2019

A Monte Carlo tree search algorithm for optimization of load scalability in database systems

Allan Odhiambo Omondi
Faculty of Information Technology
Strathmore University

A Monte Carlo Tree Search Algorithm for Optimization of Load Scalability in Database Systems

Allan Odhiambo Omondi

Thesis submitted in partial fulfilment of the requirements for the Degree of Doctor of Philosophy in Information Technology at Strathmore University

Faculty of Information Technology
Strathmore University
Nairobi, Kenya

June, 2019

Declaration and Approval

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

……[*Name of Candidate*]

……[*Signature*]

……[*Date*]

Approval

The thesis of Allan Odhiambo Omondi was reviewed and approved by the following:

Prof. Ismail Ateya Lukandu,
Faculty of Information Technology,
Strathmore University

Prof. Gregory Wabuke Wanyembi,
School of Computing and Informatics,
Mount Kenya University

Dr. Joseph Orero,
Dean, Faculty of Information Technology

Dean, School of Graduate Studies

Abstract

Variable environmental conditions and runtime phenomena require developers of complex business information systems to expose configuration parameters to system administrators. This allows system administrators to intervene by tuning the bottleneck configuration parameters in response to current changes or in anticipation of future changes in order to maintain the system's performance at an optimum level. However, these manual performance tuning interventions are prone to error and lack of standards due to varying levels of expertise and over-reliance on inaccurate predictions of future states of a business information system. The purpose of this research was therefore to investigate on how to design an algorithm that proactively reconfigures bottleneck parameters without over-relying on an accurate model of a stochastic environment. This was done using a comparative experimental research design that involved quantitative data collection through simulations of different algorithm variants. The research built on the theoretical concepts of control theory and decision theory, coupled with the estimation of unknown quantities using principles of simulation-based inferential statistics. Subsequently, Monte Carlo Tree Search, with a variant of the selection stage, was used as the foundation of the designed algorithm. The selection stage was variated by applying a "lean Last Good Reply with Forgetting" (lean-LGRF) strategy and first tested in the context of a strategy board game, Reversi. The lean-LGRF selection strategy applied over 1,000 playouts against the baseline Upper Confidence Bound applied to Trees (UCT) selection strategy recorded the highest number of wins. On the other hand, the Progressive Bias selection strategy had a win-rate of 45.8% against the UCT selection strategy. Lastly, as expected, the UCT selection strategy had a win-rate of 49.7% (an almost 50-50 win-rate) against itself. The results were then subjected to a Chi-square (χ^2) test which provided evidence that the variation technique applied in the selection stage of the algorithm had a significantly positive impact on its performance. The superior selection variant was then applied in the context of a distributed database system. This also provided compelling results that indicate that applying the algorithm in a distributed database system resulted in a response-time latency that was 27% lower than the average response-time latency and a transaction throughput that was 17% higher than the average transaction throughput.

Keywords: *Database Theory, Performance Tuning, Decision Theory, Monte Carlo Tree Search, Autonomic Computing.*

Table of Contents

Acknowledgements

My foremost gratitude is to God Almighty.

To the scholars who paved the path before me, upon whose shoulders I stood and continue to stand on. Thank you.

I am indebted and most grateful to my research supervisors, Prof. Ismail Ateya and Prof. Gregory Wanyembi, for their time, sage advice, unwavering support, mentorship, and valuable and constructive feedback. I can never express enough appreciation and respect to my supervisors.

I extend my gratitude to my friends and esteemed Faculty of Information Technology colleagues at Strathmore University for their collegial guidance and support. I extend a special and deep gratitude to the Strathmore University Doctoral Academy, for its amazing support in workload regulation, training, and encouragement throughout this journey.

To all the obstacles, chaos, hard-times, distractions, challenges and stress that I encountered at various points in the journey; for the transformative effect they had in moulding my personality to be even more determined, passionate, industrious, and focused.

Last but not least, my sincere gratitude to my parents, Mr. Romulus Omondi Odhiambo and Dr. Everlyn Atieno Omondi; and my siblings, Mrs. Cynthia Achieng Agak and Miss Paulette Akello Omondi, for their patience and endless support for almost 3 decades. A special thank you to my beloved wife, Dr. Tabitha Adhiambo Odero, for being my accountability partner, for her encouragement and support, and for her understanding during the numerous and regular sessions dedicated to working on the PhD research on a daily basis.

Dedication

I dedicate this work to the strengthening of the technological foundations for a knowledge economy.

List of Figures

List of Tables

List of Equations

List of Models

List of Algorithms

Abbreviations/Acronyms

AI	–	Artificial Intelligence
CBM	–	Condition-Based Maintenance
CobRA	–	Control-Based Requirements-Oriented Adaptation
DIKAR	–	Data-Information-Knowledge-Action-Reward
DMAIC	–	Define-Measure-Analyse-Improve-Control
EDW	–	Enterprise Data Warehouse
GDP	–	Gross Domestic Product
IT	–	Information Technology
IBM	–	International Business Machines Corporation
MAPE	–	Monitor-Analyse-Plan-Execute
MDP	–	Markov Decision Process
MRP	–	Markov Reward Process
MTTF	–	Mean-Time-To-Failure
ODE	–	Ordinary Differential Equation
OLAP	–	Online Analytical Processing
OLTP	–	Online Transaction Processing
PLA	–	Proactive Latency-Aware Adaptation
QoS	–	Quality of Service
QEP	–	Query Execution Plan
rpm	–	Revolutions per Minute
RUL	–	Remaining Useful Life
SyRS	–	System Requirements Specification

Operational Definition of Terms

Adaptation latency – The time it takes for an adaptation (a change in system configurations) to become effective in a system (Moreno, Papadopoulos, Angelopoulos, Cámara & Schmerl, 2017).

Adaptive algorithm – An algorithm that changes its behaviour based on information available at the time it is run (Su, Chen, Feng, Rosenblum & Thiagaranj, 2016).

Algorithm – A set of unambiguously defined, finite instructions that if executed in the correct order (based on predefined controls that define how it should be executed) and in a finite amount of time will computationally process 0 or more input (instructions and/or data) to produce 1 or more desired outcomes (logic that defines what should be done) based on specific assumptions (Kitchin, 2017).

Autonomic computers – Computing systems that can manage themselves given high-level objectives from administrators (Kephart & Chess, 2003).

Data warehouse – A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data, techniques, and technologies that together provide a systematic and pragmatic approach to solve the end user problem of accessing information that is distributed in different systems inside an organization (Dewan, Aggarwal & Tanwar, 2013).

Load scalability – While a variety of definitions of the term load scalability have been suggested, this research will use the definition as "the ability of a system to expand/contract its resource pool to handle increases in workloads without negatively impacting on its responsiveness to execute any action within a given time interval." (Shahapure & Jayarekha, 2014)

Chapter 1: Introduction

1.1 Background

Automation forms a critical foundation required to make progress for the evolution of human beings. It enables human beings to free their minds from mundane tasks in order to concentrate on previously unachievable tasks. As research by Autor (2015) indicated, automation essentially saves on human resource as it reduces labour requirements per unit of output produced. However, recent studies have established that a directly proportional relationship can be observed between the automation of previously unachievable tasks and the level of complexity of the autonomic system as a whole. If not done correctly, the level of complexity ends up being an unavoidable by-product which leads to systems that are increasingly difficult to manage. A study by Li et al. (2017) cited the growing complexity of Information Technology (IT) infrastructure as a threat to the very benefits that IT has to offer. The study further pointed out that with the current rates of technological progress, there will not be enough IT personnel who have the skills needed to keep the world's computer-based business information systems running. This poses a hindrance in embracing new technologies required for growth.

A number of authors have been inspired by the homeostasis of the human body, whereby an equilibrium is established such that the human mind is freed to think about what should be done physically as opposed to the internal details of how to do it (Kephart & Chess, 2003; Li et al., 2017; Moreno et al., 2017). The demand for systems that can manage themselves given high-level objectives from system administrators has subsequently led to the advent of autonomic computing. While a variety of definitions of the term autonomic computing have been suggested, this research uses the definition first suggested by its original proponents, Kephart and Chess (2003), who defined it as systems that are expected to automatically adapt by reacting to variable environmental conditions and runtime phenomena. This can be achieved through adjusting their configuration parameters either reactively or proactively.

A study by Moreno et al. (2017) pointed out that reactive autonomic computing systems are appropriate in situations where the time it takes for an adaptation to become effective in the system, that is, the adaptation latency, is low. However, the same is not true for systems which have a high adaptation latency. The study further indicated that systems that have a high adaptation latency can end up in a situation where the effects of adjusting

configurations in response would be felt after the conditions that warranted the adjustments in the first place are no longer present. A possible workaround to this situation is anticipating conditions that warrant a change in configurations and proactively beginning to act gradually ahead of time before all the conditions are met (Bousdekis, Magoutas, Apostolou & Mentzas, 2015). There is potential to implement this through Model Predictive Control (MPC) which is an approach to control theory that makes use of models of the system to anticipate future behaviour.

One possible technique that can be used to apply MPC is called Proactive Latency-Aware Adaptation (PLA). PLA relies on stochastic analysis to create an explicit representation of the predicted environment behaviour. The merit of this technique lies in its ability to enable a system to proactively adapt to variable environmental conditions while simultaneously taking into consideration the adaptation latency (the time it takes for the adaptation to take effect) (Moreno et al., 2017). However, in the case of a system which experiences variable environmental conditions and runtime phenomena, creation of an explicit (detailed and complete) representation to predict environment behaviour would lead to a weak and inaccurate decision-making foundation. This is because of the inaccuracy involved in predicting based on a large prediction horizon. An alternative to the PLA technique is the Control-Based Requirements-Oriented Adaptation (CobRA) technique. Unlike PLA, CobRA makes use of an optimal estimation algorithm to estimate an implicit near-future system state when it cannot be measured directly (Cámara, Moreno & Garlan, 2014). This estimation enables it to function even in the presence of noise or incomplete observations.

Variables such as work metrics (transaction throughput or response-time latency), resource metrics (resource availability, utilization, saturation, or resource-related error-rates) and in some cases, event metrics, can be used to create an implicit representation of the system state when applying the CobRA or PLA techniques. This can be done using Ordinary Differential Equations (ODEs), Kalman filters, or even linear regression (Ranadip, 2018). Herein lies an opportunity to apply theoretical concepts of MPC for self-optimization to achieve load scalability. It is interesting to note that reinforcement learning concepts such as the use of state-action-reward logs have the potential to be applied at this point in such a way that over-reliance on prediction of the far distant, future is reduced (Duan, Chen, Houthooft, Schulman & Abbeel, 2016). This research provided an

investigation of these opportunities.

1.2 Problem Statement

As the MPC approach to control theory correctly states, it is useful to anticipate the future state of a system and to proactively adapt its configurations as the workload or other variable environmental conditions and runtime phenomena change (Moreno et al., 2017). However, several studies acknowledge difficulty in obtaining an accurate prediction of the future required to construct a profile of the system (Cheng & Garlan, 2012; Jimoh & McCluskey, 2016; Kim et al., 2016; Taleb & Blyth, 2011). This subsequently leads to the problem that the research focused on, which is the fact that load scalability in computer-based information systems is still heavily reliant on manual performance tuning interventions from system administrators (Kephart & Chess, 2003; Li et al., 2017). These manual performance tuning interventions are in turn based on inaccurate prediction of long-term future states which do not support the use of a system profile to guide optimization efforts. The manual performance tuning interventions also have a significant contribution towards substandard load scalability due to factors such as fatigue, long reaction times and inconsistent expertise amongst system administrators (Li et al., 2017).

1.3 Aim

To design an adaptive algorithm that proactively reconfigures bottleneck parameters without over-relying on an accurate, long-term model of a stochastic environment. This was done in order to achieve self-optimization required for load scalability in database systems.

1.4 Research Objectives

(i) To identify the parameters that need to be considered when performing configuration optimization in storage servers

(ii) To identify optimization techniques that can be used to achieve load scalability in servers

(iii) To review selection variants that can improve the performance of a Monte Carlo Tree Search algorithm

(iv) To design a latency-aware algorithm that proactively reconfigures bottleneck

parameters without over-relying on an accurate model of an unpredictable stochastic environment

(v) To perform experiments to test the designed algorithm using a game-theory based simulation of a strategy board-game

(vi) To apply the designed algorithm in the context of a distributed database system using a simulation of business-oriented ad-hoc queries on a test-bed

1.5 Research Questions

(i) What are the parameters that need to be considered when performing configuration optimization in storage servers?

(ii) Which optimization techniques can be used to achieve load scalability in servers?

(iii) How can the selection stage of the Monte Carlo Tree Search algorithm be variated to improve the overall performance?

(iv) How can a latency-aware algorithm be designed to proactively reconfigure bottleneck parameters without over-relying on an accurate model of an unpredictable stochastic environment?

(v) How can experiments be performed to test the designed algorithm using a game-theory based simulation of a strategy board-game?

(vi) How can the designed algorithm be applied in the context of a distributed database system using a simulation of business-oriented ad-hoc queries on a test bed?

1.6 Justification and Significance

Developing economies globally aim to transition into knowledge economies (ICTA, 2016). This means an economy in which growth is dependent on the quantity, quality, and accessibility of information available to be used for innovation, as opposed to dependency on traditional factors of production defined in classical economics as land, labour, capital and entrepreneurship (MoICT, 2014). Given the importance of access to quality information in the right quantity, the outcome of this research was expected to contribute towards the greater societal goal by laying the necessary technological foundations required for a knowledge economy. The potential in the research output was implemented

by ensuring that the effect of an increased workload did not drastically affect the ability of a database system to execute any action that pertains to access or manipulation of data or information within a given time interval.

1.6.1 Significance to IT Practitioners

Continuous impact-driven research in the field of database theory underpins fundamental advances in numerous database application programs in the society. For example, high performance criminal records databases to aid in policing, supporting accessibility to textual academic research in repositories, better provision of health through efficient storage and access of health records in a database, analysis of historical telecommunications data for fraud detection, amongst many other civilian and defence applications. Online Analytical Processing (OLAP) business information systems found in the areas of financial modelling (budgeting and planning), sales forecasting, exception reporting, variance analysis, promotion planning, and market share analysis also stand a chance of directly benefiting from the results of this research.

A key point arises that the separation of interface and implementation makes it difficult to know all possible applications of an algorithm. This is because numerous different interfaces can be developed to access the implementation of the designed algorithm. In spite of this fact, the running theme common across all possible applications of the algorithm designed in this research is the role it plays in promoting access to information by implementing principles of self-optimization to support system administrators.

1.6.2 Significance to Researchers

The amalgamation of control theory, decision theory and the estimation of unknown quantities using principles of simulation-based inferential statistics formed a key contribution of literature to academia. The amalgamation of these theoretical concepts was used as a framework to create a proposition which was subsequently confronted with data observed from experiments. The tested proposition, the procedure of conducting the experiments and the actual results of the experiments form a foundation for reapplication and further scrutiny by researchers in academia.

1.6.3 Significance to Policy-Makers

Policy-makers require evidence to make decisions on which policies to formulate, review, or abolish. The link between research and public policy can be strengthened by making academic research physically, intellectually, and socially accessible. When this link is strong, policy-makers combine their understanding of the communities they are in direct contact with with evidence that is generated by researchers. This leads to evidence-informed policy. Policy-makers can use the results of this research to make policies on performance tuning best-practices in the IT industry and also on the best practices or standards of interaction between system developers and system administrators. An example of a potential application area is the Government of Kenya's ICT standard on electronic data and records management. This subsequently has the potential to positively impact the acquisition of quality data in the right quantity.

1.7 Assumptions

(i) The main assumption made in this research was that the quantity of data stored in the database system is continuously increasing and as the quantity of data increases, the workload demands also increase. As a study by Chaudhuri and Narasayya (2007) indicates, this warrants the need for load scalability because an increase in workload demands has a significant impact on a system's performance. The designed algorithm, will therefore take this into consideration as it is executed.

(ii) Factors such as fatigue, long reaction times, and inconsistent expertise amongst system administrators have a significant impact on substandard performance-based scalability (Van Aken, Pavlo, Gordon & Zhang, 2017).

(iii) The tools to be used to appraise optimization techniques and Monte Carlo Tree Search (MCTS) algorithm variants are unbiased

(iv) The tools used to design the algorithm and test it through various experiments are unbiased

(v) The objectivity of the experiment results support the decision rule that was applied.

(vi) The architecture of the experiment test-bed will form a foundation for related

algorithms that intend to achieve a similar aim of self-optimization

1.8 Scope and Limitations

1.8.1 Scope

Artificial Intelligence (AI) can be used to create an autonomic agent that can make procedural decisions on its own based on a consideration of the cost and value of applying specific configurations. This research applied principles of Reinforcement Learning as a way of implementing the autonomic approach to vertical partitioning. Other branches of AI-based autonomic agents such as supervised learning and unsupervised learning were not considered.

There are numerous enterprise-grade Database Management Systems that can be used. For example, Oracle Database Management System, Microsoft SQL Server, IBM DB2, SAP Adaptive Server Enterprise (ASE), PostgreSQL, MariaDB, and MySQL DBMS. This research focused on MariaDB and went a step further to implement it as a MariaDB Galera synchronous multi-master distributed database made up of a cluster of 3 nodes. This was based on a relational data model as the context for applying the designed algorithm.

1.8.2 Limitations

Firstly, accessibility of real, confidential data to conduct experiments was a limitation. This is a limitation also acknowledged by several studies (Kim et al., 2016; Van Aken et al., 2017). However, to overcome this limitation, the research used a data generation tool to create the data. That is, the American National Standards Institute (ANSI) Structured Query Language (SQL) Standard Scalable and Portable (AS^3AP) Benchmark that was implemented using Benchmark Factory version 8.1. This was capable of simulating a real-world environment where the data grows exponentially and was then used to design the desired algorithm.

Secondly, there were limitations that were purely based on the fact that the research involved the appraisal of existing algorithms. This is a limitation also acknowledged by recent studies (Seaver, 2013). For example, the fact that many landmark algorithms are not open to scrutiny because their source code is not available to the public. However, the countermeasure applied was to appraise equivalent open-source algorithms in such cases.

Thirdly, heterogeneity of algorithms was a limitation with regard to appraisal and

design of algorithms. Even though it can be argued that code is understandable by nature, a study by Seaver (2013) was of a contrary opinion. The study argued that algorithms are essentially assemblages made up of hundreds of other algorithms woven together to create an algorithmic system. The pertinent issue is therefore that the algorithms are designed by different teams, each with different approaches. Therefore, according to the study by Seaver (2013), appraising an algorithm pertains understanding an algorithmic system made up of hundreds of other algorithms. This increases the complexity involved. A possible antidote to this, as a study by Bucher (2012) pointed out, is that rather than appraising the algorithmic system as a whole, only relevant sections that provide useful answers to the research question need to be appraised.

Fourthly, the advent of machine learning was a limitation encountered in this research and also acknowledged as a limitation by recent studies (Kitchin, 2017). Adaptive algorithms re-write themselves independent of their creators. This is because of its ability to randomly learn based on empirical runtime observations and rewards. Appraising such algorithms thus becomes a challenge because the instance of the algorithm during the point of appraisal may not be the same as the instance of the algorithm after its adaptation based on what it has learnt. A possible antidote to this is to appraise adaptive algorithms across multiple mutations.

Fifthly, predicting the behaviour of an algorithm is not a straightforward task. An algorithm is capable of producing different results given the same input. This is the case with algorithms that are dependent on the context of time. For example, Google's autocomplete search algorithm as highlighted in a study by Kathuria, Nagpal and Duhan (2016). A limitation thus arises of not being able to appraise an algorithm in all possible contexts. However, the research applied the countermeasure of coming up with a representative sample of all possible contexts and appraising the algorithm in a manageable sample of contexts.

Sixthly, with regard to sampling limitations, this research took the algorithms through a specific treatment multiple times. This repetition was done with the aim of increasing the statistical accuracy of the experiments and subsequently the validity of the research. It is also important to note that the research limitation of limited generalizability can also be experienced if the state of a subject prior to a treatment, say treatment B, is affected by the results of applying a previous treatment, say treatment A. However, the

research enjoyed the countermeasure caused by using inanimate subjects. It was therefore able to reset the state of the experiment test-bed to its initial state prior to applying and observing the effects of any treatment.

The generalizability of these results is subject to certain limitations. For instance, the computing resources available in the experiment test bed and the licensing requirements limited the number of concurrent virtual users that could be used to 20. These concurrent virtual users were used to generate and submit transactions and their inputs simultaneously to the database system. In reality, a medium or large enterprise will have much more than 20 concurrent users of a business information system. Notwithstanding the relatively limited number of concurrent virtual users, the study offers valuable insights on how automatic performance tuning can be conducted to achieve self-optimization in a system. These insights are particularly useful to small enterprises that are not yet able to afford to hire a permanent Database Administrator.

Lastly, the scope of this study was also limited in terms of the data model of the database under investigation. The only data model was a relational data model. This scope did not provide an opportunity to investigate performance tuning of other models such as graph, document, column-family, array/matrix, hierarchical, and network data models. This includes the data models in Enterprise Data Warehouses (EDWs) as explained in Appendix D. In spite of the scope limitation, the study certainly adds to our understanding of how to use an MCTS algorithm to automatically tune database systems based on relational data models that apply an InnoDB storage engine. The findings reported in this study also shed new light on the need for and how to automate the performance tuning process. Researchers can add on to this knowledge by extending and applying it to different data models.

Chapter 2: Literature Review

2.1 Introduction

This literature review provides an analytical perspective of key themes in the research required to formulate a coherent argument. The following seven questions are used as points of departure in order to present a systematic review:

(i) How can control theory be applied to ensure a system continuously operates at a desired performance level? (Addressed in Section 2.2.5)

(ii) How can systems use control theory concepts to automatically and proactively adapt to environmental changes and runtime phenomena that may occur in the future? (Addressed in Section 2.2.4)

(iii) How can systems use decision theory and reinforcement learning to decide which action should be performed to obtain the highest reward? (Addressed in Sections 2.2.6 and 2.3.4)

(iv) What are the parameters that need to be considered when optimizing the performance of a database system? (Addressed in Section 2.4)

(v) Which optimization techniques can be used to achieve load scalability in servers? (Addressed in Section 2.5)

(vi) What are the merits and demerits of optimization techniques that can be used to achieve load scalability in servers? (Addressed in Section 2.5)

(vii) Is there a significant advantage in applying machine learning as opposed to linear optimization when performing self-optimization? (Addressed in Sections 2.3.4 and 2.5.5)

2.2 Control Theory and Decision Theory

This research applied control theory and decision theory as the core theories. Control theory is based on the foundations of feedback theory and linear systems analysis. It integrates the concepts of network theory and transactional-model based decentralized communication theory in order to develop systems that have self-organizing, adaptive, robust, and optimum qualities. It is widely used in engineering to control energy use in industry and to stabilize and connect loads evenly to gain fuel economy (Moreno, Cámara, Garlan & Schmerl, 2018). However, recent studies have indicated that the same theories can be applied in numerous contexts such as, in the control of agricultural systems, auto-

mobile engine controllers, automatically controlled aids for the disabled, humanoid robots, and, as this research did, in the control of the performance of database systems (Cheng et al., 2016).

The main objective of control theory is to use a controller (*C*) to control a system in such a way that its output at time *t*, that is, the feedback signal *y(t)*, follows a desired control signal, *r(t)*. The desired control signal is known as the reference whereas the system being controlled is known as the plant or process, *P*. A controller would then obtain an error signal, *e*, which is the difference between the reference and the actual output, *r(t) - y(t)*. In order to tend towards obtaining the reference, the error signal informs the feedback applied using an actuator, *A*, in a closed-loop feedback control system to inform the input, *u(t)*, of the system (Lewis, Vrabie & Vamvoudakis, 2012). This can be represented as a block diagram of a single-input-single-output (SISO) closed-loop negative feedback control system as shown in Figure 2.1 below.

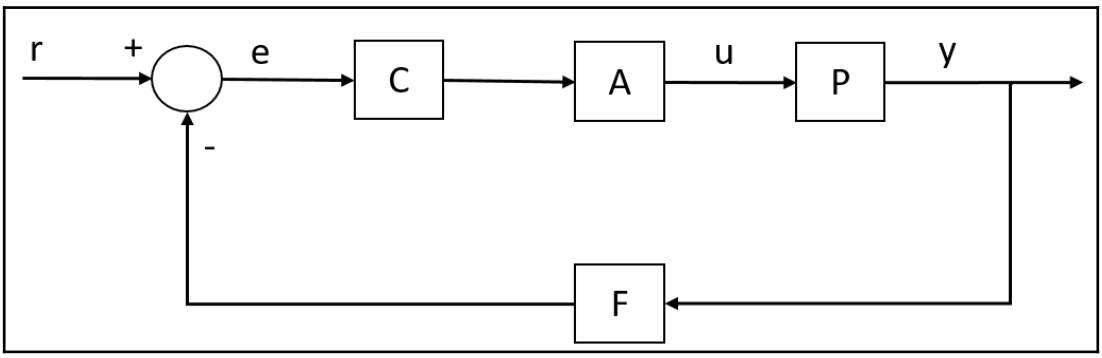

Figure 2.1: Block diagram of a closed-loop negative feedback control system

2.2.1 Establishment of Control Goals and Variables to be Controlled

The control goal in this research was to "design a system to regulate the transaction throughput and response-time latency of a database system given any workload, by controlled bottleneck reconfiguration in order to achieve load scalability". The variables to be controlled (manipulated) were subsequently the set of all configuration parameters exposed to system administrators by the developers of the database system. Appendix E provides a list of these parameters and Section 4.2 explains how the parameters were selected. In this thesis, the terms "feature", "configuration parameter" and "variable to be controlled" are used interchangeably to refer to the same thing. Based on this, the System Requirements Specification (SyRS) informed by the ISO/IEC/IEEE 29148:2011 standard

are explained in Appendix A.

2.2.2 System Definition and Modelling

Even though a variety of definitions of the term model have been suggested, this research will use the definition of a model as a description of a system using mathematical or computational concepts and language (Astrup, Coates & Hall, 2008). A model can be either a mathematical model or a computation model, or in some cases a hybrid of a mathematical and a computational model. A mathematical model is defined as a set of variables and a set of equations that establish relationships between the variables. Whereas a computational model is defined as a computer program that implements computational techniques for example, automata theory, petri nets, or artificial neural networks to describe a system. Models are necessary to control complex systems and to understand the phenomena related to them. More specifically, models are necessary to:

(i) Predict: Make testable predictions of events expected to happen in future
(ii) Explain: Reveal underlying mechanisms or rule out proposed explanations
(iii) Discover: Propose new questions as starting points of research
(iv) Guide: Guide the collection of data or the design of experiments

Two of the many methods that can be used to create models are differential equations and statistical methods. Differential equations include Ordinary Differential Equations (ODEs) and partial differential equations. On the other hand, statistical methods include linear regression, multilevel models, structural equation models, and Kalman filters (Ranadip, 2018). Regardless of the method, Figure 2.2 shows the flow of the general modelling process of a dynamic system.

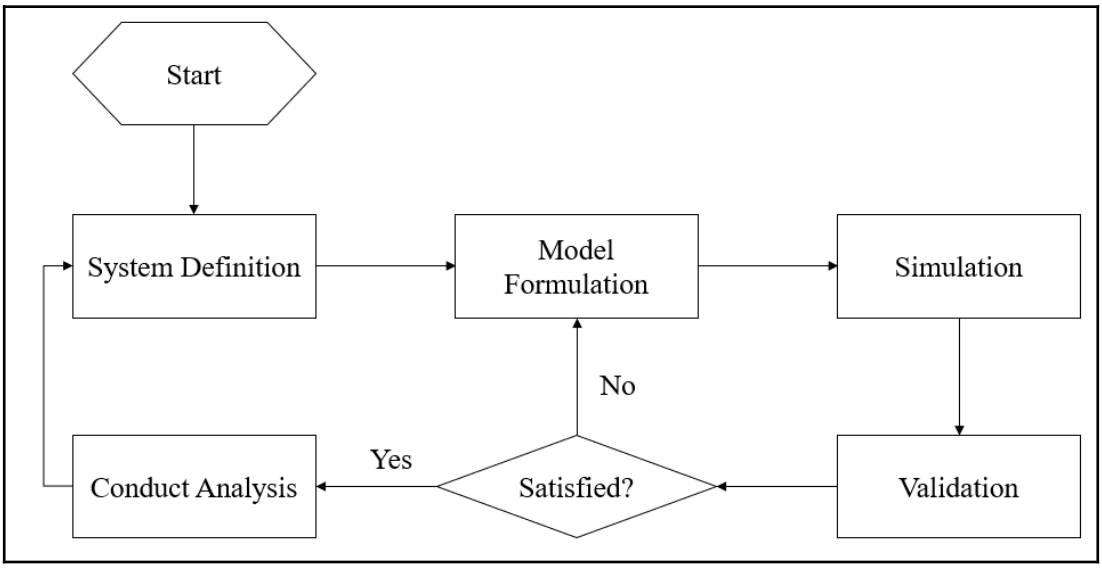

Figure 2.2: Dynamic system modelling process

Figure 2.3 further extends the dynamic system modelling process by placing it in the context of a broader research. This represents an inductive-deductive scientific approach to research.

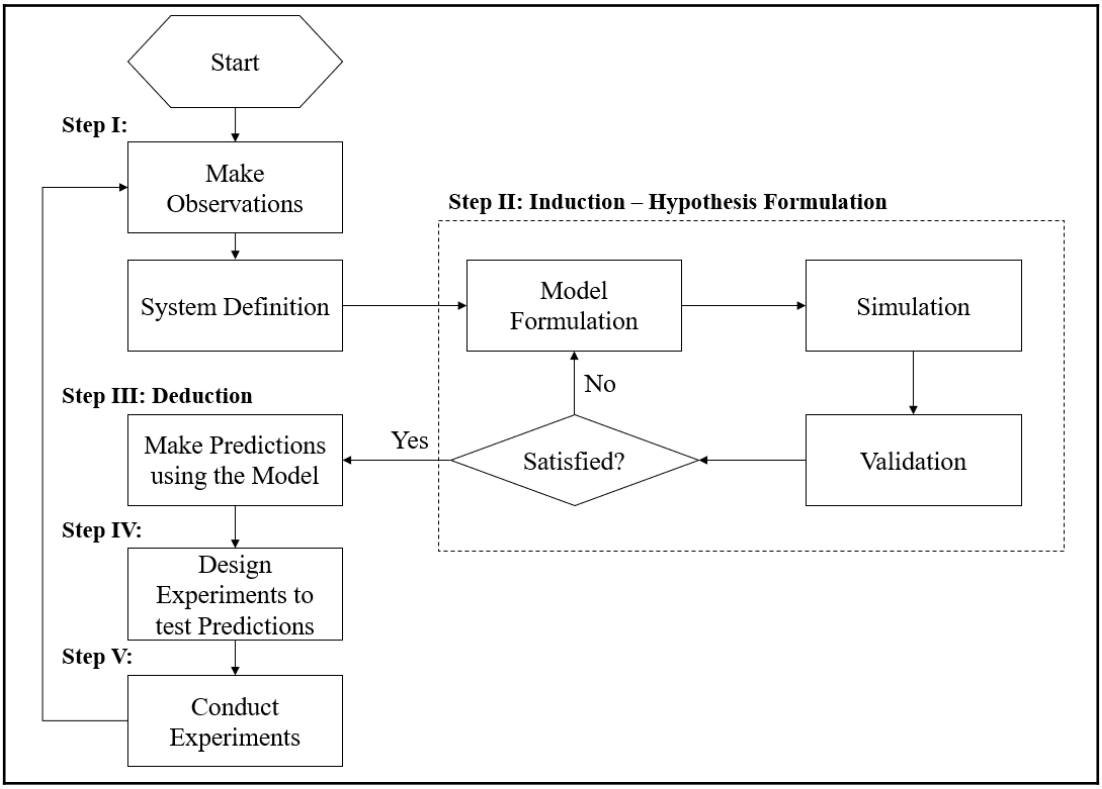

Figure 2.3: System modelling in the context of a broader research (adapted from Astrup et al., 2008; Mykoniatis, 2015)

2.2.2.1 Ordinary Differential Equations

Unlike a static system whose output depends on the present input values, the output of dynamic systems depends on the present and past values of inputs. Figure 2.4 further depicts this difference. In the case of this research, the system under consideration is a dynamic system and therefore differential equations can play a major role in system modelling. Differential calculus cuts the performance of the database system into small pieces to find out how it changes over time. Whereas integral calculus joins the small pieces together to find out how much change there is (Astrup et al., 2008).

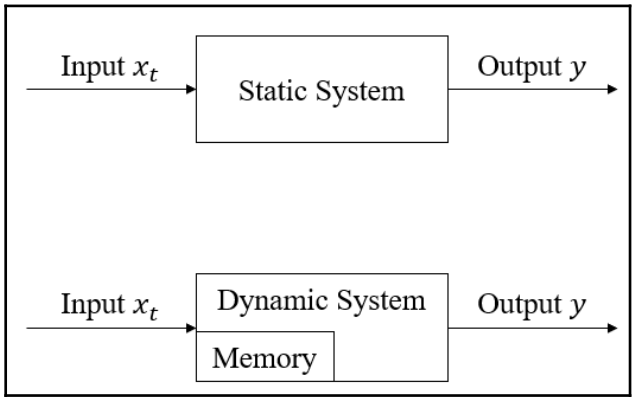

Figure 2.4: Difference between static systems and dynamic systems

With the assumption that the quantity of data in the database system does not decrease, the following ODE models the behaviour of the system according to how the performance changes over time:

Feasibility constraint in Equation (2.1):

$$\frac{dP}{dx} = \sum_{p \in T} \left(\frac{(t_p) y_w + 1}{(r_p) z_w + 1 + e_p + a_p} \right) \times \left(\frac{1}{qps + u} \right) \qquad \textit{(2.1)}$$

Where each parameter, p, in a tactic, T, $p \in T$ is characterized by:

(i) its effect on the transaction throughput (t)

(ii) its effect on the response-time latency (r)

(iii) its negative effect on the hardware resources of the distributed database (e)

(iv) its adaptation latency (a)

(v) the number of active concurrent users (u)

(vi) queries per second submitted to the database system by all concurrent users (qps)

The effect in (iii) can be measured using resource metrics as explained in Appendix F.

And:

y_w, z_w

Such that:

$$y_w = \begin{cases} 1 & \text{workload } w \text{ is an Online Transaction Processing (OLTP) workload} \\ 0 & \text{otherwise} \end{cases}$$

$$z_w = \begin{cases} 1 & \text{workload } w \text{ is an OLAP workload} \\ 0 & \text{otherwise} \end{cases}$$

Further details regarding how the ODE and its integral were obtained in the process of system modelling are provided in Appendix B.

2.2.2.2 Kalman Filters

A Kalman filter is also an ideal way of modelling dynamic systems that are continuously changing (dynamic systems). It is capable of combing estimates of system states even in the presence of noise (Ranadip, 2018). The speed of Kalman filters makes them well-suited for real-time problems and embedded systems. This speed is attributed to the fact that Kalman filters do not keep a historical record of all states. They instead keep track of only the most recent state.

Specific metrics can be used to define the state of a database system. The results of the current study confirmed that samples from measurements of random values of metrics that were independently measured have a Gaussian distribution. This result holds provided that there are sufficient samples according to the law of large numbers. The centre of the Gaussian distribution of each metric is its mean while its variance represents the uncertainty present in its estimated value. Figure 2.5 depicts this by showing a 3D and 2D representation of a Gaussian distribution with a mean of 0 and a standard deviation of 1. As shown in the 2D representation, points further away from the mean are blurred representing their sparsity. The Kalman filter treats this blurred area as an area of uncertainty. The closer a value is to the mean, the higher the certainty that it is a correct value (Ranadip, 2018). In the context of this research, the mean is a matrix that contains an estimation of the values of metrics that collectively define the state of the database system.

Kalman filters attempt to extract as much information from the uncertain values of the metrics that are used to define the state of the database system. Part of the information extracted includes the covariance which represents the joint variability of random variables. The Kalman filter makes use of a covariance matrix to represent the degree of correlation.

