtjohnson.me/2014/06/18/raspberry-pi-mysql-workbench/ Date of access: 15 Oct. 2016.

Johnson, P. & Harris, D. 2002. Qualitative and quantitative issues in research design. London: Sage.

Kant, K. 2007. Microprocessors and microcontrollers: architecture, programming and system design. New Delhi: PHI Learning.

Kimball, R., Ross, M., Thornthwaite, W., Mundy, J. & Becker, B. 2008. The data warehouse lifecycle toolkit. 2nd ed. Indianapolis: Wiley Publishing.

Koomey, J.G., Belady, C., Patterson, M., Santos, A. & Lange, K. 2009. Assessing trends over time in performance, costs, and energy use for servers. http://www.intel.com/assets/pdf/general/servertrendsreleasecomplete-v25.pdf Date of access: 16 Mar. 2015.

Linksprite.com. 2015. Mini PC + Arduino. http://store.linksprite.com/ Date of access: 25 Mar. 2015.

Linux Information Project. 2006. Computer definition. http://www.linfo.org/computer.html Date of access: 29 Mar. 2015.

List, J. 2017. Intel discontinues Joule, Galileo, and Edison product lines. https://hackaday.com/2017/06/19/intel-discontinues-joule-galileo-and-edison-product-lines/ Date of access: 11 Oct. 2017.

Lyons. 2015. A history of the Raspberry Pi. http://novadigitalmedia.com/history-raspberry-pi/ Date of access: 10 Oct. 2017.

McGregor, S.L.T. & Murnane, J.A. 2010. Paradigm, methodology and method: intellectual integrity in consumer scholarship. *International Journal of Consumer Studies*, 34(4):419-427.

Meier, J.D., Farre, C., Bansode, P., Barber, S. & Rea, D. 2007. Load-testing web applications. https://msdn.microsoft.com/en-us/library/bb924372.aspx Date of access: 13 Mar. 2015.

Microsoft. 2017. Hardware and software requirements for installing SQL Server. https://docs.microsoft.com/en-us/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server Date of access: 3 Mar. 2018.

Mohammed, A. 2016. Choosing the right database management system. http://www.computerweekly.com/feature/Choosing-the-right-database-management-system Date of access: 10 Oct. 2017.

Morris, S.A., Coronel, C. & Rob, P. 2013. Database principles: fundamentals of design, implementation, and management. 10th ed. Boston: Cengage Learning.

Mullins, C.S. 2010. Defining database performance. http://www.dbta.com/Columns/DBA-Corner/Defining-Database-Performance-70236.aspx Date of access: 13 Mar. 2015.

Mullins, C.S. 2013. Database administration: the complete guide to DBA practices and procedures. 2nd ed. New Jersey: Addison-Wesley.

Munsell, A. 2013. Getting started with Raspberry Pi: installing Raspbian. https://www.andrewmunsell.com/blog/getting-started-raspberry-pi-install-raspbian/ Date of access: 9 Oct. 2016.

Myers, M. 1997. Qualitative research in information systems. *Management information systems quarterly*, 21(2):241-242.

Myers, M.D. 2013. Qualitative research in information systems. http://www.qual.auckland.ac.nz/ Date of access: 14 May. 2017.

MySQL. 2017a. Supported platforms: MySQL database. https://www.mysql.com/support/supportedplatforms/database.html Date of access: 1 Aug. 2017.

MySQL. 2017b. Why MySQL? https://www.mysql.com/why-mysql/ Date of access: 1 Oct. 2017.

MySQL. 2017c. MySQL TCO savings calculator. https://www.mysql.com/tcosavings/ Date of access: 1 Oct. 2017.

MySQL. 2017d. MySQL Enterprise: system requirements. https://dev.mysql.com/doc/mysql-monitor/4.0/en/system-prereqs-reference.html Date of access: 3 Mar. 2018.

Noller, A. 2014. The 4 types of NoSQL database. https://dzone.com/articles/4-types-nosql-database Date of access: 10 Oct. 2017.

Noor, K.B.M. 2008. Case study: a strategic research methodology. *American journal of applied sciences*, 5(11):1602-1604.

OECD. 2002. Frascati manual: proposed standard practice for surveys on research and experimental development. 6th ed. Paris: OECD Publishing.

Oracle.com. 2017. Oracle MySQL. https://www.oracle.com/mysql/index.html Date of access: 1 Oct. 2017.

Pcduino.com. 2015. LinkSprite pcDuino. http://www.pcduino.com/ Date of access: 13 Mar. 2015.

Privitera, G.J. 2014. Research methods for the behavioural sciences. London: Sage.

Ramakrishnan, R. & Gehrke, J. 2003. Database management systems. 3rd ed. New York: McGraw-Hill.

Ramey, K. 2013. What are the disadvantages and advantages of computers in a business? https://www.useoftechnology.com/disadvantages-advantages-computers/ Date of access: 14 Oct. 2017.

Raparthi, C. 2014. Flat file database. http://sqlversity.com/flat-file-dbms/ Date of access: 10 Oct. 2017.

Raspberrypi.org. 2015a. What is a Raspberry Pi? http://www.raspberrypi.org/help/what-is-a-raspberry-pi/ Date of access: 14 Mar. 2015.

Raspberrypi.org. 2015b. About us. http://www.raspberrypi.org/about/ Date of access: 25 Mar. 2015.

Raspberrypi.org. 2015c. Raspberry Pi 3 Model B. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ Date of access: 12 Feb. 2017.

Raspberrypi.org. 2015d. Projects. http://www.raspberrypi.org/forums/ Date of access: 28 Mar. 2015.

Raspberrypi.org. 2015e. Downloads. https://www.raspberrypi.org/downloads/ Date of access: 19 Aug. 2017.

RaspberryPi.org. 2016. Installing operating system images. https://www.raspberrypi.org/documentation/installation/installing-images/ Date of access: 9 Oct. 2016.

Riecktron. 2017. pcDuino3B: dev board. https://www.riecktron.co.za/en/product/3519?search=pcduino Date of access: 14 Sep. 2017.

RS Components. 2017a. Raspberry Pi 3 model B SBC. http://za.rs-online.com/web/p/processor-microcontroller-development-kits/8968660/ Date of access: 14 Sep. 2017.

RS Components. 2017b. BeagleBone Black, revision C. http://za.rs-online.com/web/p/processor-microcontroller-development-kits/1252411/?sra=pmpn Date of access: 14 Sep. 2017.

RS Components. 2017c. Intel Edison board for Arduino. http://za.rs-online.com/web/p/processor-microcontroller-development-kits/8330895/ Date of access: 14 Sep. 2017.

Samiksha, S. 2016. Models of database architecture: hierarchical, network and relational models. http://www.yourarticlelibrary.com/database/models-of-database-architecture-hierarchical-network-and-relational-models/10389/ Date of access: 10 Oct. 2017.

SAS. 2005. Concepts of experimental design: design institute for six sigma. https://support.sas.com/resources/papers/sixsigma1.pdf Date of access: 3 May. 2015.

Saunders, M., Lewis, P. & Thornhill, A. 2009. Research methods for business students. 5th ed. Harlow: Pearson.

Seymour, G. 2010. Introduction to database management systems. https://hostway.com/blog/introduction-to-database-management-systems/ Date of access: 10 Oct. 2017.

Siegle, D. 2015. Introduction to correlation research. http://researchbasics.education.uconn.edu/correlation/ Date of access: 20 May. 2017.

Sipral, S. 2007. Basic computer concepts. http://vfu.bg/en/e-Learning/Computer-Basics--computer_basics2.pdf Date of access: 25 Mar. 2015.

Soltan, L. 2016. Digital divide: the technology gap between the rich and poor. http://www.digitalresponsibility.org/digital-divide-the-technology-gap-between-rich-and-poor/ Date of access: 24 Sep. 2017.

Soni, A. 2010. High availability principle: request queueing. https://ideasoni.wordpress.com/tag/concurrency-control/ Date of access: 12 Feb. 2017.

Srivastava, T. 2014. Types of database management system and their evolution. http://www.analyticsvidhya.com/blog/2014/11/types-databases-evolution/ Date of access: 23 Jan. 2016.

SSH Communications Security. 2017. SSH protocol. https://www.ssh.com/ssh/protocol/ Date of access: 17 Sep. 2017.

Stackoverflow.com. 2009. Just what is 'a big database'? https://stackoverflow.com/questions/647210/just-what-is-a-big-database Date of access: 3 Mar. 2018.

Statistics Solutions. 2013. ANOVA. http://www.statisticssolutions.com/manova-analysis-anova/ Date of access: 9 Oct. 2017.

Stewright.me. 2014. Tutorial: install MySQL server on Raspberry Pi. https://www.stewright.me/2014/06/tutorial-install-mysql-server-on-raspberry-pi/ Date of access: 15 Oct. 2016.

Suehring, S. 2002. MySQL Bible. New York: Wiley Publishing.

Vaishnavi, V. & Kuechler, W. 2004. Design science research in information systems. http://desrist.org/desrist/content/design-science-research-in-information-systems.pdf Date of access: 27 Mar. 2015.

Visual Paradigm. 2011. Relational database design with ERD. http://www.visual-paradigm.com/tutorials/databasedesign.jsp Date of access: 11 Feb. 2017.

Wieërs, D. 2016. Dstat: versatile resource statistics tool. http://dag.wiee.rs/home-made/dstat/ Date of access: 15 Jun. 2017.

Williams, C. 2007. Research methods. *Journal of business & economic research*, 5(3):65-72.

Yount, W.R. 2006. Research design and statistical analysis in Christian ministry. http://www.napce.org/documents/research-design-yount/13_experiment_4th.pdf Date of access: 3 May. 2015.

(P*AGE INTENTIONALLY LEFT BLANK)*

APPENDIX A

A.1 Preparation of computing devices

This section describes how each device is prepared in terms of hardware, operating system (OS) and database management system (DBMS).

A.1.1 Hardware preparation

The hardware preparation is standard with devices only requiring a power connection. All MPDBs are powered with a 5V 2AMP USB adapter and the PC with the normal kettle-plug cord that most desktop PCs use:

- The BeagleBone Black has a Mini USB port as its power source and on-board 2GB eMMC storage on which the OS is installed.
- To combine the Intel Edison chip with the Arduino breakout board, the Intel chip was mounted on the Arduino breakout board. The Intel Edison has 4GB on-board eMMC storage onto which the OS can be installed. The board and chip are powered by a Micro USB port.
- The Raspberry Pi 3 is powered by a Micro USB port and requires a MicroSD card onto which the OS must be loaded.
- The PC requires a plugged in power cable, monitor, keyboard and mouse.

A.1.2 Operating system installation

The same OS was installed on all devices to minimise uncontrollable variables that relate to the OS environment. The chosen OS is Linux Debian 8, which is compatible with all computing devices. Note that each device manufacturer maintains its own distribution of Debian, which ensures hardware-OS compatibility.

A.1.2.1 First steps followed on BeagleBone Black and Raspberry Pi 3

These steps only relate to the BeagleBone Black (eLinux.org, 2015) and Raspberry Pi 3 (RaspberryPi.org, 2016).

1. Download the Linux Debian 8 image compatible with the particular MPDB.
2. Extract the compressed image file with an extractor application such as 7zip.
3. Insert the MicroSD card into the Windows development PC using a Micro-SD-to-USB adapter.
4. Write the image to the MicroSD card using an application called Win32 Disk Imager.
5. Insert the MicroSD card into the unplugged MPDB.

The following MPDB specific steps were performed after the steps above were completed.

A.1.2.2 BeagleBone Black

The steps listed below was acquired and followed from a guide provided by eLinux.org (2015).

