Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 9, No. 1, 2021, 99-108.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Probabilistic Reasoning and Markov Chains as Means to Improve Performance of Tuning Decisions under Uncertainty

Allan Odhiambo Omondi*, Ismail Ateya Lukandu and Gregory Wanyembi

Department of Information Technology, Strathmore University, Nairobi, Kenya.

Article Info

Article History:
Received 17 September 2019
Revised 08 January 2020
Accepted 17 March 2020

DOI:10.22044/jadm.2020.8920.2027

Keywords:
Database Theory, Auto-Tuning, Decision Theory, Bayes' Theorem, Reinforcement Learning, Monte Carlo Simulation, Autonomic Computing.

**Corresponding author: aomondi@strathmore.edu (A. O. Omondi).*

Abstract

The variable environmental conditions and runtime phenomena require the developers of complex business information systems to expose the configuration parameters to the system administrators. This allows them to intervene by tuning the bottleneck configuration parameters in response to the current changes or in anticipation of the future changes in order to maintain the system performance at an optimum level. However, these manual performance tuning interventions are prone to error and lack of standards due to fatigue, varying levels of expertise, and over-reliance on inaccurate predictions of future states of a business information system. The purpose of this research work is to investigate that how the capacity of probabilistic reasoning to handle uncertainty can be combined with the capacity of Markov chains to map the stochastic environmental phenomena to ideal self-optimization actions. This is done using a comparative experimental research design that involves quantitative data collection through simulations of different algorithm variants. This provided compelling results, which indicate that applying the algorithm to a distributed database system improves the performance of tuning decisions under uncertainty. The improvement is measured quantitatively by a response-time latency 27% lower than the average and a transaction throughput 17% higher than the average.

1. Introduction

The argument $p \longrightarrow q$ is valid when it is impossible for the premise, *p*, to be true, while the conclusion, *q*, is false. The same argument is considered to be sound when the premise, *p*, is confirmed to be true. It is not always possible to gain 100% confidence regarding the truthfulness of a premise. The probabilistic reasoning is useful when it represents an uncertain knowledge in a case where we are not sure about the truthfulness of the premises of an argument [1].

The Bayes' theorem enables us to determine the probability of an event with an uncertain knowledge. This is made possible by relating the conditional probabilities to the marginal probabilities of two random events [2]. Given that $P(A|B)$ represents "the probability of A under the conditions of B", $P(B)$ represents "the marginal probability of B" and $P(A \wedge B)$ represents the joint probability of both A and B; it is well known, as shown in (1) that:

$$P(A|B)=\frac{P(A \wedge B)}{P(B)} \quad (1)$$

The Bayes' theorem extends the lemma in (1) further based on the product rule and the conditional probability of event B with a known event A, as shown in (2).

$$P(A \wedge B)=P(B\,|A)P(A) \quad (2)$$

This is then substituted in the original equation, and it essentially enables us to calculate the value of $P(A\,|B)$ with the knowledge of $(B\,|\,A)$, $P(A)$, and $P(B)$ as formally defined in the Bayes' theorem shown in (3).

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)} \quad (3)$$

This can also be stated as shown in (4).

$$P(cause|effect)=\frac{P(effect\ |cause)P(cause)}{P(effect)} \quad (4)$$

or in the research terms, as shown in (5).

$$P(hypothesis|evidence)=\frac{P(evidence\ |hypothesis)P(hypothesis)}{P(evidence)} \quad (5)$$

This forms a basic truism for most modern Artificial Intelligence (AI) systems that involve a probabilistic inference.

1.1. Bayes' Theorem and Markov Reward Process

Given that an observation, O_t, can be made at time t, an action performed based on the observation made, and a reward received based on the action performed, we can have a history H_t such that $H_t = O_1, A_1, R_1, \ldots, O_t, A_t, R_t$. Figure 1 shows the relationship between the observation, the action, and the reward as well as the managed element and the autonomic manager.