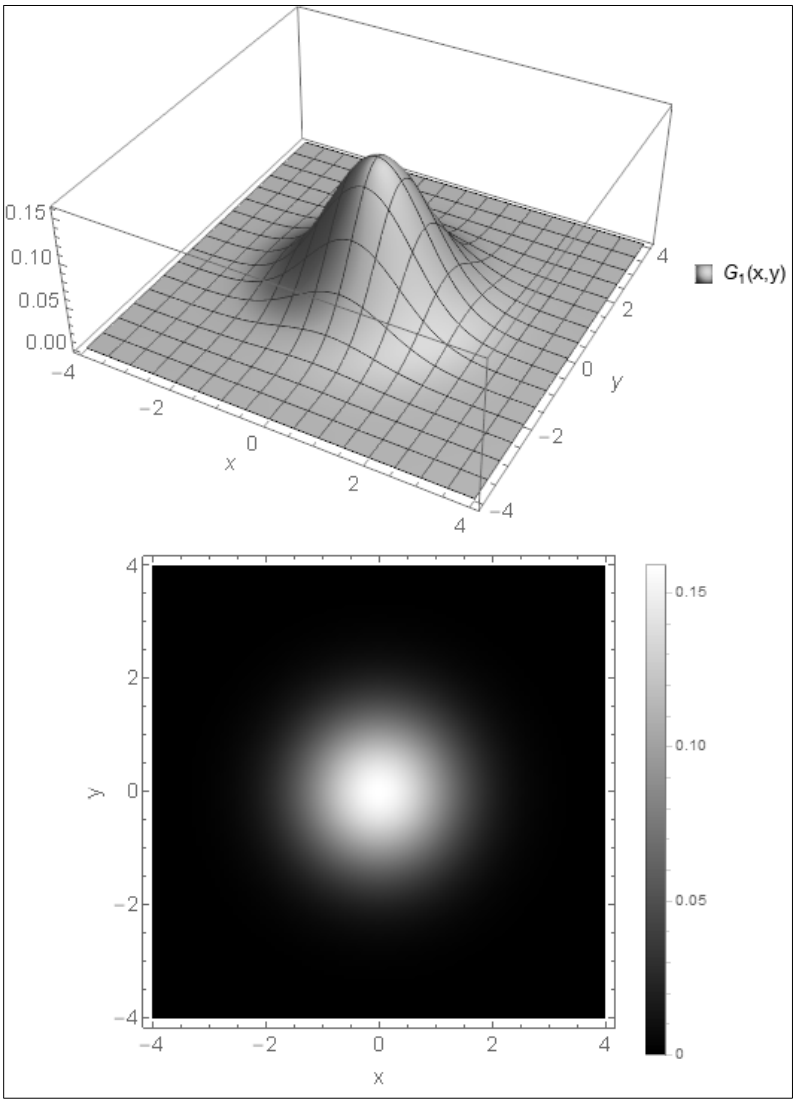

Figure 2.5: 3D and 2D Gaussian distribution
(adapted from Sellner, 2019)

The Kalman filter requires:

(i) **A state** $\hat{x}_k$ **:** the best estimate of the state of the database system as defined by the value of its metrics at time *k*. This tends as close as possible to the mean of the database system's state according to a Gaussian distribution.

(ii) **A covariance matrix (** P_k **):** to represent the degree of correlation between the different values of metrics that define the state of the database system

(iii) **The prediction matrix (** F_k **):** that represents the future state (with all the observable metrics) of the database system at time *k*. This is derived by using the covariance matrix, P_k , and the original estimate of the state of the database

system at time *k-1*, that is, $\hat{x}_k$. Using the covariance matrix, P_k , on the past state, x_{k-1} , will lead to the future state x_k if and only if x_{k-1} was a correct estimate as shown in Equation (2.2). Such that:

$$\hat{x}_k = F_k \hat{x_{k-1}} \quad \textbf{(2.2)}$$

The covariance matrix is updated as shown in Equation (2.3):

$$P_k = F_k P_{k-1} F_k^T \quad \textbf{(2.3)}$$

Environmental variables and runtime phenomena triggered by external influences can also form a basis for the system's state to change. If we know additional information about the external influences, then we can record this information into a control vector $\vec{u_k}$. All the control vectors are in turn recorded in a control matrix B_k . The new estimated future state is thus made from the previous estimated state while making corrections for known external influences of the database system as shown in Equation (2.4):

$$\hat{x}_k = F_k \hat{x_{k-1}} + B_k \vec{u}_k \quad \textbf{(2.4)}$$

However, in reality, there can be spikes in workload and we may be uncertain of what triggered the spike. A possible explanation for this uncertainty is that we are not keeping track of what caused the trigger. This uncertainty can be modelled by adding a new uncertainty after every prediction step. This implies treating the uncertainty as noise that has a Gaussian distribution with its covariance as Q_k . The new uncertainty (recorded in the covariance matrix P_k) can then be predicted from the old covariance matrix but with the consideration of the additional noise from the environment as shown in Equation (2.5).

$$P_k = F_k P_{k-1} F_k^T + Q_k \quad \textbf{(2.5)}$$

The Kalman filter is also able to reconcile the estimated state and the predicted state by using the actual state that has been observed by recording the metrics that define the state of the database system. Reconciling by multiplying the two matrices (that are both Gaussian distributions) gives us the region where both the estimated state and the actual state are most likely to occur. That is, the intersection/overlap of the estimate and the actual state. This new region is in turn a new Gaussian distribution with its own mean and

variance as shown in Figure 2.6.

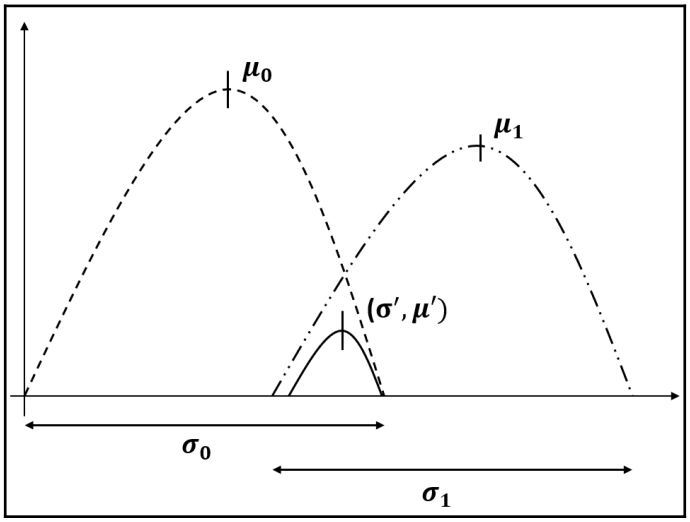

Figure 2.6: Actual (μ_0) versus estimate (μ_1) overlap for reconciliation

At this point, the prediction of an estimated state and the updating of the covariance matrix proceed iteratively as we continuously improve the estimation. This is as shown in Figure 2.7.

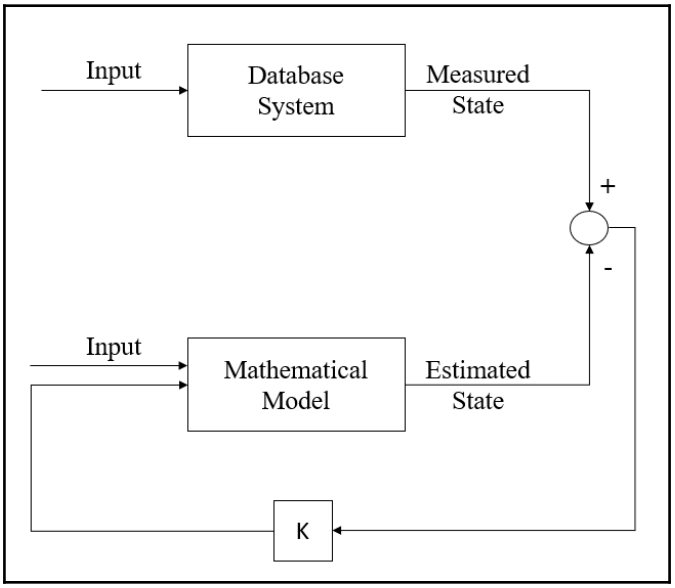

Figure 2.7: Application of control theory in predict-update iteration

2.2.2.3 Linear Regression

The aim of linear regression is to estimate the value of a response (Y) when only the predictors (X) are known. This is achieved through the use of regression coefficients and an error term (Ma & Liu, 2016). In the case of this research, the predictors are the measurable internal and external factors while the response is the performance of the database system. The internal factors include parameter configurations in the database system. External factors include the number of concurrent users and the quantity of queries per second. The performance can then be measured in terms of transaction throughput for OLTP workloads and response-time latency for OLAP workloads.

The integral of the ODE, explained in Section 2.2.2.1 and in Appendix B, can be used to formulate the linear regression. Given that OLTP and OLAP workloads sometimes require conflicting parameters to be adjusted, the linear equation for OLTP and that for OLAP need to be separated (Shahapure & Jayarekha, 2014). This implies conducting performance tuning for OLTP is not the same as conducting performance tuning for OLAP. The linear regression for OLTP will be as shown in Equation (2.6):

$$y = \beta_0 + \beta_1\left(\frac{t_T^2}{2}\right) y_w \times \left(\beta_2 \ln|e_T| + \beta_3 \ln|a_T|\right) + \beta_4 \ln|qps| + \beta_5 \ln|u| + \epsilon \quad \textbf{(2.6)}$$

The linear regression for OLAP will be as shown in Equation (2.7):

$$y = \beta_0 + \beta_1 \ln|(r_T) z_w| \times \left(\beta_2 \ln|e_T| + \beta_3 \ln|a_T|\right) + \beta_4 \ln|qps| + \beta_5 \ln|u| + \epsilon \quad \textbf{(2.7)}$$

Whereby each tactic, T, is characterized by:

(i) its effect on the transaction throughput (t)
(ii) its effect on the response-time latency (r)
(iii) its negative effect on the hardware resources of the distributed database (e)
(iv) its adaptation latency (a)
(v) queries per second submitted to the database system by all concurrent users (qps)
(vi) the number of active concurrent users (u)

The effect in (iii) can be measured using resource metrics explained in Appendix F.
And:

y_w, z_w

Such that:

$$y_w = \begin{cases} 1 & \text{workload } w \text{ is an OLTP workload} \\ 0 & \text{otherwise} \end{cases}$$

$$z_w = \begin{cases} 1 & \text{workload } w \text{ is an OLAP workload} \\ 0 & \text{otherwise} \end{cases}$$

2.2.3 Control System Design

When the performance of the database system begins to settle at a level below expectations, the control can be introduced iteratively in order to automatically maximize it. Subsequently, Figure 2.1 can be extended as shown in Figure 2.8.

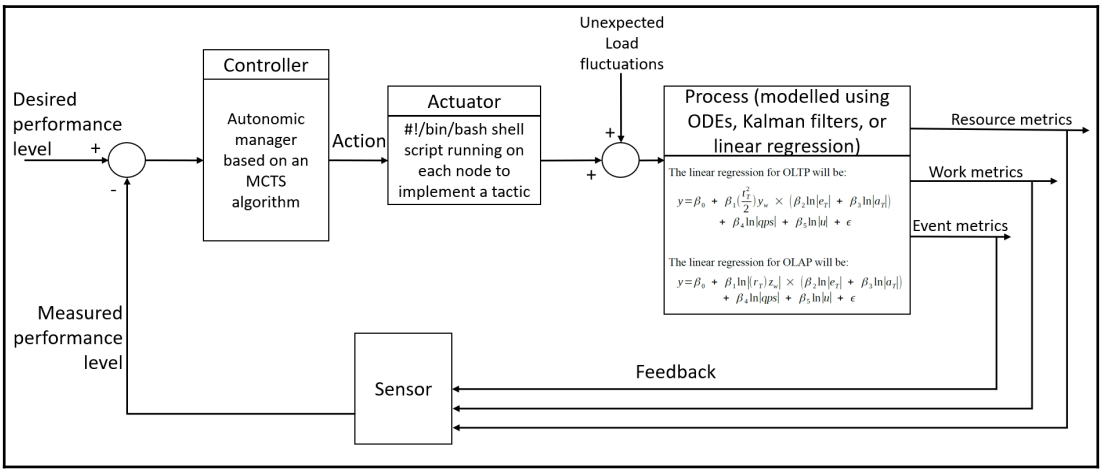

Figure 2.8: Block diagram of the closed-loop negative feedback control system

2.2.4 Model Predictive Control

A study by Moreno et al. (2017) indicated that predictive control describes an approach (as opposed to an algorithm) for how to think about designing control systems. The keyword in predictive control is anticipation. This implies that all the future obstacles (environmental variables and runtime phenomena) and performance targets are anticipated and used to determine the most effective strategy the controller should apply in the current timeslot. A model of the environment is required to support this anticipation. As a result, Model Predictive Control (MPC) is used to optimize the current timeslot while taking into

consideration future timeslots that are yet to be realized. Three key concepts arise in the realization of this action: (i) the use of models to predict future system states and behaviours (ii) the computation of a sequence of possible control actions to be performed with a commitment to only the first action in the sequence, and (iii) the computation of a sequence of control actions after the first control action has been performed (receding horizon).

A dynamic state of the system modelled by a block diagram, performance measurements from parametric sensors, and the desired performance targets and limits are all used to calculate the anticipated future behaviour of the storage server. Once the behaviour is known, then the objective becomes to maintain the database system as close to the predefined targeted performance level as possible through the execution of appropriate actions. However, the manner in which this MPC process is undertaken can vary based on the framework applied by the control system (Moreno et al., 2018). Two possible MPC frameworks that can be applied in non-physical systems include: CobRA, a requirements-based approach that applies control theory, and PLA, an architecture-based approach that applies stochastic analysis. Fundamental differences exist between these two frameworks as shown in Table 2.1.

Table 2.1: Comparative analysis of CobRA and PLA MPC frameworks

Attribute	CobRA	PLA
Anticipation of System's Behaviour	Applies an implicit representation of the system behaviour to compute the corresponding optimal control strategy. This representation is based on an optimal estimation algorithm such as the Kalman filter, an ODE, or a linear regression.	Applies an explicit representation of the system's behaviour to compute the corresponding optimal control strategy. This representation is based on stochastic analysis performed using algorithms such as Gillespie's Stochastic Simulation Algorithm.
Actuation	Applies control strategies through the use of tactics in a discrete manner. It is therefore better suited to deal with discrete control strategies.	Applies control strategies incrementally and in a continuous manner and relies on setting different control parameter values for actuation. It is therefore better

		suited to deal with continuous control inputs.
Goal model	Attempts to keep the values of indicators around reference values obtained from the goal model	Optimization is driven by a utility function (the reward) that captures goals

Recent studies have shown that CobRA performs comparatively better than PLA in certain contexts. The choice of action to perform in relation to moving the system from one state to the next is always the most optimum action in the case of CobRA. However, CobRA, being an iterative process, requires parameter tuning before it can converge to a stable solution. It is for this reason that PLA is better suited to deal with continuous control strategies, whereas CobRA is better suited to deal with discrete control strategies. Unlike CobRA, PLA is over-reliant on an accurate prediction of the environment. The environment and system model therefore need to be able to foresee multiple events, such as abrupt increase in workload. However, obtaining an accurate prediction is not always possible because not all variables can be observed (Taleb & Blyth, 2011). These findings, therefore, discredit PLA as a realistic approach to applying MPC in optimization of load scalability in database systems.

2.2.5 Condition-Based Maintenance

A study by Bousdekis, Magoutas, Apostolou and Mentzas (2015) proposed a framework for proactive prognosis and decision-making in the context of Condition-Based Maintenance (CBM). The study argued that through proactive decision making in an enterprise, would control occurrence of undesired future events, such as unplanned downtime by using short-term prediction and automated decision-making technologies. This is as opposed to being in a reactive state which does not allow an enterprise to be aware of what might happen. It also prevents an enterprise from being able to optimize its behaviour in order to achieve what should happen. Figure 2.9 graphically depicts the proactive decision-making framework using a block diagram.

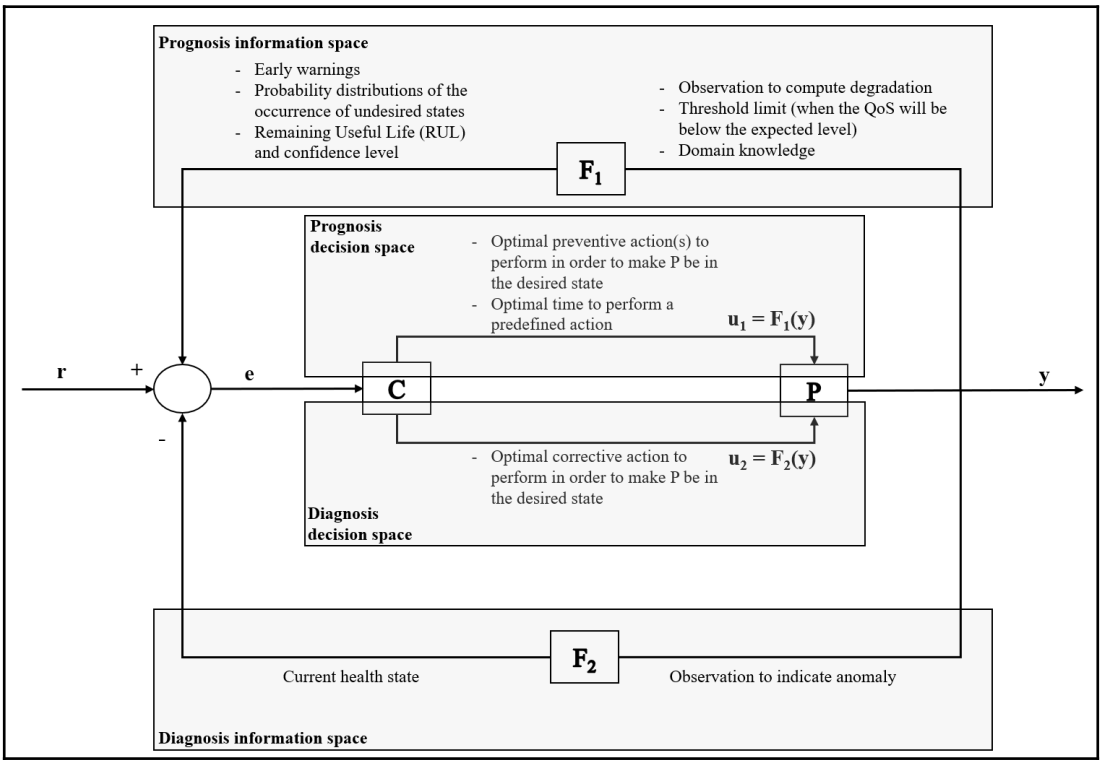

Figure 2.9: Proactive decision making framework for maintenance

(adapted from Bousdekis et al., 2015, p. 1241)

A combination of concepts of ODEs for system modelling and CobRA principles for Model Predictive Control formed a strong foundation required for the expected output of this research.

2.2.6 Decision Theory and the Lucid Fallacy

Decision theory, also known as the theory of choice, is a discipline concerned with the study of the reasons underlying an individual agent's choices under uncertainty. This is unlike game theory which extends this concept further by studying the interaction of multiple agents whose decisions affect each other (Hartmann, 2017). Whereas normative decision theory is concerned with providing advice on how an agent can make the best decision given a set of uncertain values, descriptive decision theory is concerned with the analysis of how agents make the actual decisions.

The concept of expected reward is central to decision theory (Chajewska, Koller & Parr, 2000). Expected reward involves multiple possible actions to choose from. Each action in turn has multiple outcomes with various probabilities of occurring. The first step

in making a rational decision regarding which action to perform will involve identifying the actual value of each outcome and the probability that performing the action will lead to a specific outcome (Pynadath & Marsella, 2005). The probability and the value are then multiplied to obtain the expected value. As later discussed in the current thesis, this forms a critical foundation for optimization theory and for Monte Carlo Tree Search based algorithms.

Prediction is an exercise in futility (Taleb & Blyth, 2011). This is true because of the lack of accuracy involved in making predictions. However, an agent needs to know the possible outcomes of actions before it can decide which action to perform, hence resulting in a conundrum. A possible workaround to this, as this research applied, is to award the immediate outcomes a higher weight in comparison to distant outcomes that will be manifested in the future. This essentially promotes the reduction of the long-term prediction horizon of an agent and reduces its over-reliance on an inaccurate model of future behaviour of a stochastic system. As explained further by the lucid fallacy, there are inevitable imperfections in modelling the real world and over-reliance on models blinds the user to their limits (Taleb & Blyth, 2011). A significant portion of their limits is attributed to the fact that there can be unknowns that are not captured in a model. In such cases, even the error term in linear regression is inadequate.

2.3 Autonomic Computing

As computing systems get more optimized, the complexity involved in managing them increases rapidly and this can result in a barrier to further growth (Li et al., 2017). Autonomic computing, if implemented effectively, enables such systems to adapt to unpredictable changes while hiding intrinsic complexities. Today's High Availability/Disaster Recovery (HA/DR) requirements on complex mission-critical systems put greater demands on computing systems to have self-management features. These self-management features are needed in order to maintain a desirable Quality of Service (QoS) in the presence of system faults, variable environmental conditions and runtime phenomena, dynamic user expectations, cyber attacks, system integration, and system installation (Cheng & Garlan, 2012; Jimoh & McCluskey, 2016). This further supports the need for autonomic computing because self-management is the essence of autonomic computing systems. Figure 2.10 categorizes the self-management features of

autonomic computing into 4 paradigms.

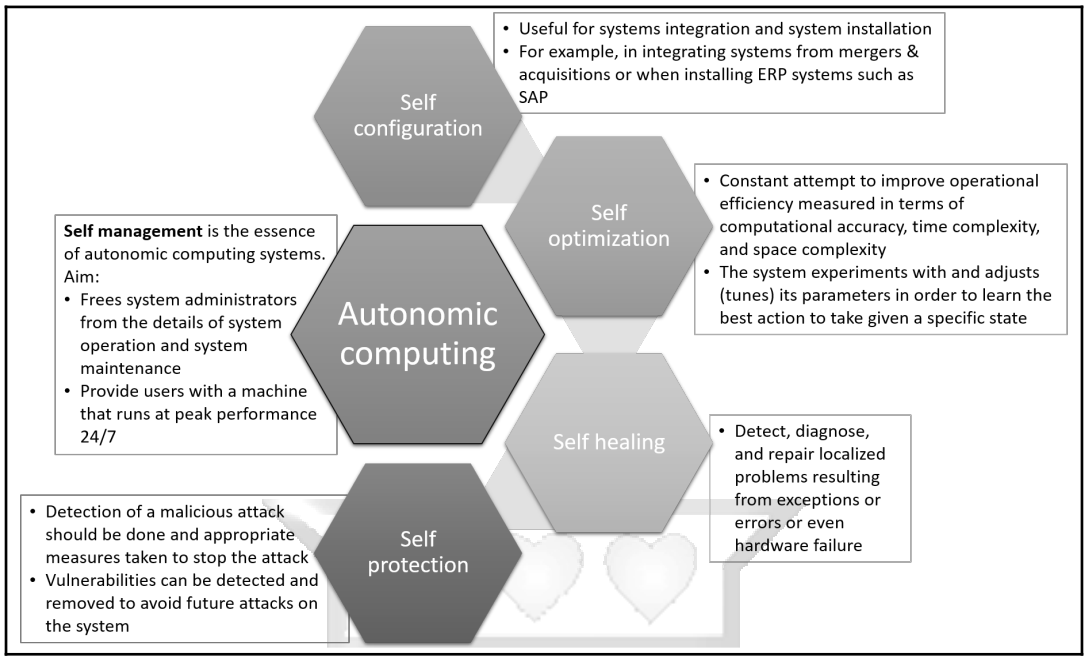

Figure 2.10: Categories of autonomic computing

Even though system administrators are better at understanding the overall problem context than computers, they are prone to long reaction times, fatigue, errors, and varying and potentially inconsistent expertise (Cheng & Garlan, 2012). The results of this research showed that it is possible to maintain the desired QoS, measured in terms of transaction throughput and response-time latency from a distributed database, by using an algorithm to automate routine but complex tasks that system and database administrators would otherwise have to perform.

The original intent of self-management and its 4 paradigms was to free system administrators from the details of system operation and maintenance and to provide users with a machine that runs at peak performance 24/7 (Kephart & Chess, 2003). The first paradigm is self-configuration. This is useful when an enterprise is performing systems integration, for example, during a merger and acquisition (M&A) (Henningsson, 2015). Self-configuration is also useful during system installation, for example, complex configurations involved in ERP and e-commerce business information system installations. The second branch of autonomic computing is self-optimization. In this paradigm, an autonomic system constantly strives to find ways in which it can improve its operational

efficiency. Operational efficiency in this case can be quantitatively measured through the use of work metrics. The third branch of autonomic computing, self-healing, involves the diagnosis and repair of localized problems resulting from exceptions or errors related to software or hardware failure. A possible way to implement self-healing is by matching diagnosis against known software patches and then automatically installing the appropriate patch. Last but not least is the branch of self-protection. An autonomic system designed for self-protection should diagnose a malicious attack and stop it from causing further damage to a system. It should also mitigate against potential vulnerabilities.

In the case of self-optimization, exception handling code embedded within the system can be used to maintain the desirable QoS. This would work by coding the system to throw exceptions if the QoS falls below a certain threshold and then handling the thrown exceptions using exception handlers. However, the occurrence of runtime phenomena is stochastic in nature and can be asynchronous with respect to the flow of the application logic. For this reason, it is preferable to gather the complex adaptation logic and profile formulation into a component that is distinct from the managed system (Su et al., 2016).

This distinction can be implemented by combining features of the Data-Information-Knowledge-Action-Reward (DIKAR) model, the Define-Measure-Analyse-Improve-Control (DMAIC) data-driven strategy and the IBM autonomic Monitor-Analyse-Plan-Execute (MAPE) reference model as shown in Figure 2.11 (Autor, 2015).

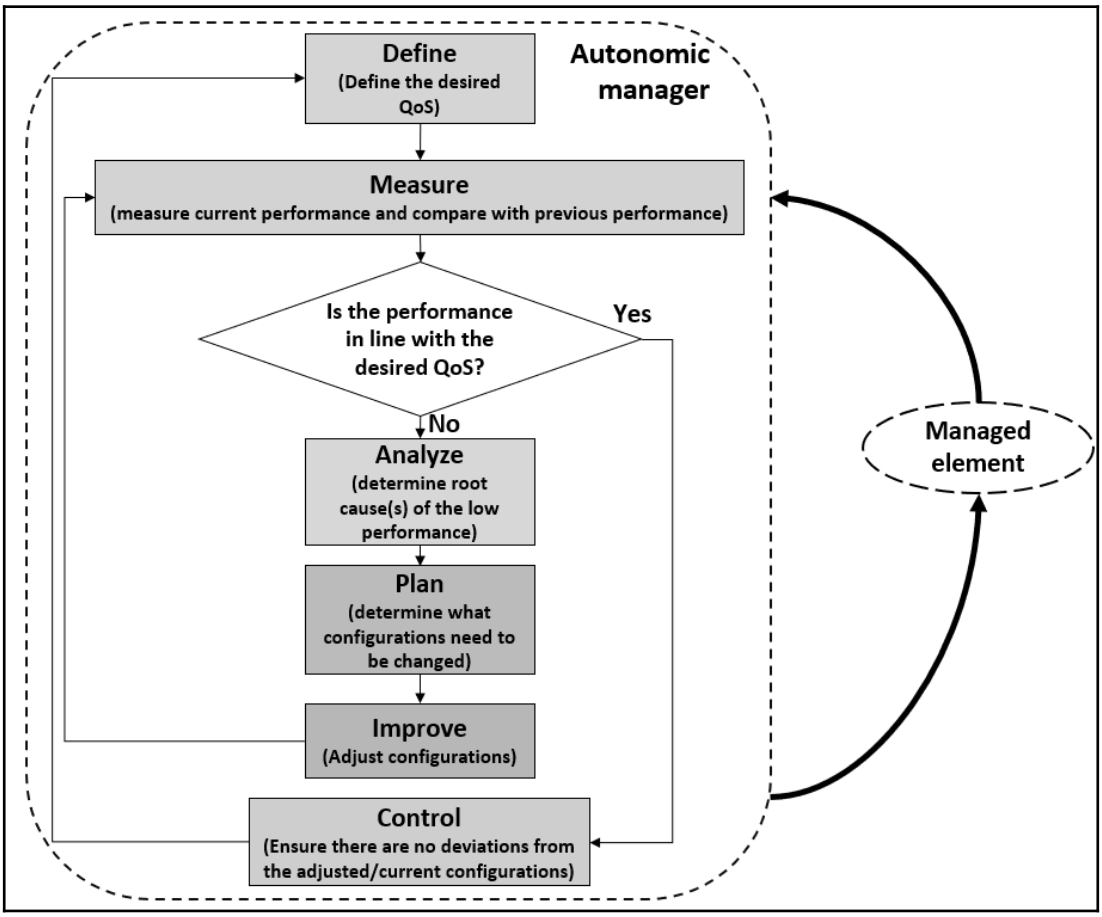

Figure 2.11: Adapted combination of the DIKAR model, the DMAIC data-driven strategy and the IBM MAPE reference model

2.3.1 Strategy Definition

The algorithm was designed to proactively reconfigure bottleneck parameters without over-relying on an accurate model of a stochastic environment. This was done in order to achieve self-optimization required for load scalability in database systems. Various primitive operations can be performed to achieve this aim. In the case of MPC's CobRA technique, multiple successive operations are involved. This results in a decision process whereby a specific collection of primitive operations (bottleneck reconfigurations) are performed and the observed/measured effect of these operations are in turn used as feedback to decide what to do next. It is therefore a continuous, process that keeps adapting based on the feedback it receives.

Measuring the effect of each primitive operation would make the overall speed of adaptation too slow because of the wait-time incurred while obtaining measurements and

too sensitive because of the response to each feedback received. The high sensitivity would result in a higher demand for compute resources which is one of the factors that this research aimed to address. It is better to observe the effect of executing a collection of primitive operations, referred to herein as a tactic (Duan et al., 2016). Various tactics can then be applied to tend towards the achievement of the predefined goal. The collection of the chosen tactics would then form a strategy that can subsequently be expressed as a decision tree.

2.3.2 Optimization Problem

In order to arrive at the desired goal of maximizing the transaction throughput and minimizing the response-time latency, the tactics that form the strategy must be chosen carefully so as to select only the best amongst the set of feasible solutions. Various factors that can determine what are the best possible tactics include the defined objectives and priorities of the enterprise, past successes and failures, as well as the relative costs and benefits associated with the tactic (Lewis et al., 2012). This can be modelled mathematically as an optimization model whose aim would be to find the variables (configuration parameters) that need to be changed in order to maximize the transaction throughput and minimize the response-time latency.

This results in complex dynamic reasoning based on the number of factors that have to be considered simultaneously in order to decide which tactics are the best (Chajewska et al., 2000). Three key questions follow based on this complex dynamic reasoning: (i) How can the complex dynamic reasoning be expressed in the autonomic manager? (ii) How can the autonomic manager ensure that it makes only the best decisions without being explicitly informed of what the best decisions are in every possible state? (iii) How can the autonomic manager know which decisions had a positive reward and which ones had a negative reward?

It is at this juncture that answering these questions solicits the need to amalgamate control theory, decision theory, and the branch of self-optimization in autonomic computing with machine learning theories from the AI community arises. One possible AI paradigm that is in line within the foundational theories that informed this research is reinforcement learning. The following section defines the mathematical model of the optimization problem. This forms a pre-requisite to be used in the subsequent review of

reinforcement learning.

2.3.3 Mathematical Model of the Optimization Problem

Let $C=\{p_1, p_2, ..., p_n\}$ be the set of all possible configuration parameters in the distributed database cluster's identical nodes. It consists of a total of *n* configuration parameters. Since not all *n* configuration parameters may be optimal in terms of their ability to maximize the transaction throughput and the response time latency, then some can be considered and others can be left out. We can then define a subset of configuration tactics from all possible configuration parameters $T \subseteq C$ such that each element in the subset $p \in T$ can be used to maximize the transaction throughput and response time latency.

Given that the chosen configuration parameters for each node in the cluster can be grouped into a set T_k we can have *k* sets of *T* each with unique combinations of parameters as members of the set. Each parameter $p \in T$ is characterized by:

(i) its actual value which has an effect on a specific hardware resource, i (V_{pi})
(ii) its effect on the transaction throughput (*t*)
(iii) its effect on the response-time latency (*r*)
(iv) its negative effect on the hardware resources of the distributed database (*e*)
(v) its adaptation latency (*a*)
(vi) the capability of the database system's hardware under consideration (K_i)

The objective is therefore to find the subset of parameters in T that:

(i) maximizes the transaction throughput (*t*)
(ii) minimizes the response-time latency (*r*)
(iii) minimizes its negative effect on other parameters in the distributed database (*e*)
(iv) has a minimum adaptation latency (*a*)
(v) has a value (*v*) that is less than or equal to the hardware capabilities of the server (*K*)

Decision variables:

x_p, y_w, z_w

Such that:

$$x_p = \begin{cases} 1 & \text{a decision has been made to reconfigure parameter } p \\ 0 & \text{otherwise} \end{cases}$$

$$y_w = \begin{cases} 1 & \text{workload } w \text{ is an OLTP workload} \\ 0 & \text{otherwise} \end{cases}$$

$$z_w = \begin{cases} 1 & \text{workload } w \text{ is an OLAP workload} \\ 0 & \text{otherwise} \end{cases}$$

Feasibility constraint in Equation (2.8):

$$\sum_{p \in T} V_{pi} \leq K_i \tag{2.8}$$

In other words, if a parameter, p, causes the value, V, of a specific hardware i (V_{pi}) to change, then the change should be less than or equal to what the database system's hardware is capable of handling for the hardware under consideration, K_i .

Objective function in Equation (2.9):

$$\sum_{p \in T} \left(\frac{(t_p) y_w + 1}{(r_p) z_w + 1 + e_p + a_p} \right) x_p \tag{2.9}$$

The mathematical model of the optimization problem will therefore be as shown in Model (2.1):

$$\begin{aligned} &\text{Maximize:} && \sum_{p \in T} \left(\frac{(t_p) y_w + 1}{(r_p) z_w + 1 + e_p + a_p} \right) x_p \\ &\text{Subject to:} && \sum_{p \in T} V_{pi} x_p \leq K_i \\ &\text{Such that:} && x_p, y_w, z_w \in \{0,1\}, p \in T \end{aligned} \tag{2.1}$$

2.3.4 Reinforcement Learning

Reinforcement learning sits at the intersection between many fields of science as the study of the most optimal way to make the best decisions (Duan et al., 2016). These fields include machine learning in computer science, operations research in mathematics,

optimal control in engineering, bounded rationality in economics, classical and operant conditioning in psychology, and the reward system in neuroscience. This research focused on the computer science field whereby reinforcement learning, supervised learning, and unsupervised learning form the three paradigms of machine learning.

When it comes to finding the most optimal way to make the best decision, reinforcement learning uses feedback to define how well an agent is performing towards achieving a goal at any point in time. This is known as a reward at time t, R_t . The agent's goal is thus to maximize the reward. In the case of this research, this is as defined in Model (2.1). Figure 2.12 further depicts fundamental reinforcement learning concepts as applied in this research.