1. Connect a keyboard, mouse, monitor and Ethernet cable to the BeagleBone Black (BBB).
2. While holding down the boot button on the BBB, apply power to the board. Continue to hold the boot button until the USER LEDs start flashing.
3. The image is then automatically flashed to the eMMC from which the board boots.
4. When the flash process is complete, the USER LEDs will stop flickering.
5. Disconnect the power, remove the MicroSD, and then re-apply power to the board.
6. When the board has booted, the command line terminal for Debian 8 is displayed with a login prompt; the default login credentials are:
   Username: debian
   Password: temppwd

A.1.2.3 Raspberry Pi 3

The Raspberry Pi 3 (RPi3) OS setup guide provided by RaspberryPi.org (2016) and Munsell (2013) was followed:

1. Connect a keyboard, mouse, monitor and Ethernet cable to the RPi3 and connect the power cable to the board.
2. Once powered, OS settings such as language will be requested.
3. After these settings are specified the RPi3 boots to the graphical user interface.
4. In the menu pane, the terminal command line can be selected, which opens the Debian 8 command line terminal. The default login credentials are:

Username: pi
Password: raspberry

A.1.2.4 Intel Edison Arduino

The Intel Edison Arduino (IEA) does not have any type of display port. The installation process therefore requires the board to be connected to a PC. In order for the Windows development PC to interact with the IEA, software needs to be installed on the PC. The Debian 8 and Windows software installation are detailed in the steps below, acquired from a guide by Hymel (2015):

1. Download and install the Intel Edison drivers on Windows from Intel's website.
2. Download the DFU utility which is required for the uploading of firmware to USB connected devices.
3. Download the Ubilinux (embedded Linux distribution based on Debian) image to be flashed to the board.
4. Extract the compressed Ubilinux image and the DFU utility with 7zip.
5. Copy "dfu-util.exe" to the contents of the extracted Ubilinux image and execute "flashall.bat", which will open a command prompt window requesting for the device to be connected to the PC.
6. Toggle the micro switch on the board towards the two Micro USB ports (J16 and J3) to enable communication to the PC through these ports. See Figure A-1.
7. Connect the J16 port to the PC with a Micro-USB-to-USB cable. This port powers the board and reads/writes to the on-board eMMC flash memory. See Figure A-1.
8. Connect the J3 port to the PC as well; this port is used for shell access via serial communication. A visual representation is shown in Figure A-1.

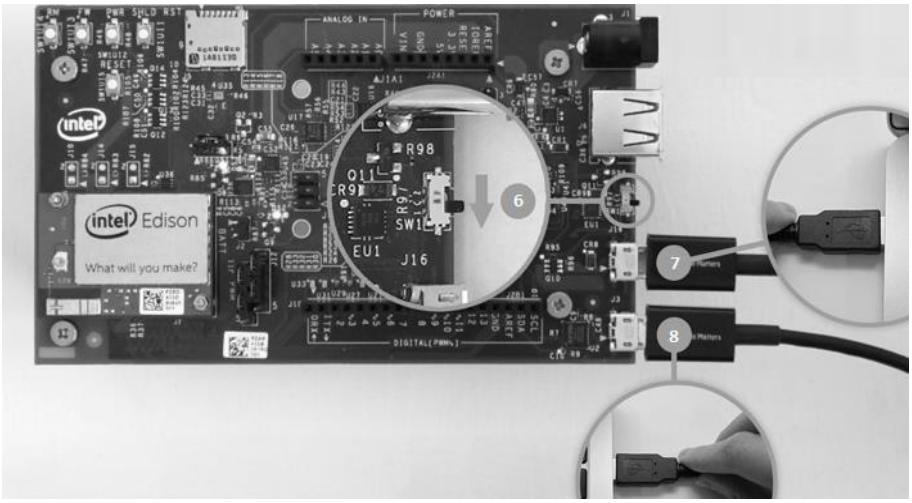


Figure A-1: Intel Edison Arduino setup

9. The installation then starts automatically and after the command prompt window closes, the device is left plugged in for two minutes.
10. Download and install the Virtual COM port drivers supplied by FTDI.
11. Debian 8 can now be accessed through the USB serial port with the use of a serial terminal application such as PuTTY.
12. Once the connection is opened, the command line terminal for Debian 8 is displayed with the login prompt; the default login credentials are:
    Username: edison
    Password: edison

A.1.2.5 Commodity personal computer

Due to restrictions in resources available to the study, a low-range commodity PC was not available. The high-end development PC was, however, available to simulate a low-range PC. VMWare Workstation is an application that allow OSs to host additional OSs on top of the main OS. The application also allows the user to specify the number of resources allocated from the main OS to the virtual OS.

Using VMWare Workstation, a new virtual system is created, allocated with two 3.5GHz cores and 8GB of RAM. The network of the virtual machine is bridged from the physical PC, which is equipped with a gigabit Ethernet connection.

A disk image of Linux Debian 8 was downloaded from the official Debian website. The “Console Only” version of Debian 8 was chosen, since it conforms to the standard of the MPDBs. The OS is installed on the virtual machine by selecting the disk image via VMWare Workstation and following the on-screen installation process. The only reason for the use of the virtual machine in this study is to limit the CPU clock speed and amount of RAM of the development PC to simulate a low-range PC, which is accurately done through VMware Workstation and does not threaten the validity of the study.

Once the installation is completed and the system booted, the user is presented with the command line terminal for Debian 8 with the login prompt. The default login credentials are:

Username: debian
Password: debian

A.1.3 Database management system installation

The DBMS, MySQL Server, is installed on all devices and the same installation steps are followed on all devices.

A.1.3.1 Installation

The installation required the following CLI commands on each device (Stewright.me, 2014):

1. Update the OS to the latest available version:
   ```
   sudo apt-get update && sudo apt-get upgrade
   ```
2. Install MySQL Server:
   ```
   sudo apt-get install mysql-server --fix-missing
   ```
3. The user is then prompted for a password for the default user “root”.
4. Install MySQL Client that allows connections to MySQL Server via the CLI:
   ```
   sudo apt-get install mysql-client
   ```
5. Connect to the MySQL Server instance:
   ```
   mysql -uroot -hlocalhost –p
   ```
6. Enter the password from Step 3.
7. Create a new user that will be used to connect to MySQL Server from the SSH tunnel:
   ```
   CREATE USER 'mscuser'@'%' IDENTIFIED BY 'password';
   GRANT ALL PRIVILEGES ON *.* TO 'mscuser'@'%';
   FLUSH PRIVILEGES;
   ```

A.1.3.2 Storage setup

A guide provided by a tech blog is followed to set up the storage locations of the databases on the different devices (DevRandom, 2014).

Each MPDB is set up to store the database data locally on a MicroSD card. The same approach is followed with the commodity PC which uses an internal hard drive as local storage.

In the pilot, the BBB and RPi3 were each set up with an external hard drive that connects to an USB port on the device. The database data is stored on the external hard drives. This approach was followed to prove that there is practically no restriction on storage space for these devices. The IEA does not consist of a client USB port; the database data was therefore set up to be stored to a MicroSD card connected to the device. The commodity PC was set up to use an internal hard drive.