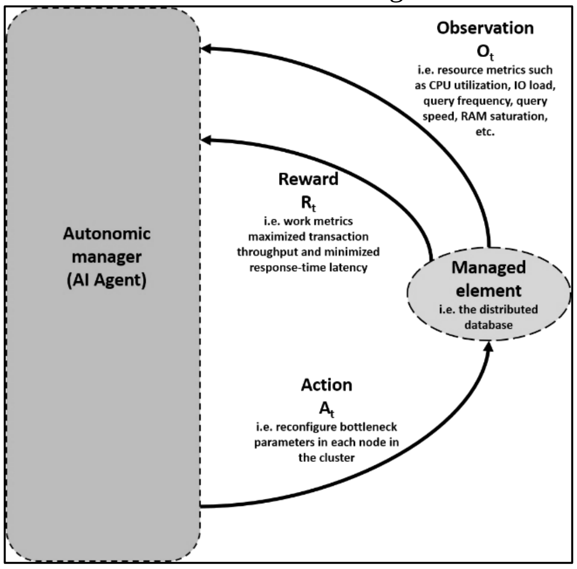

Observation
O_t
i.e. resource metrics such as CPU utilization, IO load, query frequency, query speed, RAM saturation, etc.
Reward
R_t
i.e. work metrics maximized transaction throughput and minimized response-time latency
Autonomic manager (AI Agent)
Managed element
i.e. the distributed database
Action
A_t
i.e. reconfigure bottleneck parameters in each node in the cluster

Figure 1. Reinforcement learning concepts applied in the autonomic manager.

In this case, the autonomic manager is the AI agent, and it is tasked with making decisions of which actions to perform under uncertainty. It is essential to note that a dynamical system has a direct relationship between the amount of computation performed and the quality of the output given. This is such that the more computations that are performed, the fewer the number of resources available to compute and produce the output at the required pace [3]. As demonstrated further in our previous work, it would be contradictory to have an intelligent agent that seeks to make decisions on how to improve the performance of a system but it simultaneously leads to a reduction in the system performance [4]. With this in mind, the autonomic manager was made distinct from the managed element.

One of the ways a non-compute intensive autonomic manager can be realized is by not storing and processing the history since time $t=1$. Instead of this, a summary of the history can be obtained in the form of $S_t = f(H_t)$ such that S_t is the state at time t. 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. We can, therefore, deduce that a Markov state defines the future as independent from the past given the present, as shown in (6).

$$H_{1:t} \to S_t \to H_{t+1:\infty} \quad (6)$$

An algorithm that promotes decision-making under uncertainty should be able to estimate future states. This is done in order to determine the expected reward if a certain action or sequence of actions are performed in the current state. This can be modelled as a value function, as shown in (7).

$$\begin{aligned} v(s) &= \mathrm{E}[R_{t+1}+\gamma R_{t+2}+\gamma^2 R_{t+3}+\ldots|S_t=s] \\ &= \mathrm{E}[R_{t+1}+\gamma(R_{t+2}+\gamma R_{t+3}+\ldots)|S_t=s] \\ &= \mathrm{E}[R_{t+1}+\gamma v(S_{t+1})|S_t=s] \end{aligned} \quad (2)$$

This implies that the value of a state, s, is the immediate reward that is received from being in that state, R_{t+1}, plus the value of all the other states in the future $v(S_{t+1})$ through recursion. γ is considered as the discount factor in order to ensure that the reward at time t is much higher than the reward at time $t+1$, thus giving a higher priority to immediate rewards than to future rewards. One reason for giving a less priority to future rewards is because there is uncertainty in the future [5,6]. It also makes it mathematically valid by avoiding a summation to infinity.

Given that at each state the autonomic manager can have multiple options of subsequent states that can traverse to, then we can assign probabilities to each subsequent state in the form depicted in Figure 2. We can, therefore, adjust the autonomic manager's value function to be the immediate value derived from being in a state, say s, plus the discounted value of the subsequent state, say $s``$, multiplied by the probability of transitioning to that subsequent state under the conditions of the current state that is $P_{ss``}$.