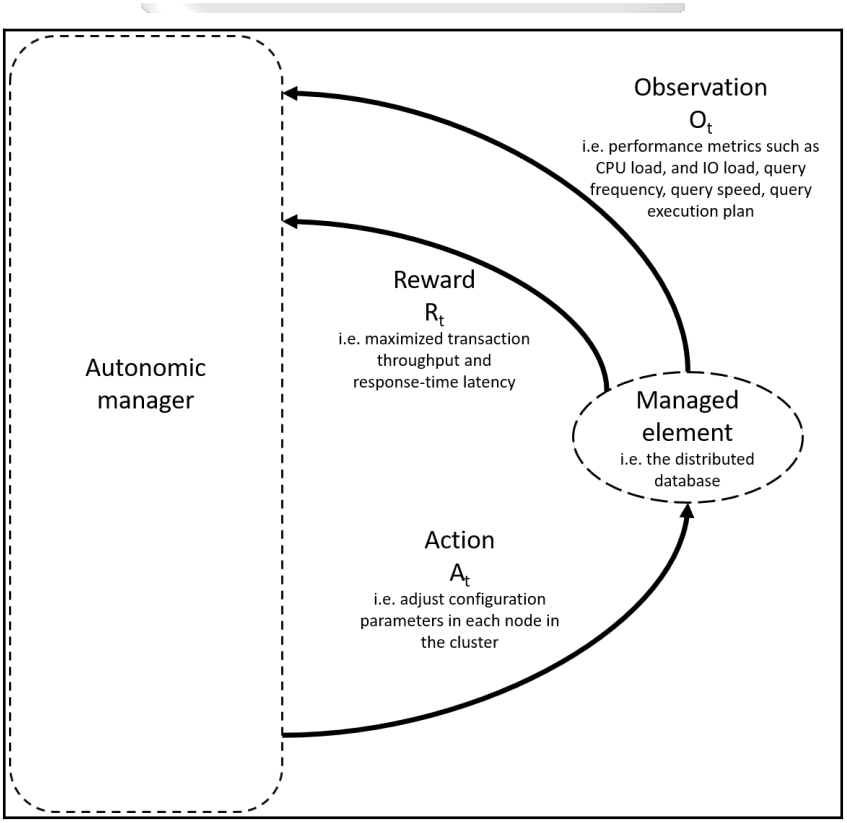

Figure 2.12: Reinforcement learning concepts applied in the autonomic manager

This results in a continuous loop that strives for constant improvement. This is directly dependent on the data that it receives in the form of observations and rewards from the managed element (the distributed database). A sequence of observation, action, and

rewards thus forms a history, H_t such that Equation (2.10) is true.

$$H_t = O_{1,} A_{1,} R_{1,} \ldots, O_t, A_t, R_t \quad \textbf{(2.10)}$$

The agent's next action then depends on H_t . This results in a mapping from the history, H_t , to the next action. This mapping can be done through the use of a reinforcement learning algorithm. However, going through the entire history can be computationally demanding. A more resourceful alternative is to consider only the current state that the distributed database is in. This state is actually a summary of the history. The current state is therefore a function of the history as shown in Equation (2.11).

$$S_t = f(H_t) \quad \textbf{(2.11)}$$

This state can in turn be in two forms: a state of the environment S_t^e , and the agent's internal representation S_t^a . S_t^e is the information that determines what should happen next from the environment's point of view. It may not always be possible for the agent to have all the information that constitutes S_t^e . This is true in the case of this research whereby the environment is made up of a stochastic distributed database system that has workloads that cannot be predicted with an acceptable level of accuracy. However, in the case of this research, it is possible for the autonomic manager (the agent) to measure work metrics (high-level observations of transaction throughput and response-time latency), resource metrics (hardware availability, utilization, saturation, and error-rate), and event metrics (occurrences that are asynchronous to the external environment). Although this allows the agent to be aware of the current state of the database system, it is problematic to create an internal representation of the future states based on a long prediction horizon.

Observations, O_t , from the environment can also be processed and a decision is made on what information to remember and what information to discard. These observations are based on the measurement of work metrics, resource metrics, and event metrics without considering their associated actions and rewards. The information is then stored in S_t^a as a summary of what has happened to the agent so far. Equation (2.12) is true in the case of a fully observable environment:

$$O_t = S_t^a = S_t^e \quad \textbf{(2.12)}$$

Either Equation (2.10) or Equation (2.11) or Equation (2.12) can then be used by reinforcement learning algorithms to determine what action to perform next in order to maximize the reward, R_t . This research ruled out Equation (2.10) and Equation (2.12) in favour of Equation (2.11).

According to the Markov chain concept, the probability of moving to the next state given the current state that the agent is in, is the same as the probability of moving to the next state given all of the previous states that the agent has been in (Su et al., 2016). This can be expressed mathematically as shown in Equation (2.13).

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \dots, S_t] \quad \textbf{(2.13)}$$

This implies that all the previous states can be discarded and only the representation of the current state considered when the agent is deciding what action to perform next. A Markov state therefore defines the future as independent of the past given the present as shown in Equation (2.14).

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty} \quad \textbf{(2.14)}$$

2.3.4.1 Reinforcement Learning Agents

Reinforcement learning agents are made up of three main components: a policy, a value, and a model. The policy defines how the agent will determine which action to perform based on the state that it is currently in. It is therefore a map from a state to an action. The policy therefore defines the behaviour of an agent. In certain environments, the current state can determine the action to be performed in order to move to a desired state. This can be modelled as a deterministic policy shown in Equation (2.15).

$$a = \pi(s) \quad \textbf{(2.15)}$$

It is also possible to have a stochastic policy such that a certain degree of randomness is considered when deciding which action to perform given a specific state. A stochastic policy distributes actions over events in the form shown in Equation (2.16). This research implemented both a deterministic policy and a stochastic policy during the design of the algorithm.

$$\pi(a|s) = P[A_t = a|S_t = s] \quad \textbf{(2.16)}$$

A reinforcement learning agent also has a value function. The value function uses the level of reward to determine which action is the best or which resulting state is the best in the long-term. It is therefore an evaluation of the goodness or the badness of states. It can subsequently be expressed as the total expected future reward in a stochastic environment as shown in Equation (2.17):

$$v_{\pi}(s)=\boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1}+\gamma R_{t+2}+\gamma^{2} R_{t+3}+...|S_t=s] \quad \textbf{(2.17)}$$

Simplified as shown in Equation (2.18):

$$E=[G(t)|S_t=s] \quad \textbf{(2.18)}$$

Such that Equation (2.19) is true.

$$G(t)=\sum_{k=0}^{\infty}\gamma^{k} R_{t+k+1} \quad \textbf{(2.19)}$$

Where G(t) is the return (goal).

The value function $v(s)$ is dependent on the policy of the agent. It is therefore indexed by pi as $v_{\pi}(s)$. The discount factor, γ , ensures that the reward at time *t* is much higher than that at time *t+1* thus giving a higher priority to immediate rewards than to future rewards that have a long prediction horizon. One of the reasons why it is important to give less priority to future rewards is because there is uncertainty in the future (Taleb & Blyth, 2011). Another reason is to avoid a summation to infinity. The value of the discount factor is designed to tend to zero. The discount factor is therefore the present value of future rewards expressed as $\gamma\in[0,1]$ such that the value of the discount factor is between 0 and 1 (exclusive). However, as this research argues, this favours exploitation of known good rewards and neglects the exploration of rewards that are not immediately good, but will lead to a good reward in the future. The algorithm designed in this research provided an enhanced method of calculating the reward value in an exploration-exploitation balance.

The third main component of a reinforcement learning agent is a model. This is a prediction of what the environment will do next and it can also be used to determine what the agent should do next. It is useful to have a reinforcement learning agent (the autonomic

manager) that has a model. The model in this case is made up of two parts: the transition probability and the rewards probability. The transition probability relates to the dynamics of the environment and predicts the next state that the environment will be in. For example, if the current time is tending towards the end of the financial year, then the transition probability can predict that certain types of workloads will be expected while getting End of Year (EOY) reports from the distributed database that supports a business information system. The rewards probability on the other hand can predict that if the agent is in a specific state and performs a specific action, then it will get a specific reward. The transition probability and the rewards probability can be depicted as shown in Equation (2.20) and Equation (2.21) respectively:

$$P_{ss'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a] \tag{2.20}$$

$$R_s^a = E[R_{t+1} | S_t = s, A_t = a] \tag{2.21}$$

2.3.4.2 Definition of the Markov Chain

The transition probability can further be defined in a state transition matrix as shown in Equation (2.22).

$$P = \begin{array}{c} \text{from} \\ \\ \\ \end{array} \begin{array}{c} \text{to} \\ \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & & \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \end{array} \tag{2.22}$$

This defines the transition probabilities from all states S_t to all successor states S_{t+1}. A Markov chain can subsequently be defined using a set of states, S, that the agent can be in and all transition probabilities that define the probability of moving from one state to the next. In addition to this, rewards and a discount factor can also be added to the definition in the form of matrices (Duan et al., 2016). This requires the value function $v_{\pi}(s)$ to be first decomposed into two parts using Bellman equation rules as follows:

$$
\begin{aligned}
v(s) &= \boldsymbol{E}[G_t | S_t = s] \\
&= \boldsymbol{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s] \\
&= \boldsymbol{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \ldots) | S_t = s] \\
&= \boldsymbol{E}[R_{t+1} + \gamma (G_{t+1}) | S_t = s] \\
&= \boldsymbol{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s]
\end{aligned}
\qquad \textit{(2.23)}
$$

The resulting Bellman equation can thus be represented as as shown in Equation (2.24).

$$v = \boldsymbol{R} + \gamma \boldsymbol{P} v \qquad \textit{(2.24)}$$

Such that it can be combined for the entire environment as shown in Equation (2.25).

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}(1) \\ \vdots \\ \boldsymbol{R}(n) \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & & \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} \qquad \textit{(2.25)}$$

This forms a Markov Reward Process (MRP). By adding actions to the MRP, the agent (autonomic manager in the case of this research) can decide what to do at each state in order to maximize the reward (Su et al., 2016). This results in a Markov Decision Process (MDP).

2.3.5 Markov Decision Processes

An MDP is made up of the state, the transition probability matrix, the reward, the discount factor, and the action. The transition probability matrix in this case is directly dependent on the action that the agent performs. It is therefore dynamic in nature because the agent can decide to perform different actions. The decision on which action to perform lies on the policy that defines the agent's behaviour.

As a result, an action-value function can be defined such that the value that an agent gets after performing action a in state s is different from the value that the agent will get if it performed action b in state s. In this case, action a and action b are each defined by policies π. That is as shown in Equation (2.26).

$$q_\pi(s, a) = \boldsymbol{E}_{\boldsymbol{\pi}}[G_t | S_t = s, A_t = a] \qquad \textit{(2.26)}$$

It is therefore necessary to find the optimal path through the system which will

define the best policy (behaviour) that the agent should have in order to get the maximum return (Duan et al., 2016). By solving this, the MDP is also solved and the agent can be loosely classified as a rational, autonomous agent in an inanimate context. The solution is represented as shown in Equation (2.27).

$$q_*(s,a) = \max_{\pi} q_\pi(s,a) \qquad \textbf{(2.27)}$$

where $\max_{\pi}$ gets the maximum possible reward while performing an action a defined by policy π. The corresponding Bellman optimality equation is therefore as shown in Equation (2.28).

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} \boldsymbol{P}_{ss'}^{a} \max_{*} q_*(s',a') \qquad \textbf{(2.28)}$$

2.4 Profile-Guided Optimization

The ability to compare the quality of automated physical design solutions remains an elusive task. One of the reasons it is elusive is because there is no standard model to come up with the cost of using a certain automated physical design. Despite this, a study by Lee et al. (2016) indicated that accurate estimates of the overall progress of execution of an SQL query are still valuable to database administrators as well as autonomic managers/agents. Currently, their main value lies in their use in determining whether a long-running, resource-intensive query should be terminated or allowed to run to completion.

The progress estimation of queries can be done after the query has finished executing. However, as Lee et al. (2016) indicated, applying online, operator-level progress information enables a database administrator to identify potential issues much more quickly. A study by Damasio, Mierzejewski, Szlichta and Zuzarte (2016) did not contradict this. It instead added on to it by indicating that the analysis of Query Execution Plans (QEPs) as well as other performance data is the most common technique applied in query progress estimation.

Four key areas of a database system that can be monitored for performance and resource utilization include: query throughput, query execution performance, number of concurrent user connections, and buffer pool usage. The first 2 areas are under work metrics, while the last 2 are under resource metrics. It is important to monitor whether the

database system is doing its work as expected or not. This involves first identifying what the expected work of a database system is. The expected work can be summarized by the Create-Read-Update-Delete (CRUD) functionality. The amount of work will naturally rise and fall, but it is worthwhile to alert on sudden changes in query volume. These include drastic drops in throughput, which can indicate a serious problem within the database system (Van Aken et al., 2017). In the case of a MariaDB Galera synchronous multi-master distributed database, a profile of how it is processing the work assigned can be measured using the work metrics outlined in Appendix F. In the event that the work metrics signal unusual symptoms, a diagnosis of the possible causes of the unusual symptoms can be performed through the use of resource metrics which are also outlined in Appendix F. Resource metrics involve the saturation, errors, utilization, and availability of relevant hardware resource. In the case of database systems, the most relevant hardware resources include primary memory and secondary storage.

An analogy to this approach is medical diagnosis (abbreviated as D_X or D_S). This involves the identification of symptoms that manifest themselves in a patient. A medical doctor would go through a systematic analysis to identify the most probable disease that explains a patient's symptoms (Hahn-Goldberg et al., 2014). However, a symptom or set of symptoms can be associated with many possible diseases. The smaller the set of symptoms, the harder it is to classify and correlate the symptoms to a particular disease. This would require the medical doctor to profile the patient in order to identify additional symptoms (Hahn-Goldberg et al., 2014). A systematic analysis at this point where the patient's symptoms have been adequately identified (profiling) can still lead to multiple possible diseases that are the cause of the symptoms. In such cases, differential diagnosis (DD_X) is performed to compare and contrast possible diseases, thus ruling out those that are not probable.

During the systematic analysis, vital organs are examined to determine whether their saturation, error-rate, utilization, and availability are at a standard level. For example, profiling of a patient can identify excessive drowsiness, unexplained shortness of breath, persistent nausea, chest pains, and swelling of ankles and feet as the set of observable symptoms. A medical doctor would then examine an internal vital organ such as the kidney to determine whether it is functioning appropriately. In a similar fashion, a database administrator would profile a database system using work metrics to identify a set of

symptoms. If these symptoms are abnormal, then the database administrator would conduct further systematic analysis to determine whether the underlying IT infrastructure (the hardware resources, referred to as "vital organs" in the analogy) that serves the database system is functioning appropriately. This involves the use of resource metrics to determine the saturation, error-rate, utilization, and availability of the resources. The database administrator would then administer an appropriate treatment to return the database system's resources back to an appropriate or optimum level of operation.

2.5 Appraisal of Optimization Techniques

The following Sections provide a review of literature on the most common optimization techniques. These are based on greedy algorithms and the branch & bound and relaxation concepts. Constraint Programming as well as Mixed Integer Programming are reviewed as techniques that guarantee high quality optimization. Local Search on the other hand is reviewed as a technique that guarantees scalability in optimization. The last Section then submits an approach that is based on probabilistic reasoning. The key advantage of its ability to model a stochastic environment is highlighted.

2.5.1 Greedy Algorithms

Greedy algorithms make a locally optimal choice with the hope that this choice will lead to a globally optimal solution. They are easy to design (for simple problems) and they can arrive at a locally optimal choice within a short period of time (Qian, Yu & Tang, 2018). However, greedy algorithms sometimes fail to find the globally optimal solution because they make commitments to certain choices too early thereby preventing them from finding the best overall solution later.

There are numerous improvements to the traditional, pure greedy algorithm. Two such improvements are the addition of the branch & bound concept and the relaxation concept (Ma & Liu, 2016). Decision-making problems involve the task of choosing "the best" amongst alternatives. Consequently, the act of choosing involves the concept of searching through numerous alternatives depending on the problem. These numerous alternatives can be organized in the form of a tree, hence the concept of a "tree search". It is possible (although computationally expensive) to conduct an exhaustive tree search in the process of finding the most optimum choice to make. However, the branch & bound

concept improves on this by applying pruning to focus only on the most promising area of the search tree (it reduces the search space). The branching splits the problem into several sub-problems while the bounding finds an optimistic estimate of the sequence of choices made.

On the other hand, the concept of relaxation involves making the problem easier to solve. It is through relaxation that a bigger portion of the search tree can be pruned before applying the branch & bound concept. The following three Sections describe further improvements to the traditional, pure greedy algorithm that apply principles of branch & bound as well as relaxation.

2.5.2 Constraint Programming

Constraint Programming is a paradigm that defines the process of optimizing an objective function with respect to some variables in the presence of constraints. These constraints are in the form of hard limits placed on the value of a variable. For example, limitations on the possible values of a hardware's configuration by stating that it cannot be above what that hardware can handle. It therefore constrains the possible values that can be assigned during the process of optimization. This can be represented graphically using a search engine and a constraint store as shown in Figure 2.13.

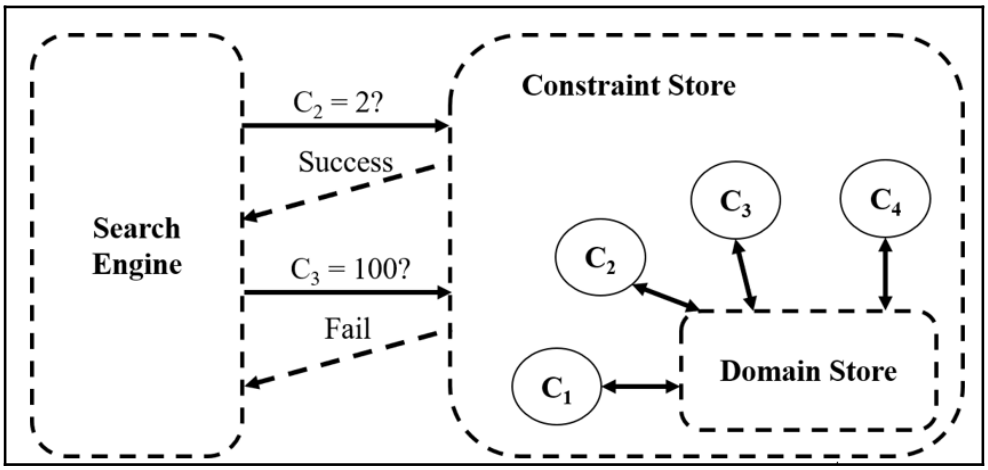

Figure 2.13: Graphical representation of constraint programming

(adapted from Omondi, Lukandu & Wanyembi, 2018)

There is continuous interaction between the search engine and the constraint store.

The search engine continuously probes the constraint store to check if the value it has found for a variable is within the limits. Given adequate time to continuously probe, Constraint Programming will find an optimal solution to an optimization problem (or conclude that there is no optimal solution). This qualifies it as a complete method and not a heuristic.

The computational paradigm of Constraint Programming is based on the concept of branching and the concept of pruning. In this case, pruning involves the use of constraints to remove values that cannot belong to any solution. This is done through the process of feasibility checking and results in the reduction of the search space (Hahn-Goldberg et al., 2014). Unlike pure branch & bound which focuses on bounding, Constraint Programming focuses on feasibility checking. This enables its key benefit to be realized, that is, its ability to capture complex, idiosyncratic constraints.

2.5.3 Mixed Integer Programming

Mixed Integer Programming (MIP) borrows several concepts from linear programming. However, unlike linear programming, MIP allows for some of the constraints to be integers. In order to create a MIP model, decision variables, constraints, and an objective function are all required. Binary values are preferred when assigning values to these variables. Similar to other greedy algorithm improvements, MIP requires good linear relaxation in order to conduct effective pruning. However, a study on chemotherapy outpatient scheduling provided evidence that showed that Constraint Programming outperforms MIP (Hahn-Goldberg et al., 2014).

2.5.4 Local Search

Local Search (LS) works with complete assignments to decision variables and continuously modifies them as it tends towards finding the optimum solution. The optimum solution in this case is defined by a local minima, that is, a position where every neighbour is worse off than the value under consideration. This is unlike Constraint Programming which works with partial assignments to constraints and continuously checks to see if these assignments can be modified. In order to accomplish this, LS starts with suboptimal (infeasible) solutions and moves towards more optimal (feasible) solutions by performing local moves. A common approach to solving LS optimization problems is

based on the max/min conflict concept as described below.

Table 2.2: An approach to solving LS optimization problems based on the max/min conflict concept

Step	Description
I	Choose the decision variable that appears the most in violations
II	Change the value in order to decrease the number of violations
III	Keep changing until the number of violations is the least (until you reach a local minima)
IV	Use the hypothesis to make predictions (deduction)

2.5.5 Probabilistic Reasoning for Decision-Making

Sullivan (2003) proposed a systematic approach to software tuning that can be applied to an arbitrary software system. This methodology was based on the use of probabilistic and decision-making techniques that have been developed by researchers in Artificial Intelligence (AI), operations research, and other related fields. One of the distinct characteristics of the methodology is the interaction with domain experts during the initial stages to determine how the variables under consideration are inter-dependent or related to their parent and to their ancestors (conditional independencies). The methodology applies the acquisition of knowledge from domain experts as well as from intuitive notions of causality regarding how changing one variable affects other variables in the environment or decision situation. The methodology also applies probabilistic reasoning modelled by influence diagrams and thus outperforms the use of regression models which do not capture elements of the decision maker's objective function (what to maximize or minimize).

Influence diagrams can be used as a compact, graphical and mathematical representation of the decision situation. Influence diagrams are also becoming a preferred alternative to traditional decision trees (Chajewska et al., 2000). This is because decision trees suffer from exponential growth in the number of branches with each variable modelled (Hansen, Shi & Khaled, 2016). In order to achieve the above function, a monitor, periodically or in real-time checks the system for any significant changes. If a significant change is detected, the monitor feeds the detected changes to the tuner. The tuner can then

use an influence diagram together with a description of the current state (workload characteristics) to determine the necessary adjustments to each of the configuration settings. Figure 2.14 shows the relationship between the system, monitor, and the tuner.

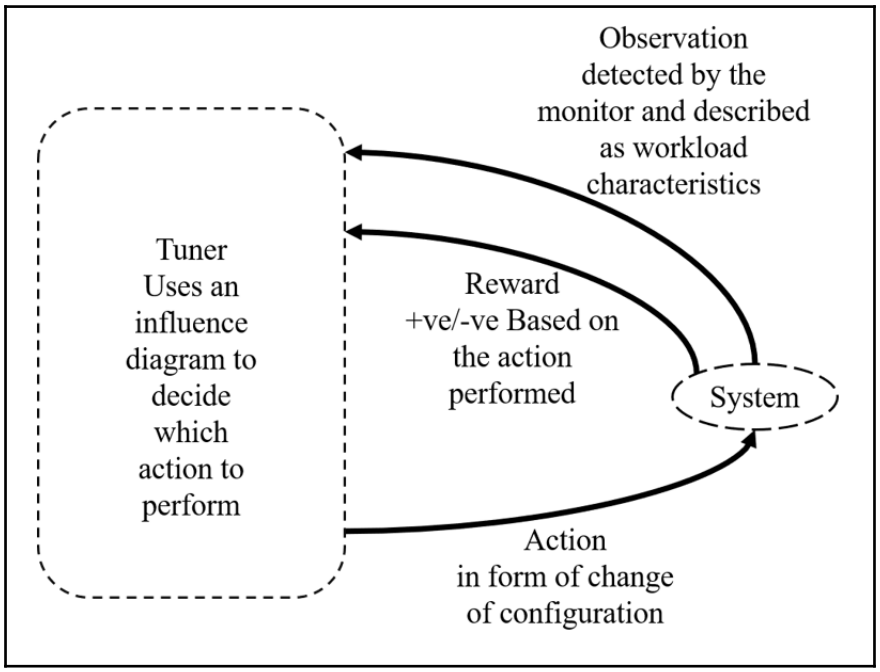

Figure 2.14: System-monitor-tuner relationship

2.6 Conceptual Framework

The literature reviewed highlights research gaps in current propositions. This contributes towards establishing the importance and need of the research founded upon the problem statement and research questions. Amalgamating concepts from various theories provided an opportunity to investigate on how to design an adaptive algorithm that proactively reconfigures bottleneck parameters without over-relying on an accurate, long-term model of a stochastic environment. The theories include the branch of model predictive control in control theory, decision theory, the branch of self-optimization in autonomic computing, and the branch of reinforcement learning in artificial intelligence. The paradigm in Figure 2.15 vividly depicts what the conceptual framework conveys.

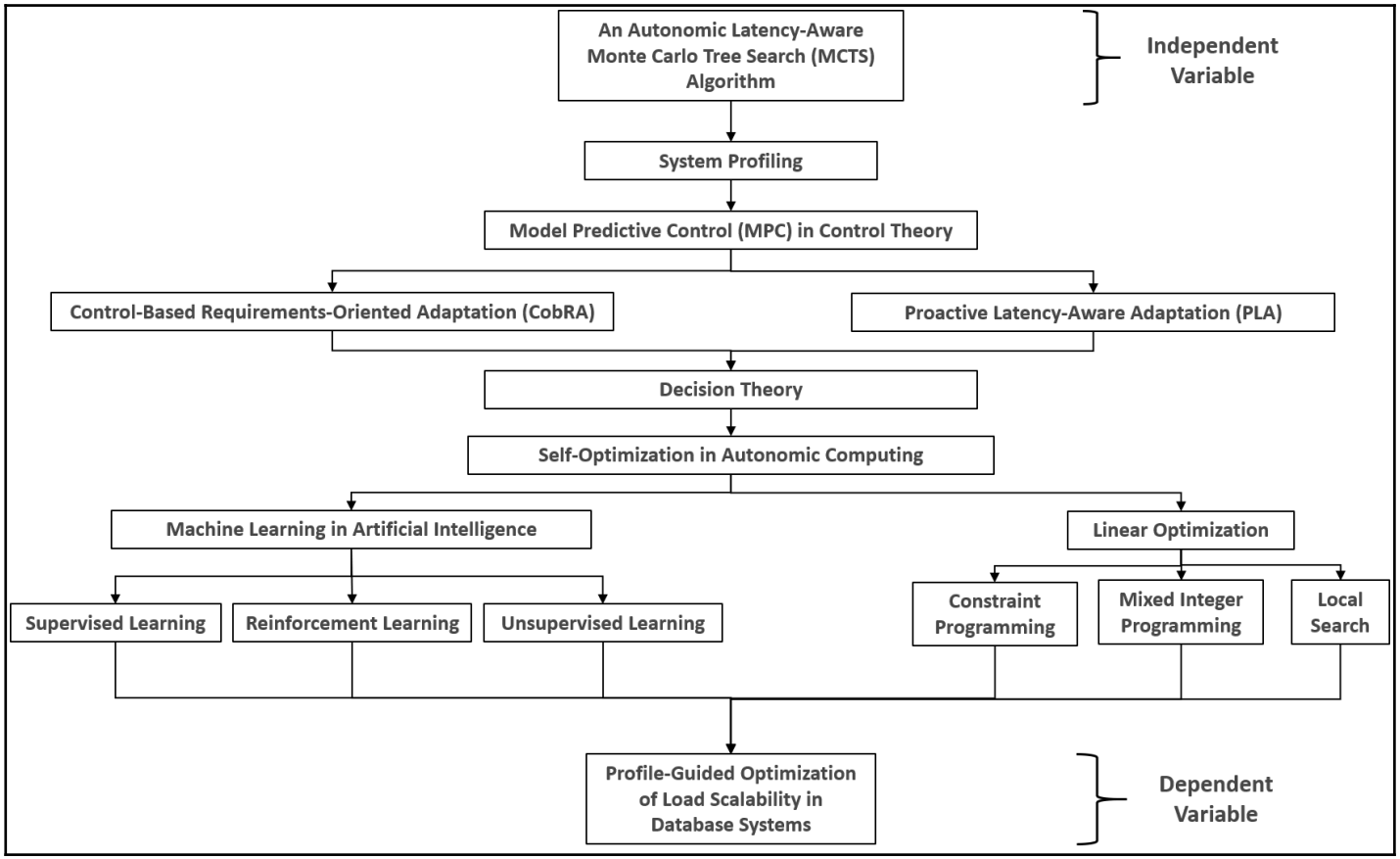

Figure 2.15: The research's paradigm

(adapted from Ameri, 2016; Chaudhuri & Narasayya, 2007)

Chapter 3: Research Methodology

3.1 Introduction

A research design aims to link the research questions and the theoretical concepts by detailing how the research will be conducted. This provides guidance for decisions concerning the process and criteria for identifying the true answers to the research questions. Based on the fact that the purpose of this research is best categorized as an experimentation, the blueprint must be one that minimizes bias and maximizes the reliability and validity of the data collected and analysed.

3.2 Philosophical Assumptions

3.2.1 Ontology

Ontology relates to the study of the categorical structure of reality. It reflects an individual's interpretation about what constitutes a fact. The assumptions made with regards to ontology predicate all other assumptions made in relation to the research methodology that was applied in this research. The following dichotomy exists and can be used to explain these assumptions: ontological materialism and ontological idealism (Lemke, 2015).

Ontological materialism involves the belief that material things are more real than immaterial phenomena; it is objective. This implies that reality exists regardless of the human observer concerned with its existence. Ontological idealism on the other hand, involves the belief that immaterial phenomena are more real than material things; it is subjective. This implies that reality is constructed in the mind of the observer. This research adopted the former assumption of ontological materialism such that the results of the research exist regardless of the researcher's intuitive perception.

The reason for choosing ontological materialism is that the research questions do not seek to understand the dynamic and subjective reality of social actors (database or system administrators) in order to make sense of their motives and actions. Instead, the research assumes that the object under discussion, that is a scalable database system, has a reality that is separate from the social actors that act upon it within the context of an enterprise, hence supporting the choice of ontological materialism.

3.2.2 Epistemology

Epistemology involves the study of knowledge in general. Questions that arise in epistemology include: What does knowledge mean? How does a person get to know something? and What is the basis for acceptable knowledge? One of the fundamental underpinnings of this research endeavour is to create new knowledge. Knowledge in this case involves a claim that has been justified as true by the knower. Epistemology therefore also forms a critical foundation for the research as it attempts to study the criteria by which an individual classifies what does and does not constitute knowledge.

There are four main categories of knowledge that can be considered (Hjørland, 2005). First, intuitive knowledge, whereby human feelings or intuition play a greater role in comparison to measured facts. Second, authoritative knowledge, which relies on information obtained from articles in reputable journals, peer-reviewed books, and domain experts. Third, logical knowledge, which involves the creation of new knowledge through the application of logical reasoning. Lastly, empirical knowledge, which relies on objective facts that have been established and can be demonstrated. This research inevitably integrates multiple categories of knowledge: intuitive knowledge in problem formulation, authoritative knowledge in literature review, the identification of the theories and the formulation of the research's paradigm, logical knowledge in hypothesis testing and data analysis which then forms the prerequisite required to produce empirical knowledge in the presentation and discussion of results.

Two main mutually exclusive branches of epistemology arise based on the categories of knowledge: empiricism and rationalism. Empiricism (also known as positivism or the positivistic epistemology), which is primarily founded on input from sensors, involves an emphasis on using observations in order to justify claims (empirical knowledge). Rationalism (also known as interpretivism or the interpretivist epistemology) on the other hand, emphasizes on reason and involves the use of ideas as the primary source for justifying claims (intuitive knowledge). Rationalism rejects the ontological materialism view that meaning resides within the world independent of consciousness.

Other branches include pragmatism and realism. Pragmatism supports the idea that there is no single point of view that can give the entire picture and that there may be multiple realities. Pragmatics are capable of combining both empiricism and rationalism within the scope of a single research. Realism on the other hand relies on the idea of

independence of reality from the human mind. This can be either direct realism, which adopts a "what you see is what you get" approach, or critical realism, which acknowledges that perception of the real world can be deceptive, thus, a multi-level perception is necessary.

Both of the two main branches (empiricism and rationalism) have clear principles, have a method, and rely on evidence; they are thus both scientific. However, the empiricism adopts more of a hard science (natural sciences) approach whereas the rationalism adopts more of a soft science (social sciences) approach (Hjørland, 2005). This research, being scientific research in Information Technology, adopted empiricism as the epistemological form. This places emphasis on applying a value-free axiology whereby the researcher is independent of the data and maintains an objective stance. This is in line with ontological materialism. The choice of positivism in this research is further justified by the fact that the research uses existing theory to develop a hypothesis which is then subjected to factual data to determine whether or not to reject it. The existing theories that were used in this research are model predictive control in control theory, decision theory, the branch of self-optimization in autonomic computing, and the branch of reinforcement learning in artificial intelligence.

A tendency towards replacing *phronesis* (knowledge derived from practice and deliberation) and *metis* (knowledge based on experience) with *episteme* (scientific knowledge) and *teche* (practical instrumental knowledge) is noted. A study by Parsons (as cited in Kitchin, 2017) pointed out that this is indeed the case with research involving the design of algorithms.

Given that materialism is adopted as the ontological form and empiricism is adopted as the epistemological form, it can be deduced that the philosophical foundation of this research is objectivism (where reality exists independent of the observer) as opposed to subjectivism (where reality is what the interpreter perceives, that is, the subject).

3.2.3 Approach

Based on the prior assumptions made with regards to the ontology, epistemology, and axiology, deductive reasoning that applies a mono-method, quantitative methodological choice was amongst the most appropriate methodological choices for this research. Deductive reasoning in this case involves developing a hypothesis based on

existing theory as the starting point to conduct an inquiry. The hypothesis is then used to form a testable proposition between the two variables concerned in the study; these were, “profile-guided optimization of load scalability in database systems”, which in turn depends on the “the autonomic latency-aware MCTS algorithm”. The hypothesis was then tested by confronting its proposition with factual data. If the propositions made based on the hypothesis are true based on the test results, then the research’s conclusion will also be true.

Inductive reasoning on the other hand starts with the collection of data followed by the development of theories as a result of the patterns identified in the data. This kind of reasoning requires an abundance of time (longitudinal research) to identify patterns in the data which lead to theory. Consequently, inductive reasoning was not an appropriate approach to this research which had a time constraint. The time constraint on this research implied a cross-sectional time horizon.

Lastly, abductive reasoning, commonly used by pragmatists, combines both deductive and inductive reasoning. It starts with surprising data or a puzzle that cannot be explained by existing theory. A back and forth progression between deductive and inductive reasoning then ensues with the aim of finding the most logical theory amongst many alternatives to explain the surprising data or puzzle. This research does not apply abductive reasoning given the complexity in its approach. Moreover, the research does not seek to describe/explain a puzzle as is the case with inductive reasoning. On the contrary, it is an applied scientific research and therefore deductive reasoning is the most appropriate approach.

A quantitative approach involves using scientific or mathematical numerical data derived from experiments, simulations, or inferential databases. Whereas a qualitative approach involves a more social methodology such as non-numerical data derived from subjective assessment of attitudes, opinions, and behaviour. This research applies a quantitative approach based on its alignment to ontological materialism, a positivistic epistemology, a value-free axiology, and deductive reasoning.

Quantitative data collection techniques were used in the research to provide data required to answer the research questions. These were based on the quantitative measurement of the results of the experiments. This results in a mono-method, quantitative choice as opposed to a multi-method or mixed-methods choice. The mono-method

quantitative choice that was applied in the case of IT-based research can further be sub-classified into either inferential, experimental, or simulation approaches. An experimental research design is characterised by much greater control over the research environment such that the independent variables can be manipulated in order to observe their effect on other variables. This was indeed the case in this research whereby different variants were used to manipulate the autonomic latency-aware algorithm in order to observe their effect on promoting profile-guided load scalability in database systems.

3.3 Research Design

This research aimed to investigate on how to design an adaptive algorithm that proactively reconfigures bottleneck parameters without over-relying on an accurate, long-term model of a stochastic environment. This was done in order to achieve self-optimization required for load scalability in database systems. Consequently, the dependent variable was "profile-guided optimization of load scalability in database systems" which depended on the independent variable that was "an autonomic latency-aware MCTS algorithm".

However, apart from an autonomic latency-aware MCTS algorithm, profile-guided optimization of load scalability in database systems can also be affected by the subjective actions of social actors such as system/database administrators. Since this was not the main focus of the research, then the extraneous variable was identified as the subjective reality of social actors on the database systems. The study was able to control this by applying an experimental approach that supported the customization of the experiment's test bed in an artificial environment.

3.3.1 Experiment Procedure

Figure 3.1 shows the 4-layer experiment infrastructure. This consists of the experiment test bed, basic services required by the experiment, orchestration of experiments, and the experiment methodology as the 4th layer.