A.2 MySQL server variables

These server variables are uniform across all devices.

```
auto_increment_increment.............................. 1
auto_increment_offset................................. 1
autocommit............................................ ON
automatic_sp_privileges............................... ON
back_log.............................................. 50
big_tables............................................ OFF
binlog_cache_size..................................... 32768
binlog_direct_non_transactional_updates............... OFF
binlog_format......................................... STATEMENT
binlog_stmt_cache_size................................ 32768
bulk_insert_buffer_size............................... 8388608
character_set_client.................................. latin1
character_set_connection.............................. latin1
character_set_database................................ latin1
character_set_filesystem.............................. binary
character_set_results................................. latin1
character_set_server.................................. latin1
character_set_system.................................. utf8
collation_connection.................................. latin1_swedish_ci
collation_database.................................... latin1_swedish_ci
collation_server...................................... latin1_swedish_ci
completion_type....................................... NO_CHAIN
concurrent_insert..................................... AUTO
connect_timeout....................................... 10
date_format........................................... %Y-%m-%d
datetime_format....................................... %Y-%m-%d %H:%i:%s
```

```
default_storage_engine.............................. InnoDB
default_week_format................................. 0
delay_key_write..................................... ON
delayed_insert_limit................................ 100
delayed_insert_timeout.............................. 300
delayed_queue_size.................................. 1000
div_precision_increment............................. 4
engine_condition_pushdown........................... ON
event_scheduler..................................... OFF
expire_logs_days.................................... 10
flush............................................... OFF
flush_time.......................................... 0
foreign_key_checks.................................. ON
ft_boolean_syntax................................... + -><()~*:""&|
ft_max_word_len..................................... 84
ft_min_word_len..................................... 4
ft_query_expansion_limit............................ 20
ft_stopword_file.................................... (built-in)
general_log......................................... OFF
group_concat_max_len................................ 1024
have_compress....................................... YES
have_crypt.......................................... YES
have_csv............................................ YES
have_dynamic_loading................................ YES
have_geometry....................................... YES
have_innodb......................................... YES
have_ndbcluster..................................... NO
have_openssl........................................ DISABLED
have_partitioning................................... YES
have_profiling...................................... YES
have_query_cache.................................... YES
have_rtree_keys..................................... YES
have_ssl............................................ DISABLED
have_symlink........................................ YES
hostname............................................ arm
ignore_builtin_innodb............................... OFF
innodb_adaptive_flushing............................ ON
innodb_adaptive_hash_index.......................... ON
innodb_additional_mem_pool_size..................... 8388608
innodb_autoextend_increment......................... 8
innodb_autoinc_lock_mode............................ 1
innodb_buffer_pool_instances........................ 1
innodb_buffer_pool_size............................. 134217728
innodb_change_buffering............................. all
innodb_checksums.................................... ON
innodb_commit_concurrency........................... 0
innodb_concurrency_tickets.......................... 500
innodb_doublewrite.................................. ON
innodb_fast_shutdown................................ 1
innodb_file_format.................................. Antelope
innodb_file_format_check............................ ON
innodb_file_format_max.............................. Antelope
```

```
innodb_file_per_table.............................. OFF
innodb_flush_log_at_trx_commit..................... 1
innodb_flush_method.................................
innodb_force_load_corrupted........................ OFF
innodb_force_recovery.............................. 0
innodb_io_capacity................................. 200
innodb_large_prefix................................ OFF
innodb_lock_wait_timeout........................... 50
innodb_locks_unsafe_for_binlog..................... OFF
innodb_log_buffer_size............................. 8388608
innodb_log_file_size............................... 5242880
innodb_log_files_in_group.......................... 2
innodb_max_dirty_pages_pct......................... 75
innodb_max_purge_lag............................... 0
innodb_mirrored_log_groups......................... 1
innodb_old_blocks_pct.............................. 37
innodb_old_blocks_time............................. 0
innodb_open_files.................................. 300
innodb_print_all_deadlocks......................... OFF
innodb_purge_batch_size............................ 20
innodb_purge_threads............................... 0
innodb_random_read_ahead........................... OFF
innodb_read_ahead_threshold........................ 56
innodb_read_io_threads............................. 4
innodb_replication_delay........................... 0
innodb_rollback_on_timeout......................... OFF
innodb_rollback_segments........................... 128
innodb_spin_wait_delay............................. 6
innodb_stats_method................................ nulls_equal
innodb_stats_on_metadata........................... ON
innodb_stats_sample_pages.......................... 8
innodb_strict_mode................................. OFF
innodb_support_xa.................................. ON
innodb_sync_spin_loops............................. 30
innodb_table_locks................................. ON
innodb_thread_concurrency.......................... 0
innodb_thread_sleep_delay.......................... 10000
innodb_use_native_aio.............................. ON
innodb_use_sys_malloc.............................. ON
innodb_write_io_threads............................ 4
interactive_timeout................................ 28800
join_buffer_size................................... 131072
keep_files_on_create............................... OFF
key_buffer_size.................................... 16777216
key_cache_age_threshold............................ 300
key_cache_block_size............................... 1024
key_cache_division_limit........................... 100
large_files_support................................ ON
large_page_size.................................... 0
large_pages........................................ OFF
lc_messages........................................ en_US
lc_time_names...................................... en_US
```

```
license..... GPL
local_infile..... ON
lock_wait_timeout..... 31536000
locked_in_memory..... OFF
log..... OFF
log_bin..... OFF
log_bin_trust_function_creators..... OFF
log_output..... FILE
log_queries_not_using_indexes..... OFF
log_slave_updates..... OFF
log_slow_queries..... OFF
log_warnings..... 1
long_query_time..... 10.000000
low_priority_updates..... OFF
lower_case_file_system..... OFF
lower_case_table_names..... 0
max_allowed_packet..... 16777216
max_binlog_cache_size..... 18446744073709547520
max_binlog_size..... 104857600
max_binlog_stmt_cache_size..... 18446744073709547520
max_connect_errors..... 100
max_connections..... 1000
max_delayed_threads..... 20
max_error_count..... 64
max_heap_table_size..... 16777216
max_insert_delayed_threads..... 20
max_join_size..... 18446744073709551615
max_length_for_sort_data..... 1024
max_long_data_size..... 16777216
max_prepared_stmt_count..... 16382
max_relay_log_size..... 0
max_seeks_for_key..... 4294967295
max_sort_length..... 1024
max_sp_recursion_depth..... 0
max_tmp_tables..... 32
max_user_connections..... 0
max_write_lock_count..... 4294967295
metadata_locks_cache_size..... 1024
min_examined_row_limit..... 0
multi_range_count..... 256
myisam_data_pointer_size..... 6
myisam_max_sort_file_size..... 2146435072
myisam_mmap_size..... 4294967295
myisam_recover_options..... BACKUP
myisam_repair_threads..... 1
myisam_sort_buffer_size..... 8388608
myisam_stats_method..... nulls_unequal
myisam_use_mmap..... OFF
net_buffer_length..... 16384
net_read_timeout..... 300
net_retry_count..... 10
net_write_timeout..... 300
```

```
new.......................................................... OFF
old.......................................................... OFF
old_alter_table.............................................. OFF
old_passwords................................................ OFF
open_files_limit............................................. 5000
optimizer_prune_level........................................ 1
optimizer_search_depth....................................... 62
optimizer_switch.............................................
index_merge=on,index_merge_union=on,index_merge_sort_union=on,index_merge_intersection
=on,engine_condition_pushdown=on
performance_schema........................................... OFF
performance_schema_events_waits_history_long_size..... 10000
performance_schema_events_waits_history_size.......... 10
performance_schema_max_cond_classes.......................... 80
performance_schema_max_cond_instances........................ 1000
performance_schema_max_file_classes.......................... 50
performance_schema_max_file_handles.......................... 32768
performance_schema_max_file_instances........................ 10000
performance_schema_max_mutex_classes......................... 200
performance_schema_max_mutex_instances....................... 1000000
performance_schema_max_rwlock_classes........................ 30
performance_schema_max_rwlock_instances...................... 1000000
performance_schema_max_table_handles......................... 100000
performance_schema_max_table_instances....................... 50000
performance_schema_max_thread_classes........................ 50
performance_schema_max_thread_instances...................... 1000
port......................................................... 3306
preload_buffer_size.......................................... 32768
profiling.................................................... OFF
profiling_history_size....................................... 15
protocol_version............................................. 10
query_alloc_block_size....................................... 8192
query_cache_limit............................................ 1048576
query_cache_min_res_unit..................................... 4096
query_cache_size............................................. 16777216
query_cache_type............................................. ON
query_cache_wlock_invalidate................................. OFF
query_prealloc_size.......................................... 8192
range_alloc_block_size....................................... 4096
read_buffer_size............................................. 131072
read_only.................................................... OFF
read_rnd_buffer_size......................................... 262144
relay_log_purge.............................................. ON
relay_log_recovery........................................... OFF
relay_log_space_limit........................................ 0
report_port.................................................. 3306
report_user.................................................
rpl_recovery_rank............................................ 0
secure_auth.................................................. OFF
server_id.................................................... 0
skip_external_locking........................................ ON
skip_name_resolve............................................ OFF
```

```
skip_networking..................................... OFF
skip_show_database.................................. OFF
slave_compressed_protocol........................... OFF
slave_exec_mode..................................... STRICT
slave_max_allowed_packet............................ 1073741824
slave_net_timeout................................... 3600
slave_skip_errors................................... OFF
slave_transaction_retries........................... 10
slave_type_conversions...............................
slow_launch_time.................................... 2
slow_query_log...................................... OFF
sort_buffer_size.................................... 2097152
sql_auto_is_null.................................... OFF
sql_big_selects..................................... ON
sql_big_tables...................................... OFF
sql_buffer_result................................... OFF
sql_log_bin......................................... ON
sql_log_off......................................... OFF
sql_low_priority_updates............................ OFF
sql_max_join_size................................... 18446744073709551615
sql_mode............................................
sql_notes........................................... ON
sql_quote_show_create............................... ON
sql_safe_updates.................................... OFF
sql_select_limit.................................... 18446744073709551615
sql_slave_skip_counter.............................. 0
sql_warnings........................................ OFF
ssl_ca..............................................
ssl_capath..........................................
ssl_cert............................................
ssl_cipher..........................................
ssl_key.............................................
storage_engine...................................... InnoDB
stored_program_cache................................ 256
sync_binlog......................................... 0
sync_frm............................................ ON
sync_master_info.................................... 0
sync_relay_log...................................... 0
sync_relay_log_info................................. 0
system_time_zone.................................... UTC
table_definition_cache.............................. 400
table_open_cache.................................... 400
thread_cache_size................................... 8
thread_concurrency.................................. 10
thread_handling..................................... one-thread-per-connection
thread_stack........................................ 196608
time_format......................................... %H:%i:%s
time_zone........................................... SYSTEM
timed_mutexes....................................... OFF
tmp_table_size...................................... 16777216
transaction_alloc_block_size........................ 8192
transaction_prealloc_size........................... 4096
```

```
tx_isolation..................................... REPEATABLE-READ
unique_checks.................................... ON
updatable_views_with_limit....................... YES
wait_timeout..................................... 28800
```

A.3 Configuration files

The configuration files shown in this section are used by the Multi-Client Simulator and are of file type, ini. The content of the configuration files can be understood with the use of the following legend:

```
[Section]
<keyname>=<value>
```

A.3.1 Experiment

This section details the configuration files for the BeagleBone Black, Intel Edison Arduino, Raspberry Pi 3 and commodity personal computer. The following sections are identical in all four configuration files:

```
[LOG]
DiagnosticLogging=false
LogDbLines=false

[STORED_PROCEDURES]
1=0_C_Product
2=0_C_Client
3=0_C_Order
4=0_R_ClientDetails
5=0_R_OrderDetails
6=0_R_ClientOccupation
7=0_U_Product
8=0_U_Order
9=0_U_Client
10=0_U_OrderLine
11=0_D_Client
12=1_R_ClientOrders
13=1_R_ClientOrdersPaid
14=1_R_ClientOrdersCancelled
15=2_R_ClientNotes
16=2_R_ClientNotesCity
17=3_R_CreditCheck
18=3_R_ClientNotesOrders
```

```
19=3_R_AmountOfOrdersDate
20=4_R_SoldPerProduct

[PHASE_STORED_PROCEDURES]
IndexDB=4,5,6,12,13,14,15,16,17,18,19,20
1=4
2=5
3=12
4=4,5
5=14
6=4,5,6
7=1,2,3,11
8=7,8,9,10
9=2,9,11
10=13,14
11=17
12=18
13=4,5,6
14=2,4,9,11
15=20
16=17,18
17=12,13,14
18=18,19
19=15,20
20=17,18,19

[PHASE_NO_OF_THREADS]
1=5
2=25
3=30
4=25
5=40
6=10
7=7
8=12
9=30
10=10
11=60
12=100
13=150
14=200
15=150
16=45
17=150
18=60
19=50
20=100
```

The settings that follow are unique to each device’s configuration file.

A.3.1.1 BeagleBone Black

```
[MPDBCONNECTION]
Type=BBB
Host=192.168.1.170
Port=22
Username=bbb
Password=router127
RootPassword=router127
SleepDuration=1000

[DBCONNECTION]
Server=192.168.1.170
Database=fabrics
Username=mscuser
Password=router127
CommandTimeout=60

[EXPERIMENT_SETTINGS]
IndexDB=true
NumberOfPhases=15
NumberOfIterations=30
```

Note that the BeagleBone Black only performed 15 phases due to the device becoming unreachable and failing all database queries in phases 16 to 20.

A.3.1.2 Intel Edison Arduino

```
[MPDBCONNECTION]
Type=IEA
Host=192.168.1.160
Port=22
Username=edison
Password=router127
RootPassword=router127
SleepDuration=1000

[DBCONNECTION]
Server=192.168.1.160
Database=fabrics
Username=mscuser
Password=router127
CommandTimeout=60

[EXPERIMENT_SETTINGS]
IndexDB=true
NumberOfPhases=20
```

```
NumberOfIterations=30
```

A.3.1.3 Raspberry Pi 3

```
[MPDBCONNECTION]
Type=RPi3
Host=192.168.1.150
Port=22
Username=pi
Password=router127
RootPassword=router127
SleepDuration=1000

[DBCONNECTION]
Server=192.168.1.150
Database=fabrics
Username=mscuser
Password=router127
CommandTimeout=60

[EXPERIMENT_SETTINGS]
IndexDB=true
NumberOfPhases=20
NumberOfIterations=30
```

A.3.1.4 Commodity personal computer

```
[MPDBCONNECTION]
Type=PC
Host=192.168.1.180
Port=22
Username=pc
Password=router127
RootPassword=router127
SleepDuration=1000

[DBCONNECTION]
Server=192.168.1.180
Database=fabrics
Username=mscuser
Password=router127
CommandTimeout=60

[EXPERIMENT_SETTINGS]
IndexDB=true
NumberOfPhases=20
```

```
NumberOfIterations=30
```

A.3.2 Pilot

The pilot required one configuration file per phase for each device. The following four configuration file sections are populated with the relevant stored procedures from each phase:

```
[SP_CREATE]
SPs=fabrics.0_C_Client(), 0_C_Order()

[SP_READ]
SPs=fabrics.0_R_ClientDetails()

[SP_UPDATE]
SPs=fabrics.0_U_Client()

[SP_DELETE]
SPs=fabrics.0_D_Client()
```

The settings that follow do not include the settings mentioned above. Note that although the above settings are not included below, they were present in all configuration files of the pilot.