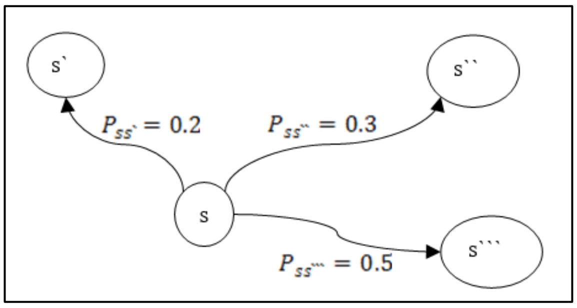


Figure 2. Probability distribution of transitioning from state, *s*, to subsequent states.

The probability in this case can be obtained through the Monte Carlo simulations. This gives us the equation shown in (8), where R is the immediate reward, γv is the discounted future reward, and P is the probability of the system transitioning from the current state to the next state.

$$v = R + \gamma P v \tag{8}$$

Inductively applying this in a real context can be done through the use of matrices. The real context in this case would involve hundreds of possible states that the system can be in. This gives us the Markov reward process, as shown in (9).

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R(1) \\ \vdots \\ R(n) \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} \tag{9}$$

1.2. Bayes' Theorem and Markov Decision Processes

The previous section defined the reward that the autonomic manager derived from being in various states as well as the probability of transitioning to those states. At this juncture, we can develop this foundation further by assigning agency to the autonomic manager, as demonstrated by [7]. By assigning agency, the Markov reward process depicted in (9) becomes a Markov decision process. It is this agency that allows the autonomic manager to make decisions on which actions to perform in order to move to a specific state that has a desired value. The possible actions that our autonomic manager can perform involve deciding which parameter settings should be tuned and how to tune them. Once it performs this action, it observes the environment, and subsequently, receives the reward of performing the specific action. If the reward is adequate, then it can conclude that it is on the right track based on the reinforcement learning principles. The reward in the case of this study was quantitatively measured as the transaction throughput and the response-time latency in a distributed database.

The aim is, therefore, to find the action that has a high probability of enabling the autonomic manager to get the highest reward. It can be modelled as shown in (10)

$$q_\pi(s,a) = \mathrm{E}_\pi[R_{t+1} + \gamma v(S_{t+1}) | S_t = s, A_t = a] \tag{3}$$

such that $q_\pi(s,a)$ is the action-value function that defines the value that the agent will get if it performs action a (defined by a policy π) given that it is in state s while performing the action. Herein lies an opportunity to employ the Bayes' theorem and Monte Carlo simulations to estimate $P(s|a)$.

The final solution is, therefore, to find the policy that has a sequence or set of actions, which if performed in specific states, is likely to yield the maximum benefit possible. This is the solution to the Markov decision process and is subsequently modelled as shown in (11).

$$q_*(s,a) = \max_\pi \left(q_\pi(s,a)\right), \tag{11}$$

where, $q_*(s,a)$ is the most optimum action-value pair, which is a solution to an optimization problem.

This leads to the research question that formed the starting point of this inquiry: *"How can the capacity of probabilistic reasoning to handle uncertainty be combined with the capacity of Markov chains to map the stochastic environmental phenomena to ideal self-optimization actions?"* Subsequently, the research hypotheses are as what follow.

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

Alternative hypothesis (H_1): Distributed database systems that apply to the designed algorithm, on average, have a faster transaction throughput.

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

$$H_0: \mu = \mu_{H_0}$$
$$H_1: \mu > \mu_{H_0}$$
$$H_2: \mu < \mu_{H_0}$$

Section 2 in this paper presents the details with the methodology applied to conduct the research work. It specifies the philosophical assumptions made, experiment procedure, test-data, and test bed as well as the data analysis methods used. Section 3 then presents the results of the research work highlighting the designed algorithm and empirical results of applying the algorithm in the context of a distributed database system. This is followed by Section 4 that provides an objective explanation of the facts that are supported by the discovered results. Section 5 then concludes the paper and provides recommendations for further research works.