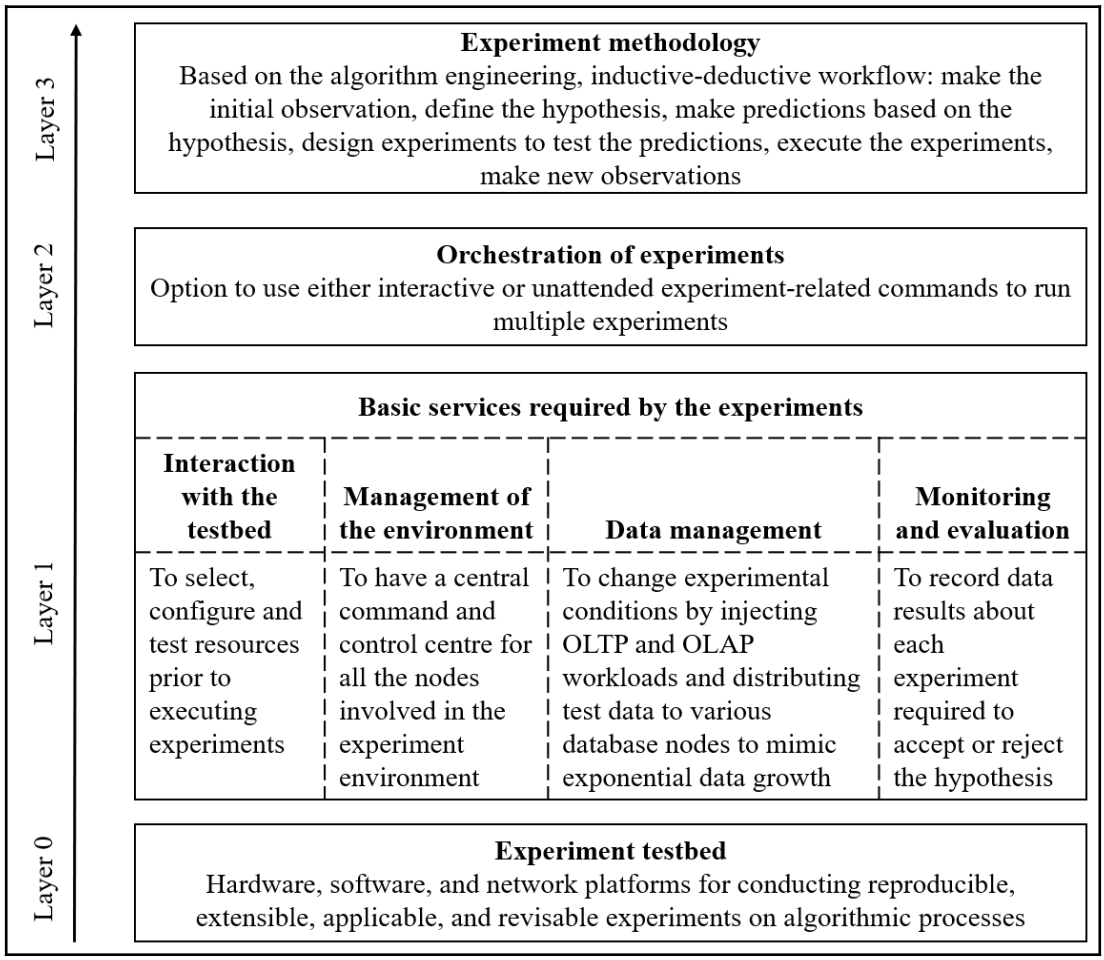

Figure 3.1: Experiment infrastructure

(adapted from Desprez et al., 2012)

The research applied a systematic and scientific approach to designing the algorithm. This was made possible through the use of experiments. The traditional approach to development of algorithms stemmed from mathematics. However, as a study by Sanders (2009) indicated, it can also be universally observed that applying traditional algorithm design methods is a slow process. This research proposes to apply a different approach that is partly based on Popper's scientific method to design the algorithm. An overview of the general scientific method is as depicted in Figure 3.2.

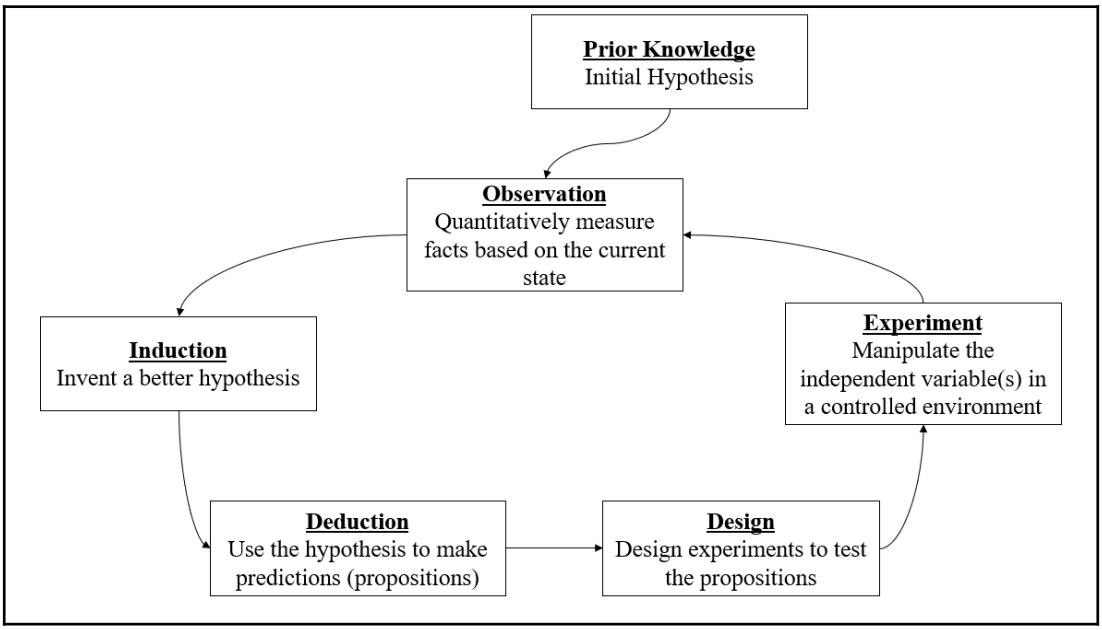

Figure 3.2: Popper's scientific method

The approach is expected to reduce the amount of time taken to derive the algorithm and as a result, reduce the gap between theory in academia and practical application of the algorithm in industry. Figure 3.3 provides a graphical representation of the proposed algorithm engineering, inductive-deductive workflow inspired by Popper's scientific method.

Research objectives (iv), (v) and (vi), that is, "to design a latency-aware algorithm that proactively reconfigures bottleneck parameters without over-relying on an accurate model of an unpredictable stochastic environment", "to perform experiments to test the designed algorithm using a game-theory based simulation of a strategy board-game", and "to apply the designed algorithm in the context of a distributed database system using a simulation of business-oriented ad-hoc queries on a test-bed" were achieved through the methods described in the following workflow.

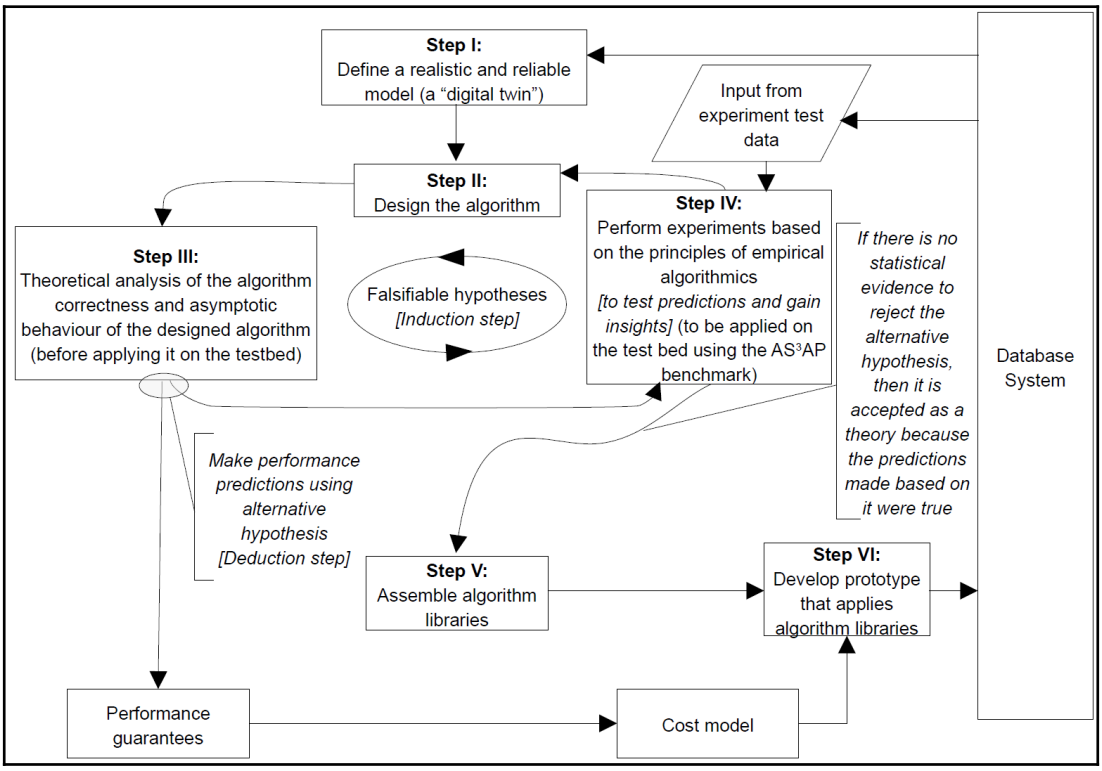

Figure 3.3: Algorithm engineering inductive-deductive workflow

(adapted from Sanders, 2009)

The experiments will subsequently be constituted of the following steps:

Step i. **Define a realistic and reliable model of the problem as well as the underlying database system (essentially a "digital twin").** In this case, the model of the underlying database system consists of work metrics that measure the amount of work the system is performing per unit time, the number of active concurrent users, and the effect that a tactic has on the transaction throughput and response time latency. This is further explained in Section 2.2.2 and Appendix B. Linear regression was used in favour of Kalman filters and ODEs. The metrics should have a ±5% precision range in comparison to the metrics found in an enterprise. This should form an adequate image of reality such that the expectation is not to have a perfect model. Section 3.5.3 further highlights the importance of setting up a realistic model with regards to ecological validity.

Step ii. **Design the algorithms** to take into consideration asymptotic behaviour, simplicity, implementability as a library in programming languages,

implementability on actual hardware, and supportability of code reuse. The code reuse was an advantage with regards to designing various flavours of the algorithm for different experiments. Principles of reflexive production of code (explained further in Appendix D) were applied. This involves the analysis of the algorithm's objective, followed by an identification of the required tasks needed to achieve the objective, and the conversion of the results of the analysis into actual pseudo-code. The pseudo-code is then eventually converted into actual code to be used in Step iv. The Perl high-level, general purpose, interpreted, dynamic programming language, in conjunction with bash, a Unix shell and command language, were used due to their ability to manipulate textual configuration files in servers.

Step iii. **Theoretically analyse the algorithm correctness and asymptotic behaviour of the designed algorithm.** The algorithm correctness was done by proving the presence of a stop property and total correctness property in the algorithm as explained in Chapter 4. On the other hand, analysis of the asymptotic behaviour of the algorithm was done by stating its running time as a function relating the input length to time complexity and space complexity. This provided an approximation of the amount of resources needed to execute the algorithm. It is however acknowledged that a full, complete analysis was difficult to achieve because of the fact that numerous dependencies need to be taken into account in such a case. The Bachmann-Landau notation (Big O notation) was used to analyse the algorithm.

Step iv. **Perform experiments based on the principles of empirical algorithmics (explained further in Appendix C) to test predictions made based on the hypothesis and to gain insights into the behaviour of the designed algorithm.** These insights serve the role of complementing the results of the theoretical analysis of the algorithm done in Step iii. The insights also serve as feedback to the prior step (Step ii.) of the experiment. This step is further broken down into a number of sub-steps in the form of a controlled experiment. The sub-steps were repeated

to obtain a sample size of 27 to increase the statistical accuracy and also to apply a distribution that can be analysed using t-scores because the standard deviation of the population was unknown. Each of the 27 experiments involved the database system processing an average of 5,300 transactions per second from 20 concurrent users. The sub-steps are as follows:

(a) Identify the experimental question

(b) Formulate a falsifiable hypothesis based on the expected behaviour of the algorithm under investigation

(c) Re-establish the causality between the independent variable (the cause) and dependent variable (the effect); all other extraneous variables held constant with the help of control variables

(d) Manipulate the independent variable (the autonomic latency-aware MCTS algorithm) using the candidate algorithm designed in Step ii. Each manipulation forms a treatment that can be applied with the aim of obtaining the desired effect (profile-guided optimization of load scalability in database systems)

(e) Establish a measurement scale to measure the results of each treatment on the experimental group. This measurement must be valid and reliable in line with the research quality. Transaction throughput and response-time latency were used as the key indicators of load scalability when the number of concurrent users was increased.

(f) Use the AS^3AP benchmark to model a gradual increase in workload based on the addition of concurrent users.

(g) Measure the state of the database system in the experimental group ($Pre\text{-}Test_{exp}$) and the database system in the control group ($Pre\text{-}Test_{control}$) before applying any treatment. They should be equal at this point. Note that the measurement of the state is conducted over time and includes a vector that

stores the average, maximum, and minimum state values.

(h) Execute the experiment, and ensure that the experimental group goes through all treatments (different variants of the autonomic latency-aware MCTS algorithm).

(i) Measure the state of the database system in the experimental group (Post-Test_{exp}) and the state of the database system in the control group ($\text{Post-Test}_{control}$) after applying the treatments. The measurement is a vector that contains the average, maximum, and minimum state values.

(j) Get the difference between the pre-test and post-test measurements for both the experimental and control groups:

$\text{Diff}_{exp} = \text{Post-Test}_{exp} - \text{Pre-Test}_{exp}$

$\text{Diff}_{control} = \text{Post-Test}_{control} - \text{Pre-Test}_{control}$

(k) Calculate the overall effect of the experiment by comparing the results of the strength of applying each of the treatments against not applying it:

$\text{Effect} = \text{Diff}_{exp} - \text{Diff}_{control}$

(l) Statistically analyse the results of the effects of applying various treatments. This analysis can then be used as a feedback to the prior steps if need arises. A chi-square (χ^2) test can be used to compare the level of variance between the treatments applied on the independent variable.

(m) Using the same treatment, and the same sample in its default state, repeat the experiment from Step iv.(a) to obtain a sample size of 27. This repetition is done to increase the statistical accuracy and thus improve the reliability of the research.

(n) At the end of the experiment, based on the hypothesis stated in Step iv.(b) , apply the decision rule using the t-statistic that will determine whether to reject or fail to reject the alternative hypothesis. If rejected, go back to Step iii. This serves a critical role in the scientific approach such that each

designed algorithm is subject to revision or even falsification using the same methodology that was used to establish it in the first place.

Step v. The algorithm engineering process continues after the core experiment with an algorithm library creation step. This step aims to **assemble the most optimum algorithms chosen through the experiments into an algorithm library**. The result, as supported by Sanders (2009), should be an efficient, generalizable, easy to use, well-documented, and portable implementation of behaviour that has a well-defined interface by which the behaviour is invoked. This is done with the aim of reducing the gap between theory and practice that is sometimes caused by the complexity involved in the theoretical research of algorithms.

Step vi. **Create a simple application that implements the designed algorithm coded in the algorithm library.** The algorithm that is part of the result of this research is derived in such a way that it can then be applied by database administrators. Due to the separation of the interface and the implementation, it is not possible to know all the applications that can use the algorithm for their benefit. However, this forms a critical justification for undertaking this research as mentioned in Section 1.6.

3.3.2 Experiment Test Data

The American National Standards Institute (ANSI) Structured Query Language (SQL) Standard Scalable and Portable (AS^3AP) benchmark is designed to compare relational database systems with vastly different architectures and capabilities over a variety of workloads. AS^3AP is capable of defining a runtime ordering of the queries in the workload to prevent the data of one query from being memory resident as a consequence of the previous query. This avoids lengthy operations that would otherwise be needed to flush the buffers. It consists of single-user tests and multi-user tests. The single-user AS^3AP workloads focus on basic functions that a relational DBMS must support. These include:

(i) Utilities for loading and structuring the database, building clustered and secondary indices, checking for referential integrity and performing backups

(ii) User queries that include selections, projections and sorting, joins (theta joins,

natural joins, outer joins, and semi-joins), aggregation and grouping operations, complex relational divisions, join-aggregates, and recursive queries, single-tuple updates, and bulk updates

On the other hand, the multi-user AS^{3}AP workloads focus on establishing the maximum throughput for OLTP transactions and measuring degradation in response-time latency for OLAP queries. Both of these measurements are taken as a function of the workload profile (response-time latency for read-intensive workloads or transaction throughput for write-intensive workloads), the quantity of data accessed, the system's compute-overhead caused by the algorithm, and the number of concurrent users. Consequently, multi-user AS^{3}AP workloads include:

(i) Mixed OLTP and OLAP workloads that include a balance of write-intensive transactions (*oltp_update* with Level 3 isolation) as well as read-intensive analytical queries (*ir_select* with Level 0 isolation).

The justification for applying the AS^{3}AP benchmark is its combination of OLTP and OLAP workloads in a single experiment. This is unlike TPC-E and TPC-H which separate OLTP and OLAP workloads respectively. This separation is not always ideal given the presence of business applications that are a hybrid of OLTP and OLAP.

In order to simulate real-world user interactions, a latency of 1 second of think-time was added. Think-time is used to simulate the amount of time required "to think" about the results of a previous transaction. In addition to this, the time phase was divided into pre-sampling time and sampling time. The pre-sampling time is the length of time the virtual users continuously send workloads to the database system in order to reach a steady state before statistics are collected. While sampling time refers to the length of time to collect statistics during the continuous sending of workloads to the database system. The research used 1/3 of the total experimentation time as pre-sampling time and the remaining 2/3 for sampling time. Lastly, the virtual users were added continuously at a rate of 1 virtual user every 2 seconds. The tool used to orchestrate the experiment was Benchmark Factory version 8.1 which, together with the test bed's hardware capabilities, limited the maximum number of concurrent virtual users to 20. However, this limitation did not reduce the ecological validity of the test bed to model a small to medium -size enterprise because of the tool's ability to orchestrate intensive workload submitted simultaneously by 20 virtual users.

3.3.3 Experiment Test Bed

Numerous test executions environments configured for testing exist, for example, Grid'5000, Open Cirrus, Planet Lab, Future Grid, Distem, ModelNet, SimGRID, GridSIM, and Linpack. However, these publicly available test beds face significant challenges. One such challenge is ineffective planning for resource usage amongst testing teams. This leads to unstable results because running a test case in the same test scenario may produce different results if the shared resources have not been properly sandboxed, for example fluctuating shared network bandwidth at the backbone. Another significant challenge is working with remote environments. This leads to heavy reliance on the test bed's support team in cases where the remote node requires a firmware upgrade or a build upgrade or any other physical support. This causes considerable delays in the testing schedule.

For these reasons, this research created its own test bed such that the researcher maintained absolute authority over the experiments and their environment. Table 3.1 below specifies the details of the test bed.

Table 3.1: Technical specifications of the experiment test bed

Hardware	Bare Metal	• Processor: Intel® Core™ i7-6700HQ CPU @ 2.60GHz × 8 • Memory: 16GB • Storage: 931.51GB
	3 Distributed database Virtual Machines	• Processor: Intel® Core™ i7-6700HQ CPU @ 2.60GHz × 2 • Memory: 3GB • Storage: 50GB
	1 Load balancer virtual machine	• Processor: Intel® Core™ i7-6700HQ CPU @ 2.60GHz × 1 • Memory: 2GB • Storage: 50GB
	Secondary Storage	• Manufacturer: Standard, fixed, internal hard disk drive • Model: ScanDisk SD8SNAT – 128G – 1006 • Size: 931.51 GB

		• Total cylinders: 15,566 • Total sectors: 250,067,790 • Total tracks: 3,969,330 • Bytes/Sector: 512 • Sectors/Track: 63 • Tracks/Cylinder: 255
Software	Operating Systems	• Bare metal: Microsoft Windows – 64-bit Windows 10 Pro • Virtual machines: 64-bit Ubuntu Server 16.04 LTS
	Distributed Database Management Systems (DDBMSs)	• MariaDB 10.2.14 Galera synchronous multi-master Distributed Database Management System • Load balancer based on a "least connections" balancing solution using HAProxy software
	Hypervisor	• Oracle VirtualBox version 5.2.24 r128163 (Qt5.6.2)
	Virtual Machines (VMs)	• Shared-nothing architecture (each node had its own CPU, memory, and storage) • Total VMs were 4: 3 master nodes (with no slaves) that form the distributed database and 1 load balancer
Network	Fast Ethernet (100Mbps), dual band Wireless-AC 3165	

The experiment test-bed was made up of a distributed database with Maria DB 10.2.14 installed as the Distributed Database Management System (DDBMS). There were 3 nodes in the cluster, each configured as a master with no slaves and there was synchronous replication between all the 3 nodes. The synchronous replication was made possible through the use of the Write-Set REPlication (WSREP) Application Programming Interface (API). The WSREP API implements an eager replication whereby nodes in the cluster synchronize their states (database content) with all other nodes by updating the replicas through a single transaction. A load balancer based on a least connections balancing solution was also configured. The least connections balancing solution worked by forwarding connections to the server with the least number of connections. The distributed system was based on a shared-nothing architecture such that each of the 3 nodes had their own CPU and storage as Virtual Machines (VMs). All the 3 nodes plus the load

balancer were running a 64-bit Ubuntu Server 16.04 LTS as the Operating System. Figure 3.4 shows the architecture of the test-bed.

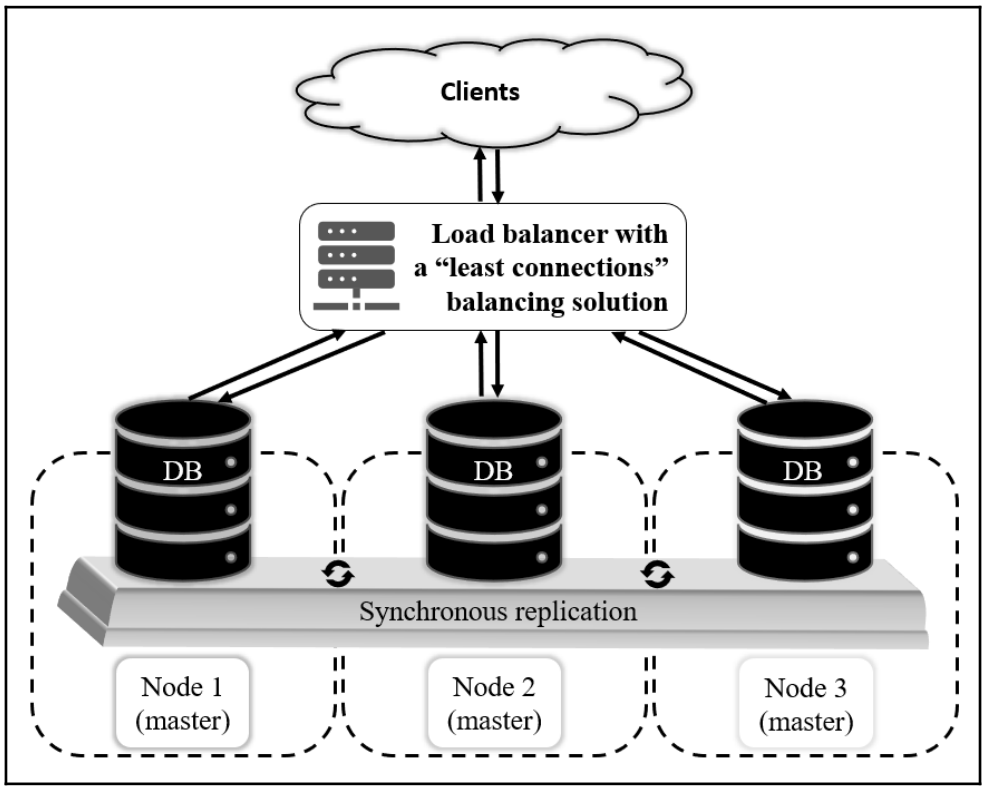

Figure 3.4: Architecture of the experiment test bed

The test-bed aimed to model a real-world environment whereby the normal architecture is that of a distributed database for the sake of High Availability/Disaster Recovery (HA/DR) features. This was done in order guarantee the ecological validity of the research. Ecological validity subsequently contributes towards generalizability of the results of the study to a population as part of external validity.

3.4 Data Analysis Methods

3.4.1 Algorithm Appraisal Methods

In order to achieve research objective (iii) stated in Section 1.4, this research applied reflexive production of code. Reflexive production of code involves the analysis of the algorithm's objective, followed by an identification of the required tasks needed to achieve the objective. Once the specific, actionable tasks are identified, they are converted into pseudo-code and subsequently into actual code using a programming language of choice. Appendix C provides further explanation of algorithm appraisal methods under the

discipline of empirical algorithmics.

3.4.2 Null Hypothesis and Alternative Hypothesis

The research submitted the following composite hypothesis as a starting point for performing a relevant investigation:

Null hypothesis (H_0): Distributed database systems that apply the designed autonomic latency-aware algorithm on average have the same transaction throughput and response-time latency.

Alternative hypothesis (H_1): Distributed database systems that apply the designed autonomic latency-aware algorithm on average have a faster transaction throughput.

Alternative hypothesis (H_2): Distributed database systems that apply the designed autonomic latency-aware algorithm on average have a slower response-time latency.

$$H_0: \mu = \mu_{H_0}$$

$$H_1: \mu > \mu_{H_0}$$

$$H_2: \mu < \mu_{H_0}$$

The following sections describe the data analysis techniques applied based on the stated composite hypothesis.

3.4.3 Level of Significance

This research desires to reject (disprove) the null hypothesis when it is proved to be false. The risk of rejecting the null hypothesis when it should be accepted as true is referred to as α (the level of significance), also known as a Type I error. If α = 5% then the probability of rejecting H_0 when it is true is 0.05. This means that the research is willing to take a maximum risk of 5% for rejecting the null hypothesis when it is true. Represented as shown in Equation (3.1) when testing the 2nd alternative hypothesis.

$$P(\mu < \mu_{H_0} \mid H_0 \text{ is true}) < 0.05 \qquad \textbf{(3.1)}$$

On the other hand, a Type II error implies failing to reject the H_0 when it should have been rejected. This is denoted by β. By reducing a Type I error, the probability of

committing a Type II error increases. However, there is not much of a difference between the penalties involved in obtaining a Type I error versus obtaining a Type II error. Both are negative.

3.4.4 Decision Rule

The criterion that was applied to reject or fail to reject the alternative hypothesis was that if out of all the experiments executed for each treatment or variant of the algorithm, 95% result in a faster transaction throughput and a slower query response time, then the alternative hypothesis will not be rejected. This does not necessarily imply that the alternative hypothesis is true, however it implies that there would be no statistical evidence to reject it. However, given that this is not adequate for hypothesis testing, further analysis is required as explained in Section 3.4.5.

3.4.5 One-Tailed Test

A one-tailed t-test is a technique used to compute the statistical significance of a parameter inferred from the results of an experiment. This is done in form of a test statistic. A one-tailed test (right-tail for testing the 1st alternative hypothesis and left-tail for testing the 2nd alternative hypothesis) involving a T-score was used to measure the level of difference between the results and what was expected. The T-score supports the transformation of an individual score into a standardized form for easier comparison. The greater the difference from the expected T-score, the more evidence there is that the results of an experiment are significantly different from the average expected results. Given that the null hypothesis represents the expected results, then the null hypothesis cannot be true when the actual results are different from the expected results. The decision rule can therefore be extended to state, by using a one-tailed test with a significance level of 5% and a sample size of 27 experiment results:

(i) Reject H_0 if $T_{score_{calculated}} > T_{score_{tabular}}$ in the case of the 1st alternative hypothesis, H_1 and

(ii) Reject H_0 if $T_{score_{calculated}} < T_{score_{tabular}}$ in the case of the 2nd alternative hypothesis, H_2

3.4.6 Sampling Distribution

This research applied the student's T distribution in the calculation of the T score used in hypothesis testing. This was based on the fact that the standard deviation of the population of database systems was unknown. Furthermore, the sample size of 27 was less than 30 and thus considered a small sample size. The formula used to calculate the t-score was:

$$t = \frac{\bar{x} - \mu}{\sigma_s / \sqrt{n}} \quad \text{with } d.f. = (n\text{-}1) \qquad (3.2)$$

and

$$\sigma_s = \sqrt{\frac{\sum (X_i - \bar{X})^2}{(n-1)}}$$

Where:

$\bar{x}$ = the sample mean

μ = a representation of the population mean

σ_s = the sample standard deviation

n = the sample size ($n < 30$)

d.f. = the degrees of freedom

3.5 Research Quality Methods

Reliability and validity are rooted in the positivistic epistemology. They are both vital tools used in establishing truth hence being fundamental cornerstones of the scientific method. Reliability demands two main requirements: an accurate representation of the total population under study and stable results or measurements that can be reproduced by applying the same methodological approach. Validity on the other hand determines whether the research truly measures what it is supposed to measure as well as whether the means of measurement are accurate. This is further divided into internal validity and external validity. Reliability is therefore a critical ingredient for determining the overall validity of a scientific experiment.

3.5.1 Reliability

It is desired to obtain significant results that are not a one-off finding but are instead

inherently repeatable. This research therefore applied the experimental design in such a way that the application of treatments was simulated 1,000 times in each of the 27 experiments. This increased the statistical accuracy and subsequently the reliability of the research. This then formed an objective pre-requisite to the research's hypothesis establishing itself as an accepted, scientific truth.

The research also used the AS^3AP benchmark. This modelled a real-world environment that guarantees the reliability of the research. It subsequently justified the use of a simulation to conduct the experiment as shown in Figure 3.5. Hypothesis testing using t-tests and Chi-square (χ^2) tests were then used to measure how significantly different an outcome or a category/group of outcomes was from its expected value.

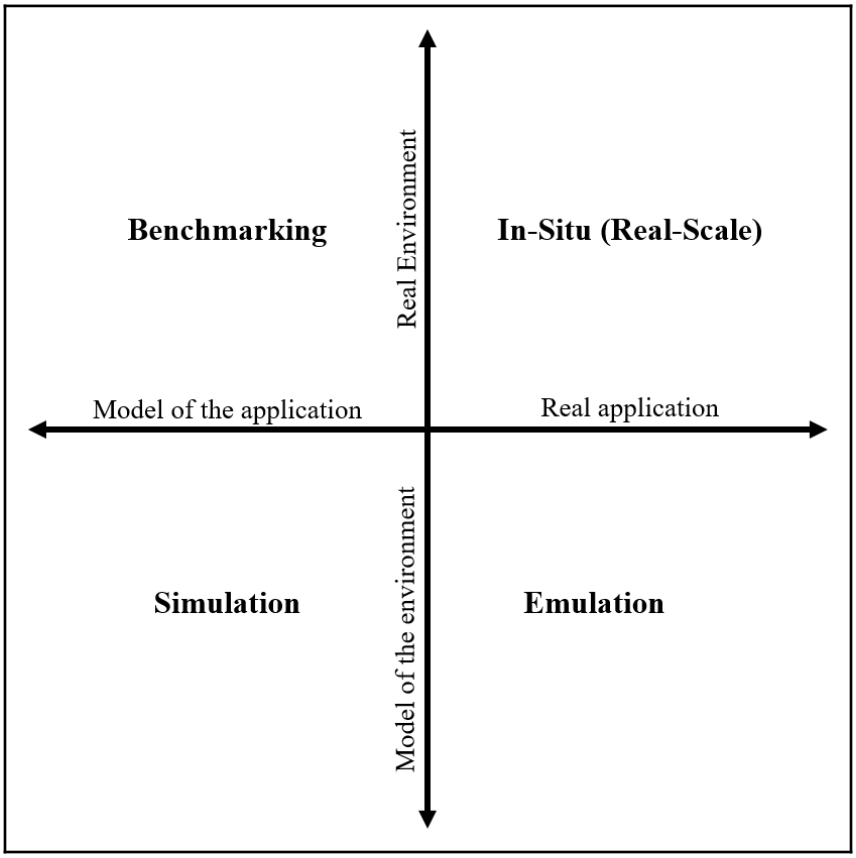

Figure 3.5: Experimental Computer Science

(adapted from Desprez et al., 2012)

3.5.2 Internal Validity

When applied to an experimental research design, internal validity refers to how confidently one can conclude that the change in the dependent variable was produced

solely by the independent variable and not by extraneous variables. It encompasses all of the steps of the scientific research method by dictating how the experimental design should be structured. The key question that internal validity would address is “Could there be other variables, apart from the designed autonomic latency-aware MCTS algorithm, that could cause the optimization of load scalability in database systems?” As expected, attaining internal validity can indeed be a complex task.

The study conducted a thorough literature review in order to increase the internal validity. By doing so, a temporal precedence was established. Studying previous research, as expected, helped to determine which of the two variables (cause and effect) comes first. The expectation is that a scalable storage server is directly influenced by an autonomic latency-aware algorithm.

3.5.3 External Validity

The external validity refers to the extent to which the results of this research can be generalized to other settings. External validity can be further divided into two: ecological validity and population validity. Ecological validity focuses on how the artificial environment in which the experiments have been conducted influences the generalizability of the results of the research to a population. On the other hand, population validity focuses on how the sample used influences the generalizability of the results of the research to a population.

The risk of conducting the experiments in a live environment in the real-world is significantly high because of the possibility of the algorithm changing the wrong parameters during its scientific design stage. A model of the environment therefore had to be created for the experiments to be conducted in. According to ecological validity, the results of these experiments should be generalizable to the actual environment in the real-world. It was therefore critical for the experiments to resemble the real world situation as much as possible.

Multiple treatments are also capable of causing limited generalizability. However, this research enjoys the lack of restriction caused by using an inanimate subject (a database system). Even though the subject was taken through a number of successive treatments in experiments, this research reset the subject to its initial state before taking it through another treatment. This implied that the effect of conducting treatment B after treatment A

(on the same subject) was not affected by the results of treatment A. Thus increasing the validity of the research.

3.6 Ethical Considerations

True data is the catalyst through which progress towards truth can be made. This is in line with the positivistic epistemology whereby the researcher gets to know things through sense experience in experiments rather than through intrinsic interpretations. It is therefore paramount that the recording and analysis of the results of an experiment is done accurately and truthfully. This should be such that fabrication, falsification, and plagiarism of data are not engaged in at all costs. In order to achieve this, the pre-determined outline provided in Chapter 3 was applied.

The results stand to not only benefit SMEs in administration of their database systems for improved acquisition of data from their databases, but to also benefit the society at large in numerous ways. Social responsibility was therefore taken into consideration. This was such that the results can be applied to automatically maintain an optimum performance in database systems that support not only transaction processing systems, but also analytical processing systems. These include, but are not limited to:

(i) Analytical processing systems used to analyse agricultural data for improved efficiency in farming. This can benefit organizations such as the Kenya Agricultural and Livestock Research Organization (KALRO)

(ii) Analytical processing systems used to analyse transaction records for fraud prediction, and surveillance data for security. This can benefit organizations such as the Kenyan National Police Service and the Kenyan Directorate of Criminal Investigations' Banking Fraud Investigation Unit.

(iii) Analytical processing systems used to analyse meteorological data for weather forecasting to benefit the Kenya Meteorological Department. This in turn benefits farmers in the agricultural sector which is the largest contributor to Kenya's Gross Domestic Product (GDP).

(iv) Analytical processing systems used to analyse resource utilization data for energy optimization in developing countries. Organizations such as the Kenya Power and the Nairobi City Water and Sewerage Company can benefit from this.

(v) Analytical processing systems used to analyse pharmaceutical data for drug discovery. This can benefit pharmaceutical companies regulated by the Kenya Pharmacy and Poisons Board.

(vi) Analytical processing systems in businesses used for financial modelling (budgeting and planning), sales forecasting, and market share analysis. This can benefit businesses seeking to compete effectively by making data-driven decisions.

The separation of interface and implementation makes it difficult to know all possible applications of an algorithm. This is because numerous different interfaces can be developed to access the implementation of the algorithm to be derived. However, the running theme common across all possible implementations of the algorithm is the impact it can have in promoting improved performance when acquiring data from databases even as the volume of data scales up.

Chapter 4: Algorithm Design

4.1 Introduction

Shopping online, travelling, saving lives, online dating, preparing a meal and many other activities in our modern world involve a series of step-by-step instructions which guide us in solving problems. These step-by-step instructions are called algorithms, and they have become central to our daily lives as human beings. A significant number of algorithms, both modern and ancient, tap into the mathematical order that governs our world.

Designing these algorithms requires a considerable amount of effort towards creativity. This is unlike using them which is a matter of following instructions, hence the reason why algorithms are ideal for computers. Computers are capable of performing repetitive, unambiguously defined tasks at phenomenal speeds. The beauty of using and designing algorithms is the opportunity they provide to aspire for solutions that are elegant and as efficient as possible.

This Chapter applies the algorithm engineering inductive-deductive workflow described in Section 3.3.1. It starts by explaining the importance of feature selection and its use towards reducing the curse of dimensionality in Section 4.2. This is followed by Section 4.3 which proceeds to categorize the features selected into tactics and then uses an influence diagram to represent the complexity of the relationships between the features. Section 4.4 then provides a detailed, step-by-step description of how the algorithm identifies the optimum tactic to implement in order to move the database system towards a desired state. Section 4.5 provides a proof of the algorithm's correctness using the stop property and total correctness property. It also states the asymptotic behaviour of the algorithm based on time and space complexity. Lastly, Section 4.6 provides a summary of the key points of this Chapter.