A.3.2.1 BeagleBone Black

```
[LOG]
DiagnosticLogging=false
LogDbLines=false

[MPDBSTATUS]
;Time in seconds to read status
TimerFrequency=2
RamUsage=true
CpuLoad=true
CpuTemperature=true

[MPDBCONNECTION]
Type=BBB
Host=192.168.1.170
Port=22
Username=bbb
Password=router127
RootPassword=router127
```

```
[DBCONNECTION]
Server=192.168.1.170
Database=fabrics
Username=mscuser
Password=router127
CommandTimeout=60
```

A.3.2.2 Intel Edison Arduino

```
[LOG]
DiagnosticLogging=false
LogDbLines=false

[MPDBSTATUS]
;Time in seconds to read status
TimerFrequency=2
RamUsage=true
CpuLoad=true
CpuTemperature=true

[MPDBCONNECTION]
Type=IEA
Host=192.168.1.160
Port=22
Username=edison
Password=router127
RootPassword=router127

[DBCONNECTION]
Server=192.168.1.160
Database=fabrics
Username=mscuser
Password=router127
CommandTimeout=60
```

A.3.2.3 Raspberry Pi 3

```
[LOG]
DiagnosticLogging=false
LogDbLines=false

[MPDBSTATUS]
;Time in seconds to read status
TimerFrequency=2
RamUsage=true
CpuLoad=true
```

```
CpuTemperature=true

[MPDBCONNECTION]
Type=RPi3
Host=192.168.1.150
Port=22
Username=pi
Password=router127
RootPassword=router127

[DBCONNECTION]
Server=192.168.1.150
Database=fabrics
Username=mscuser
Password=router127
CommandTimeout=60
```

A.3.2.4 Commodity personal computer

```
[LOG]
DiagnosticLogging=false
LogDbLines=false

[MPDBSTATUS]
;Time in seconds to read status
TimerFrequency=2
RamUsage=true
CpuLoad=true
CpuTemperature=false

[MPDBCONNECTION]
Type=PC
Host=192.168.1.180
Port=22
Username=pc
Password=router127
RootPassword=router127

[DBCONNECTION]
Server=192.168.1.180
Database=fabrics
Username=mscuser
Password=router127
CommandTimeout=60
```

APPENDIX B

B.1 Script files for importing table data

The import script for the pilot is shown in Section B.1.1 and is followed by the experiment import scripts in Section B.1.2. Note that columns not referenced in these scripts are populated from extracts of the imported data after this process, for example; the “Phase_Duration_Sec” column in the LOAD_FACT table are obtained from the “Log_Line” column.

B.1.1 Pilot

```
use [Data_Analysis]

IF OBJECT_ID('tempdb..#DirectoryTree') IS NOT NULL
      DROP TABLE #DirectoryTree;

CREATE TABLE #DirectoryTree (
       Id int IDENTITY(1,1)
      ,Subdirectory nvarchar(512)
      ,Depth int
      ,Isfile bit);

IF OBJECT_ID('dbo.xTempLogTable') IS NOT NULL
      DROP TABLE dbo.xTempLogTable;

CREATE TABLE dbo.xTempLogTable
(Time Time,
Thread INT,
Log_Line VARCHAR(2000))
GO


INSERT #DirectoryTree (Subdirectory,Depth,Isfile)
EXEC master.sys.xp_dirtree 'C:\LOG\LOG',0,1;

declare @fileName varchar(50) = ''
                ,@directory varchar(50) = ''
                ,@filenameID int = 0
                ,@path varchar(100) = ''
                ,@insertQuery varchar(1000) = ''
                ,@deviceID varchar(10) = ''
                ,@phase int = 0
                ,@indexOfP int = 0
                ,@indexOfDot int = 0
                ,@startDate date
                ,@startTime time
                ,@date date
                ,@time time
                ,@hour int
                ,@minute int
                ,@second int
                ,@millisecond int
                ,@timeID int
                ,@threadNo int
                ,@logLine varchar(2000)

while ((select COUNT(*) from #DirectoryTree) > 0)
begin

        --Log file selection
        set @directory = (select top 1 Subdirectory from #DirectoryTree where Isfile = 0 order by Id)
```

```
		set @fileName = (select top 1 Subdirectory from #DirectoryTree where Isfile = 1 order by Id)
		set @filenameID = (select top 1 Id from #DirectoryTree where Isfile = 1 order by Id)
		set @deviceID = (select Device_ID from dbo.Device_Dimension where Device = @directory)

		delete from #DirectoryTree where Id = @filenameID

		--Phase
		set @indexOfP = CHARINDEX('P', @fileName)
		set @indexOfDot = CHARINDEX('.', @fileName)
		set @phase = CAST((SUBSTRING(@fileName, @indexOfP + 1, (@indexOfDot - @indexOfP - 1))) as int)

		--Current file's path
		set @path = ('C:\LOG\LOG\' + @directory + '\' + @fileName)


		--Insert logs into a temporary table
		set @insertQuery = 'BULK INSERT dbo.xTempLogTable FROM ''' + @path + ''' WITH(FIELDTERMINATOR =
''';'',ROWTERMINATOR = ''\n'')'
		exec (@insertQuery)

		--Start date & time
		set @startDate = (SUBSTRING(@fileName,1,8))
		set @hour = (SUBSTRING(@fileName,10,2))
		set @minute = (SUBSTRING(@fileName,12,2))
		set @second = (SUBSTRING(@fileName,14,2))
		set @startTime = CAST(@hour as char(2)) + ':' + CAST(@minute as char(2)) + ':' + CAST(@second as
char(2))

		while ((select COUNT(*) from dbo.xTempLogTable) > 0)
		begin
			select top 1 @time = [Time], @threadNo = [Thread], @logLine = [Log_Line] from
dbo.xTempLogTable

			set @date = @startDate
			set @hour = DATEPART(HOUR, @time)
			set @minute = DATEPART(MINUTE, @time)
			set @second = DATEPART(SECOND, @time)
			set @millisecond = DATEPART(MILLISECOND, @time)

			--Create Time_Dimension record
			if ((select COUNT(*) from [dbo].[Time_Dimension] where [Date] = @date and [Hour] = @hour
and [Minute] = @minute and [Second] = @second) = 0)
			begin
				insert into [dbo].[Time_Dimension]
				([Date]
				,[Hour]
				,[Minute]
				,[Second])
				values
				(@date
				,@hour
				,@minute
				,@second)

				set @timeID = (select IDENT_CURRENT('[dbo].[Time_Dimension]'))
			end
			else
				set @timeID = (select top 1 [Time_ID] from [dbo].[Time_Dimension] where [Date] =
@date and [Hour] = @hour and [Minute] = @minute and [Second] = @second)

			if (@threadNo = 0)
			begin
				insert into [dbo].[Load_Fact]
					([Device_ID]
					,[Phase_ID]
					,[Time_ID]
					,[Millisecond]
					,[Log_Line])
				values
					(@deviceID
					,@phase
					,@timeID
					,@millisecond
					,@logLine)
```

```
                end
                else
                begin
                        insert into [dbo].[Logs_Fact]
                                   ([Device_ID]
                                        ,[Thread_ID]
                                        ,[Time_ID]
                                        ,[Millisecond]
                                        ,[Log_Line])
                        values
                                   (@deviceID
                                   ,@threadID
                                   ,@timeID
                                   ,@millisecond
                                   ,@logLine)
                end

                delete from dbo.xTempLogTable where [Time] = @time and [Log_Line] = @logLine and [Thread]
= @threadNo
        end
end
```