2. Methodology

2.1. Philosophical Assumptions

The philosophical assumptions made in this work predicate all the choices made concerning the research methodology. The research question does not seek to understand the dynamic and subjective reality of the social actors (system administrators) in order to make sense of their motives and actions. Based on this premise, the study applied an ontological materialism, which was objective in nature. This objectivity matches with positivism as the epistemological approach because positivism emphasizes on the use of observations in order to justify the claims [8]. Given a cross-sectional time horizon, deductive reasoning moves from the existing theoretical knowledge to formation of a testable proposition (a hypothesis), to acceptance/rejection of the proposition by confronting it with the factual data [9]. This leads to a positivism that uses a mono-method quantitative choice that can be applied in the form of an experimental research design.

2.2. Experimental Procedure

5 steps outline the experimental procedure that was followed.

Step I: Define a realistic and reliable model of the underlying database system. This model should consist of work metrics to measure the amount of work the system is performing per unit time, a measure of the number of active concurrent users, and the quantitative effect that a series of configurations has on the transaction throughput and response time latency.

Step II: Design an algorithm that effectively achieves the pre-defined objective. This was done through the reflexive production of a code. It involves the analysis of the algorithm’s objective, followed by an identification of the required tasks required to achieve the objective, and the conversion of the results of the analysis into a pseudo-code. The pseudo-code is then converted into an actual code depending on the programming language. The Perl high-level, interpreted programming language, in conjunction with bash, a Unix shell and command language, were used during the implementation due to their ability to manipulate the textual configuration files in servers.

Step III: Theoretically analyze the asymptotic behavior of the designed algorithm. This is done in order to measure the level of algorithm correctness, time complexity, and space complexity. The algorithm should have a running time proportional to either a linear function or an n-log-n function because these are considered to be efficient.

Step IV: Complement the theoretical analysis by conducting controlled experiments in the form of empirical algorithmics. This is done using the treatments that manipulate the algorithm and measurements that identify the effect of the manipulation. Each experiment was repeated 30 times based on a manipulated form of the algorithm. Repetitions above 30 did not provide any significant change in the average value of the results. The decision rule should then be applied at this point to determine whether to reject or fail to reject the null hypothesis. Go back to Step II if there is no reason to reject the null hypothesis; otherwise, proceed to Step V if the null hypothesis is rejected in favor of an alternative hypothesis.

Step V: Assemble the best-performing algorithm variations into an algorithm library. The result, as supported by [10] and [11], should be an efficient, generalizable, easy to use, well-documented, and portable implementation of a behavior that has a well-defined interface by which the behavior 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 [11].

2.3. Experiment Test-Data

The American National Standards Institute (ANSI) Structured Query Language (SQL) Standard Scalable and Portable (AS^{3}AP) benchmark was designed to compare the performance of relational database systems with vastly different architectures and capabilities over a variety of workloads. One of the key advantages of AS^{3}AP is its ability to define a runtime ordering of the queries in the workload in order 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 required to flush the buffers. It consists of the single-user tests and the multi-user tests. The single-user AS^{3}AP workloads focus on the basic functions that a relational Database Management System (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, 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 the Online Transaction Processing (OLTP) system transactions and measuring degradation in response time latency for the Online Analytical Processing (OLAP) system 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 background programs, and the number of concurrent users. Consequently, multi-user AS^{3}AP workloads include 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).

Another justification for applying the AS^{3}AP benchmark is its combination of OLTP and OLAP workloads in a single experiment. This is unlike Transaction Processing Performance Council's (TPC's) 'E' and 'H' Benchmarks, which are also the testing tools used to compare the performance of relational database systems that have different architectures. TPC-E and TPC-H separate the OLTP and OLAP workloads, respectively. This separation is not always ideal given the presence of business applications that process a hybrid of OLTP and OLAP workloads.