4.2 Feature Selection

Handling thousands of possible combination of values of configuration parameters inevitably leads to the curse of dimensionality. This phenomenon creates sparsity in the data which in turn is a problem when it comes to obtaining statistical significance (Bach, 2017). The sparsity is caused by the dramatic increase in size of the space of possible

combination of variables in different dimensions. Bach (2017) argued that exponentially many observations are needed to obtain optimal generalization performances in such cases of sparsity. An exhaustive search of the space is therefore required yet this exhaustive search is computationally intractable. To avoid the curse of dimensionality in this study, some configuration parameters or features that were either non-essential or redundant, were discarded without negatively impacting the algorithm. This was done through feature selection which enhances generalization of the algorithm by reducing over fitting and simultaneously strives to get the most outcome with the fewest number of variables. Unlike feature extraction which creates new features from functions of the original feature set, feature selection returns a subset of features from the original feature set. In addition to this, the current study also made use of tactics, which were defined as categorical groups of configuration parameter variables and their values according to general database administration best-practices.

The three common methods for performing feature selection are: wrapper methods, filter methods, and embedded methods (Ishibashi, Iwasaki, Otomasa & Yada, 2016). Filter methods, although considered a pre-processing step, select features on the basis of their score in statistical tests that measure the level of correlation between the feature and the dependent variable. The statistical tests to measure the level of correlation that can be used in filter methods include Pearson's Correlation Coefficient (PCC), Linear Discriminant Analysis (LDA), Analysis of Variance (ANOVA), and the Chi-Square (χ^2) test among other similar tests. Table 4.1 shows the most appropriate statistical test depending on whether the feature or its response is continuous or categorical.

Table 4.1: Statistical tests for feature selection in filter methods

	Response (dependent variable)	
Feature	**Continuous**	**Categorical**
Continuous	Pearson's Correlation Coefficient (PCC)	Linear Discriminant Analysis (LDA)
Categorical	Analysis of Variance (ANOVA)	Chi-Square (χ^2) Test

Wrapper methods on the other hand are based on a continuous iterative search problem whereby the results of the previous search generate a subset of features which is

subsequently searched to generate an even smaller or more significant subset of features. This iteration continues until an ideal subset is obtained. Wrapper methods can use either feed-forward selection, backward selection, or recursive feature elimination techniques. This study applied a wrapper method based on a feed-forward selection technique. The feed-forward selection technique starts with an empty set of features. It continues adding the features that best improve the model of the environment's behaviour until a point when additional features do not have any significant impact on the model. Stepwise regression was used to implement the feed-forward selection technique in the wrapper feature selection method. The criterion, as informed by literature, used to decide which feature was retained in the model upon each feed-forward iteration of the stepwise regression was that the feature's β-coefficient should have the highest absolute t-score (Ishibashi et al., 2016; Sadatrasoul, Gholamian & Shahanaghi, 2015). The initial set of configuration parameters was obtained from the review of literature by domain experts. This was then reduced to a set of 5 key features. Appendix E provides the raw data of the stepwise regression runs that resulted in the selection of 5 features (configuration parameters).

As explained in Section 2.5.5, influence diagrams can be used to represent the intertwined complexity of a decision situation. This is unlike traditional decision trees which suffer from exponential growth in the number of branches caused by multiple variables (Chajewska et al., 2000; Hansen et al., 2016). Figure 4.1 thus graphically depicts the influence diagram of features identified by domain experts and the subset of those selected using stepwise regression. In addition to this, Appendix G provides a detailed description of each of the 15 parameters. These results were obtained from conducting a total of 81 experiment. Each of the 81 experiments involved the database system processing an average of 5,300 transactions per second from 20 concurrent users. The 81 experiments were divided into 3 categories based on the 3 tactics applied and measured for OLTP workloads and OLAP workloads.

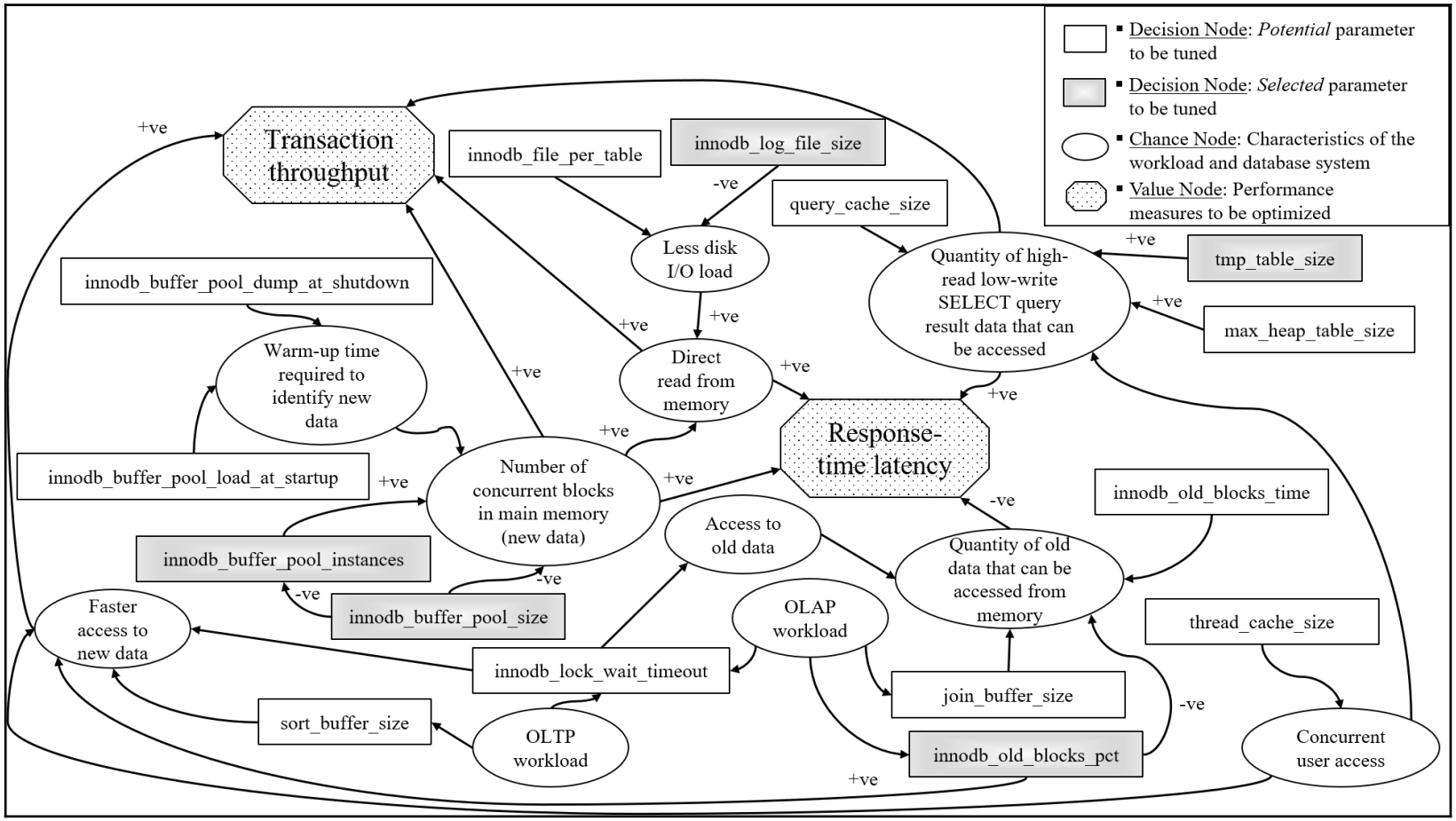

Figure 4.1: Influence diagram of features (configuration parameters) selected to be tuned

On the other hand, the backward selection technique of the wrapper method starts with a set of all possible features (Chajewska et al., 2000). It iteratively eliminates the least significant features that do not improve the model of the environment's behaviour until a point when no removal of features has any significant impact on the model.

Lastly, the recursive feature elimination technique of the wrapper method repeatedly creates models of the environment and keeps track of each model's value. Each creation of a model involves the use of leftover features from the previous creation of a model (Ishibashi et al., 2016). The features in each model are also simultaneously ranked based on their order of elimination. The recursive feature elimination then applies a greedy algorithm to select the features with the highest values (the feature values are based on their order of elimination).

Another method for performing feature selection is the use of embedded methods. Embedded methods combine the qualities of filter methods and wrapper methods. Algorithms that have their own built-in feature selection methods implement embedded methods, hence the name embedded (Chajewska et al., 2000). Examples of embedded methods are regularized trees, memetic algorithms, random multinomial logit algorithms, amongst others.

4.3 Configuration Tactics and Recommendations

Working with sets of parameters, as opposed to individual parameters, supports the construction of decision trees used by algorithms required to make complex decisions. Given the numerous and diverse configuration values of parameters, even after performing feature selection and obtaining 5 parameters to focus on, the study categorized the parameters into sets referred to as tactics. Implementing one tactic, for example the tactic "medium", therefore involved the simultaneous implementation of numerous configuration parameter values that belong to the same tactic. The tactics were categorized into low, medium and high based on database administrator best-practices as detailed in Table 4.2.

Table 4.2: Categorization of parameter values into tactics

Key Parameter	Range of Values			Notes on Database Administrator Best-Practices
	Low	**Medium**	**High**	
innodb_old_blocks_pct (x_3)	95%	50%	5%	The default is 37%
tmp_table_size (x_12) = max_heap_table_size (x_13)	64M	128M	192M	$=64\,M \times RAM_{\text{size}}(Gigabytes)$ The default is 64M
innodb_buffer_pool_instances (x_2)	64 instances	33 instances	2 instances	$=\frac{\text{innodb\_buffer\_pool\_size}(Megabytes)}{1024\,M}$ The default is 64 instances. It should divide the buffer pool size into instances of at least 1GB each. The default **innodb_buffer_pool_size** should be changed from 1504M to **80% of RAM** (2458M in this case) to support this.
innodb_buffer_pool_size (x_1)	1504M	1981M	2458M	$=(70..80)\%\times RAM_{\text{size}}(M)$ The default is 1504M
innodb_log_file_size (x_8)	376M	496M	615M	$=25\%\times \text{innodb\_buffer\_pool\_size}$ The default is 512M

The 3 most relevant features for low-read high-write OLTP workloads according to the feature selection results were:

(i) **innodb_old_blocks_pct** (x_3): It specifies the maximum percentage of the buffer pool dedicated to storing "old" blocks of data and indexes that were frequently used in the past. The value can range between 5% and 95% of the entire buffer pool. A smaller percentage value enables faster eviction of less frequently used old blocks of data and indexes from the buffer pool. This eviction in turn creates room for more frequently used "new" blocks of data to be stored in the buffer pool. It subsequently reduces the bottleneck caused by disk IO and has a direct impact on transaction throughput for OLTP workloads. This parameter works directly at the resource level focusing on primary memory.

(ii) **tmp_table_size** (x_12): It specifies the size allocated for temporary tables. Temporary tables are used when processing complex queries that involve joins and sorting. The size allocated to tmp_table_size must be the same as the size

allocated to **max_heap_table_size**. The results showed that this parameter has a direct impact on transaction throughput for OLTP workloads. The results of the stepwise regression were contrary to the expectation that the parameter has a direct impact on response time for OLAP workloads. This parameter works directly at the resource level focusing on primary memory.

(iii) **innodb_buffer_pool_instances** (x_2): It divides the buffer pool into equal-sized instances each of which manages its own data. It subsequently reduces concurrency problems caused by a shared buffer pool. According to the study's stepwise regression results, it has a direct impact on the transaction throughput of high-write low-read, concurrent OLTP workloads. This parameter works directly at the resource level focusing on primary memory.

Whereas, the 2 most relevant features for high-read low-write OLAP workloads according to the feature selection results were:

(i) **innodb_buffer_pool_size** (x_1): It specifies the amount of RAM that should be set aside to store frequently used blocks of data and indexes. The results showed that it has a significant impact on the response time of high-read OLAP workloads. This parameter works directly at the resource level focusing on primary memory.

(ii) **innodb_log_file_size** (x_8): It reduces the disk I/O because of less flushing of checkpoint activity. It also increases the speed of database writes and the durability of transactions. According to the stepwise regression results, this parameter has a direct impact on the response-time of OLAP workloads. This parameter also works directly at the resource level focusing on primary memory.

A more detailed description of each of the 5 parameters is provided in Appendix G.

4.4 Algorithm Design

The study quantitatively defined the current state of the database by measuring work metrics such as transaction throughput, response-time latency per schema, and number of concurrent users. Given the definition of the current state, it is necessary to determine the action to perform that will lead to a subsequent state that has the highest possible transaction throughput and the lowest possible response-time latency. This is true even though the state with the highest possible value is not the immediate next state. The

challenge is that this action is not known, given that there are variable numerous combinations of environmental conditions and runtime phenomena.

Monte Carlo simulations enable us to estimate the value of subsequent states through numerous random simulations. This enables us to identify the action that, if performed, will lead to the state with the highest possible value. Past research has demonstrated that combining the generality of random simulations and the precision of a tree search in the form of a Monte Carlo Tree Search, has a significantly positive effect on the effectiveness of an autonomic agent (Gelly, Wang, Munos & Teytaud, 2006; Hartmann, 2017). The following Sections describe the stages of the Monte Carlo Tree Search with the variants that the research implemented. The term "node" in the following Sections is used to refer to the state of the database system.

4.4.1 Variated Selection Stage

The selection stage selects a node according to a predefined utility function. The selection continues through the search tree until it encounters a node that has not been fully expanded to allow its children to be explored. In the context of this research, this means that the selection stage proceeds until it reaches the state of a database system whose subsequent states are still unknown. Recent studies have shown that the default MCTS selection stage faces the challenge of focusing too much on the most promising nodes while neglecting nodes whose immediate reward is inadequate but may turn out to lead to a superior reward in the long-run (Gelly et al., 2006; Kocsis & Szepesvári, 2006).

This research demonstrated that it is possible to variate the selection stage by combining Upper Confidence Bound applied to Trees (UCT) and lean Last Good Reply with Forgetting (lean-LGRF). UCT makes use of confidence intervals to balance between exploitation of known good nodes and exploration of nodes that are not currently the best but may lead to good nodes along the tree. Therefore, no potential node is starved of selections, and at the same time, favourable nodes are selected more often than their counterparts. This is essentially a form of balancing between order (exploitation) and chaos (exploration). Chaos is necessary to provide an adequate amount of challenge required for growth (Peterson & Djikic, 2003). The UCT formula is as shown in Equation (4.1):

$$\underset{i \in \text{children of the root node}}{argmax} \left(\frac{y(n_i)}{x(n_i)} + \sqrt{c \times \frac{\log(t)}{x(n_i)}} \right) \qquad \textbf{(4.1)}$$

Where:

$\gamma(n_i)$ is the total discounted value of selecting node *i*. It is discounted because we cannot get to a terminal state that determines a final win or loss. If *i* is an OLTP workload, the value is based on transaction throughput and if it is an OLAP workload, the value is based on response-time latency. Section 2.3.4 provides a more detailed explanation on the benefits of discounting future reward values.

$x(n_i)$ stands for the total number of simulations that have occurred for the node *i*

$\frac{\gamma(n_i)}{x(n_i)}$ is therefore the value of the node and forms the exploitation parameter of the UCT formula.

c stands for the exploration parameter. An increase in the value of *c* results in more exploration. The study used a value of $c=5{,}000$ as the baseline

t stands for the total number of simulations that have occurred for the parent of node *i*

Lean-LGRF on the other hand works by randomly simulating scenarios and keeping track of the reactions that resulted in a high reward at the end of the simulation. Stankiewicz, Winands, Uiterwijk and Jos (2011) provide a deeper insight of LGRF. This can be expressed as shown in Figure 4.2.

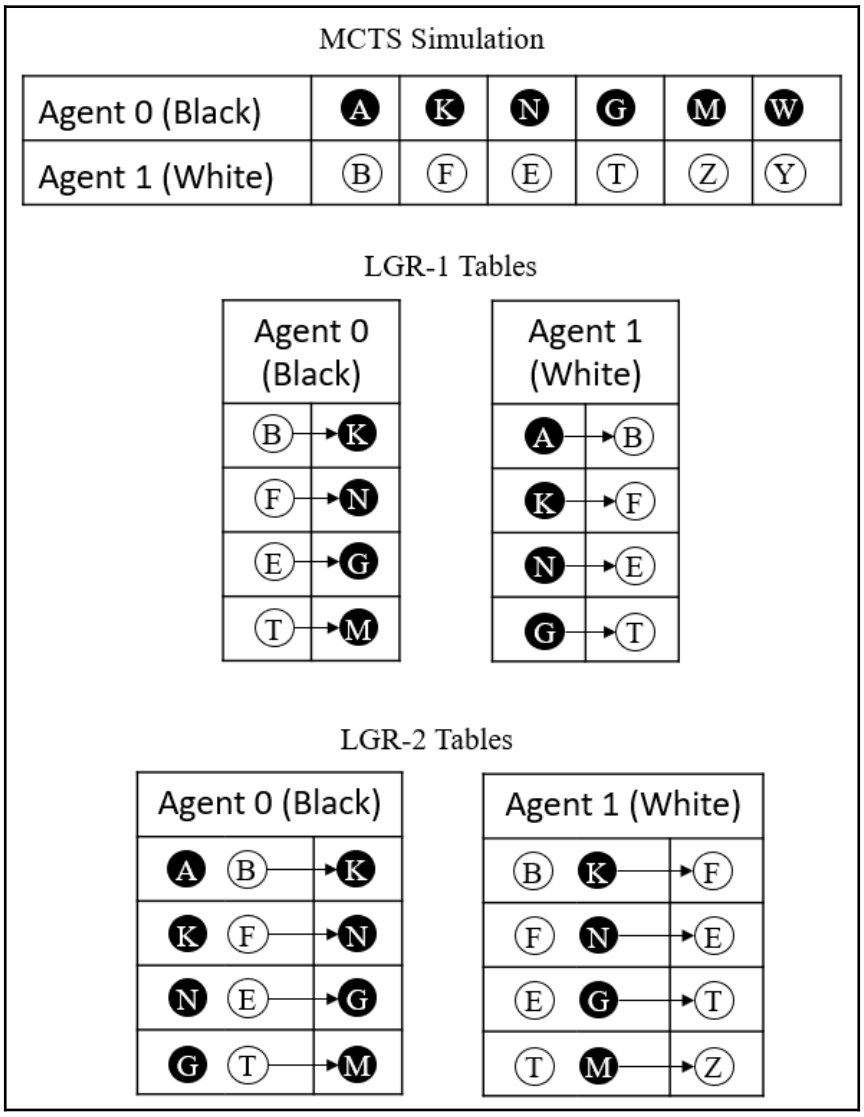

Figure 4.2: Illustration of Last Good Reply with Forgetting (LGRF)

A full LGRF algorithm contains both LGR-1 and LGR-2 tables. However, the study applied only the LGR-1 table due to its speed, affordable computational cost, and low memory footprint. Hence the concept of "lean-LGRF". UCT and lean-LGRF are thus combined as a variant to the selection stage in the designed algorithm as shown in Equation (4.2).

$$\underset{i \in \text{children of the root node}}{armax} \left(\frac{w_i}{x(n_i)} + \text{lean\_LGRF}(n_0, n_i) + \sqrt{c \times \frac{\log(t)}{x(n_i)}} \right) \qquad \textbf{(4.2)}$$

Where:

lean_LGRF(n_0, n_i) is a function that returns the value of the node that led to the highest reward given that the previous node was n_0 and the current node under consideration is n_i . This implies that it is twice as valuable to select a node that leads to the state with the highest possible value.

4.4.2 Expansion Stage

The expansion phase of the MCTS algorithm checks the selected node to determine if it has any unexplored child nodes. If an unexplored child node is found, it is added to the search tree. Previous studies have demonstrated implementation of the expansion stage by adding one node to the search tree at a time, or by adding multiple nodes at the same time, as commonly done in multi-threaded architectures (Browne et al., 2012; Den Teuling & Winands, 2011; Hartmann, 2017; Kocsis & Szepesvári, 2006). This study variated the expansion stage by adding all the possible children of the unexplored child node before proceeding to the simulation stage. This was made possible due to the categorization of parameters into tactics which narrowed the number of possible actions that can be performed in order to lead to a child node.

4.4.3 Simulation Stage

The simulation stage randomly simulates possible scenarios in a replica virtual environment. This replica virtual environment can be referred to as the "digital twin" of the database system. The study used a total of 1,000,000 simulations for the simulation stage which allowed it to maintain the law of large numbers. This meant that the more the simulations, the less the degree of uncertainty regarding the value of a state and the tactic(s) required to reach that state.

Unlike the default MCTS algorithm which simulates one child at a time, the study found that variating the simulation stage by simulating all the children simultaneously at the simulation stage produced remarkable results. This was made possible due to the categorization of parameters into 3 tactics. The random simulations were continuously carried out until a predefined computational budget was reached. The randomness was applied during the simulation process in order to reduce the level of over-reliance on a mathematical model and to reduce the bias that can be introduced by a human-designed

algorithm. The results of these simulations are then recorded and used to calculate the value of subsequent nodes along the tree.

4.4.4 Back-Propagation Stage

The back-propagation stage is the least demanding stage and involves returning the discrete reward value of each node (obtained through simulation as explained in Section 4.4.3) back up the search tree (back-propagation). Figure 4.3 provides a graphical representation of the full, variated MCTS algorithm that formed the foundation for the design of the autonomic latency-aware algorithm in this research.

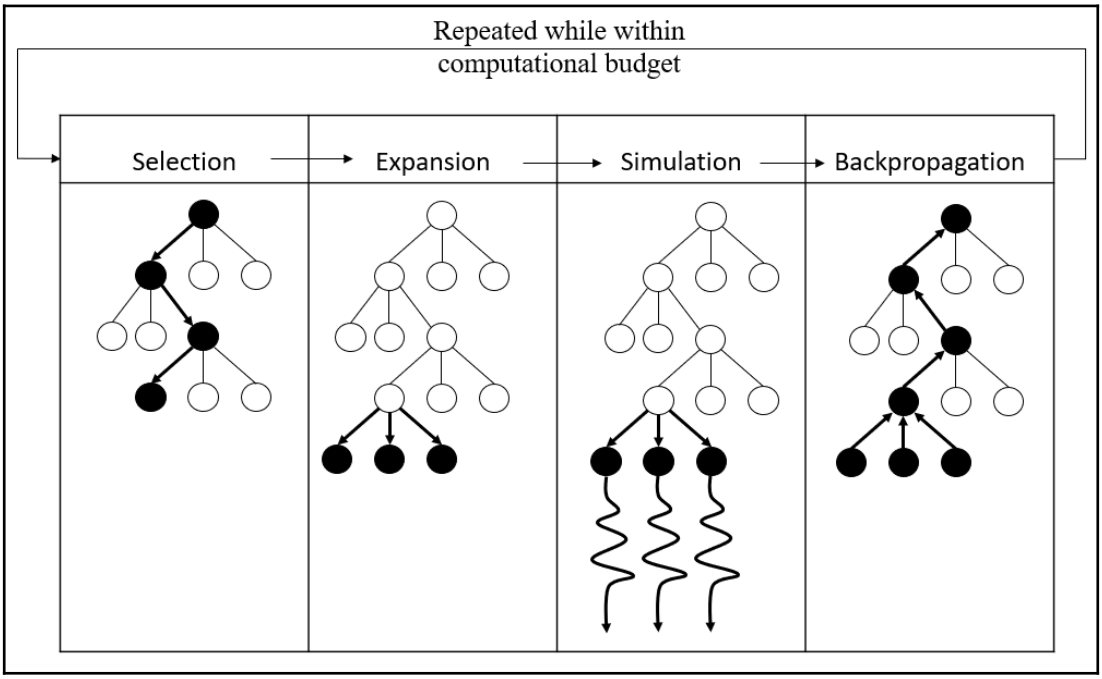

Figure 4.3: Representation of the Monte Carlo Tree Search (MCTS) algorithm

Algorithm 4.1 provides an overview of the basic Monte Carlo Tree Search (MCTS) algorithm. Algorithm 4.2 details the variated selection stage and the default expansion stage. Lastly, Algorithm 4.3 details the variated simulation and back-propagation stages of the MCTS algorithm.

Algorithm 4.1: The Basic Monte Carlo Tree Search Algorithm

Input: The state, $\mathbf{s_0}$, of the current root node, $\mathbf{n_0(s(n_0))}$

Output: The action, **a**, that leads to the most optimum child node that has a desirable state

1 **Function** MCTS ($\mathbf{s_0}$)

2 create root node n_0 with state s_0;

3 **while** *within computational budget* **do**

/*The **SELECTEXPAND** function is used to select a child node according to a predefined utility function: "UCT lean-LGRF" in the case of this study. This selected child node, $\mathbf{n_i}$, is then added to the search tree in the process of expansion. The variated expansion stage goes a step further to add all the 3 possible children of the child node $\mathbf{n_i}$.*/

4 $\mathbf{n_i} \leftarrow$ SELECTEXPAND ($\mathbf{s(n_0)}$)

/*The **SIMULATEBACKPROPAGATE** function is used to randomly simulate possible scenarios from the newly expanded node $\mathbf{n_i}$. A reward value is calculated by executing each of the possible actions available from the newly expanded node. This is then stored in Δ and back-propagated up the tree to update the node values.*/

5 $\Delta \leftarrow$ SIMULATEBACKPROPAGATE ($\mathbf{s(n_i)}$);

/*The result of the overall search, a(SELECT($\mathbf{n_0}$)), is the action, **a**, that leads to the best child of the root node, $\mathbf{n_0}$, where the exact definition of "best" is defined by the implementation. In the case of this study, "best" is the node with the highest discounted transaction throughput (for OLTP) or the lowest discounted response-time latency (for OLAP).*/

6 **return** a(SELECT($\mathbf{n_0}$))

Algorithm 4.1: The basic Monte Carlo Tree Search (MCTS) algorithm

Algorithm 4.2: Variated Selection and Variated Expansion Stages of the Monte Carlo Tree Search Algorithm

Input: The state, $\mathbf{s_0}$, of the current root node, $\mathbf{n_0(s(n_0))}$

Output: The child node $\mathbf{n_i}$ that has been selected through either exploitation or exploration. In addition to $\mathbf{n_i}$, all its 3 children are also specified as output.

1 **Function** SELECTEXPAND(n_0)
2 **while** *n_0 is non-terminal* **do**
3 **if** *n_0 has been fully expanded* **then**
4 **return** $\Delta \leftarrow$ SELECT ($\mathbf{n_0}$)
5 **else**
6 **return** EXPAND ($\mathbf{n_0}$);

7 **Function** SELECT ($\mathbf{n_0}$)

/*Determines the best child, Δ, to select out of all the other possible children of node $\mathbf{n_0}$. This is based on the value of the UCT lean-LGRF variation to the selection stage.*/

8 **return** $\underset{i \in \text{children of the root node}}{armax} \left(\frac{w_i}{x(n_i)} + \text{lean\_LGRF}(n_0, n_i) + \sqrt{c \times \frac{\log(t)}{x(n_i)}} \right)$

9 **Function** EXPAND($\mathbf{n_0}$)
10 Perform a ∈ *untried actions* from $\mathbf{A(s(n_0))}$;
11 add new child $\mathbf{n_i}$ to $\mathbf{n_{i-1}}$
with $\mathbf{a(n_i)} \leftarrow \mathbf{a}$
and $\mathbf{s(n_i)} \leftarrow f(\mathbf{s(n_{i-1})}, \mathbf{a})$;
12 add all children of $\mathbf{n_i}$ to $\mathbf{n_i}$;
13 **return** $\mathbf{n_i}$, children of $\mathbf{n_i}$;

Algorithm 4.2: Variated selection and variated expansion stages of the MCTS algorithm

Algorithm 4.3: Variated Simulation and Back-Propagation Stages of the Monte Carlo Tree Search Algorithm

Input: The state, $\mathbf{s_i}$, of the current node, $\mathbf{n_i}(\mathbf{s}(\mathbf{n_i}))$

Output: The reward value of node $\mathbf{n_i}$

1 **Function** SIMULATEBACKPROPAGATE($\mathbf{s}(\mathbf{n_i})$)
2 **while** *$A(s(n_i))$ is not empty and $x(n_i) < 1{,}000{,}001$* **do**
3 perform $\mathbf{a} \in \mathbf{A}$ ($\mathbf{s}(\mathbf{n_i})$) at random;
4 $\mathbf{s}(\mathbf{n_i}) \leftarrow f(\mathbf{s}(\mathbf{n_{i-1}}), \mathbf{a})$;
/*$\mathbf{x}(\mathbf{n_i})$ is the number of times node $\mathbf{n_i}$ has been traversed*/
5 $\mathbf{x}(\mathbf{n_i}) \leftarrow \mathbf{x}(\mathbf{n_i}) + 1$;
6 **if** *result is a win* **then**
/*The number of simulated wins that have occurred when node $\mathbf{n_i}$ was traversed; where a "win"" is defined by the implementation. In the case of this study, "win"" is the node with the highest discounted transaction throughput (for OLTP) or the least response-time latency (for OLAP).*/
7 $\mathbf{w_i} \leftarrow \mathbf{w_i} + 1$;
8 SIMULATEBACKPROPAGATE(for all the children of $\mathbf{n_i}$);
9 BACKPROPAGATE($\mathbf{n_i}$);

10 **Function** BACKPROPAGATE ($\mathbf{n_i}$)
11 **while** *n_i is not null* **do**
12 update *rewardValueofNode* $\boldsymbol{n_i}$;
/*$\mathbf{w_i}$ = discounted transaction throughput (for OLTP) or the least response-time latency (for OLAP) for node $\mathbf{n_i}$ and all of its children*/
13 update $\mathbf{w_i}$;
/*$\mathbf{x_i}$ = number of traversals (visits) made on node $\mathbf{n_i}$ and all of its children*/
14 update $\mathbf{x_i}$;
15 update *rewardValueofNode* $\boldsymbol{n_i} = \dfrac{y(n_i)}{x(n_i)}$;
16 $\mathbf{n_i} \leftarrow$ parent of $\mathbf{n_{i-1}}$;

Algorithm 4.3: Variated simulation and back-propagation stages of the MCTS algorithm

4.5 Algorithm Correctness and Time and Space Complexities

For an argument to be considered valid, the conclusion must always coherently follow from the premises. In other words, $premise \rightarrow conclusion$ (if premise, then conclusion). The algorithm can similarly be distilled to the form of a conditional in order to prove it.

Theorem: If s_0 is the state of the current root node, then the action, *a*, recommended by the variated MCTS algorithm is the action that leads to the most optimum child node that has a desirable state: $s_o \rightarrow a_0$

Proof: Assume that the most optimum child node that has a desirable state, s_1, is the node that has the maximum value over all other nodes at time $t+1$, that is, $\max_{t+1}(\forall s) = s_1$. This means that the action that led to the node at time $t+1$, is defined by the action to be taken when the state of the root node was s_0, at time t_0. That is, $q_*(s_0, a_0)$, where q_* represents the variated MCTS algorithm, will result in s_1. Therefore, $q_*(s_0, a_0) \rightarrow s_1$ and $q_*(s_1, a_1) \rightarrow s_2$ whereby the numerous simulations imply that the law of large numbers reduces the level of uncertainty in these conditionals. Thus $s_0 \rightarrow a_0$ and $s_1 \rightarrow a_1$. □

The variated MCTS algorithm has a stop property implemented by a computational budget which is based on a pre-defined number of simulations (1,000,000 simulations). This can be observed by the fact that the number of simulations is defined as a constant, finite integer.

Algorithm 4.3 receives 3 children and updates the value of the 3 children and their respective parents and ancestors. The child with the highest value is the focus. The assumption made is that the SIMULATEBACKPROPAGATE($s(n_i)$) must receive the state of at least 1 child that was selected in the previous MCTS stage. The partial correctness property can be proved by induction in this case.

This begins with the loop invariant which is a condition that must hold as true during the entire execution of the loop. The loop invariant in this case is:

$max \leftarrow Max \{ 0 \leq k < i \mid values[k] \}$ which in other words means that for any given

value, *i*, the variable named "max" will contain the maximum values of elements with indexes smaller than *i*. This is true at the beginning of the simulation when *i=1* and at the end of the simulation when *i=1,000,001*.

The inner if statement in the loop dictates that if and only if the simulation resulted in a "win", then the value of the current child is incremented by 1 unit as shown in line 7 of Algorithm 4.3: $max \leftarrow Max \{ 0 \leq k < i+1 \mid values[k] \}$. This means that the value of "max" is the maximum value of elements up to this point (with indexes smaller than *i+1*).

The loop invariant will therefore hold when the next iteration is initiated and by inductive reasoning:

$N = 1{,}000{,}000$ (computation budget)

$max \leftarrow Max \{ 0 \leq k < N \mid values[k] \}$

The algorithm will therefore be able to know if a "win" has occurred and subsequently record the number of wins. Thus proving the "total correctness property" of the algorithm. □

The theoretical analysis of the algorithm shows that the Big-O notation of the algorithm's time complexity is linear, that is, the algorithm's time complexity has an order of n: $O(n)$. On the other hand, space complexity is a measure of the amount of working storage an algorithm requires. The main concern is how space requirements grow in Big-O terms as the size of *n* (input) grows. The Big-O notation of the algorithm's space complexity is therefore also linear. That is, the algorithm's space complexity has an order of n: $O(n)$.

4.6 Summary

The use of stepwise regression as a wrapper technique in feature selection was demonstrated. This is a critical step that helps to avoid the curse of dimensionality which is attributed to a large number of interconnected variables. The use of tactics to categorize the selected parameter values was also explained as an important step that has a significant effect on the performance of the algorithm. This positive effect was observed in the number of possible actions (tactics) that can be performed to move from one state to the next. This subsequently reduces the number of child nodes that need to be simulated in the simulation stage. The algorithm design was based on variating critical stages in the MCTS algorithm. These include the selection, expansion, and simulation stages. The back-

propagation stage formed the last stage of the algorithm. A proof by induction was provided for the algorithm's behaviour taking into consideration the simulation loop invariant. The simulation loop was also analysed to conclude the presence of a linear time and space complexity based on the Bachmann-Landau notation. The algorithm designed was then applied in the context of a database system as well as a strategy board game, known as Reversi, and it produced interesting results. Section 5.2 and Section 5.3 of this thesis present these results.

Chapter 5: Experiment Results

5.1 Introduction

The desire to know (the truth) inevitably leads to the questioning of old or current theories. The philosophical assumptions made in this research dictate the use of true data, that is neither fabricated nor plagiarised, to discover the truth. This involves the formulation of propositions or predictions based on hypothesis and then confronting them with factual data to determine whether they can be accepted as the truth or not. Accepting a proposition as the truth in turn enables it to be used to answer research questions.

This Chapter presents the factual data required in the truth-discovery process. It starts with Section 5.2 which analyses the different categories of selection variants. Chi-square (χ^2) tests are conducted and used to identify the most superior variation. They are also used to show that variating the selection stage has a significant impact on the MCTS algorithm as a whole. This is followed by Section 5.3 which presents the results of the empirical algorithmics conducted using the superior selection variant. The raw data presented in this section forms the foundation required to apply t-tests that then either provide statistical evidence to reject the null hypothesis, or do not provide any statistical evidence to reject the null hypothesis.

5.2 Selection Variant Results

The process of designing the algorithm involved the application of treatments on the independent variable and observing the effect associated with it. The treatments applied were in the form of variants to the selection, expansion, and simulation stages of the baseline (default) MCTS algorithm. Emphasis was placed on the variation at the selection stage because the selection stage has an impact on all the subsequent stages of the algorithm. In order to identify the most beneficial selection variant, the study made use of 3 categories of treatment: the Upper Confidence Bound applied to Trees (UCT) baseline, the designed Upper Confidence Bound applied to Trees with “lean-Last Good Reply with Forgetting” (UCT with “lean-LGRF”), and lastly, the Progressive Bias (PB) selection variant. The observed level of variance between the three categories was compared with the expected, theoretical level of variance by using a Chi-square (χ^2) test.

It was necessary to discretely identify which variant was the best. Game theory was

used to accomplish this because of its ability to distinguish between good (a win) and bad (a loss) depending on the rules of the game. This is unlike decision theory applied in the context of performance tuning whereby the distinction between a good and a bad QoS or performance can be subjective.

Reversi, also known as Othello, a strategy board game played by two players usually on an 8×8 board, was used as the test bed to apply and test the 3 categories of selection variants of the algorithm. 64 disks with a top side and a bottom side, such that each side has a different colour (usually one side black and the other side white), are used to play the game of Reversi. It is a sequential game whereby players take turns to place disks on the board. A player places a disk on the board with their assigned colour facing up. When the opposing player's disk, say a white disk, is directly surrounded by the other player's disk, say a black disk, then the white disk is flipped and becomes a black disk. A player can flip any number of opposing disks so long as they have a disk on either side of the consecutive opposing disks. If a player cannot place a disk so that it is on either side of the opposing player's disk, then that player loses their turn. The game continues until either all spaces are occupied, or no player can make a move. The rules are simple, however strategy is what enables a player to win. Figure 5.1 shows an example of a strategy-based Reversi move whereby the player using the black disks attempts to occupy the corners as early as possible. An autonomic agent is able to determine through multiple simulations that occupying the corners is valuable because it leads to a win.