B.1.2 Experiment

The first script is run to import MCS log files:

```
use [Data_Analysis_2]

IF OBJECT_ID('tempdb..#DirectoryTree') IS NOT NULL
      DROP TABLE #DirectoryTree;

CREATE TABLE #DirectoryTree (
       Id int IDENTITY(1,1)
      ,Subdirectory nvarchar(512)
      ,Depth int
      ,Isfile bit);

IF OBJECT_ID('dbo.xTempLogTable') IS NOT NULL
      DROP TABLE dbo.xTempLogTable;

CREATE TABLE dbo.xTempLogTable
(Time Time,
Thread INT,
Log_Line VARCHAR(2000))
GO


INSERT #DirectoryTree (Subdirectory,Depth,Isfile)
EXEC master.sys.xp_dirtree 'C:\LOG\LOG',0,1;


declare @fileName varchar(50) = ''
                ,@directory varchar(50) = ''
                ,@filenameID int = 0
                ,@path varchar(100) = ''
                ,@insertQuery varchar(1000) = ''
                ,@deviceID varchar(10) = ''
                ,@iteration int = 0
                ,@phase int = 0
                ,@indexOfI int = 0
                ,@indexOfP int = 0
                ,@indexOfDot int = 0
                ,@startDate date
                ,@startTime time
                ,@date date
                ,@time time
                ,@hour int
```

```
		,@minute int
		,@second int
		,@millisecond int
		,@timeID int
		,@threadNo int
		,@threadID int
		,@logLine varchar(2000)

while ((select COUNT(*) from #DirectoryTree) > 0)
begin

	--Log file selection
	set @directory = (select top 1 Subdirectory from #DirectoryTree where Isfile = 0 order by Id)
	set @fileName = (select top 1 Subdirectory from #DirectoryTree where Isfile = 1 order by Id)
	set @filenameID = (select top 1 Id from #DirectoryTree where Isfile = 1 order by Id)
	set @deviceID = (select Device_ID from dbo.Device_Dimension where Device = @directory)

	delete from #DirectoryTree where Id = @filenameID

	--Phase
	set @indexOfP = CHARINDEX('P', @fileName)
	set @indexOfDot = CHARINDEX('.', @fileName)
	set @phase = CAST((SUBSTRING(@fileName, @indexOfP + 1, (@indexOfDot - @indexOfP - 1))) as int)

	--Iteration
	set @indexOfI = CHARINDEX('I', @fileName)
	set @iteration = CAST((SUBSTRING(@fileName, @indexOfI + 1, (@indexOfP - @indexOfI - 1))) as int)

	--Current file's path
	set @path = ('C:\LOG\LOG\' + @directory + '\' + @fileName)


	--Insert logs into a temporary table
	set @insertQuery = 'BULK INSERT dbo.xTempLogTable FROM ''' + @path + ''' WITH(FIELDTERMINATOR =
'';'',ROWTERMINATOR = ''\n'')'
	exec (@insertQuery)
	delete from dbo.xTempLogTable where [Thread] = 0

	--Start date & time
	set @startDate = (SUBSTRING(@fileName,1,8))
	set @hour = (SUBSTRING(@fileName,10,2))
	set @minute = (SUBSTRING(@fileName,12,2))
	set @second = (SUBSTRING(@fileName,14,2))
	set @startTime = CAST(@hour as char(2)) + ':' + CAST(@minute as char(2)) + ':' + CAST(@second as
char(2))

	while ((select COUNT(*) from dbo.xTempLogTable) > 0)
	begin
		select top 1 @time = [Time], @threadNo = [Thread], @logLine = [Log_Line] from
dbo.xTempLogTable
		delete from dbo.xTempLogTable where [Time] = @time and [Log_Line] = @logLine and [Thread]
= @threadNo

		set @date = @startDate
		set @hour = DATEPART(HOUR, @time)
		set @minute = DATEPART(MINUTE, @time)
		set @second = DATEPART(SECOND, @time)
		set @millisecond = DATEPART(MILLISECOND, @time)

		--Log file rolled over to next day
		if (@time < @startTime)
			set @date = DATEADD(DAY, 1, @date)

		--Create Time_Dimension record
		if ((select COUNT(*) from [dbo].[Time_Dimension] where [Date] = @date and [Hour] = @hour
and [Minute] = @minute and [Second] = @second) < 1)
		begin
			insert into [dbo].[Time_Dimension]
			([Date]
			,[Hour]
			,[Minute]
			,[Second])
			values
			(@date
```

```
                        ,@hour
                        ,@minute
                        ,@second)

                        set @timeID = (select IDENT_CURRENT('[dbo].[Time_Dimension]'))
                end
                else
                        select @timeID = [Time_ID] from [dbo].[Time_Dimension] where [Date] = @date and
[Hour] = @hour and [Minute] = @minute and [Second] = @second


                set @threadID = (select [Thread_ID] from [dbo].[Thread_Dimension] where [Iteration_ID] =
@iteration and [Phase_ID] = @phase and [Thread_No] = @threadNo)

                insert into [dbo].[Logs_Fact]
                                ([Device_ID]
                                      ,[Thread_ID]
                                      ,[Time_ID]
                                      ,[Millisecond]
                                      ,[Log_Line_Subs]
                                      ,[Log_Line])
                values
                                (@deviceID
                                ,@threadID
                                ,@timeID
                                ,@millisecond
                                ,SUBSTRING(@logLine, 1, 50)
                                ,@logLine)
        end

        if ((@phase = 20) or (@phase = 15 and @directory = 'BBB'))
                select @directory AS Device, @iteration AS Iteration

        if ((@phase = 20 and @iteration = 30) or (@phase = 15 and @iteration = 30 and @directory = 'BBB'))
        begin
                delete from #DirectoryTree where Subdirectory = @directory
                select @directory AS 'Completed Device'
        end
end


IF OBJECT_ID('tempdb..#DirectoryTree') IS NOT NULL
      DROP TABLE #DirectoryTree;

CREATE TABLE #DirectoryTree (
       Id int IDENTITY(1,1)
      ,Subdirectory nvarchar(512)
      ,Depth int
      ,Isfile bit);

IF OBJECT_ID('dbo.xTempLogTable') IS NOT NULL
      DROP TABLE dbo.xTempLogTable;

CREATE TABLE dbo.xTempLogTable
(Time Time,
Thread INT,
Log_Line VARCHAR(2000))
GO


INSERT #DirectoryTree (Subdirectory,Depth,Isfile)
EXEC master.sys.xp_dirtree 'C:\LOG\LOG',0,1;

declare @fileName varchar(50) = ''
                ,@directory varchar(50) = ''
                ,@filenameID int = 0
                ,@path varchar(100) = ''
                ,@insertQuery varchar(1000) = ''
                ,@deviceID varchar(10) = ''
                ,@iteration int = 0
                ,@phase int = 0
                ,@indexOfI int = 0
                ,@indexOfP int = 0
                ,@indexOfDot int = 0
```

```
		,@startDate date
		,@startTime time
		,@date date
		,@time time
		,@hour int
		,@minute int
		,@second int
		,@millisecond int
		,@timeID int
		,@threadNo int
		,@threadID int
		,@logLine varchar(2000)
		,@duplicateTotal int = 0
		,@duplicateCounter int = 0

while ((select COUNT(*) from #DirectoryTree) > 0)
begin

	--Log file selection
	set @directory = (select top 1 Subdirectory from #DirectoryTree where Isfile = 0 order by Id)
	set @fileName = (select top 1 Subdirectory from #DirectoryTree where Isfile = 1 order by Id)
	set @filenameID = (select top 1 Id from #DirectoryTree where Isfile = 1 order by Id)
	set @deviceID = (select Device_ID from dbo.Device_Dimension where Device = @directory)

	delete from #DirectoryTree where Id = @filenameID

	--Phase
	set @indexOfP = CHARINDEX('P', @fileName)
	set @indexOfDot = CHARINDEX('.', @fileName)
	set @phase = CAST((SUBSTRING(@fileName, @indexOfP + 1, (@indexOfDot - @indexOfP - 1))) as int)

	--Iteration
	set @indexOfI = CHARINDEX('I', @fileName)
	set @iteration = CAST((SUBSTRING(@fileName, @indexOfI + 1, (@indexOfP - @indexOfI - 1))) as int)

	--Current file's path
	set @path = ('C:\LOG\LOG\' + @directory + '\' + @fileName)


	--Insert logs into a temporary table
	set @insertQuery = 'BULK INSERT dbo.xTempLogTable FROM ''' + @path + ''' WITH(FIELDTERMINATOR =
'';'',ROWTERMINATOR = ''\n'')'
	exec (@insertQuery)
	delete from dbo.xTempLogTable where [Thread] != 0

	--Start date & time
	set @startDate = (SUBSTRING(@fileName,1,8))
	set @hour = (SUBSTRING(@fileName,10,2))
	set @minute = (SUBSTRING(@fileName,12,2))
	set @second = (SUBSTRING(@fileName,14,2))
	set @startTime = CAST(@hour as char(2)) + ':' + CAST(@minute as char(2)) + ':' + CAST(@second as
char(2))

	while ((select COUNT(*) from dbo.xTempLogTable) > 0)
	begin
		select top 1 @time = [Time], @threadNo = [Thread], @logLine = [Log_Line] from
dbo.xTempLogTable

		set @date = @startDate
		set @hour = DATEPART(HOUR, @time)
		set @minute = DATEPART(MINUTE, @time)
		set @second = DATEPART(SECOND, @time)
		set @millisecond = DATEPART(MILLISECOND, @time)

		--Log file rolled over to next day
		if (@time < @startTime)
			set @date = DATEADD(DAY, 1, @date)

		--Create Time_Dimension record
		if ((select COUNT(*) from [dbo].[Time_Dimension] where [Date] = @date and [Hour] = @hour
and [Minute] = @minute and [Second] = @second) = 0)
		begin
			insert into [dbo].[Time_Dimension]
			([Date]
```

```
                        ,[Hour]
                        ,[Minute]
                        ,[Second])
                        values
                        (@date
                        ,@hour
                        ,@minute
                        ,@second)

                        set @timeID = (select IDENT_CURRENT('[dbo].[Time_Dimension]'))
                end
                else
                        set @timeID = (select top 1 [Time_ID] from [dbo].[Time_Dimension] where [Date] =
@date and [Hour] = @hour and [Minute] = @minute and [Second] = @second)


                insert into [dbo].[Load_Fact]
                                ([Device_ID]
                                      ,[Iteration_ID]
                                      ,[Phase_ID]
                                      ,[Time_ID]
                                      ,[Millisecond]
                                      ,[Log_Line_Subs]
                                      ,[Log_Line])
                values
                                (@deviceID
                                ,@iteration
                                ,@phase
                                ,@timeID
                                ,@millisecond
                                ,SUBSTRING(@logLine, 1, 50)
                                ,@logLine)

                if ((select COUNT(*) from dbo.xTempLogTable where [Time] = @time and [Log_Line] =
@logLine and [Thread] = @threadNo) > 1)
                begin
                        select @duplicateTotal = COUNT(*) from dbo.xTempLogTable where [Time] = @time and
[Log_Line] = @logLine and [Thread] = @threadNo
                        set @duplicateCounter = 1

                        while (@duplicateCounter < @duplicateTotal)
                        begin
                                insert into [dbo].[Load_Fact]
                                            ([Device_ID]
                                                  ,[Iteration_ID]
                                                  ,[Phase_ID]
                                                  ,[Time_ID]
                                                  ,[Millisecond]
                                                  ,[Log_Line_Subs]
                                                  ,[Log_Line])
                                        values
                                            (@deviceID
                                            ,@iteration
                                            ,@phase
                                            ,@timeID
                                            ,@millisecond
                                            ,SUBSTRING(CONCAT(@logLine, '_', CAST(@duplicateCounter as
varchar(1))), 1, 50)
                                            ,@logLine)

                                        set @duplicateCounter = @duplicateCounter + 1
                        end
                end

                delete from dbo.xTempLogTable where [Time] = @time and [Log_Line] = @logLine and [Thread]
= @threadNo
        end

        if ((@phase = 20) or (@phase = 15 and @directory = 'BBB'))
                select @directory AS Device, @iteration AS Iteration

        if ((@phase = 20 and @iteration = 30) or (@phase = 15 and @iteration = 30 and @directory = 'BBB'))
        begin
                delete from #DirectoryTree where Subdirectory = @directory
```

```
                select @directory AS 'Completed Device'
        end
end
```

The second script is run to import Dstat log files:

```
use [Data_Analysis_2]

IF OBJECT_ID('tempdb..#DirectoryTree') IS NOT NULL
      DROP TABLE #DirectoryTree;

CREATE TABLE #DirectoryTree (
       Id int IDENTITY(1,1)
      ,Subdirectory nvarchar(512)
      ,Depth int
      ,Isfile bit);

IF OBJECT_ID('dbo.xTempLogTable') IS NOT NULL
      DROP TABLE dbo.xTempLogTable;

CREATE TABLE dbo.xTempLogTable
(Time varchar(20)
,CPU_usr float
,CPU_sys float
,CPU_idl float
,CPU_wai float
,CPU_hiq float
,CPU_siq float
,RAM_used float
,RAM_buff float
,RAM_cach float
,RAM_free float
,DISK_read float
,DISK_writ float
,LOAD_1m float
,LOAD_5m float
,LOAD_15m float)
GO


INSERT #DirectoryTree (Subdirectory,Depth,Isfile)
EXEC master.sys.xp_dirtree 'C:\LOG\STATUSLOG',0,1;


declare @fileName varchar(50) = ''
                ,@directory varchar(50) = ''
                ,@filenameID int = 0
                ,@path varchar(100) = ''
                ,@insertQuery varchar(1000) = ''
                ,@Device_ID varchar(10) = ''
                ,@Iteration_ID int = 0
                ,@Phase_ID int = 0
                ,@indexOfI int = 0
                ,@indexOfP int = 0
                ,@indexOfDot int = 0
                ,@startDate date
                ,@startTime time
                ,@recordDateTime varchar(20)
                ,@date date
                ,@time time
                ,@hour int = 0
                ,@minute int = 0
                ,@second int = 0
                ,@millisecond int = 0
                ,@Time_ID int = 0
                ,@threadNo int = 0
                ,@threadID int = 0
                ,@Log_Line varchar(2000) = ''
                ,@CPU_User float = 0.0
                ,@CPU_System float = 0.0
                ,@CPU_Idle float = 0.0
                ,@CPU_Wait float = 0.0
                ,@CPU_HwI float = 0.0
```

```
                ,@CPU_SwI float = 0.0
                ,@RAM_Use float = 0
                ,@RAM_Buffer float = 0
                ,@RAM_Cache float = 0
                ,@RAM_Free float = 0
                ,@RAM_Total float = 0
                ,@DISK_Read float = 0.0
                ,@DISK_Write float = 0.0
                ,@LOAD_1m float = 0.0
                ,@LOAD_5m float = 0.0
                ,@LOAD_15m float = 0.0

while ((select COUNT(*) from #DirectoryTree) > 0)
begin

        --Log file selection
        set @directory = (select top 1 Subdirectory from #DirectoryTree where Isfile = 0 order by Id)
        set @fileName = (select top 1 Subdirectory from #DirectoryTree where Isfile = 1 order by Id)
        set @filenameID = (select top 1 Id from #DirectoryTree where Isfile = 1 order by Id)
        set @Device_ID = (select Device_ID from dbo.Device_Dimension where Device = @directory)

        delete from #DirectoryTree where Id = @filenameID

        --Phase
        set @indexOfP = CHARINDEX('P', @fileName)
        set @indexOfDot = CHARINDEX('.', @fileName)
        set @Phase_ID = CAST((SUBSTRING(@fileName, @indexOfP + 1, (@indexOfDot - @indexOfP - 1))) as int)

        --Iteration
        set @indexOfI = CHARINDEX('I', @fileName)
        set @Iteration_ID = CAST((SUBSTRING(@fileName, @indexOfI + 1, (@indexOfP - @indexOfI - 1))) as
int)

        --Current file's path
        set @path = ('C:\LOG\STATUSLOG\' + @directory + '\' + @fileName)


        --Insert logs into a temporary table
        set @insertQuery = 'BULK INSERT dbo.xTempLogTable FROM ''' + @path + ''' WITH(FIELDTERMINATOR =
'';'',ROWTERMINATOR = ''\n'')'
        exec (@insertQuery)

        --Start date & time
        set @startDate = (SUBSTRING(@fileName,1,8))
        set @hour = (SUBSTRING(@fileName,10,2))
        set @minute = (SUBSTRING(@fileName,12,2))
        set @second = (SUBSTRING(@fileName,14,2))
        set @startTime = CAST(@hour as char(2)) + ':' + CAST(@minute as char(2)) + ':' + CAST(@second as
char(2))


        while ((select COUNT(*) from dbo.xTempLogTable) > 0)
        begin
                select top 1 @recordDateTime = [Time]
                                        ,@CPU_User = [CPU_usr]
                                        ,@CPU_System = [CPU_sys]
                                        ,@CPU_Idle = [CPU_idl]
                                        ,@CPU_Wait = [CPU_wai]
                                        ,@CPU_HwI = [CPU_hiq]
                                        ,@CPU_SwI = [CPU_siq]
                                        ,@RAM_Use = [RAM_used]
                                        ,@RAM_Buffer = [RAM_buff]
                                        ,@RAM_Cache = [RAM_cach]
                                        ,@RAM_Free = [RAM_free]
                                        ,@DISK_Read = [DISK_read]
                                        ,@DISK_Write = [DISK_writ]
                                        ,@LOAD_1m = [LOAD_1m]
                                        ,@LOAD_5m = [LOAD_5m]
                                        ,@LOAD_15m = [LOAD_15m]
                 from dbo.xTempLogTable

                delete from dbo.xTempLogTable where [Time] = @recordDateTime

                set @time = CAST(SUBSTRING(@recordDateTime,7,8) as time)
```

```
		set @date = @startDate
		set @hour = DATEPART(HOUR, @time)
		set @minute = DATEPART(MINUTE, @time)
		set @second = DATEPART(SECOND, @time)
		set @millisecond = DATEPART(MILLISECOND, @time)

		--Log file rolled over to next day
		if (@time < @startTime)
			set @date = DATEADD(DAY, 1, @date)

		--Create Time_Dimension record
		if ((select COUNT(*) from [dbo].[Time_Dimension] where [Date] = @date and [Hour] = @hour
and [Minute] = @minute and [Second] = @second) = 0)
		begin
			insert into [dbo].[Time_Dimension]
			([Date]
			,[Hour]
			,[Minute]
			,[Second])
			values
			(@date
			,@hour
			,@minute
			,@second)

			set @Time_ID = (select IDENT_CURRENT('[dbo].[Time_Dimension]'))
		end
		else
			set @Time_ID = (select top 1 [Time_ID] from [dbo].[Time_Dimension] where [Date] =
@date and [Hour] = @hour and [Minute] = @minute and [Second] = @second)



		insert into [dbo].[Load_Fact]
				([Device_ID]
				,[Iteration_ID]
				,[Phase_ID]
				,[Time_ID]
				,[Millisecond]
				,[Log_Line_Subs]
				,[CPU_User]
				,[CPU_System]
				,[CPU_Idle]
				,[CPU_HwI]
				,[CPU_SwI]
				,[RAM_Use]
				,[RAM_Buffer]
				,[RAM_Cache]
				,[RAM_Free]
				,[RAM_Total]
				,[DISK_Read]
				,[DISK_Write]
				,[LOAD_1m]
				,[LOAD_5m]
				,[LOAD_15m])
		values
				(@Device_ID
				,@Iteration_ID
				,@Phase_ID
				,@Time_ID
				,@Millisecond
				,'DSTAT Log'
				,@CPU_User
				,@CPU_System
				,@CPU_Idle
				,@CPU_HwI
				,@CPU_SwI
				,@RAM_Use
				,@RAM_Buffer
				,@RAM_Cache
				,@RAM_Free
				,@RAM_Total
				,@DISK_Read
```

```
                                        ,@DISK_Write
                                        ,@LOAD_1m
                                        ,@LOAD_5m
                                        ,@LOAD_15m)
        end

        if ((@Phase_ID = 20) or (@Phase_ID = 15 and @directory = 'BBB'))
                select @directory AS Device, @Iteration_ID AS Iteration

        if ((@Phase_ID = 20 and @Iteration_ID = 30) or (@Phase_ID = 15 and @Iteration_ID = 30 and
@directory = 'BBB'))
        begin
                delete from #DirectoryTree where Subdirectory = @directory
                select @directory AS 'Completed Device'
        end
end
```