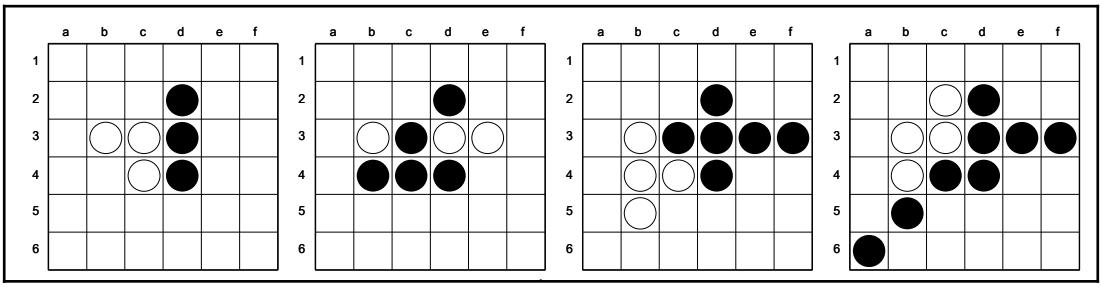

Figure 5.1: Reversi strategy of occupying the corners as early as possible

(adapted from Nair, 2017)

This provides an opportunity to apply the MCTS algorithm because each player is required to simulate multiple possible future scenarios and then choose the action that will lead to the highest possible reward (that is, a win). Table 5.1 shows the contingency table

of selection variants versus performance results measured in quantitative and objective number of wins and losses. Table 5.2 then follows with chi-square calculations for comparing the level of association in the selection variants.

Table 5.1: Contingency table of selection variants versus performance

	Selection Variants			
Performance	**UCT**	**UCT with "lean LGRF"**	**PB**	**Total**
Win	497	1000	458	**1,955**
Loss	503	0	542	**1,045**
Total	**1,000**	**1,000**	**1,000**	**3,000**

Table 5.2: Chi-Square calculations for comparing the level of association in the selection variants

Selection Variant	**Observed frequency** O_i	**Expected frequency** E_i	$O_i - E_i$	$(O_i - E_i)^2$	$\frac{(O_i - E_i)^2}{E_i}$
UCT with "lean LGRF"	1,000	729	271	73,441	100.7421
PB	458	729	-271	73,441	100.7421
				$\sum \frac{(O_i - E_i)^2}{E_i}$	**201.4842**

χ^2_{calc} (Calculated chi-square) value = 201.4842

d.f. (Degrees of freedom) = $(c-1)\times(r-1)=(2-1)\times(2-1)=1$

α (Level of significance) = 5%

$\chi^2_{tabulated}$ (Table chi-square) value = 3.841

Figure 5.2 graphically depicts the contingency table in detail by showing the number of experiments (game plays) against the average number of wins.

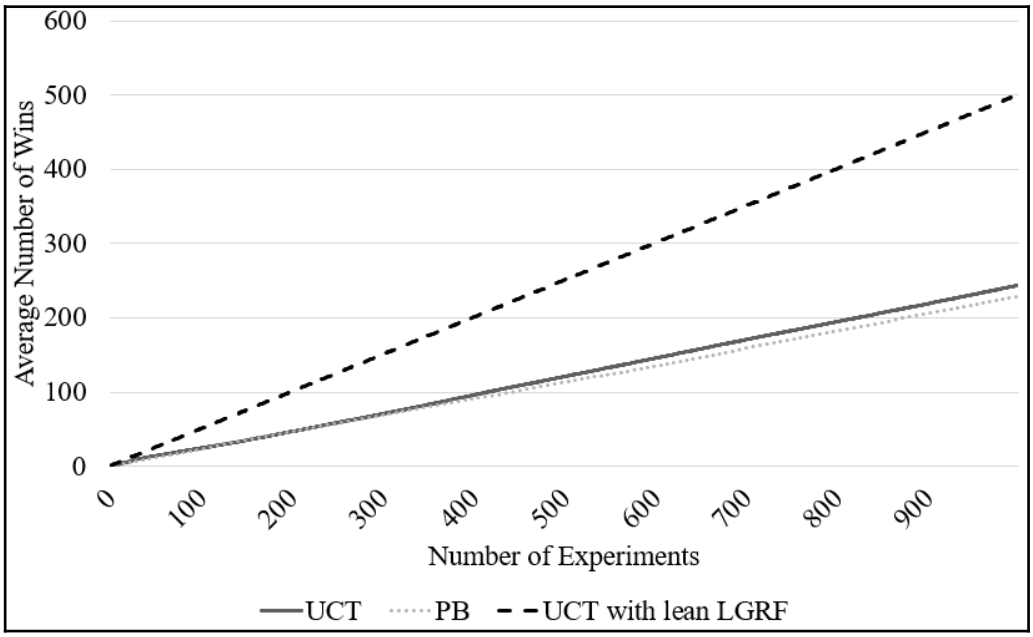

Figure 5.2: Number of experiments against average number of wins

Figure 5.3 then shows the percentage number of wins for each treatment against the baseline UCT selection variant. It is evident from the results that the designed "UCT with lean LGRF" selection variation has a superior performance.

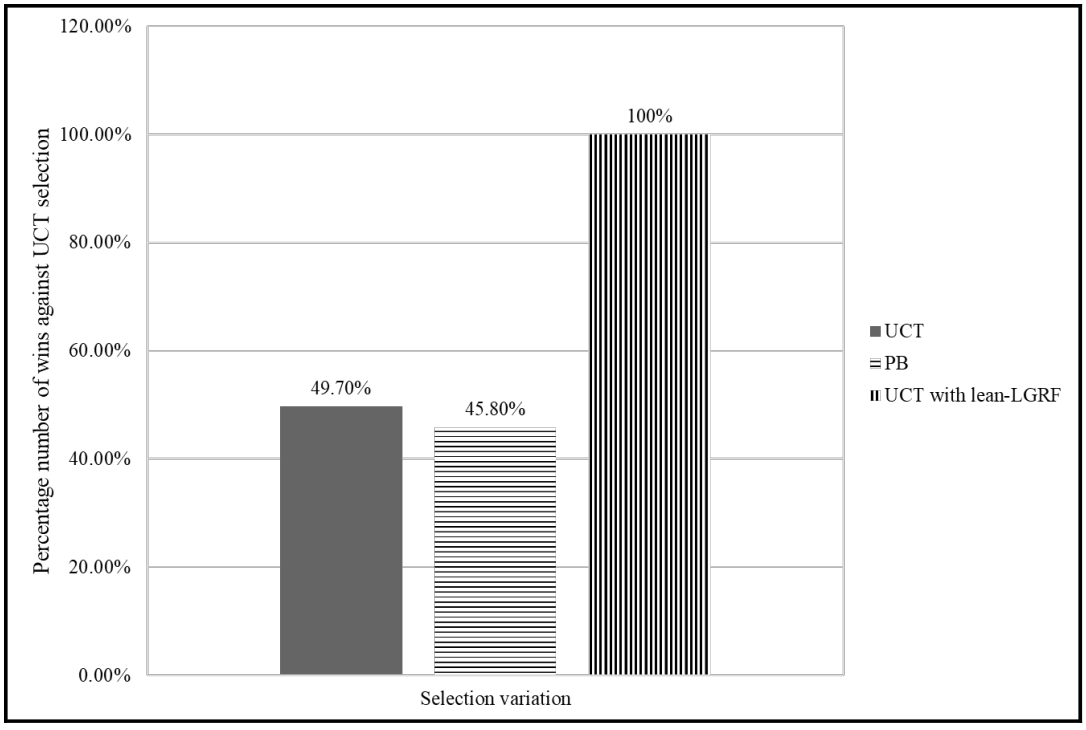

Figure 5.3: Percentage number of wins for each treatment against the baseline UCT selection variation

This was a surprising result, because the PB has the advantage of being able to combine Upper Confidence Bound applied to Trees (UCT) and heuristics from the Unvisited Legal Moves (ULM) strategy into one technique (Den Teuling & Winands, 2011; Lanctot, Wittlinger, Winands & Den Teuling, 2013). This combination should have enabled it to apply heuristic evaluation to perform a selection when the UCT parameters have not yet had enough iterations to settle as shown in Equation (5.1).

$$\underset{i}{argmax}\left(\frac{w_i}{n_i}+\sqrt{c\times\frac{\log(t)}{n_i}}\ +\ \frac{W\times H_i}{l_i+1}\right) \qquad \textbf{(5.1)}$$

Where:

w_i is the number of simulated wins that have occurred for the node *i*

n_i stands for the total number of simulations that have occurred for the node *i* ($\frac{w_i}{n_i}$ is therefore the value of the node and forms the exploitation parameter of the UCT formula)

c stands for the exploration parameter. An increase in the value of *c* results in more exploration. The study used a value of $c=2$ as the baseline.

t stands for the total number of simulations that have occurred for the parent of node *i.*

Therefore, $\sqrt{c\times\frac{\log(t)}{n_i}}$ forms the exploration parameter of the UCT formula and $\frac{w_i}{n_i}+\sqrt{c\times\frac{\log(t)}{n_i}}$ forms the UCT parameter of the PB formula.

W is a constant set to 10

H_i , is the N-Gram based heuristic value for child *i*. If a 3-Gram is used, this value

will be based on the average reward value of the move sequence consisting of the child node *i*, its parent, *p*, and the parent of node *p*.

l_i is the number of losses encountered during the simulation of node *i*. That is, $n_i - w_i$

However, when played against the UCT baseline, PB did not have a significant superiority over UCT. This was unlike the designed UCT with "lean-LGRF" selection variant that had the best overall performance.

This rather unexpected result could be attributed to the fact that the limiting computational budget of time meant that the agent applying the PB selection variation had to present a result based on the point it had reached when the time allocated had expired. The maximum time allocated to each agent to decide which move to perform was 4 seconds only. This was maintained as a constant across all selection variations.

Unlike the PB strategy which combines two techniques, the strategy applied in the Reversi game context selected one of two techniques based on a pre-defined condition. The condition was that if all the children of a node have been visited before, then the UCT selection strategy was applied. However, if not all the children have been visited before, then a variation of the LGRF strategy was used to select a child node. The negative consequence of doing this was that the search was focused on regions of the tree that contain good move sequences despite that region being fully explored.

The results provided evidence that the calculated value of χ^2 (201.4842) was much higher than its table value (3.841). The calculated value did not occur by chance; it was significant. We therefore failed to reject the hypothesis that stated that there is a significant level of variance in the selection variation that is applied in an MCTS algorithm. Because the two selection variations were not similar, then it follows that one must be superior than the other. The UCT with "lean LGRF" selection variation won all of the games played against the baseline UCT strategy unlike the PB selection variation which won only 45.8% of the games. This observation therefore suggests that a strong link exists between applying UCT with "lean LGRF" as a selection strategy and the overall performance of an MCTS algorithm.

Additional enhancements were made when applying the UCT with "lean LGRF"

selection strategy in the context of the broader research. The enhancements were that instead of choosing between UCT and “lean LGRF” depending on whether the child nodes had been visited earlier, the enhancement combined the two techniques. This combination meant that the value of a node was determined at any point in time by the sum of the UCT value and the lean LGRF value. The result of doing this was that the most valuable child was expanded and once in a while, a currently less valuable child that had a promising future along the tree was selected. Thus addressing the exploitation-exploration balance. This subsequently led to an observed improvement in performance measured by the transaction throughput and the response-time latency.

5.3 Empirical Algorithmics Results

Empirical algorithmics supports the acquisition of insights into the behavior of a designed algorithm as explained further in Appendix C (Diakopoulos, 2015; Kitchin, 2017). The designed algorithm under investigation applied variations of not only the selection stage, but also variations of the expansion, and the simulation stages of the MCTS algorithm.

Figure 5.4 shows the most suitable implementation architecture that was applied on each cluster member. A cron time-based job scheduler was used to schedule the periodic execution of the bash shell script based on time. The shell script was then used to call a Perl script from an online server. This enabled the Perl script to be updated from a central location instead of copying it to every node/member of the distributed database system cluster upon each update. The Perl script then implemented the variated MCTS-based algorithm which recommended the most appropriate tactic to implement. This was then implemented by editing the text-based configuration file on each node/member of the distributed database system cluster.

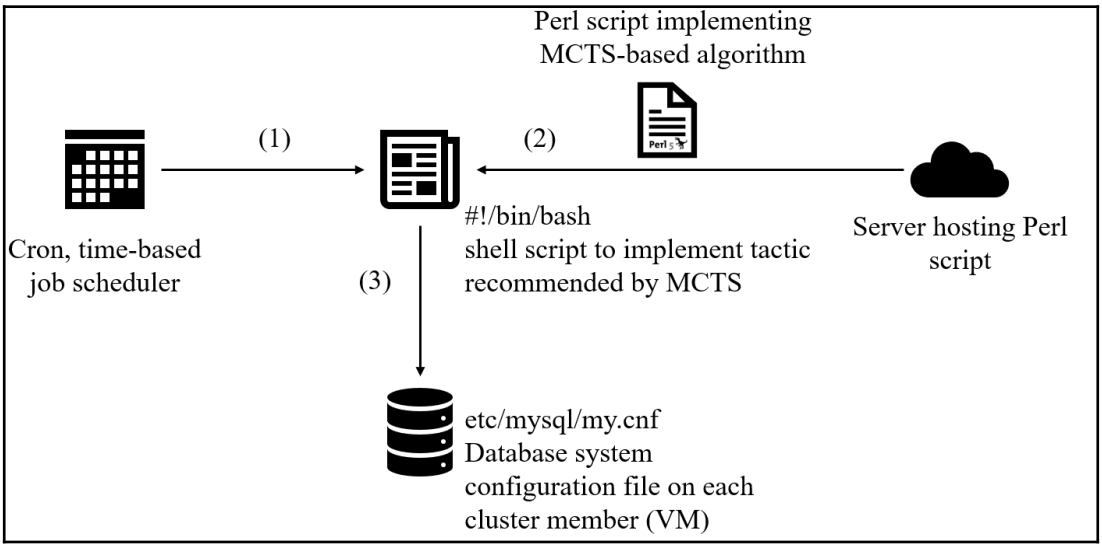

Figure 5.4: Implementation architecture on each cluster member

The experiment procedure outlined in Section 3.3.1 was used and Table 5.3 subsequently presents the experiment results.

The study applied ontological materialism, which is objective in nature. This was applied in conjunction with positivism which in turn emphasizes on revealing the truth through objective observations. It is through experiments based on empirical algorithmics that objective observations were made and recorded. This section of the thesis presents the results of the experiments based on empirical algorithmics in the context of profile-guided optimization of load scalability in distributed database systems applying an InnoDB storage engine.

An initial objective of the study was to design a latency-aware algorithm that proactively reconfigures bottleneck parameters without over-relying on an accurate model of an unpredictable stochastic environment. Each experiment conducted involved the identification of bottleneck parameters that have a significant impact on the system's performance, and proactively reconfiguring them using the designed algorithm so that they can adapt to the current workload. The results presented in this study indicate that all of the experiments conducted with the algorithm running resulted in a transaction throughput that was higher than the average transaction throughput in an environment running using default configurations.

Table 5.3: Empirical algorithmics results

	Default	**Low**	**Medium**	**High**	**Adaptive** (Automatically switches between high, medium, and low tactics based on the MCTS algorithm) **(μ)**
Average Transaction Throughput (Transactions Per Second)	4,958.46 ($\sigma = 350.74$)	5,272.92 ($\sigma = 210.94$)	5,413.56 ($\sigma = 303.06$)	5,784.91 ($\sigma = 435.78$)	5,812.75 ($\sigma = 249.11$)
Average Response-Time Latency (Microseconds)	3.50 ($\sigma = 0.61$)	3.04 ($\sigma = 0.19$)	3.02 ($\sigma = 0.14$)	3.07 ($\sigma = 0.38$)	2.56 ($\sigma = 0.50$)
Maximum & Minimum Transaction Throughput (Transactions Per Second)	5,298.36 and 2,935.70	5,648.19 and 4,809.30	5,912.09 and 4,721.93	6,074.99 and 3,665.23	6,454.89 and 5,379.13
Maximum & Minimum Response-Time Latency (Microseconds)	6.00 and 3.00	4.00 and 3.00	4.00 and 3.00	5.00 and 3.00	3.00 and 2.00

The results further indicate that all of the experiments conducted with the algorithm running resulted in a response-time latency that was lower than the average response-time latency in an environment running using the default configurations.

The probability of finding that the average transaction throughput for database systems that apply the designed autonomic latency-aware algorithm is greater than the average transaction throughput for distributed database systems that do not apply the designed autonomic latency-aware algorithm given that the null hypothesis is true is less than 0.05.

$$P(\mu > \mu_{H0} | H_0 \text{ is true}) < 0.05$$

In other words, the study adopted a 5% level of significance. This can be standardized using a t distribution such that the area under the t distribution curve that is greater than a t-score of 1.705618, given that there are 26 degrees of freedom, represents the 5% probability of occurring. Therefore, if a t-score is greater than 1.705618, then we cannot continue to claim that the null hypothesis is true. In such a case, we should reject the null hypothesis in conformance with the study's decision rule. That is,

Reject H_0 in favour of H_1 *if* $Tscore_{calculated} > Tscore_{tabular}$.

$Tscore_{calculated} = 17.81918823$ and $Tscore_{tabular} = 1.705618$ in the case of transaction throughput. Based on this evidence, the null hypothesis should be rejected in favour of the proposition that "distributed database systems that apply the designed autonomic latency-aware algorithm on average have a faster transaction throughput".

The probability of finding that the average response-time latency for distributed database systems that apply the designed autonomic latency-aware algorithm is less than the average response-time latency for distributed database systems that do not apply the designed autonomic latency-aware algorithm given that the null hypothesis is true is also less than 0.05.

$$P(\mu < \mu_{H0} | H_0 \text{ is true}) < 0.05$$

The area under the t distribution curve that is less than a t score of -1.705618 given that there are 26 degrees of freedom represents the 5% probability of occurring. Therefore, if a T-score is less than -1.705618, then we cannot continue to claim that the null hypothesis is true. The H_0 should be rejected in favour of H_2 if $Tscore_{calculated} < Tscore_{tabular}$. Given that $Tscore_{calculated} = -9.738191311$ and $Tscore_{tabular} = -1.705618$, then the null hypothesis should be rejected in favour of the

proposition that “distributed database systems that apply the designed autonomic latency-aware algorithm on average have a slower response-time latency.”

The results therefore provide no statistical evidence to reject the hypothesis that “distributed database systems that apply the designed autonomic latency-aware algorithm on average have a faster transaction throughput and a slower response-time latency.” These are particularly encouraging results because they draw our attention to the importance of considering an autonomic latency-aware Monte Carlo Tree Search (MCTS) algorithm for profile-guided optimization of load scalability in database systems applying an innoDB storage engine.

Figure 5.5 and Figure 5.6 display the average-max-min charts of the transaction throughput results and the response-time latency results respectively.

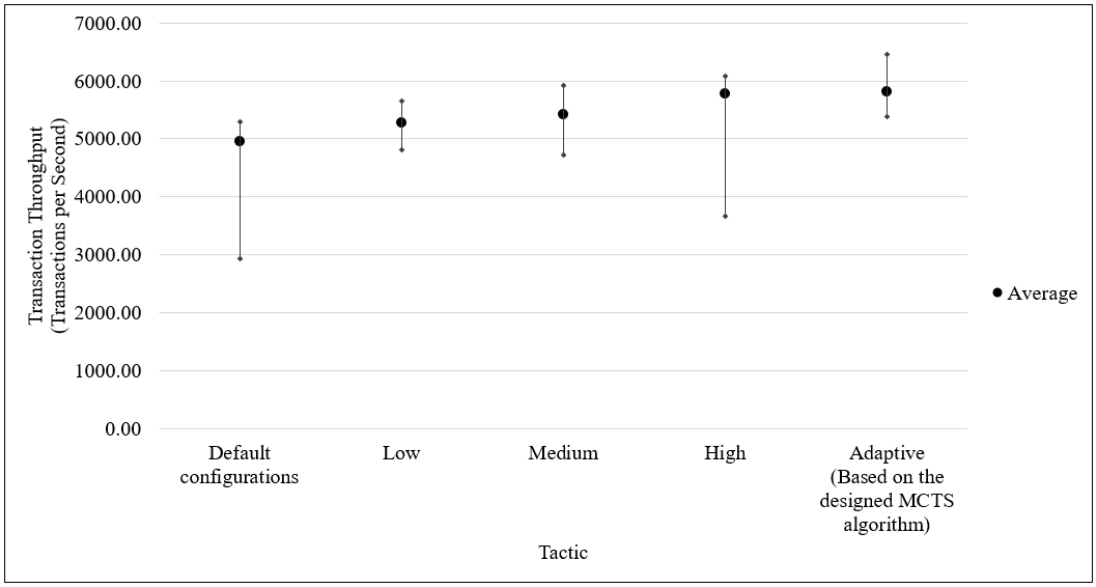

Figure 5.5: Average-Max-Min chart of the transaction throughput results

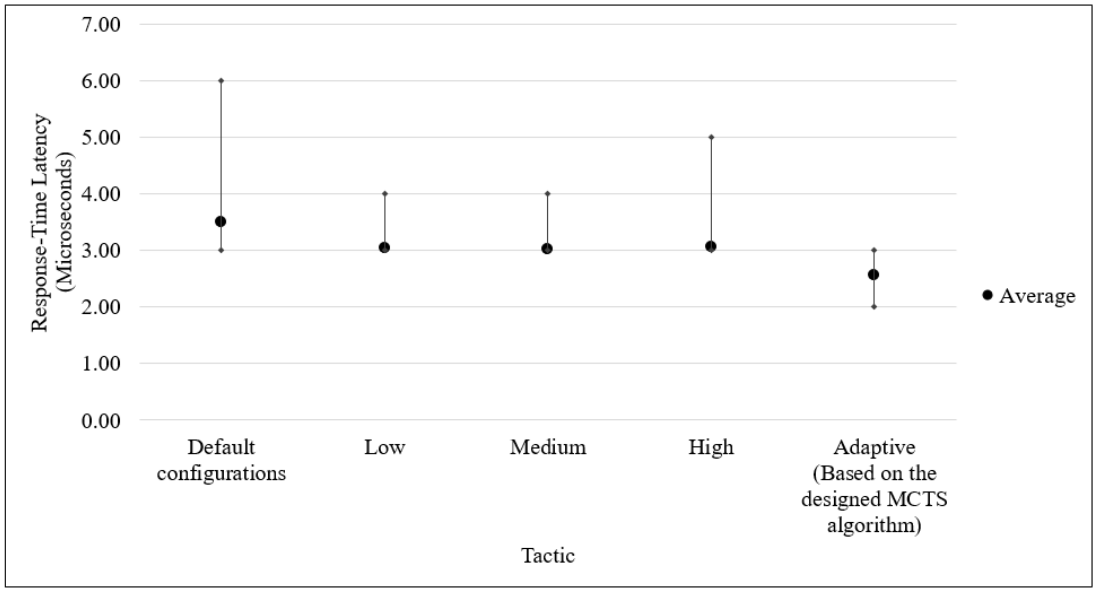

Figure 5.6: Average-Max-Min chart of the response-time latency results

5.4 Summary

The results of the experiments presented were used to rank the selection variations in order of superiority. A strategy board-game known as Reversi was used for this purpose. This was because game theory supports the objective and quantitative distinction between a good state (win) and a bad state (loss). This distinction was necessary to show that using the superior selection variation resulted in the autonomic agent winning all its games. The superior selection variation was then applied in the context of a distributed database system using concepts of empirical algorithmics. This supported compelling results that proved that the autonomic latency-aware MCTS algorithm had a significant impact on profile-guided optimization of load scalability in database systems. The following Chapter provides a discussion of what these results means in the context of the research as a whole including the problem statement. The discussion is presented in comparison with past and current research in the same area.

Chapter 6: Discussion of Experiment Results

6.1 Introduction

This Chapter seeks to answer fundamental questions related to the research. One of the key questions is "How does this research fit in the broader context of literature?" This Chapter therefore presents the answers to this question through the use of various Sections. Section 6.2 provides background information that refers to the literature review as well as the purpose of the study. This is followed by Section 6.3 which presents a summary of expected and unexpected results. Section 6.4 then specifies literature that supports the results documented in Chapter 5 of this thesis. It also specifies literature that seem to contradict some of the current results. Section 6.5 follows by providing an in-depth interpretation of what the current results mean as well as possible implications to IT departments in an enterprise. A highlight on Predictive Maintenance (PdM) is provided with the hidden potential of "digital twins" that can be modelled through ODEs or Kalman Filters. Section 6.6 then concludes the Chapter by providing a summary.

6.2 Background Information

6.2.1 Why the Research was Conducted

The motivation as to why this research was conducted was to provide an opportunity to tap into the potential benefits associated with achieving a knowledge economy. The definition of a knowledge economy as adopted by the Government of Kenya's MoICT (2014) is an economy in which growth is dependent on the quantity, quality, and accessibility of information available to be used for innovation. This is as opposed to dependency on scarce traditional factors of production defined in classical economics as land, labour, and capital. Prior studies have noted the on-going transition from agricultural-intensive and manual labour-intensive economies to knowledge economies that are not over-reliant on depleting natural resources. Such economies are capable of producing products and services that command a high premium and this can have a positive effect on a country's GDP. Section 3.6 provides additional examples of the benefits of a knowledge economy.

Data, information, and knowledge are different. As explained in the DIKAR model, the process starts with collection of raw data, which is then processed by computers using

basic maths into information. The information is then subjected to further analysis using probabilistic and statistical formulae to form knowledge. This knowledge can then be applied in form of action in order to obtain a desired result. If the desired result is not ideal or it is changed with time, then the process iterates starting from the collection of raw data. Therefore, a database that operates at its most optimum level of performance plays a significant role in this entire process. The exponential growth of data and the growing need for actions that are informed by knowledge, requires database systems to be periodically tuned to expand or contract their resources to handle increases in workload without negatively impacting on their responsiveness to execute any action within a given time interval. Current literature refers to this as load scalability (Shahapure & Jayarekha, 2014). In reviewing the literature, the study found that manual tuning interventions from system administrators have a significant impact on substandard load scalability due to factors such as fatigue, long reaction times, inconsistent expertise amongst system administrators and the over-reliance on inaccurate prediction of future states as explained further in Section 1.2 of this thesis.

6.2.2 How the Research was Conducted

With this in mind, the present study reviewed theoretical concepts based on the theories of model predictive control in control theory, decision theory, the branch of self-optimization in autonomic computing, and the branch of reinforcement learning in artificial intelligence. This was discussed in Chapter 2 of this thesis. These were used to inform the formulation of the hypothesis that "distributed database systems that apply the designed autonomic latency-aware algorithm on average have a faster transaction throughput and a slower response-time latency". The hypothesis was then used to make testable propositions which were subsequently subjected to factual data. The factual data was used to either reject or fail to reject the hypothesis. The source of this factual data was experiments in a controlled environment based on the steps outlined in detail in Section 3.3.1 of this thesis. This involved enabling the algorithm to automatically tune the database system as the amount of workload received increased. The workload was a mixture of transactions to manipulate the data, and analytical processing queries to analyse the data. Transaction throughput measured in the number of transactions per second and response-time latency measured in microseconds were respectively used to measure the performance of the

database system as the workload was increased.

6.2.3 What the Research Accomplished

It then follows that the research performed an investigation on how to design an adaptive algorithm that proactively reconfigures bottleneck parameters without over-relying on an accurate, long-term model of a stochastic environment. As stated in Section 1.3, this was done in order to achieve self-optimization required for load scalability in database systems. The use of probabilistic reasoning that relies on multiple simulations to confidently decide the best action to perform was used. Chapter 4 explains how MCTS principles were applied in a variated manner to customize the algorithm for the context of performance tuning in a database system. Chapter 5 of this thesis presents detailed results of the experiments conducted in the process of designing the algorithm and the following Sections discuss what these results possibly imply.

6.3 Expected and Unexpected Results

6.3.1 Increased Transaction Throughput and Reduced Response-Time Latency

It was interesting to note that the variation technique applied in the selection stage of the algorithm had a significantly positive impact on its performance. Implementing the UCT with “lean-LGRF” technique in the selection stage enabled the search to focus on promising areas of the tree while maintaining an exploitation-exploration balance. This was somewhat surprising given the fact that recent studies claim that the Progressive Bias selection variation is superior (Den Teuling & Winands, 2011). Figure 6.1 attempts to provide a graphical depiction of a dendrogram that focuses on the most promising area of a tree. In this case, the most promising area is the right-side of the tree and this results in a relatively in-depth extension of this area.

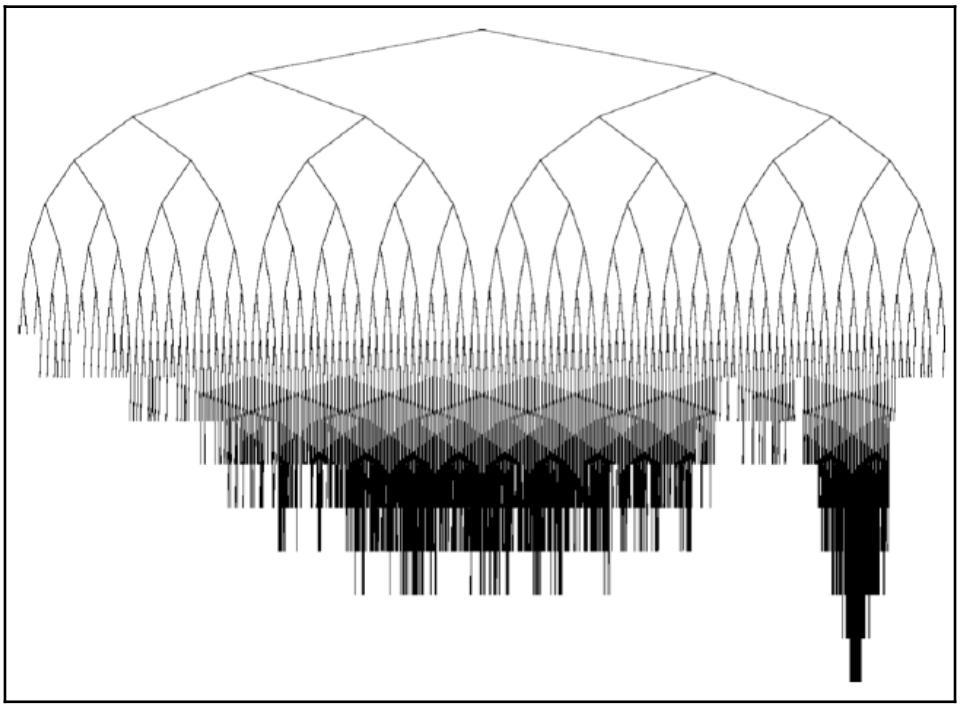

Figure 6.1: A dendrogram showing an intentional search focus on the most promising area of a tree

This selection variation was applied in conjunction with variations to the expansion and simulation phases of the MCTS algorithm. The variations enabled the algorithm to be customized in order to work in the context of performance tuning in a database system. Perhaps the most striking finding was that the variated MCTS algorithm indeed had a significant impact on the load scalability of the database system. This was measured based on transaction throughput and response-time latency as the amount of workload and the number of concurrent users increased gradually.

6.3.2 Sustainable Energy

What is surprising and unexpected is that even though sustainable development was not the main focus of this research, the results showed an unexpected positive outcome in this area. In as much as development is critical for the expansion of current human capabilities, sustainable development emphasizes that it is important to ensure that this development expands human capabilities without compromising the ability of future generations to meet their own needs and expand their capabilities (Saha, 2018). For this to happen, we need to replace natural resources used in the process of development with

resources of equal or greater value. Energy is the natural resource in IT and data centres. In the majority of cases where we are not able to replace natural resources with resources of greater value, then prudent use of currently available resources is necessary. In a case where demand drives supply, this implies consuming only that which we need in the energy consumption stage. It also implies sharing energy and its infrastructure among different loads through reuse in energy consumption. Lastly, it implies capturing and recovering wasted energy through recycling in the energy consumption stage.

A currently active area of research is the design of servers that have a positive correlation between the power it consumes and the rate at which work is done (Armbrust et al., 2010). This falls under the discipline of energy proportional computing and such servers are said to be energy proportional. From the stepwise regression results provided in Appendix E, the research found that the resources that had the most significant impact on a database system were the storage and memory. Changing configuration parameters of how these resources are used, after identifying symptoms using work metrics, caused a change in the performance of the database system. This is supported by results from studies that show that improvements in aggregate energy proportionality can be accomplished largely with software reconfiguration, requiring minimal changes to the underlying hardware (Barroso & Hölzle, 2007).

Contrary to common misconceptions, the CPU is no longer the biggest consumer of power in computers. This is attributed to innovations in low-power technologies, circuits, and microarchitectures that enable today's CPU to scale its power relatively well in relation to its utilization (Barroso & Hölzle, 2007). Memory, storage, and networks on the other hand are known to be energy disproportional and contribute to poor cluster and data centre level energy proportionality. The study however found that increasing the utilization and availability of the buffer pool in memory and reducing the focus on storage, contributed towards making the database system more energy efficient. A possible explanation of this is that transistor counts and densities in memory increase static power. Static power in this case is defined as the amount of energy consumed by a component even when the component is idle. Power is required to retain volatile data in memory. Non-volatile storage also requires power to maintain an interface required for access on demand. In the case of Hard Disk Drives (HDDs), the Revolutions per Minute (rpm) are a direct cause of static energy. Solid State Drives (SSDs) that use flash memory have also

shown signs of energy disproportionality.

Studies have shown that using the amount of work to ration the amount of power supplied to memory (using Dynamic Voltage Scaling [DVS]) or reducing the frequency of memory refresh cycles (using Dynamic Frequency Scaling [DFS]) are possible options to consider towards sustainable energy in data centres (David, Fallin, Gorbatov, Hanebutte & Mutlu, 2011; Deng, Ramos, Bianchini, Meisner & Wenisch, 2012). However, even though DVS and DFS (which can be combined together as DVFS) technology has worked well in CPUs, it is considered too risky to apply in memory because it can result in data corruption. Application of DVFS to scale memory power with throughput is still an active area of research.

In the meantime, this research submits the proposition that ensuring that memory is utilized as much as possible can tend towards sustainable energy in the data centre. This should involve the use of performance tuning algorithms that focus on how the software uses existing hardware in a more energy efficient manner. Figure 6.2 shows the memory utilization without the algorithm running and Figure 6.3 shows the memory utilization with the algorithm running. The fluctuations in temperature can be explained based on the use of a temperature threshold. When the temperature of the memory modules reaches a threshold, the cooling mechanism lowers it to an acceptable level. However, one of the major components of a data centre that consumes energy is cooling. This provides an opportunity for further research to investigate on how energy proportional computing can be achieved in the memory and storage of database systems while controlling the extraneous variable of temperature, which requires energy for cooling.

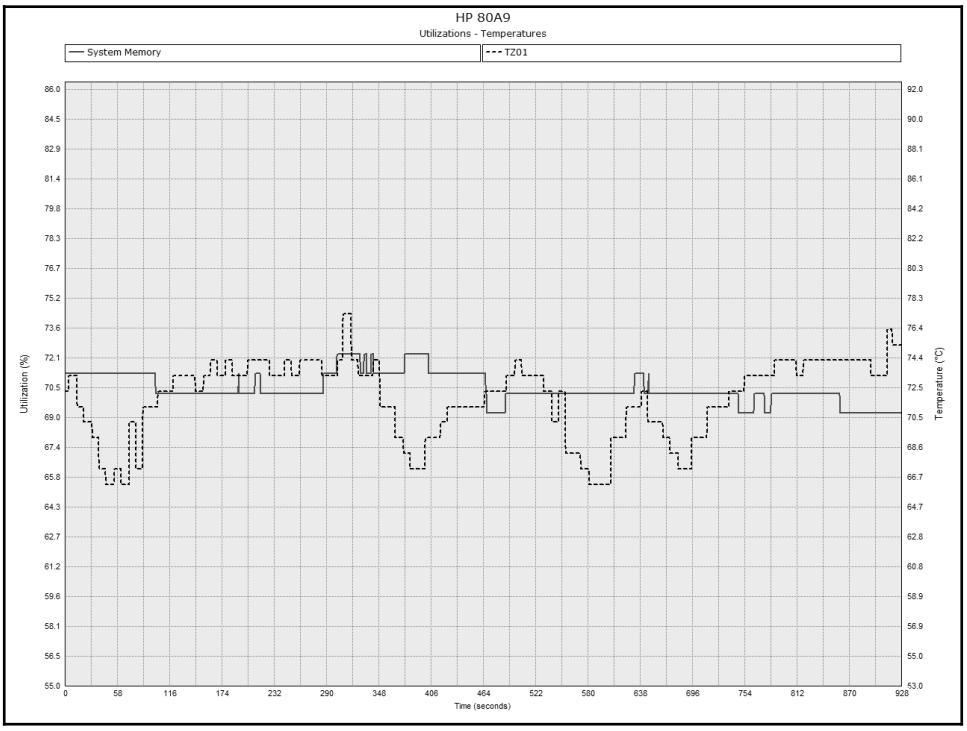

Figure 6.2: Memory utilization and temperature against time without the algorithm

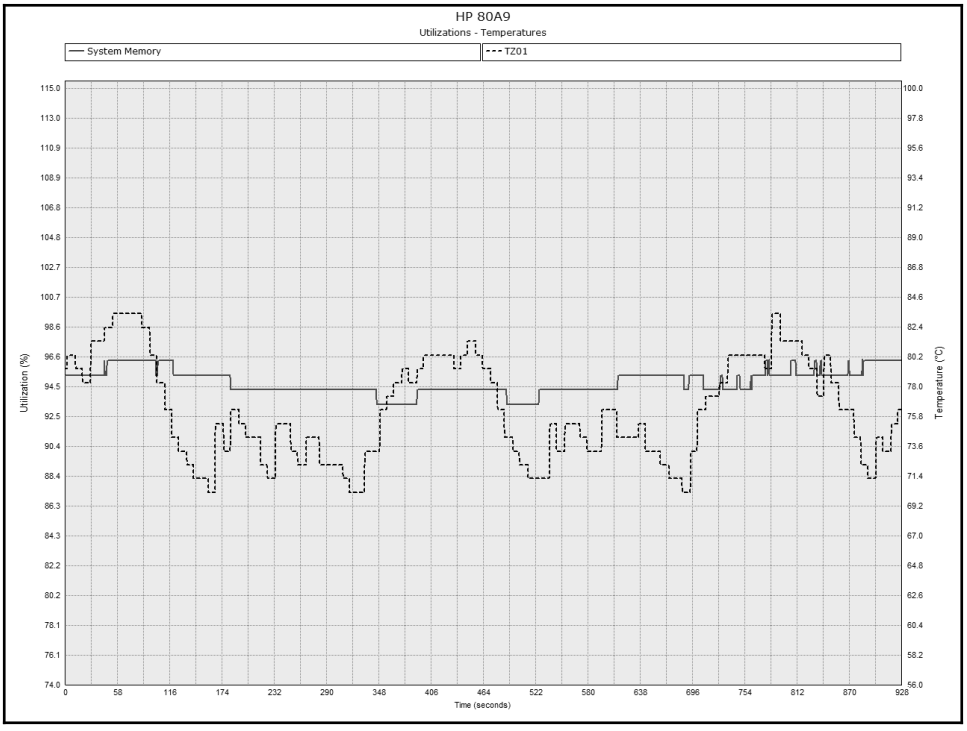

Figure 6.3: Memory utilization and temperature against time with the algorithm

6.4 Supporting and Contradicting Literature

This study confirms that optimization of load scalability in database systems is affected by an autonomic latency-aware algorithm. This temporal precedence finding is consistent with that of Van Aken et al. (2017) who reported that an automated approach that leverages past experience and simultaneously learns new information can be used to conduct performance tuning of database systems. The study highlighted performance tuning as an essential aspect of any database-intensive application.

A comparison of the study by Van Aken et al. (2017) with a previous study by Cheng and Garlan (2012) accords with our initial observation that although humans are better at understanding the overall problem context than computers, they are prone to long reaction times, fatigue, errors, and varying and potentially inconsistent expertise. This is in line with an even earlier seminal study by Kephart and Chess (2003) which championed the concept of self-management in computing in order to automate previously unachievable tasks or tasks that are performed in a sub-standard manner. The research is however keen to caution that this does not imply automation in order to replace database administrators. On the contrary, the findings in this study propose the use of automation to enable human beings to free their minds from mundane tasks in order to concentrate on previously unachievable tasks. This corroborates the findings from a study by Autor (2015) which investigated on the history and future of workplace automation.

The findings in this study also broadly support the work of other studies in this subject that combine the generality of random simulation and the precision of a tree search to form a Monte Carlo Tree Search (MCTS) (Coulom, 2006; Gelly et al., 2006). The Monte Carlo Tree Search was then applied in the current study to promote evidence-based, probabilistic reasoning in an Artificial Intelligent agent to identify which parameters need to be tuned and how to tune them. This is in accord with recent studies which highlight the value of probabilistic reasoning (Sullivan, 2003). In order to do this, it was necessary to variate and enhance the default MCTS algorithm. Browne et al. (2012) support this by presenting a comprehensive review of different variations and enhancements to the MCTS algorithm and subsequently show the benefits of going beyond the default version of the MCTS algorithm.

The study found that there is a significant level of variance in the different types of variations applied in the selection stage of the MCTS algorithm. The results provided

evidence that the designed novel Upper Confidence Bound applied to Trees in conjunction with lean Last Good Reply with Forgetting (UCT with “lean-LGRF”) was superior to the current Upper Confidence Bound applied to Trees (UCT) and Progressive Bias (PB) selection variations. This outcome is contrary to that of Den Teuling and Winands (2011) which claimed that PB was superior.

The findings of the current study seem to contradict the findings by Kim et al. (2016) which applies vertical and horizontal partitioning of data to promote scalability. The current study’s results also seem to contradict the findings by Chaudhuri and Narasayya (2007) which applies the creation of cost-driven indexes to promote scalability. Although indexes and partitioning are beneficial at the software level, they fail to provide an in-depth, lasting solution to the underlying scalability challenge, which should be focused on how the software makes use of the underlying hardware resources, for example, memory and storage.

6.5 Interpretation and Implication of the Results

The results of the current study provided adequate statistical evidence to accept the alternative hypothesis stated in Section 3.4.2 that “distributed database systems that apply the designed autonomic latency-aware algorithm on average have a faster transaction throughput and a slower response-time latency”. The prediction (proposition) that was made using the hypothesis was that if a distributed database system is running with the algorithm implemented in each node, then the transaction throughput will be higher and the response time latency will be slower. The one-tailed (right tail) t-score meant that there was a 95% chance that transaction throughput is faster when using the algorithm if the t-score of the experiment results was greater than 1.705618 on the t distribution. The one-tailed (left tail) t-score meant that there was a 95% chance that response-time latency is slower when using the algorithm if the t-score of the experiment results was less than -1.705618 on the t distribution. Figure 6.4 graphically depicts how the decision rule was applied in this research.

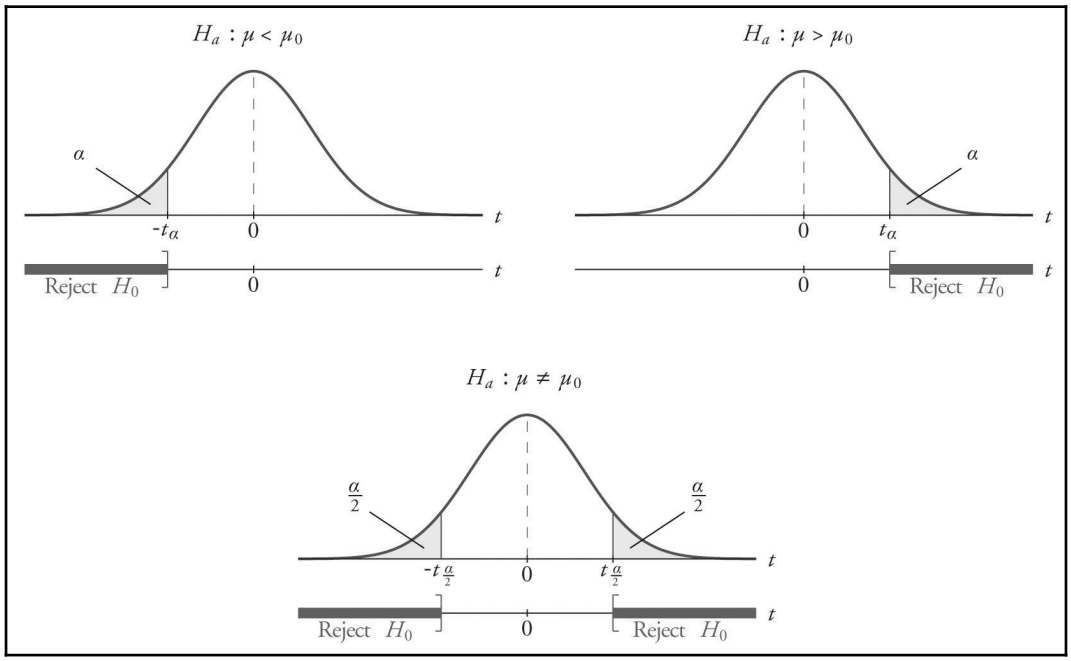

Figure 6.4: Interpretation of the research's decision rule based on a t-distribution

6.5.1 Implication of the Results to the IT Industry

Businesses rely on computer-based information systems that act as enablers of business processes. Unlike computer science, which is primarily concerned with the engineering of technologies that make up computer-based information systems, Information Technology (IT) is concerned with the practical application of computer-based information systems. This application can be in the context of a business or enterprise to support the storage and manipulation of business-related data as well as the processing and analysis of information to generate knowledge. The knowledge is then used by the decision-makers in the business to inform the creation of policies for business processes which are in turn used to implement appropriate actions.

The role of IT, therefore, goes beyond engineering technologies and focuses on the actual, useful implementation of these technologies often in the context of a business. As the data from the current study show, variable environmental phenomena and runtime conditions imply that these systems periodically either breakdown or require maintenance. Implementation and maintenance of systems therefore forms a key role of IT departments in a business. Once a database system has been implemented, its performance needs to be maintained at an acceptable level. Hence the concept of performance tuning in database systems.

Unplanned downtime remains a constant threat to businesses. It not only causes considerable financial losses, but also inability to deliver to clients, losses in production/productivity, vulnerability to security threats, unbudgeted expenses to fix the system, and loss of valuable data. Figure 6.5 shows that reactive maintenance exposes businesses to unplanned downtime.

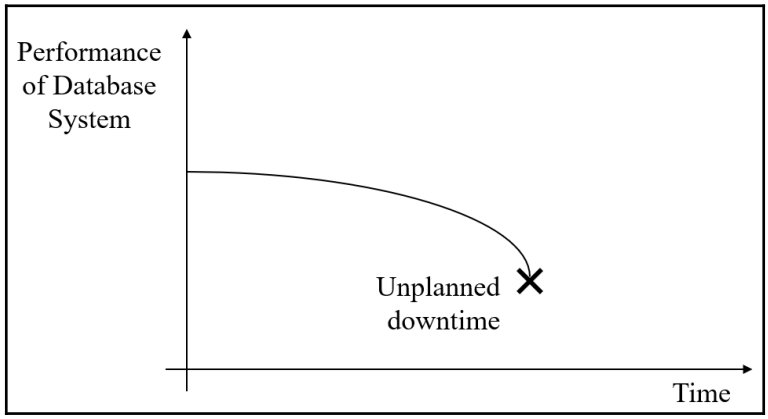

Figure 6.5: Reactive maintenance and unplanned downtime

An antidote to minimize unplanned downtime and maximize the time when the database system's performance is at an optimum level, is to conduct preventive maintenance as shown in Figure 6.6.

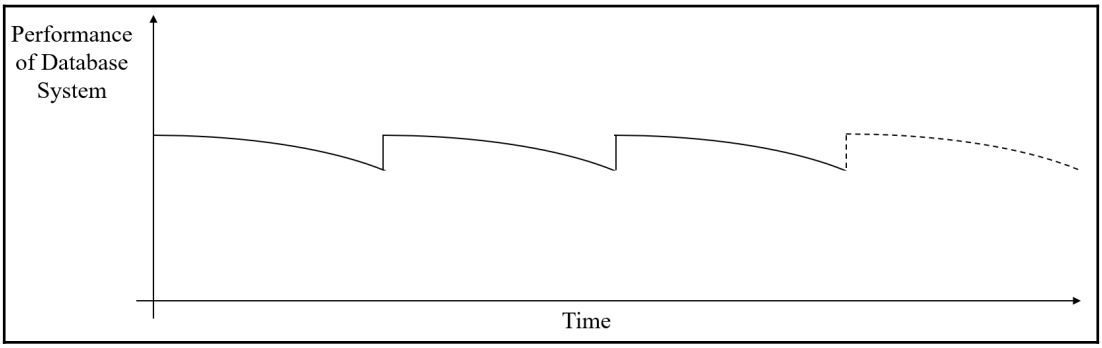

Figure 6.6: Preventive maintenance without unplanned downtime

However, one of the biggest challenges with preventive maintenance is determining when to do the maintenance. The current study implies that using multiple simulations takes advantage of probabilistic reasoning to estimate the Mean-Time-To-Failure (MTTF). As shown in Figure 6.7, there are multiple possible estimates of the time when the database

system's performance will be at an unacceptable level. The higher the number of simulations using the mathematical model (essentially a digital twin) of the database system, the more confident one can be of when and how to tune the database system. This results in picking one of the three possible trajectories shown in Figure 6.7.

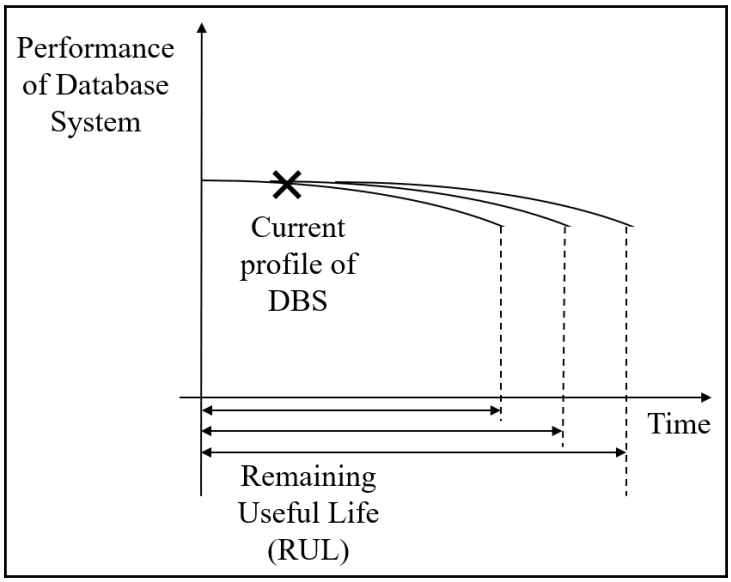

Figure 6.7: Use of multiple simulations to gain confidence

The results of this study subsequently promote the application of the process outlined in Figure 6.8. As opposed to running this process with the aim of predicting the future state of the database system, the results of the research favour the application of the process in a chronological manner based on time in order to determine what to do at the present moment to control or maintain the performance of the database system. The driving force behind this perspective is the claim that *"prediction based on a large time horizon is an exercise in futility"*.

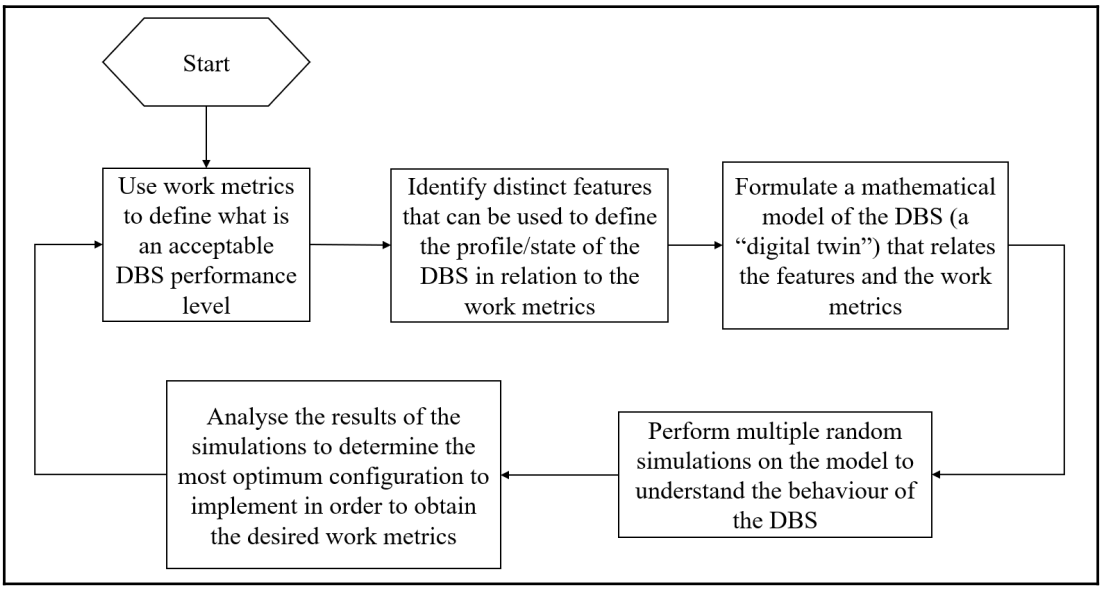

Figure 6.8: Performance tuning maintenance process in database systems

6.6 Summary

A discussion providing an interpretation and implication of the results of the experiment provides an important section of this thesis. Placing the research in the context of performance tuning of database systems contributes towards realizing its value. The value is further emphasized by comparing and contrasting it with similar research in various fields including autonomic computing and MCTS. Literature supporting the potential benefits of applying MCTS is reviewed alongside contradicting literature of performance tuning techniques in database systems. The contradicting literature advocates for the use of concepts such as horizontal and vertical partitioning, horizontal and vertical scaling, and the use of EDWs (Appendix D) to achieve load scalability. This research shows that performance tuning based on probabilistic reasoning implemented by an MCTS-based algorithm is a viable option as opposed to the propositions made in the contradicting literature. The research also obtained unexpected results that favour sustainable energy. A discussion of the value of these unexpected results is discussed. Lastly, the use of digital twins in predictive maintenance (PdM) of IT infrastructure in enterprises is submitted as a necessary step going forward. The following Chapter states the recommendations to IT practitioners, policy-makers, and scholars. It concludes the thesis by providing suggestions for future research.

Chapter 7: Conclusions and Recommendations

7.1 Conclusions

The study set out to investigate on how to design an algorithm that proactively reconfigures bottleneck parameters without over-relying on an accurate model of a stochastic environment. This was done in order to achieve self-optimization required for load scalability in database systems. The study concludes that the use of self-optimizing systems can mitigate against factors such as fatigue, long reaction times, and inconsistent expertise amongst system administrators. This mitigation can prevent sub-standard load scalability in database systems and simultaneously enable system administrators to free their minds from mundane tasks in order to concentrate on previously unachievable or difficult tasks. Doing this essentially saves (it does not eliminate) on human resource as it reduces labour requirements per unit of output produced. As explained in the current study, this is a critical foundation required to make progress in the evolution of human beings.

One of the most significant findings to emerge from this study is that applying a combination of UCT and "lean LGRF" to variate the selection stage of an MCTS algorithm results in superior performance. The experiments confirmed that it is indeed possible to reduce occurrences of sub-standard performance tuning through the use of a variated MCTS algorithm that implements principles of self-optimization.

7.2 Recommendations

7.2.1 Recommendations for Policy Makers

Systems inevitably experience numerous changes in environmental variables as well as changes in runtime phenomena. This includes unexpected input in form of a workload. Leaving a system to run without continuously changing its default configurations either reactively or proactively is not good because it prevents the system from handling unexpected workloads. No single configuration caters for every possible workload. This particular finding points to the need for policies that ensure developers of complex systems expose configuration parameters to system administrators so that once a bottleneck is identified, an appropriate treatment can be applied.

The identification of bottlenecks also relies on the ability to profile a system in order to gather symptoms. It is therefore necessary to establish a policy that requires system developers to expose profiling tools to system administrators. These profiling tools

should be able to gather work metrics and resource metrics. By having access to a system's profile, system administrators and IT practitioners can be more aware of when systems are due for maintenance. This inevitably reduces unplanned downtime.

7.2.2 Recommendations for IT Practitioners

The study recommends that if a business can automate the response to a technical system issue, by all means, it should consider doing so. This is partly in relation to the high cost of calling system administrators to address the issue during non-working hours. As demonstrated in the study, this can be made possible through the use of Artificially Intelligent (AI), Reinforcement Learning algorithms that are based on a variated Monte Carlo Tree Search.

It is also important to note that not all technical system issues are emergencies. It is not worth alerting system administrators on technical system issues that are only internally noticeable and that may revert to normal levels without intervention. If not careful, alert fatigue can cause a serious issue to be ignored. However, in the event that it is a genuine emergency and the system administrators need to be informed urgently, it is preferable to inform them on symptoms and not causes of the symptoms. Symptoms in this case are a manifestation of issues that may have a number of different causes. The study recommends that symptoms related to work metrics (high-level observations) should be categorized into either unacceptable levels of throughput, or response-time latency, or workload-related error-rates. The notification of the symptoms must then be followed by a diagnosis to identify the root cause or causes.

The study further recommends that resource metrics and event metrics should be collected periodically to support the reconstruction of detailed snapshots of the system's state. It is these snapshots that can then be used to diagnose the cause of the symptoms identified through work metrics. Resource metrics should be based on the following resources: compute (CPU), primary storage/memory (RAM), secondary storage (HDD/SSD), and network interfaces such that memory and secondary storage are the most critical resources for database systems. The symptoms collected for these resources should subsequently be categorized into either unacceptable levels of resource availability, or resource utilization, or resource saturation, or resource-related error-rate.

On the other hand, event metrics automatically capture an action or occurrence that

originates asynchronously from the external environment and is recognized by the system. For example, internally generated alerts that communicate the status of critical work such as completion of a task like data backup. The definition of what can be considered as critical work in this case is dependent on the system administrator.

Once the work metrics have been observed, and the resource and event metrics used to gain further insight, then the next stage that the study recommends is to diagnose the cause of the symptoms observed. This is often the least structured stage and is largely driven by subjective decisions. IT practitioners should also use the observations to make a prognosis of the likely course of the system's state and to be proactive. Wrapper techniques such as, but not limited, to step-wise regression, can be used to identify the bottlenecks (cause of the symptoms). The study therefore submits the use of self-optimization techniques as a means to standardize the diagnosis and prognosis stages of performance tuning. It is at this point that the study's key contribution, the designed algorithm, can be used to determine the appropriate treatment to be administered to the bottlenecks.

The study recommends the channelling of OLAP and OLTP workloads to different servers. This can work best using the architecture described in Section 3.3.3. In such a case, either the load balancer or the actual business information system can be used to channel OLAP workloads to a specific member(s) of the cluster and OLTP workloads to another member(s) of the cluster. The automatic latency-aware algorithm should then run on each member of the cluster and configure the node based on the characteristic workload it serves. This recommendation is made in view of the complexities of majority of business information systems that have a hybrid of OLTP and OLAP systems. Also, most of the bottlenecks that affect OLTP workloads are different from those that affect OLAP workloads. A possible explanation for this is that the two types of workloads can use different storage engines. It will be counterproductive to readjust configurations based on dynamic and unpredictable characteristics of workloads because it is often the case that OLTP-friendly configurations contradict OLAP-friendly configurations and vice versa. The findings of the study provide compelling evidence that the profile of the database system as a whole is a better foundation to determine which configurations to adjust. This is as opposed to a profile of the workload submitted to the database system.

Lastly, given the numerous and diverse configuration values of parameters, the study recommends categorizing parameter values into tactics. Implementing one tactic can

therefore involve the implementation of numerous configuration parameters that belong to the tactic. Working with tactics as opposed to individual parameters supports the construction of a reasonable decision tree or influence diagrams required by AI-based algorithms that make complex decisions.

7.2.3 Recommendation for Researchers

Numerous opportunities to extend the work further in multi-disciplinary research exist. Further research could analyse the possibility of applying the theoretical concepts in non-linear adaptive control in the aerospace industry, non-linear adaptive control of communication systems in the telecommunications industry, and in exploration of the potential benefits of adaptive control in Software Defined Everything (SDx). The study therefore recommends that scholars can apply a multi-disciplinary approach because it combines expertise from various fields. This can lead to creative high-impact research. However, a multi-disciplinary perspective should be approached with caution because of the lack of the potential meaningful evaluation from the team. Domain-specific concepts tend to be accepted without question or rejected without constructive criticism in multi-disciplinary research.

7.3 Suggestions for Future Research

A natural progression of this work is to investigate on self-optimization of NoSQL-based implementations such as graph, document, and column-family data models. An investigation of self-optimization of array/matrix data models would also help us to establish a greater degree of generalizability for AI and machine-learning workloads. This is particularly useful for analytical processing applied in business intelligence and big data applications. It would also be interesting for future research to investigate on the application of self-optimization in legacy systems that use hierarchical and network data models. This would be useful due to the bottleneck caused by the use of legacy systems to access data archives.

Further research on application of PB selection variation in the context of a relaxed computational time budget also needs to be conducted. The level of relaxation in this case can be determined by Service Level Agreements (SLAs) or by the researcher.

One of the surprising results obtained in this study was the fact that the

“tmp_table_size” configuration parameter affects OLTP workloads instead of OLAP workloads. It will be interesting to conduct further research to obtain an explanation for this odd phenomenon. In addition, future research can also focus on identifying additional metrics that can be used to define the state of a database system. This can go beyond software related database system work metrics to focus on mechanical, electrical and other physical engineering states of the server’s hardware when modelling and defining the system’s profile. The use of Kalman filters to perform this modelling as opposed to ODEs should also be explored further.

References

Ameri, P. (2016). *Challenges of index recommendation for databases: With specific evaluation on a NoSQL database.* Presented at the 28th GI-Workshop on Foundations of Databases, Nörten-Hardenberg, Germany.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., et al. (2010). A view of cloud computing. *Communications of the Association of Computing Machinery*, *53*(4), 50–58. https://doi.org/10.1145/1721654.1721672

Astrup, R., Coates, K. D. & Hall, E. (2008). Finding the appropriate level of complexity for a simulation model: An example with a forest growth model. *Forest Ecology and Management*, *256*(10), 1659–1665. https://doi.org/10.1016/j.foreco.2008.07.016

Autor, D., H. (2015). Why are there still so many jobs? The history and future of workplace automation. *The Journal of Economic Perspectives*, *29*(3), 3–30. https://doi.org/10.1257/jep.29.3.3

Bach, F. (2017). Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, *18*(19), 1–53.

Barroso, L. A. & Hölzle, U. (2007). The case for energy-proportional computing. *Computer*, *40*(12), 33–37. https://doi.org/10.1109/MC.2007.443

Bousdekis, A., Magoutas, B., Apostolou, D. & Mentzas, G. (2015). A proactive decision making framework for condition-based maintenance. *Industrial Management & Data Systems*, *115*(7), 1225–1250. https://doi.org/10.1108/imds-03-2015-0071

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., … Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, *4*(1), 1–43.

Bucher, T. (2012). 'Want to be on the top?' Algorithmic power and the threat of invisibility on Facebook. *New Media and Society*, *14*(7), 1164–1180.

Cámara, J., Moreno, G. A. & Garlan, D. (2014). Stochastic Game Analysis and Latency Awareness for Proactive Self-adaptation. In *Proceedings of the 9th International*

Symposium on Software Engineering for Adaptive and Self-Managing Systems (pp. 155–164). New York, NY, USA: ACM. https://doi.org/10.1145/2593929.2593933

Chajewska, U. K., Koller, D. & Parr, R. (2000). Making rational decisions using adaptive utility elicitation. In *American Association for Artificial Intelligence* (pp. 363–369). Retrieved from http://www.aaai.org/Papers/AAAI/2000/AAAI00-056.pdf

Chaudhuri, S. & Narasayya, V. (2007). Self-Tuning Database Systems: A Decade of Progress. In *Proceedings of the 33rd international conference on Very Large Databases* (pp. 3–14).

Cheng, H., Hao, L., Luo, Z. & Wang, F. (2016). Establishing the connection between control theory education and application: An arduino based rapid control prototyping approach. *International Journal of Learning and Teaching*, *2*(1), 67–72.

Cheng, S. W. & Garlan, D. (2012). Stitch: A language for architecture-based self-adaptation. *Journal of Systems and Software*, *85*(12), 2860–2875.

Coulom, R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. *Computers & Graphics*, 72–83. Turin, Italy.

Damasio, G., Mierzejewski, P., Szlichta, J. & Zuzarte, C. (2016). Query performance problem determination with knowledge base in semantic web system OptImatch. In *International Conference on Extending Database Technology* (pp. 515–526). Bordeaux, France. Retrieved from http://edbticdt2016.labri.fr/

David, H., Fallin, C., Gorbatov, E., Hanebutte, U. R. & Mutlu, O. (2011). Memory power management via Dynamic Voltage/Frequency Scaling. In *Proceedings of the 8th ACM International Conference on Autonomic Computing* (pp. 31–40). Karlsruhe, Germany: ACM. https://doi.org/10.1145/1998582.1998590

Den Teuling, N. G. P. & Winands, M. H. M. (2011). *Monte-Carlo Tree Search for the simultaneous move game Tron*. Netherlands: University of Maastricht. Retrieved from https://dke.maastrichtuniversity.nl/m.winands/documents/Tronpaper.pdf

Deng, Q., Ramos, L., Bianchini, R., Meisner, D. & Wenisch, T. (2012). Active low-power

modes for main memory with MemScale. *IEEE Micro*, *32*(3), 60–69. https://doi.org/10.1109/MM.2012.21

Desprez, F., Fox, G., Jeannot, E., Keahey, K., Kozuch, M., Margery, D., … Richard, O. (2012). *Supporting experimental computer science* (Technical Memo No. 362). Retrieved from Argonne National Laboratory website: https://hal-hcl.archives-ouvertes.fr/GRID5000/hal-00720815v1

Dewan, S., Aggarwal, Y. & Tanwar, S. (2013). Review on Data Warehouse, Data Mining and OLAP Technology: As Prerequisite aspect of business decision-making activity. *International Journal of Research in Information Technology*, *1*(10), 30–39.

Diakopoulos, N. (2015). Algorithmic Accountability: Journalistic investigation of computational power structures. *Digital Journalism*, *3*(3), 398–415. https://doi.org/10.1080/21670811.2014.976411

Duan, Y., Chen, X., Houthooft, R., Schulman, J. & Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In M. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd International Conference on Machine Learning (ICML)* (Vol. 48). New York, NY, USA: Association of Computing Machinery.

Gelly, S., Wang, Y., Munos, R. & Teytaud, O. (2006). *Modification of UCT with Patterns in Monte-Carlo Go*. Institut National de Recherche en Informatique et en Automatique.

Information and Communication Technology Authority (ICTA). (2016). *Electronic Records and Data Management Standard* (Standard No. ICTA-4.001:2016) (pp. 1–88). Kenya: Information and Communication Technology Authority. Retrieved from http://icta.go.ke/standards/electronic-records-management-standard/

Ministry of Information Communications and Technology (MoICT). (2014). *The Kenya national ICT Masterplan 2014-2017: Towards a Digital Kenya* (Master Plan) (pp. 1–130). Nairobi, Kenya: Ministry of Information Communications and Technology. Retrieved from http://icta.go.ke/national-ict-masterplan/

Hahn-Goldberg, S., Beck, J. C., Carter, M. W., Trudeau, M., Sousa, P. & Beattie, K. (2014). Solving the chemotherapy outpatient scheduling problem with constraint programming. *Journal of Applied Operational Research*, *6*(3), 135–144.

Hansen, E. A., Shi, J. & Khaled, A. (2016). A POMDP approach to influence diagram evaluation. In *Twenty-Fifth International Joint Conference on Artificial Intelligence* (pp. 3124–3132). Palo Alto, CA, USA: AAAI Press / International Joint Conferences on Artificial Intelligence. Retrieved from https://www.ijcai.org/proceedings/2016

Hartmann, D. (2017). Let's catch the train to Monte-Carlo. *International Computer Games Association Journal*, *39*(1), 44–46.

Henningsson, S. (2015). Learning to acquire: How serial acquirers build organisational knowledge for information systems integration. *European Journal of Information Systems*, *24*(2), 121–144. https://doi.org/0960-085X/14

Hjørland, B. (2005). Empiricism, rationalism and positivism in library and information science. *Journal of Documentation*, *61*(1), 130–155. https://doi.org/10.1108/00220410510578050

Ishibashi, K., Iwasaki, T., Otomasa, S. & Yada, K. (2016). Model selection for financial statement analysis: Variable selection with data mining technique. In *Procedia Computer Science* (Vol. 96, pp. 1681–1690). York, United Kingdom: Elsevier. https://doi.org/10.1016/j.procs.2016.08.216

Jimoh, F. & McCluskey, T. L. (2016). Self-management in urban traffic control: An automated planning perspective. In A. Kotsialos, J. Müller, F. Klügl, O. Rana & R. Schumann (Eds.), *Autonomic Road Transport Support Systems* (pp. 29–46). Switzerland: Springer International Publishing.

Kathuria, M., Nagpal, C. K. & Duhan, N. (2016). Journey of Web Search Engines: Milestones, Challenges & Innovations. *International Journal of Information Technology and Computer Science*, *12*, 47–58.

Kephart, J. O. & Chess, D. M. (2003). The vision of autonomic computing. *Computer*,

36(1), 41–50.

Kim, J. W., Cho, S. H. & Kim, I. M. (2016). Workload-Based column partitioning to efficiently process data warehouse query. *International Journal of Applied Engineering Research*, *11*(2), 917–921.

Kitchin, R. (2017). Thinking critically about and researching algorithms. *Information, Communication & Society*, *20*(1), 14–29. https://doi.org/10.1080/1369118X.2016.1154087

Kocsis, L. & Szepesvári, C. (2006). Bandit based monte-carlo planning. Presented at the European conference on machine, Heidelberg, Berlin. Retrieved from https://web.engr.oregonstate.edu/~afern/classes/cs533/notes/uct.pdf

Lanctot, M., Wittlinger, C., Winands, M. H. M. & Den Teuling, N. G. P. (2013). Monte-Carlo Tree Search for the simultaneous move game Tron. In *BNAIC 2013: Proceedings of the 25th Benelux Conference on Artificial Intelligence* (pp. 104–111). Delft, Netherlands: Delft University of Technology. Retrieved from https://dke.maastrichtuniversity.nl/m.winands/documents/Tronpaper.pdf

Lee, K., König, A., Narasayya, V., Ding, B., Chaudhuri, S., Ellwein, B., … Naughton, J. (2016). Operator and query progress estimation in Microsoft SQL Server live query statistics. *Proceedings of the 2016 International Conference on Data*, 1753–1764. Retrieved from http://sigmod2016.org/

Lemke, T. (2015). Varieties of materialism. *BioSocieties*, *10*(4), 490–495. https://doi.org/10.1057/biosoc.2015.41

Lewis, F. L., Vrabie, D. & Vamvoudakis, K. G. (2012). Reinforcement Learning and Feedback Control: Using Natural Decision Methods to Design Optimal Adaptive Controllers. *IEEE Control Systems Magazine*, *32*(6), 76–105. https://doi.org/10.1109/MCS.2012.2214134

Li, T., Chunqiu, Z., Jiang, Y., Zhou, W., Tang, L., Liu, Z. & Huang, Y. (2017). Data-Driven Techniques in Computing System Management. *ACM Computing Surveys*, *50*(3), 45–88. https://doi.org/10.1145/3092697

Ma, X. G. & Liu, X. (2016). A new branch and bound algorithm for integer quadratic programming problems. *Journal of Nonlinear Science and Applications*, 1153–1164.

Moreno, G. A., Papadopoulos, A. V., Angelopoulos, K., Cámara, J. & Schmerl, B. (2017). Comparing Model-Based Predictive Approaches to Self-Adaptation: CobRA and PLA. In *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)* (pp. 42–53). Buenos Aires, Argentina: IEEE/ACM. https://doi.org/10.1109/SEAMS.2017.2

Moreno, Gabriel A., Cámara, J., Garlan, D. & Schmerl, B. (2018). Flexible and Efficient Decision-Making for Proactive Latency-Aware Self-Adaptation. *ACM Trans. Auton. Adapt. Syst.*, *13*(1), 3:1–3:36. https://doi.org/10.1145/3149180

Mykoniatis, K. (2015). *A Generic framework for multi-method modeling and simulation of complex systems using discrete event, system dynamics and agent based approaches.* (University of Central Florida). Retrieved from https://stars.library.ucf.edu/etd/1391/

Nair, S. (2017, December 29). Simple Alpha Zero. Retrieved 12 February 2019, from A Simple Alpha(Go) Zero Tutorial website: https://web.stanford.edu/~surag/posts/alphazero.html

Omondi, A. O., Lukandu, I. A. & Wanyembi, G. W. (2018). Scalability and Nonlinear Performance Tuning in Storage Servers. *International Journal of Research Studies in Science, Engineering and Technology*, *5*(9), 7–18. Retrieved from http://ijrsset.org/pdfs/v5-i9/2.pdf

Peterson, J. B. & Djikic, M. (2003). You can neither remember nor forget what you do not understand. *Religion and Public Life*, *33*, 85–118.