(P*AGE INTENTIONALLY LEFT BLANK)*

APPENDIX C

C.1 Load metric ANOVA post hoc multiple-device comparisons complete tables

C.1.1 Phase 1 to 15

Table C-1: ANOVA post hoc multiple-device comparisons for Phases 1-15 using the load metric

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
1	BBB	IEA	0.498	0.016	0.000	0.456	0.541
		PC	0.870	0.017	0.000	0.826	0.914
		RPi3	0.835	0.016	0.000	0.792	0.877
	IEA	BBB	-0.498	0.016	0.000	-0.541	-0.456
		PC	0.372	0.017	0.000	0.328	0.415
		RPi3	0.336	0.016	0.000	0.294	0.379
	PC	BBB	-0.870	0.017	0.000	-0.914	-0.826
		IEA	-0.372	0.017	0.000	-0.415	-0.328
		RPi3	-0.035	0.017	0.155	-0.079	0.008
	RPi3	BBB	-0.835	0.016	0.000	-0.877	-0.792
		IEA	-0.336	0.016	0.000	-0.379	-0.294
		PC	0.035	0.017	0.155	-0.008	0.079
2	BBB	IEA	0.191	0.020	0.000	0.140	0.242
		PC	0.444	0.020	0.000	0.393	0.495
		RPi3	0.414	0.020	0.000	0.362	0.465
	IEA	BBB	-0.191	0.020	0.000	-0.242	-0.140
		PC	0.253	0.020	0.000	0.202	0.304
		RPi3	0.222	0.020	0.000	0.171	0.274
	PC	BBB	-0.444	0.020	0.000	-0.495	-0.393
		IEA	-0.253	0.020	0.000	-0.304	-0.202
		RPi3	-0.030	0.020	0.411	-0.082	0.021
	RPi3	BBB	-0.414	0.020	0.000	-0.465	-0.362
		IEA	-0.222	0.020	0.000	-0.274	-0.171
		PC	0.030	0.020	0.411	-0.021	0.082
3	BBB	IEA	0.379	0.032	0.000	0.295	0.463
		PC	0.690	0.032	0.000	0.606	0.774
		RPi3	0.590	0.032	0.000	0.506	0.674
	IEA	BBB	-0.379	0.032	0.000	-0.463	-0.295
		PC	0.311	0.032	0.000	0.227	0.395
		RPi3	0.211	0.032	0.000	0.127	0.295
	PC	BBB	-0.690	0.032	0.000	-0.774	-0.606
		IEA	-0.311	0.032	0.000	-0.395	-0.227
		RPi3	-0.100	0.032	0.013	-0.184	-0.016
	RPi3	BBB	-0.590	0.032	0.000	-0.674	-0.506
		IEA	-0.211	0.032	0.000	-0.295	-0.127
		PC	0.100	0.032	0.013	0.016	0.184
4	BBB	IEA	0.494	0.033	0.000	0.408	0.581
		PC	0.821	0.033	0.000	0.735	0.908
		RPi3	0.740	0.033	0.000	0.654	0.827
	IEA	BBB	-0.494	0.033	0.000	-0.581	-0.408
		PC	0.327	0.033	0.000	0.240	0.414
		RPi3	0.246	0.033	0.000	0.159	0.333
	PC	BBB	-0.821	0.033	0.000	-0.908	-0.735

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
		IEA	-0.327	0.033	0.000	-0.414	-0.240
		RPi3	-0.081	0.033	0.076	-0.168	0.006
	RPi3	BBB	-0.740	0.033	0.000	-0.827	-0.654
		IEA	-0.246	0.033	0.000	-0.333	-0.159
		PC	0.081	0.033	0.076	-0.006	0.168
5	BBB	IEA	7.761	0.131	0.000	7.420	8.102
		PC	10.667	0.131	0.000	10.327	11.008
		RPi3	9.890	0.131	0.000	9.549	10.231
	IEA	BBB	-7.761	0.131	0.000	-8.102	-7.420
		PC	2.906	0.131	0.000	2.565	3.247
		RPi3	2.129	0.131	0.000	1.788	2.470
	PC	BBB	-10.667	0.131	0.000	-11.008	-10.327
		IEA	-2.906	0.131	0.000	-3.247	-2.565
		RPi3	-0.777	0.131	0.000	-1.118	-0.437
	RPi3	BBB	-9.890	0.131	0.000	-10.231	-9.549
		IEA	-2.129	0.131	0.000	-2.470	-1.788
		PC	0.777	0.131	0.000	0.437	1.118
6	BBB	IEA	2.548	0.049	0.000	2.420	2.675
		PC	3.691	0.049	0.000	3.564	3.818
		RPi3	3.251	0.049	0.000	3.124	3.378
	IEA	BBB	-2.548	0.049	0.000	-2.675	-2.420
		PC	1.144	0.049	0.000	1.016	1.271
		RPi3	0.703	0.049	0.000	0.576	0.830
	PC	BBB	-3.691	0.049	0.000	-3.818	-3.564
		IEA	-1.144	0.049	0.000	-1.271	-1.016
		RPi3	-0.440	0.049	0.000	-0.568	-0.313
	RPi3	BBB	-3.251	0.049	0.000	-3.378	-3.124
		IEA	-0.703	0.049	0.000	-0.830	-0.576
		PC	0.440	0.049	0.000	0.313	0.568
7	BBB	IEA	0.757	0.023	0.000	0.695	0.818
		PC	1.238	0.023	0.000	1.177	1.299
		RPi3	1.066	0.023	0.000	1.005	1.127
	IEA	BBB	-0.757	0.023	0.000	-0.818	-0.695
		PC	0.482	0.023	0.000	0.420	0.543
		RPi3	0.310	0.023	0.000	0.248	0.371
	PC	BBB	-1.238	0.023	0.000	-1.299	-1.177
		IEA	-0.482	0.023	0.000	-0.543	-0.420
		RPi3	-0.172	0.023	0.000	-0.233	-0.111
	RPi3	BBB	-1.066	0.023	0.000	-1.127	-1.005
		IEA	-0.310	0.023	0.000	-0.371	-0.248
		PC	0.172	0.023	0.000	0.111	0.233
8	BBB	IEA	0.399	0.029	0.000	0.324	0.474
		PC	0.799	0.029	0.000	0.724	0.873
		RPi3	0.614	0.029	0.000	0.539	0.689
	IEA	BBB	-0.399	0.029	0.000	-0.474	-0.324
		PC	0.400	0.029	0.000	0.325	0.475
		RPi3	0.215	0.029	0.000	0.140	0.290
	PC	BBB	-0.799	0.029	0.000	-0.873	-0.724
		IEA	-0.400	0.029	0.000	-0.475	-0.325
		RPi3	-0.185	0.029	0.000	-0.259	-0.110
	RPi3	BBB	-0.614	0.029	0.000	-0.689	-0.539
		IEA	-0.215	0.029	0.000	-0.290	-0.140
		PC	0.185	0.029	0.000	0.110	0.259
9	BBB	IEA	0.558	0.038	0.000	0.458	0.658
		PC	1.051	0.038	0.000	0.951	1.151

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
		RPi3	0.744	0.038	0.000	0.643	0.844
	IEA	BBB	-0.558	0.038	0.000	-0.658	-0.458
		PC	0.493	0.038	0.000	0.393	0.593
		RPi3	0.186	0.038	0.000	0.086	0.286
	PC	BBB	-1.051	0.038	0.000	-1.151	-0.951
		IEA	-0.493	0.038	0.000	-0.593	-0.393
		RPi3	-0.307	0.038	0.000	-0.407	-0.207
	RPi3	BBB	-0.744	0.038	0.000	-0.844	-0.643
		IEA	-0.186	0.038	0.000	-0.286	-0.086
		PC	0.307	0.038	0.000	0.207	0.407
10	BBB	IEA	3.584	0.033	0.000	3.498	3.670
		PC	4.661	0.033	0.000	4.575	4.747
		RPi3	4.364	0.033	0.000	4.278	4.450
	IEA	BBB	-3.584	0.033	0.000	-3.670	-3.498
		PC	1.077	0.033	0.000	0.992	1.163
		RPi3	0.780	0.033	0.000	0.694	0.866
	PC	BBB	-4.661	0.033	0.000	-4.747	-4.575
		IEA	-1.077	0.033	0.000	-1.163	-0.992
		RPi3	-0.297	0.033	0.000	-0.383	-0.211
	RPi3	BBB	-4.364	0.033	0.000	-4.450	-4.278
		IEA	-0.780	0.033	0.000	-0.866	-0.694
		PC	0.297	0.033	0.000	0.211	0.383
11	BBB	IEA	14.477	0.157	0.000	14.067	14.887
		PC	28.710	0.157	0.000	28.300	29.120
		RPi3	23.377	0.157	0.000	22.967	23.787
	IEA	BBB	-14.477	0.157	0.000	-14.887	-14.067
		PC	14.233	0.157	0.000	13.823	14.643
		RPi3	8.900	0.157	0.000	8.490	9.310
	PC	BBB	-28.710	0.157	0.000	-29.120	-28.300
		IEA	-14.233	0.157	0.000	-14.643	-13.823
		RPi3	-5.333	0.157	0.000	-5.743	-4.923
	RPi3	BBB	-23.377	0.157	0.000	-23.787	-22.967
		IEA	-8.900	0.157	0.000	-9.310	-8.490
		PC	5.333	0.157	0.000	4.923	5.743
12	BBB	IEA	16.574	0.365	0.000	15.622	17.526
		PC	39.335	0.365	0.000	38.382	40.287
		RPi3	30.442	0.365	0.000	29.490	31.394
	IEA	BBB	-16.574	0.365	0.000	-17.526	-15.622
		PC	22.760	0.365	0.000	21.808	23.712
		RPi3	13.868	0.365	0.000	12.916	14.820
	PC	BBB	-39.335	0.365	0.000	-40.287	-38.382
		IEA	-22.760	0.365	0.000	-23.712	-21.808
		RPi3	-8.892	0.365	0.000	-9.845	-7.940
	RPi3	BBB	-30.442	0.365	0.000	-31.394	-29.490
		IEA	-13.868	0.365	0.000	-14.820	-12.916
		PC	8.892	0.365	0.000	7.940	9.845
13	BBB	IEA	1.828	0.100	0.000	1.568	2.088
		PC	4.979	0.100	0.000	4.719	5.239
		RPi3	2.912	0.100	0.000	2.652	3.172
	IEA	BBB	-1.828	0.100	0.000	-2.088	-1.568
		PC	3.150	0.100	0.000	2.890	3.410
		RPi3	1.084	0.100	0.000	0.824	1.344
	PC	BBB	-4.979	0.100	0.000	-5.239	-4.719
		IEA	-3.150	0.100	0.000	-3.410	-2.890
		RPi3	-2.067	0.100	0.000	-2.327	-1.807

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
	RPi3	BBB	-2.912	0.100	0.000	-3.172	-2.652
		IEA	-1.084	0.100	0.000	-1.344	-0.824
		PC	2.067	0.100	0.000	1.807	2.327
14	BBB	IEA	1.609	0.053	0.000	1.471	1.748
		PC	2.869	0.053	0.000	2.730	3.008
		RPi3	2.000	0.053	0.000	1.861	2.138
	IEA	BBB	-1.609	0.053	0.000	-1.748	-1.471
		PC	1.259	0.053	0.000	1.121	1.398
		RPi3	0.390	0.053	0.000	0.251	0.529
	PC	BBB	-2.869	0.053	0.000	-3.008	-2.730
		IEA	-1.259	0.053	0.000	-1.398	-1.121
		RPi3	-0.869	0.053	0.000	-1.008	-0.730
	RPi3	BBB	-2.000	0.053	0.000	-2.138	-1.861
		IEA	-0.390	0.053	0.000	-0.529	-0.251
		PC	0.869	0.053	0.000	0.730	1.008
15	BBB	IEA	27.060	1.191	0.000	23.956	30.163
		PC	38.223	1.191	0.000	35.120	41.327
		RPi3	32.625	1.191	0.000	29.521	35.728
	IEA	BBB	-27.060	1.191	0.000	-30.163	-23.956
		PC	11.163	1.191	0.000	8.060	14.267
		RPi3	5.565	1.191	0.000	2.462	8.668
	PC	BBB	-38.223	1.191	0.000	-41.327	-35.120
		IEA	-11.163	1.191	0.000	-14.267	-8.060
		RPi3	-5.599	1.191	0.000	-8.702	-2.495
	RPi3	BBB	-32.625	1.191	0.000	-35.728	-29.521
		IEA	-5.565	1.191	0.000	-8.668	-2.462
		PC	5.599	1.191	0.000	2.495	8.702

C.1.2 Phase 15 to 20

Table C-2: ANOVA post hoc multiple-device comparisons for Phases 15-20 using the load metric without BeagleBone Black

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
15	IEA	PC	11.163	0.194	0.000	10.700	11.627
		RPi3	5.565	0.194	0.000	5.102	6.028
	PC	IEA	-11.163	0.194	0.000	-11.627	-10.700
		RPi3	-5.599	0.194	0.000	-6.062	-5.135
	RPi3	IEA	-5.565	0.194	0.000	-6.028	-5.102
		PC	5.599	0.194	0.000	5.135	6.062
16	IEA	PC	14.468	0.100	0.000	14.229	14.707
		RPi3	10.490	0.100	0.000	10.251	10.729
	PC	IEA	-14.468	0.100	0.000	-14.707	-14.229
		RPi3	-3.978	0.100	0.000	-4.217	-3.739
	RPi3	IEA	-10.490	0.100	0.000	-10.729	-10.251
		PC	3.978	0.100	0.000	3.739	4.217
17	IEA	PC	9.150	0.193	0.000	8.691	9.609
		RPi3	4.327	0.193	0.000	3.868	4.786
	PC	IEA	-9.150	0.193	0.000	-9.609	-8.691

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
		RPi3	-4.823	0.193	0.000	-5.282	-4.364
	RPi3	IEA	-4.327	0.193	0.000	-4.786	-3.868
		PC	4.823	0.193	0.000	4.364	5.282
18	IEA	PC	17.049	0.195	0.000	16.583	17.514
		RPi3	10.727	0.195	0.000	10.262	11.193
	PC	IEA	-17.049	0.195	0.000	-17.514	-16.583
		RPi3	-6.321	0.195	0.000	-6.787	-5.855
	RPi3	IEA	-10.727	0.195	0.000	-11.193	-10.262
		PC	6.321	0.195	0.000	5.855	6.787
19	IEA	PC	7.446	0.122	0.000	7.156	7.735
		RPi3	4.732	0.122	0.000	4.443	5.022
	PC	IEA	-7.446	0.122	0.000	-7.735	-7.156
		RPi3	-2.713	0.122	0.000	-3.003	-2.423
	RPi3	IEA	-4.732	0.122	0.000	-5.022	-4.443
		PC	2.713	0.122	0.000	2.423	3.003
20	IEA	PC	30.961	0.196	0.000	30.493	31.430
		RPi3	20.621	0.196	0.000	20.152	21.089
	PC	IEA	-30.961	0.196	0.000	-31.430	-30.493
		RPi3	-10.341	0.196	0.000	-10.809	-9.872
	RPi3	IEA	-20.621	0.196	0.000	-21.089	-20.152
		PC	10.341	0.196	0.000	9.872	10.809

C.2 Duration metric ANOVA post hoc multiple-device comparisons complete tables

C.2.1 Phase 1 to 15

Table C-3: ANOVA post hoc multiple-device comparisons for Phases 1-15 using the duration metric

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
1	BBB	IEA	0.934	0.083	0.000	0.718	1.149
		PC	1.732	0.083	0.000	1.517	1.947
		RPi3	2.123	0.083	0.000	1.908	2.339
	IEA	BBB	-0.934	0.083	0.000	-1.149	-0.718
		PC	0.798	0.083	0.000	0.583	1.014
		RPi3	1.190	0.083	0.000	0.974	1.405
	PC	BBB	-1.732	0.083	0.000	-1.947	-1.517
		IEA	-0.798	0.083	0.000	-1.014	-0.583
		RPi3	0.391	0.083	0.000	0.176	0.607
	RPi3	BBB	-2.123	0.083	0.000	-2.339	-1.908
		IEA	-1.190	0.083	0.000	-1.405	-0.974
		PC	-0.391	0.083	0.000	-0.607	-0.176
2	BBB	IEA	2.609	0.300	0.000	1.826	3.392
		PC	7.330	0.300	0.000	6.547	8.113
		RPi3	8.144	0.300	0.000	7.361	8.927
	IEA	BBB	-2.609	0.300	0.000	-3.392	-1.826
		PC	4.721	0.300	0.000	3.938	5.504
		RPi3	5.535	0.300	0.000	4.752	6.318

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
	PC	BBB	-7.330	0.300	0.000	-8.113	-6.547
		IEA	-4.721	0.300	0.000	-5.504	-3.938
		RPi3	0.814	0.300	0.038	0.031	1.597
	RPi3	BBB	-8.144	0.300	0.000	-8.927	-7.361
		IEA	-5.535	0.300	0.000	-6.318	-4.752
		PC	-0.814	0.300	0.038	-1.597	-0.031
3	BBB	IEA	4.258	0.369	0.000	3.295	5.220
		PC	10.031	0.369	0.000	9.069	10.993
		RPi3	8.783	0.369	0.000	7.821	9.746
	IEA	BBB	-4.258	0.369	0.000	-5.220	-3.295
		PC	5.773	0.369	0.000	4.811	6.736
		RPi3	4.526	0.369	0.000	3.563	5.488
	PC	BBB	-10.031	0.369	0.000	-10.993	-9.069
		IEA	-5.773	0.369	0.000	-6.736	-4.811
		RPi3	-1.248	0.369	0.005	-2.210	-0.286
	RPi3	BBB	-8.783	0.369	0.000	-9.746	-7.821
		IEA	-4.526	0.369	0.000	-5.488	-3.563
		PC	1.248	0.369	0.005	0.286	2.210
4	BBB	IEA	3.997	0.573	0.000	2.505	5.489
		PC	12.389	0.573	0.000	10.896	13.881
		RPi3	14.138	0.573	0.000	12.645	15.630
	IEA	BBB	-3.997	0.573	0.000	-5.489	-2.505
		PC	8.392	0.573	0.000	6.899	9.884
		RPi3	10.141	0.573	0.000	8.648	11.633
	PC	BBB	-12.389	0.573	0.000	-13.881	-10.896
		IEA	-8.392	0.573	0.000	-9.884	-6.899
		RPi3	1.749	0.573	0.015	0.257	3.241
	RPi3	BBB	-14.138	0.573	0.000	-15.630	-12.645
		IEA	-10.141	0.573	0.000	-11.633	-8.648
		PC	-1.749	0.573	0.015	-3.241	-0.257
5	BBB	IEA	36.732	1.037	0.000	34.030	39.434
		PC	100.401	1.037	0.000	97.699	103.103
		RPi3	84.529	1.037	0.000	81.827	87.231
	IEA	BBB	-36.732	1.037	0.000	-39.434	-34.030
		PC	63.669	1.037	0.000	60.967	66.371
		RPi3	47.797	1.037	0.000	45.095	50.499
	PC	BBB	-100.401	1.037	0.000	-103.103	-97.699
		IEA	-63.669	1.037	0.000	-66.371	-60.967
		RPi3	-15.872	1.037	0.000	-18.574	-13.170
	RPi3	BBB	-84.529	1.037	0.000	-87.231	-81.827
		IEA	-47.797	1.037	0.000	-50.499	-45.095
		PC	15.872	1.037	0.000	13.170	18.574
6	BBB	IEA	2.132	0.423	0.000	1.029	3.235
		PC	7.333	0.423	0.000	6.229	8.436
		RPi3	7.792	0.423	0.000	6.689	8.895
	IEA	BBB	-2.132	0.423	0.000	-3.235	-1.029
		PC	5.201	0.423	0.000	4.098	6.304
		RPi3	5.660	0.423	0.000	4.557	6.763
	PC	BBB	-7.333	0.423	0.000	-8.436	-6.229
		IEA	-5.201	0.423	0.000	-6.304	-4.098
		RPi3	0.459	0.423	0.699	-0.644	1.563
	RPi3	BBB	-7.792	0.423	0.000	-8.895	-6.689
		IEA	-5.660	0.423	0.000	-6.763	-4.557
		PC	-0.459	0.423	0.699	-1.563	0.644
7	BBB	IEA	2.358	0.562	0.000	0.892	3.824