Pynadath, D. V. & Marsella, S. C. (2005). PsychSim: Modeling theory of mind with decision-theoretic agents. In *Proceedings of the 19th international joint conference on Artificial intelligence* (pp. 1181–1186). Morgan Kaufmann Publishers Inc.

Qian, C., Yu, Y. & Tang, K. (2018). Approximation Guarantees of Stochastic Greedy

Algorithms for Subset Selection. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (pp. 1478–1484). Freiburg, Germany. Retrieved from https://www.ijcai.org/proceedings/2018/

Ranadip, R. (2018). Application of adaptive extended Kalman filter algorithm in parameter estimation of wind turbine. *International Journal of Engineering Science and Computing*, *8*(8), 18768–18772.

Sadatrasoul, S., Gholamian, M. & Shahanaghi, K. (2015). Combination of Feature Selection and Optimized Fuzzy Apriori Rules: The Case of Credit Scoring. *International Arab Journal of Information Technology (IAJIT)*, *12*(2), 138–145.

Saha, B. (2018). Green computing: Current research trends. *International Journal of Computer Sciences and Engineering*, *6*(3), 467–469.

Sanders, P. (2009). Algorithm engineering: An attempt at a definition. In S. Albers, H. Alt & S. Naher (Eds.), *Efficient Algorithms: Lecture Notes in Computer Science* (pp. 321–340). Heidelberg, Berlin: Springer-Verlag. Retrieved from https://doi.org/10.1007/978-3-642-03456-5_22

Seaver, N. (2013). Knowing Algorithms. *Media in Transition*, *8*, 1–12.

Sellner, J. (2019). Introduction to the Hessian feature detector for finding blobs in an image - Milania's Blog [Blog]. Retrieved 3 February 2019, from Introduction to the Hessian feature detector for finding blobs in an image website: https://milania.de/blog/Introduction_to_the_Hessian_feature_detector_for_finding_blobs_in_an_image

Shahapure, N. H. & Jayarekha, P. (2014). Load Balancing in Cloud Computing: A Survey. *International Journal of Advances in Engineering & Technology*, *6*(6), 2657–2664.

Stankiewicz, J., Winands, M. H. M., Uiterwijk, J. W. & Jos, W. H. M. (2011). Monte-Carlo Tree Search enhancements for Havannah. In *Advances in Computer Games* (pp. 60–71).

Su, G., Chen, T., Feng, Y., Rosenblum, D. S. & Thiagaranj, P. S. (2016). An iterative decision-making scheme for Markov decision processes and its application to self-

adaptive systems. Presented at the 19th International Conference Fundamental Approaches to Software Engineering (FASE 2016), Eindhoven, Netherlands: Middlesex University. Retrieved from http://eprints.mdx.ac.uk/19199/

Sullivan, D. G. (2003). *Using probabilistic reasoning to automate software tuning* (Doctoral Dissertation, Harvard University). Retrieved from https://www.researchgate.net/profile/David_Sullivan16/publication/221595676_Using_probabilistic_reasoning_to_automate_software_tuning/links/55cb36dc08aeb975674ad1d3.pdf

Taleb, N. N. & Blyth, M. (2011). The black swan of Cairo: How suppressing volatility makes the world less predictable and more dangerous. *Foreign Affairs*, *90*(3), 33–39.

Van Aken, D., Pavlo, A., Gordon, G. J. & Zhang, B. (2017). Automatic Database Management System Tuning through Large-Scale Machine Learning (pp. 1009–1024). Presented at the ACM International Conference on Management of Data, Chicago, IL, USA: Association for Computing Machinery. https://doi.org/10.1145/3035918.3064029

Appendices

Appendix A: System Requirements Specification

A.1. System Capabilities

The system capabilities in Table A.1 outline the fundamental requirements that are needed to implement the solution.

Table A.1: System capabilities

Capability number	Capability
Cap.1	The system should be capable of maximizing the performance of a database system
Cap.2	The system should be capable of either directly measuring or estimating the state of the database system
Cap.3	The system should be capable of applying feedback from the current state to make decisions for future actions
Cap.4	The system should be capable of being autonomous without the need for human intervention

A.2. System Conditions

The system conditions in Table A.2 provide measurable quantitative characteristics stipulated for the capabilities identified in the previous section.

Table A.2: System conditions

Condition number	Reference	Condition
Cond.1	Cap.1	The performance of a database system should be based on its transaction throughput and its response time latency. These two elements should be measured in transactions per second and in microseconds respectively. A higher value of transactions per second and a lower response-time latency is preferred.
Cond.2	Cap.1	The database system should run on a dedicated server whose performance is not affected by other programs or services
Cond.3	Cap.1	The performance of the database system should be directly determined only by the designed algorithm according to the principles of temporal precedence (internal validity)
Cond.5	Cap.3	The system should give a higher priority to immediate rewards than to uncertain future rewards

Cond.4	Cap.4	The algorithm should perform the action (bottleneck parameter reconfiguration) that will yield the best result given its current state
Cond.6	Cap.4	The system should be able to apply principles of Reinforcement Learning to learn on its own which actions provide the highest rewards given a specific state

A.3. System Constraints

The system constraints in Table A.3 are imposed on the system by circumstance.

Table A.3: System constraints

Constraint number	Reference	Constraint
Cons.1	Cond.1	The performance of a system should not be lower than a threshold defined by the user. This threshold should be defined based on the desired transaction throughput, measured in transactions per second, and response-time latency, measured in microseconds.
Cons.2	Cond.2	The designed algorithm that is running on the dedicated server should make use of either a non-production database system or a virtual database system environment in order to experiment on the effect of implementing different control measures and treatments
Cons.3	Cond.6	The system should seek to gain confidence of the state that yields the maximum reward through at least 1 million simulations based on MCTS, by applying either Q-learning, SARSA, or the Temporal Difference Reinforcement Learning algorithms

***Appendix B*: System Modelling**

The following approach was adopted in modelling the dynamic system:

- ***Step i.*** Define the system and its components
- ***Step ii.*** Formulate a basic mathematical model and fundamental assumptions informed by theory
- ***Step iii.*** Obtain a detailed differential equation based on the mathematical model
- ***Step iv.*** Solve the equation for the expected output variables depending on the simulation performed
- ***Step v.*** Examine the solutions and assumptions
- ***Step vi.*** If necessary, reanalyse or redesign the system

A tuple of discrete, quantitative observations periodically sensed by a sensor can form a vector and be used to represent a specific point in the state space of a database system. An actuator can map this vector (the state) to actions, which when performed, gradually result in a different state sensed again by a sensor. The entire system operates in a continuous cycle such that a change in any aspect of the system depends on other sensed aspects of the system. An anticipation of the behaviour of the system can thus be obtained by observing how it transitioned from distinct states in the past. One can argue that breaking the system's overall process of functioning into discrete time steps may not fully capture this behaviour, however, the use of differential equations, where the transition from one step to another tends to zero ($\lim_{\Delta x \to 0}$), can be applied. Applying differential equations therefore supports the creation of an adequate model of a dynamic system as it evolves from one state to another over time. This research argued that this can result in a model that is based on internal (parameter configurations) and external (workload and the number of concurrent users) forces that shape a specific trajectory. This is as opposed to being based on the underlying mechanical, electrical or any other physical engineering state of the server's hardware.

As the number of concurrent users and the workload to be processed by the database system increases, the performance deteriorates due to poor load scalability. The results of the experiments conducted confirmed this basic fact as true when the database system was left to run in its default state without any form of adaptation or reconfiguration

of parameters. This can be represented as:

$$performance_{next} = \frac{1}{\text{workload} + \text{number of concurrent users}} \times performance_{current}$$

This can then be extended to consider internal forces composed of parameter configurations thus forming a foundation for the ODE in the form:

$$\frac{dP}{dx} = f(t_p, y_w, r_p, z_w, e_p, a_p, qps, u)$$

Such that:

$$f(t_p, y_w, r_p, z_w, e_p, a_p, qps, u) = \sum_{p \in T} \left(\frac{(t_p) y_w + 1}{(r_p) z_w + 1 + e_p + a_p} \right) \times \left(\frac{1}{qps + u} \right)$$

Where each parameter, *p*, in a tactic, *T*, $p \in T$ is characterized by:

(i) its effect on the transaction throughput (*t*)

(ii) its effect on the response-time latency (*r*)

(iii) its negative effect on the hardware resources of the distributed database (*e*)

(iv) its adaptation latency (*a*)

(v) the number of active concurrent users (*u*)

(vi) queries per second submitted to the database system by all concurrent users (q*ps*)

The effect in (iii) can be measured using resource metrics explained in Appendix F.

And:

y_w, z_w

Such that:

$$y_w = \begin{cases} 1 & \text{workload } w \text{ is an OLTP workload} \\ 0 & \text{otherwise} \end{cases}$$

$$z_w = \begin{cases} 1 & \text{workload } w \text{ is an OLAP workload} \\ 0 & \text{otherwise} \end{cases}$$

Differential calculus cuts the performance of the database system into small pieces to find out how it changes over time. Whereas integral calculus joins the small pieces together to find out how much change there is over time. The ODE thus seeks to determine the rate of change in the performance of the database system with respect to the workload

required to be processed. Although ODEs are useful in modelling a system, they are generally hard to use in an implementation. By solving them through integration, they become even more useful and easier to use in an implementation. Therefore, the solution to the ODE will be as follows:

$$F(x)=\int\left(\frac{(t_p)\,y_w+1}{(r_p)\,z_w+1+e_T+a_T}\right)\frac{1}{qps+u}\times P\,dx$$

$$F(x)=(t_p)\,y_w+1\int\frac{1}{(r_p)\,z_w+1+e_T+a_T}\times\int\frac{1}{qps+u}$$

$$F(x)=\left((\frac{t_p^2}{2})\,y_w\ \times\ \ln\left|(r_p)\,z_w\right|\right)\times\ln\left|e_T+a_T\right|\ +\ \ln\left|qps+u\right|$$

Where $F(x)$ represents the trajectory of the cumulative performance of the database system.

Appendix C: Empirical Algorithmics

Various methods can be used to advance critical understanding of algorithms that have made efforts towards implementing self-optimization in computing. This Appendix outlines common methods available for use in research in algorithms according to literature (Diakopoulos, 2015; Kitchin, 2017).

C.1. Examining Pseudo-Code or Source Code

This involves using the algorithm's source code, programmers' comments, and documentation to determine how the algorithm works to process input in order to produce output. This method requires the appraiser to have programming skills and to be familiar with the domain which the algorithm is operating in. However, it is difficult to trace how an algorithm mutates over time in the case of dynamic or adaptive algorithms.

C.2. Reflexive Production of Code

This method involves the analysis of the algorithm's objective, followed by an identification of the required tasks needed to achieve the objective, and the conversion of the results of the analysis into pseudo-code. The pseudo-code is then eventually converted into actual code. This method provides insights into how algorithms are designed. However, there is a tendency to be biased and subjective.

C.3. Reverse Engineering

Even though one of the limitations of appraising algorithms is that many are opaque and closed to public scrutiny, Kitchin (2017) points out that two openings arise: input and output. By examining what goes out based on what comes in, then the reverse engineering process of knowing what is done to the input in order to produce the output can begin. In this case, dummy data is fed as input and an observation made on what output is produced. In as much as dummy data can be used as input and based on this, the output is observed, the algorithm remains closed-source. This implies the inner workings of the algorithm are not analysed. A study by Diakopoulos (2015) showed that reverse engineering provides fuzzy glimpses of how an algorithm works in practice but not its actual constitution.

C.4. Interviewing Algorithm Creators

This involves interviewing the algorithm's creators to find out how they framed the goal of the algorithm, the tasks it must perform to tend towards achieving the goal, the instructions of how the tasks should be executed, how they converted the tasks into pseudo-code, and how they translated the pseudo-code into actual code. However, this research did not use this method due to the difficulty in finding the original authors of self-optimization algorithms.

C.5. Unpacking the Full Socio-Technical Assemblage of Algorithms

Given that algorithms form part of a wider technological stack, this method requires that the algorithm's context needs to be examined in order to fully understand the algorithm itself. The wider context that frames the algorithm includes the systems of thought, forms of knowledge, political economies, legal issues, institutions, among other contexts. This method should be undertaken by a research team rather than a single individual due its large-scale nature.

Appendix D: Comparative Analysis of Enterprise Data Warehouse Data Models

Table D.1: A comparative analysis of EDW data models

	Normalized	Dimensional	Anchor	Data Vault
Leading proponent	Edgar Frank Codd in 1970	Ralph Kimball in 1960s	Lars Rönnbäck and Olle Regardt in 2004	Dan Linstedt in 2000
Fundamental concepts	The most widely used data model for transactional systems. It is defined as a formal mathematical model based on set theory. It uses a relation to represent an object, a foreign key to represent a relationship, a domain to represent an attribute, and a primary key to represent an identifier.	The most widely used data model for analytical systems. Facts (measures) represent a set of events taking place within a business system and dimensions (context) provide descriptive information about the facts. A star schema supports redundancy while a snow flake schema supports normalization. Dimensions and facts represent objects	It was developed to be suited for information that changes over time both in structure and content. This is done in such a way that it takes advantage of the benefits of a high degree of normalization (6NF) while avoiding its drawbacks. It uses an anchor or a knot to represent an object, a tie to represent a relationship, an attribute is represented as an attribute, and lastly a	It was developed in order to enable the complete traceability of data as well as greater scalability. It is a detail-oriented hybrid approach that encompasses the best of breed between 3NF and the star schema. A hub is used to represent an object, a link represents a relationship, a satellite represents an attribute, and lastly, a business/ primary key is used to represent an identifier.

	Normalized	Dimensional	Anchor	Data Vault
		and relationships respectively. Attributes and business/primary key represent attributes and identifiers respectively.	primary key is used to represent an identifier.	
Built-in semantics	With regards to data warehousing, it does not presume any semantic constraints. It also fails to provide implicit concepts which would enable the maintaining of the history of changes of an object as well as the changes of the values of the object's attributes.	It presumes built-in semantics. The tracking of the history of the changes is based on complex rules pertaining to changes in dimensions	It presumes built-in semantics that support the tracing of the history of all concepts except knots. The semantics also define a highly normalized data model (normalized up to the sixth normal form).	It has built-in semantics that allow for an object to have its structure split into several satellites. This is based on the assumption that business objects have a stable identifier.
Resilience to change in the structure of the data sources	Attributes and relationships are built into the structure of the data model's objects (relations). Any change in the structure of the enterprise's data sources requires	The addition of new attributes in the source model requires changing the structure of the corresponding data	Its high degree of normalization supports non-destructive changes in the modelling concepts in order to capture change. This is	It explicitly separates the structural information from descriptive attributes. This enables it to be resilient to change in the business environment.

	Normalized	Dimensional	Anchor	Data Vault
	a change in the structure of the objects in the normalized data model. This is a key disadvantage of this data model.	warehouse dimension. This requires the changing of the corresponding fact in order to include a foreign key that references the newly formed dimension.	done in such a way that every previous schema is not modified and always remains as a subset of the current schema. Thus allowing the evolution of the data warehouse without causing any downtime.	
Temporal aspect	Capturing temporal aspects (time-variant properties) is dependent upon the data warehouse designer's choice. The normalized data model does not provide built-in mechanisms for this.	It supports the tracking of valid time based on the depending on the choice of the Slowly Changing Dimension (SCD) Type. The transaction time is not incorporated into the structure of the concepts representing business objects, instead, it is recorded in separated log	It supports the tying of pieces of information to points or periods in time. It uses historization of attributes and tries to capture the intervals of time in which a value is valid. The attributes in turn are used to capture the exact time point. However, tracking the history of value	The data vault model assumes that the structure of objects will change over time. A transaction time is therefore incorporated into each concept (hub, link, and satellite). The time is thus applied to the structure of an object, and not the object itself nor its relationships.

	Normalized	Dimensional	Anchor	Data Vault
		tables.	changes is not possible when the static option is used to create the model.	
Completeness of data stored	It does not implicitly provide concepts to support the recording of data originating from multiple sources. Its weakness is exacerbated when the *Single Version of the Facts* concept is applied. This leads to redundancy which goes against the normalized data model.	It does not allow or support the recording of multiple values, originating from multiple sources for a single object during the same period of time	It is not possible to record that, for a single object, two identical values originating from two different sources, are both valid at the same point in time. This is caused by a unique constraint over the identifier attribute, the *Valid Time* attribute, and the value of an attribute.	It makes no distinction between data that conforms to business rules and data that does not conform to business rules. It therefore supports a single version of facts as opposed to a single version of truths concept.
Traceability of data to the source	Does not implicitly provide concepts to support the traceability of data originating from multiple sources. This is dependent upon the data warehouse designer to factor in	It does not implicitly provide concepts to support the tracing of data back to the sources.	The metadata concept allows for traceability of data. This is because all time-variant concepts (except anchors and knots) can reference the metadata	The data vault model emphasizes on the need to trace where all the data in the data warehouse come from. It specifies that every tuple must be accompanied by record source and load date attributes.

Normalized	Dimensional	Anchor	Data Vault
mechanisms that cater for the recording of additional metadata for each tuple.		thus capturing the source of the data.	These enable an auditor to trace values back to the source.

Appendix E: Stepwise Regression Runs

The following is a list of the features and their corresponding identification:

x_1: innodb_buffer_pool_size
x_2: innodb_buffer_pool_instances
x_3: innodb_old_blocks_pct
x_4: innodb_old_blocks_time
x_5: innodb_buffer_pool_dump_at_shutdown
x_6: innodb_buffer_pool_load_at_startup
x_7: query_cache_size
x_8: innodb_log_file_size
x_9: innodb_file_per_table
x_10: innodb_lock_wait_timeout
x_11: thread_cache_size
x_12: tmp-table-size
x_13: max-heap-table-size
x_14: sort-buffer-size
x_15: join_buffer_size

Table E.1, Table E.2 and Table E.3 provide the raw data used to select the most influential parameters during the first, second, and third runs respectively.

Table E.1: OLTP workload submitted concurrently by 20 users (1st run)

1st Run: y_1 = b_0 + b_1.x_3

y_1 regressed only on:	**x_1**	**x_2**	**x_3**	**x_4**	**x_5**	**x_6**	**x_7**	**x_8**	**x_9**	**x_10**	**x_11**	**x_12**	**x_13**	**x_14**	**x_15**
b_1	3.56E-07	-1.16E+01	3.13E+01	3.87E-02				1.85E-06			1.23E-02	-4.28E-06		-6.99E-05	
t-statistic	8.16E-01	-1.49E+00	2.40E+00	7.40E-02				2.84E-01			3.75E-01	-8.41E-01		-2.06E-01	
ABS(t-statistic)	8.16E-01	1.49E+00	**2.40E+00**	7.40E-02				2.84E-01			3.75E-01	8.41E-01		2.06E-01	
p-value	4.34E-01	1.66E-01	3.75E-02	9.43E-01				7.82E-01			7.15E-01	4.20E-01		8.41E-01	

Table E.2: OLTP workload submitted concurrently by 20 users (2nd run)

2nd Run: y_1 = b_0 + b_1.x_3 + b_2.x_12

y_1 regressed only on:	x_3 + x_1	x_3 + x_2	x_3 + x_3	x_3 + x_4	x_3 + x_5	x_3 + x_6	x_3 + x_7	x_3 + x_8	x_3 + x_9	x_3 + x_10	x_3 + x_11	x_3 + x_12	x_3 + x_13	x_3 + x_14	x_3 + x_15
b_2	1.44E-07	-8.67E+00		-4.31E-01				1.69E-07			-3.24E-04	-5.74E-06		1.80E-05	
t-statistic	3.68E-01	-1.28E+00		-9.39E-01				3.00E-02			-1.10E-02	-1.42E+00		6.20E-02	
ABS(t-statistic)	3.68E-01	1.28E+00		9.39E-01				3.00E-02			1.10E-02	**1.42E+00**		6.20E-02	
p-value	1.21E-01	6.12E-02		8.52E-02				1.30E-01			1.30E-01	5.20E-02		1.30E-01	

Table E.3: OLTP workload submitted concurrently by 20 users (3rd run)

3rd Run: y_1 = b_0 + b_1.x_3 + b_2.x_12 + b_3.x_2

y_1 regressed only on:	x_12 + x_3 + x_1	x_12 + x_3 + x_2	x_12 + x_3 + x_3	x_12 + x_3 + x_4	x_1 2 + x_3 + x_5	x_1 2 + x_3 + x_6	x_1 2 + x_3 + x_7	x_12 + x_3 + x_8	x_1 2 + x_3 + x_9	x_12 + x_3 + x_10	x_12 + x_3 + x_11	x_12 + x_3 + x_12	x_12 + x_3 + x_13	x_12 + x_3 + x_14	x_12 + x_3 + x_15
b_3	3.50E-08	-7.29E+00		-3.79E-01				-6.32E-07			-6.57E-03			5.98E-05	
t-statistic	9.10E-02	-1.09E+00		-8.56E-01				-1.18E-01			-2.40E-01			2.16E-01	
ABS(t-statistic)	9.10E-02	**1.09E+00**		8.56E-01				1.18E-01			2.40E-01			2.16E-01	
p-value	1.35E-01	8.14E-02		9.84E-02				1.35E-01			1.32E-01			1.33E-01	

Table E.4 and Table E.5 provide the raw data used to select the most influential parameters during the first and second runs respectively with an OLAP workload.

Table E.4: OLAP workload submitted concurrently by 20 users (1st run)

1st Run: y_1 = b_0 + b_1.x_3

y_1 regressed only on:	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_10	x_11	x_12	x_13	x_14	x_15
b_1	-3.33E-10	-1.64E-03	-3.13E-03	-1.00E-04				8.42E-10			5.72E-06	6.77E-10		-4.37E-08	
t-statistic	-1.94E+00	-4.30E-01	-4.30E-01	-4.30E-01				2.89E-01			2.89E-01	2.89E-01		7.79E-01	
ABS(t-statistic)	**1.94E+00**	4.30E-01	4.30E-01	4.30E-01				2.89E-01			2.89E-01	2.89E-01		7.79E-01	
p-value	8.16E-02	6.76E-01	6.76E-01	6.76E-01				7.79E-01			7.79E-01	7.79E-01		7.79E-01	

Table E.5: OLAP workload submitted concurrently by 20 users (2nd run)

2nd Run: y_1 = b_0 + b_1.x_1 + b_2.x_8

y_2 regressed only on:	x_1 + x_1	x_1 + x_2	x_1 + x_3	x_1 + x_4	x_1 + x_5	x_1 + x_6	x_1 + x_7	x_1 + x_8	x_1 + x_9	x_1 + x_10	x_1 + x_11	x_1 + x_12	x_1 + x_13	x_1 + x_14	x_1 + x_15
b_2		-3.51E-03	3.34E-17	7.12E-19				4.63E-09			-3.71E-20	-4.39E-24		4.53E-22	
t-statistic		-1.04E+00	0.00E+00	0.00E+00				1.73E+00			0.00E+00	0.00E+00		0.00E+00	
ABS(t-statistic)		1.04E+00	0.00E+00	0.00E+00				**1.73E+00**			0.00E+00	0.00E+00		0.00E+00	
p-value		1.43E-01	2.39E-01	2.39E-01				6.54E-02			2.39E-01	2.39E-01		2.39E-01	

Appendix F: Database System Metrics

Table F.1: Work metrics for detecting symptoms of problems

Category – Measurement	Name of Metric	Description	SQL Command
Throughput – Create	Com_insert	Counts the number of INSERT statements that have been sent to the database system	`SHOW GLOBAL STATUS LIKE 'Com_insert';`
Throughput – Read	Com_select	Counts the number SELECT statements that have been sent to the database system	`SHOW GLOBAL STATUS LIKE 'Com_select';`
Throughput – Update	Com_update	Counts the number UPDATE statements that have been sent to the database system	`SHOW GLOBAL STATUS LIKE 'Com_update';`
Throughput – Delete	Com_delete	Counts the number DELETE statements that have been sent to the database system	`SHOW GLOBAL STATUS LIKE 'Com_delete';`
Throughput – Total number of queries	Questions	Provides the sum of statements that clients have sent to the database system. It is essentially Com_insert + Com_select	`SHOW GLOBAL STATUS LIKE 'Questions';`

Category – Measurement	Name of Metric	Description	SQL Command
		+ Com_update + Com_delete. It however includes a count of statements executed in stored procedures.	
Performance – Average Latency per Schema	sum_timer_wait and count_star in the table "events_statements_summary_by_digest" which is in the "performance_schema" database	Computes the average query runtime in microseconds for each schema. A continuous increase in latency requires further investigation on resource constraints that could be the cause.	`SELECT schema_name 'Name of Schema', SUM(count_star) 'Number of Queries Executed', ROUND((SUM(sum_timer_wait) / SUM(count_star)) / 1000000) AS 'Average Query Runtime (microseconds)' FROM performance_schema.events_statements_summary_by_digest WHERE schema_name IS NOT NULL GROUP BY schema_name;`
Errors – Total Number of Statements with Errors	sum_errors	Counts the total number of statements that generated errors and groups them per schema. A sudden increase in the number of statements with errors may indicate a	`SELECT schema_name 'Name of Schema', SUM(sum_errors) 'Number of Statements that Generated Errors' FROM`

Category – Measurement	Name of Metric	Description	SQL Command
		problem with the application programs connected to the database system or problems with the database system itself.	`performance_schema.events_statements_summary_by_digest WHERE schema_name IS NOT NULL GROUP BY schema_name;`
Performance – Total number of "slow" queries; queries that exceed the pre-defined threshold set in the variable "long_query_time"	Slow_queries	The slow queries counter increments each time a query's execution time exceeds the number of seconds specified in a predefined parameter. In the case of MariaDB Galera synchronous multi-master distributed database, the pre-specified parameter is called "long_query_time", which is 2 seconds by default.	`SHOW GLOBAL STATUS LIKE 'Slow_queries';`
Throughput – Concurrent users connected that are active	Threads_running	Shows the number of active users connected.	`SHOW GLOBAL STATUS LIKE 'Threads_running';`

Table F.2: Resource metrics for diagnosing the cause of problems

Category – Measurement	Name of Metric	Description	SQL Command
Utilization – Concurrent users connected	Threads_connected	The MariaDB Galera synchronous multi-master distributed database used in the research had a maximum number of 500 client connections by default. This figure depends on the amount of RAM available and the workload submitted by each user. However, it can be changed manually in the my.cnf configuration file. The "Threads_connected" metric counts the number of threads which are based on a 1 thread per user distribution.	`SHOW GLOBAL STATUS LIKE 'Threads_connected';`

Saturation – Denied user connection requests caused by the database system reaching the maximum	Connection_errors_max_connections	Shows the number of new client connections requests that have been denied because the database system has reached its limit of the number of concurrent users it can serve	`SHOW GLOBAL STATUS LIKE 'Connection_errors_max_connections';`
			
			Equivalent to:
			
			`SHOW GLOBAL STATUS LIKE`

Category – Measurement	Name of Metric	Description	SQL Command
number of concurrent users it can serve			`'Aborted_connects';`
Errors – Denied user connection requests caused by internal errors from the database system	Connection_errors_internal	Shows the number of denied user connection requests caused by internal errors from the database system. This can indicate the need to investigate on possible underlying causes, one of which is out-of-memory errors caused by saturation of the RAM resource.	`SHOW GLOBAL STATUS LIKE 'Connection_errors_internal';`
Utilization – Number of requests to read from the primary memory (buffer pool)	Innodb_buffer_pool_read_requests	Shows the number of read requests that clients have sent to the database system and that are expected to be served by the primary memory (buffer pool)	`SHOW GLOBAL STATUS LIKE 'Innodb_buffer_pool_read_requests';`
Availability – Number of requests that could not be served by primary memory and are therefore being	Innodb_buffer_pool_reads	Shows the number of read requests that could not be served by the buffer pool, and therefore the database system had to serve the read request by reading directly from secondary storage (hard disk drive).	`SHOW GLOBAL STATUS LIKE 'Innodb_buffer_pool_reads';`

Category – Measurement	Name of Metric	Description	SQL Command
served by secondary storage		Reading from secondary storage is much slower than reading from the buffer pool which is in primary storage. There is a positive correlation between the performance of the database system and the size of "Innodb_buffer_pool_reads".	
Utilization – Memory utilization of a section of RAM that is referred to as the buffer pool in database theory	"Innodb_buffer_pool_ pages_total" and "Innodb_buffer_pool_ pages_free"	"Innodb_buffer_pool_pages_total" shows the total number of pages that the buffer pool has. Whereas "Innodb_buffer_pool_pages_free" shows the number of pages in the buffer pool that are currently not being used to store anything.	`SELECT (SELECT VARIABLE_VALUE FROM information_schema.GLOBAL_STATUS WHERE VARIABLE_NAME = 'Innodb_buffer_pool_pages_total') AS 'Total Number of Pages in the InnoDB Buffer Pool', (SELECT VARIABLE_VALUE FROM information_schema.GLOBAL_STATUS`

Category – Measurement	Name of Metric	Description	SQL Command
			WHERE VARIABLE_NAME = 'Innodb_buffer_pool_pages_free') AS 'Number of Pages in the InnoDB Buffer Pool That are not Storing Anything', (((SELECT VARIABLE_VALUE FROM information_schema.GLOBAL_STATUS WHERE VARIABLE_NAME = 'Innodb_buffer_pool_pages_total') - (SELECT VARIABLE_VALUE FROM information_schema.GLOBAL_STATUS WHERE VARIABLE_NAME = 'Innodb_buffer_pool_pages_free')) / (SELECT VARIABLE_VALUE

Category – Measurement	Name of Metric	Description	SQL Command
			FROM information_schema.GLOBAL_STATUS WHERE VARIABLE_NAME = 'Innodb_buffer_pool_pages_total')) AS 'Buffer Pool Utilization';

***Appendix G*: Database System Parameters to be Tuned**

Table G.1 lists and describes the most critical parameters that should be tuned according to database administration best practices informed from the review of literature.

Table G.1: List of most critical parameters to be considered during performance tuning

x_	Parameter	Description
1.	innodb_buffer_pool_size	Specifies the amount of main memory that can be used to store frequently used blocks of data and indexes. The larger the value, the more the quantity of data and indexes that can be stored in memory. This subsequently reduces the bottleneck caused by disk IO. An ideal value is 70-80% of the total available memory on a dedicated database server with primarily XtraDB or InnoDB tables. However, if the value of this parameter is too large, then memory swapping can occur which makes the performance of the server even worse. The trade-off is that the larger the value of this parameter, the longer the server will take to initialize. **Affected resource:** main memory
2.	innodb_buffer_pool_instances	This parameter divides the InnoDB buffer pool into a specific number of instances such that each instance manages its own data structures and takes an equal portion of the total buffer pool size. This helps to reduce contention concurrency. An ideal value is greater than or equal to 1GB for each instance. For example, if the innodb_buffer_pool_size is 8GB, then there will be 8 instances each with a 1GB buffer pool when the innodb_buffer_pool_instances is set as 8. **Affected resource:** main memory
3.	innodb_old_blocks_pct	The InnoDB buffer pool has two sub-lists. One sub-list for recently used information, and another sub-

		list for older information. By default, 37% of the list is reserved for the old list but this value can be changed by adjusting the value of the innodb_old_blocks_pct parameter. This value can be changed to anything between 5% and 95%. A smaller old sub-list enables faster eviction of less frequently used data from the buffer pool, thus giving room for more frequently used data to be stored in the new sub-list. **Affected resource:** main memory
4.	innodb_old_blocks_time	This parameter specifies the delay in milliseconds before a block can be moved from the old sub-list to the new sub-list in an InnoDB buffer pool. The default value (in MariaDB 5.5) is 0 which implies no delay, but this value can be set to a non-zero value as well. A non-zero delay helps in situations where full table scans are performed in quick succession. For example, when performing logical backups, full table scans in quick succession are expected. In such cases, it is better to ensure that data which is accessed only once remains in the old sub-list so that it can be evicted from the buffer pool instead of being moved to the new sub-list. **Affected resource:** main memory
5.	innodb_buffer_pool_dump_at_shutdown	This parameter enables the buffer pool state to be dumped into disk before the server is shutdown. It can be set to either ON or OFF. By default, it is OFF. **Affected resource:** main memory
6.	innodb_buffer_pool_load_at_startup	This parameter works with the previous parameter, i.e. innodb_buffer_pool_dump_at_shutdown to restore the buffer pool to the state it was in before the server was shutdown. It can be set to either ON

		or OFF and by default it is OFF. Setting innodb_buffer_pool_dump_at_shutdown and innodb_buffer_pool_load_at_startup to both ON eliminates the warmup time required for the buffer pool to identify and store the most frequently accessed data because it can pick up from where it left off before the server was shutdown. **Affected resource:** main memory
7.	query_cache_size	Specifies the size in Bytes that is available for storing the results of SELECT queries. Storing these results is useful for OLAP workloads that have a high-read and low-write environment. However, the query cache cannot be enabled in MariaDB Galera cluster versions prior to "5.5.40-galera", "10.0.14-galera", and "10.1.2". An ideal value is to set query_cache_size=0 or query_cache_type=OFF and use other techniques to increase the performance of OLAP workloads, e.g. good indexing, and setting up a load balancer to spread the read load. This is because the query cache is a well-known bottleneck. **Affected resource:** cache memory **Affected workload:** OLAP
8.	innodb_log_file_size	Redo logs are used to make sure database writes are fast and durable. They are also used during a recovery from a server crash however, larger log files can cause slower recovery in the event of a server crash. In as much as they can make recovery from a server slow, larger log files mean less disk I/O due to less flushing checkpoint activity. The size can be 1MB to 512GB (>= MariaDB 10.0) or 1MB to 4GB (<= MariaDB 5.5) **Affected resource:** storage

9.	innodb_file_per_table	This parameter allows some of the database tables to be kept in separate storage devices. This can greatly improve the I/O load on the storage. Default value is innodb_file_per_table=ON (>=MariaDB 5.5) and innodb_file_per_table=OFF (<=MariaDB 5.3) **Affected resource:** storage
10.	innodb_lock_wait_timeout	This parameter sets the time in seconds that an InnoDB transaction waits for an InnoDB row lock before giving up with a "timeout exceeded" error. When the timeout is exceeded, the statement (not the transaction) is rolled back. OLAP workloads benefit from a high innodb_lock_wait_timeout. OLTP workloads on the other hand benefit from a low innodb_lock_wait_timeout. The default value is innodb_lock_wait_timeout=50 and the range is 0 to 1073741824 (>= MariaDB 10.3) and 1 to 1073741824 (<= MariaDB 10.2) **Affected resource:** CPU **Affected workload:** OLAP and OLTP
13.	thread_cache_size	This parameter sets the number of threads that the server should cache for re-use. Increasing this parameter helps servers with high volumes of connections per second so that most connections can use a cached thread as opposed to a new thread. It can range from 0 to 16384. The default value is thread_cache_size=0 (<=MariaDB 10.1) and thread_cache_size=auto (from MariaDB 10.2.0) **Affected resource:** CPU **Affected workload:** OLTP
14.	tmp-table-size and max-heap-table-size	This parameter sets the default size of a temporary table. The temporary tables are used when processing complex queries that involve joins and

		sorting. This parameter therefore helps to prevent disk writes. It should have the same size as max-heap-table-size. An ideal value is assigning 64MB for every GB of RAM on the server. **Affected resource:** main memory **Affected workload:** OLAP
15.	sort-buffer-size	This parameter specifies the amount of memory in a buffer that is to be allocated to each session performing a sort operation. This value should be minimized for OLTP workloads that are known to have many small sorts. The default value is 2M, but an ideal minimum value is 16K. **Affected resource:** buffer memory **Affected workload:** OLTP
16.	join_buffer_size	This parameter is used to set the size of the buffer used for queries that cannot use indexes and thus perform a full table scan. An ideal value is to minimize it globally and to set a high value for sessions that require large full joins. **Affected resource:** buffer memory **Affected workload:** OLAP