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
		PC	7.770	0.562	0.000	6.304	9.236
		RPi3	5.631	0.562	0.000	4.165	7.097
	IEA	BBB	-2.358	0.562	0.000	-3.824	-0.892
		PC	5.412	0.562	0.000	3.946	6.878
		RPi3	3.273	0.562	0.000	1.807	4.739
	PC	BBB	-7.770	0.562	0.000	-9.236	-6.304
		IEA	-5.412	0.562	0.000	-6.878	-3.946
		RPi3	-2.139	0.562	0.001	-3.605	-0.673
	RPi3	BBB	-5.631	0.562	0.000	-7.097	-4.165
		IEA	-3.273	0.562	0.000	-4.739	-1.807
		PC	2.139	0.562	0.001	0.673	3.605
8	BBB	IEA	2.604	1.267	0.174	-0.698	5.907
		PC	12.967	1.267	0.000	9.665	16.269
		RPi3	8.122	1.267	0.000	4.819	11.424
	IEA	BBB	-2.604	1.267	0.174	-5.907	0.698
		PC	10.363	1.267	0.000	7.060	13.665
		RPi3	5.517	1.267	0.000	2.215	8.820
	PC	BBB	-12.967	1.267	0.000	-16.269	-9.665
		IEA	-10.363	1.267	0.000	-13.665	-7.060
		RPi3	-4.845	1.267	0.001	-8.148	-1.543
	RPi3	BBB	-8.122	1.267	0.000	-11.424	-4.819
		IEA	-5.517	1.267	0.000	-8.820	-2.215
		PC	4.845	1.267	0.001	1.543	8.148
9	BBB	IEA	7.656	1.546	0.000	3.627	11.684
		PC	25.574	1.546	0.000	21.545	29.603
		RPi3	15.566	1.546	0.000	11.537	19.594
	IEA	BBB	-7.656	1.546	0.000	-11.684	-3.627
		PC	17.918	1.546	0.000	13.890	21.947
		RPi3	7.910	1.546	0.000	3.881	11.938
	PC	BBB	-25.574	1.546	0.000	-29.603	-21.545
		IEA	-17.918	1.546	0.000	-21.947	-13.890
		RPi3	-10.008	1.546	0.000	-14.037	-5.980
	RPi3	BBB	-15.566	1.546	0.000	-19.594	-11.537
		IEA	-7.910	1.546	0.000	-11.938	-3.881
		PC	10.008	1.546	0.000	5.980	14.037
10	BBB	IEA	72.672	1.750	0.000	68.110	77.234
		PC	101.891	1.750	0.000	97.329	106.453
		RPi3	95.079	1.750	0.000	90.517	99.641
	IEA	BBB	-72.672	1.750	0.000	-77.234	-68.110
		PC	29.219	1.750	0.000	24.657	33.781
		RPi3	22.406	1.750	0.000	17.844	26.968
	PC	BBB	-101.891	1.750	0.000	-106.453	-97.329
		IEA	-29.219	1.750	0.000	-33.781	-24.657
		RPi3	-6.813	1.750	0.001	-11.375	-2.251
	RPi3	BBB	-95.079	1.750	0.000	-99.641	-90.517
		IEA	-22.406	1.750	0.000	-26.968	-17.844
		PC	6.813	1.750	0.001	2.251	11.375
11	BBB	IEA	15.191	1.043	0.000	12.473	17.909
		PC	160.991	1.043	0.000	158.272	163.709
		RPi3	76.855	1.043	0.000	74.137	79.573
	IEA	BBB	-15.191	1.043	0.000	-17.909	-12.473
		PC	145.800	1.043	0.000	143.082	148.518
		RPi3	61.665	1.043	0.000	58.946	64.383
	PC	BBB	-160.991	1.043	0.000	-163.709	-158.272
		IEA	-145.800	1.043	0.000	-148.518	-143.082

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
		RPi3	-84.135	1.043	0.000	-86.853	-81.417
	RPi3	BBB	-76.855	1.043	0.000	-79.573	-74.137
		IEA	-61.665	1.043	0.000	-64.383	-58.946
		PC	84.135	1.043	0.000	81.417	86.853
12	BBB	IEA	15.326	3.411	0.000	6.434	24.218
		PC	187.339	3.411	0.000	178.447	196.231
		RPi3	55.912	3.411	0.000	47.020	64.804
	IEA	BBB	-15.326	3.411	0.000	-24.218	-6.434
		PC	172.013	3.411	0.000	163.121	180.905
		RPi3	40.585	3.411	0.000	31.693	49.477
	PC	BBB	-187.339	3.411	0.000	-196.231	-178.447
		IEA	-172.013	3.411	0.000	-180.905	-163.121
		RPi3	-131.428	3.411	0.000	-140.320	-122.536
	RPi3	BBB	-55.912	3.411	0.000	-64.804	-47.020
		IEA	-40.585	3.411	0.000	-49.477	-31.693
		PC	131.428	3.411	0.000	122.536	140.320
13	BBB	IEA	35.217	3.144	0.000	27.021	43.413
		PC	129.946	3.144	0.000	121.750	138.142
		RPi3	129.969	3.144	0.000	121.773	138.165
	IEA	BBB	-35.217	3.144	0.000	-43.413	-27.021
		PC	94.729	3.144	0.000	86.534	102.925
		RPi3	94.752	3.144	0.000	86.556	102.948
	PC	BBB	-129.946	3.144	0.000	-138.142	-121.750
		IEA	-94.729	3.144	0.000	-102.925	-86.534
		RPi3	0.023	3.144	1.000	-8.173	8.218
	RPi3	BBB	-129.969	3.144	0.000	-138.165	-121.773
		IEA	-94.752	3.144	0.000	-102.948	-86.556
		PC	-0.023	3.144	1.000	-8.218	8.173
14	BBB	IEA	68.327	5.502	0.000	53.985	82.668
		PC	252.300	5.502	0.000	237.958	266.641
		RPi3	214.829	5.502	0.000	200.487	229.171
	IEA	BBB	-68.327	5.502	0.000	-82.668	-53.985
		PC	183.973	5.502	0.000	169.631	198.314
		RPi3	146.502	5.502	0.000	132.161	160.844
	PC	BBB	-252.300	5.502	0.000	-266.641	-237.958
		IEA	-183.973	5.502	0.000	-198.314	-169.631
		RPi3	-37.471	5.502	0.000	-51.812	-23.129
	RPi3	BBB	-214.829	5.502	0.000	-229.171	-200.487
		IEA	-146.502	5.502	0.000	-160.844	-132.161
		PC	37.471	5.502	0.000	23.129	51.812
15	BBB	IEA	990.757	675.998	0.462	-771.345	2752.858
		PC	1162.492	675.998	0.318	-599.610	2924.593
		RPi3	998.615	675.998	0.455	-763.487	2760.716
	IEA	BBB	-990.757	675.998	0.462	-2752.858	771.345
		PC	171.735	675.998	0.994	-1590.366	1933.837
		RPi3	7.858	675.998	1.000	-1754.243	1769.960
	PC	BBB	-1162.492	675.998	0.318	-2924.593	599.610
		IEA	-171.735	675.998	0.994	-1933.837	1590.366
		RPi3	-163.877	675.998	0.995	-1925.979	1598.224
	RPi3	BBB	-998.615	675.998	0.455	-2760.716	763.487
		IEA	-7.858	675.998	1.000	-1769.960	1754.243
		PC	163.877	675.998	0.995	-1598.224	1925.979

C.2.2 Phase 15 to 20

Table C-4: ANOVA post hoc multiple-device comparisons for Phases 15-20 using the duration metric without BeagleBone Black

Phase	(I) DeviceCode	(J) DeviceCode	Mean Difference (I-J)	Std. Error	Sig. (p)	95% Confidence Interval	
						Lower Bound	Upper Bound
15	IEA	PC	171.735	1.726	0.000	167.620	175.850
		RPi3	7.858	1.726	0.000	3.743	11.973
	PC	IEA	-171.735	1.726	0.000	-175.850	-167.620
		RPi3	-163.877	1.726	0.000	-167.992	-159.762
	RPi3	IEA	-7.858	1.726	0.000	-11.973	-3.743
		PC	163.877	1.726	0.000	159.762	167.992
16	IEA	PC	259.353	1.448	0.000	255.899	262.806
		RPi3	150.981	1.448	0.000	147.527	154.435
	PC	IEA	-259.353	1.448	0.000	-262.806	-255.899
		RPi3	-108.372	1.448	0.000	-111.825	-104.918
	RPi3	IEA	-150.981	1.448	0.000	-154.435	-147.527
		PC	108.372	1.448	0.000	104.918	111.825
17	IEA	PC	287.847	3.344	0.000	279.873	295.821
		RPi3	166.589	3.344	0.000	158.615	174.563
	PC	IEA	-287.847	3.344	0.000	-295.821	-279.873
		RPi3	-121.258	3.344	0.000	-129.232	-113.284
	RPi3	IEA	-166.589	3.344	0.000	-174.563	-158.615
		PC	121.258	3.344	0.000	113.284	129.232
18	IEA	PC	254.401	3.060	0.000	247.104	261.698
		RPi3	47.102	3.060	0.000	39.805	54.399
	PC	IEA	-254.401	3.060	0.000	-261.698	-247.104
		RPi3	-207.299	3.060	0.000	-214.596	-200.002
	RPi3	IEA	-47.102	3.060	0.000	-54.399	-39.805
		PC	207.299	3.060	0.000	200.002	214.596
19	IEA	PC	239.194	3.746	0.000	230.261	248.126
		RPi3	70.778	3.746	0.000	61.845	79.710
	PC	IEA	-239.194	3.746	0.000	-248.126	-230.261
		RPi3	-168.416	3.746	0.000	-177.349	-159.484
	RPi3	IEA	-70.778	3.746	0.000	-79.710	-61.845
		PC	168.416	3.746	0.000	159.484	177.349
20	IEA	PC	362.257	1.871	0.000	357.796	366.719
		RPi3	15.231	1.871	0.000	10.770	19.693
	PC	IEA	-362.257	1.871	0.000	-366.719	-357.796
		RPi3	-347.026	1.871	0.000	-351.488	-342.564
	RPi3	IEA	-15.231	1.871	0.000	-19.693	-10.770
		PC	347.026	1.871	0.000	342.564	351.488

(PAGE INTENTIONALLY LEFT BLANK)

APPENDIX D

Refer to Appendix D on the CD provided with this document.

(PAGE INTENTIONALLY LEFT BLANK)

APPENDIX E

Refer to Appendix E on the CD provided with this document.

(PAGE INTENTIONALLY LEFT BLANK)

APPENDIX F

Refer to Appendix F on the CD provided with this document.

(PAGE INTENTIONALLY LEFT BLANK)

APPENDIX G

Refer to Appendix G on the CD provided with this document.