In order to simulate the real-world user interactions, a latency of 1 s of "think-time" was added. Think-time was 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 the pre-sampling time and the 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 the 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 the pre-sampling time and the remaining 2/3 for the sampling time.

Lastly, the virtual users were added continuously at a rate of 1 virtual user every 2 s. The tool used to orchestrate the experiment was the 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 an intensive workload submitted simultaneously by each one of the 20 virtual users.

2.4. Experiment Test Bed

There are many pre-defined environments dedicated for testing, for example, Grid'5000, Open Cirrus, Planet Lab, Future Grid, Distem, ModelNet, and Linpack. However, these publicly available test beds face significant challenges. One such challenge is an 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 [12]. Another significant challenge is working with remote environments. This leads to a heavy reliance on the test bed's support team in the 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 an absolute authority over the experiments and their environment. The experiment test bed was made up of a distributed database with Maria DB Galera synchronous multi-master cluster (version 10.2.14) installed as the Distributed Database Management System (DDBMS). MariaDB provides a full support for concurrent access, transaction processing, and analytical processing. There were three nodes in the cluster, each configured as a master with no slaves, and there was a synchronous replication between all the three nodes. The synchronous replication was made possible through the use of the Write-Set REPlication (WSREP) Application Programming Interface (API). WSREP API implements an eager replication, whereby the nodes in the cluster synchronize their states (database content) with all the 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 one of the three nodes had its own CPU and storage as Virtual Machines (VMs). All the three nodes plus the load balancer were running a 64-bit Ubuntu Server 16.04 LTS as the Operating System. Figure 3 shows the architecture of the test bed.

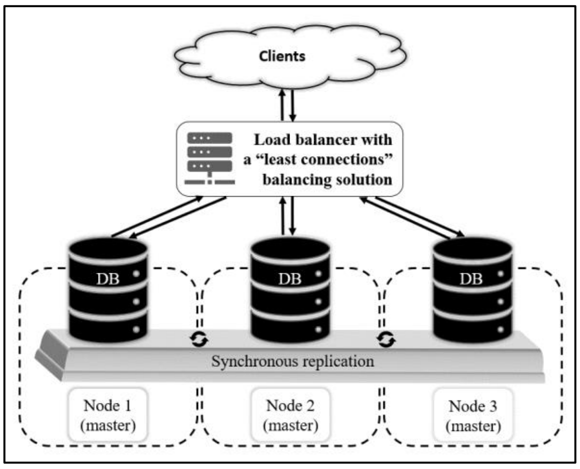


Figure 3. Architecture of the test bed.

The test bed aimed to model a real-world environment, whereby the normal architecture was 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 the generalizability of the results of the study to a population as part of external validity.

2.5. Data Analysis Methods

The study was willing to take a maximum risk of 5% for rejecting the null hypothesis when it was true (Type I error). The value of 5% was arrived at with the aim of striking an adequate balance between Type I and Type II errors, both of which were negative. Consequently, a p-value of 0.05 or less was desired when measuring how often an outcome happened over a repeated execution of experiments.

As indicated in the experimental procedure outlined in Section 2.2, the decision rule determined whether to reject or fail to reject the null hypothesis. The decision rule applied in the study states that the null hypothesis should be rejected if at least 95% of all the experiments executed for a specific treatment or variation of the algorithm result in a faster transaction throughput and a slower query response time. The transaction throughput (measured in Transactions Per Second (TPSs)) and the response time latency (measured in microseconds) were used to quantitatively define the level of optimization achieved.

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. A T-score supported the transformation of an individual score into a standardized form for an 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:

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

Reject $H_0\ if\ T\ score_{calculated} < T\ score_{tabular}$ in the case of the 2nd alternative hypothesis (H_2).

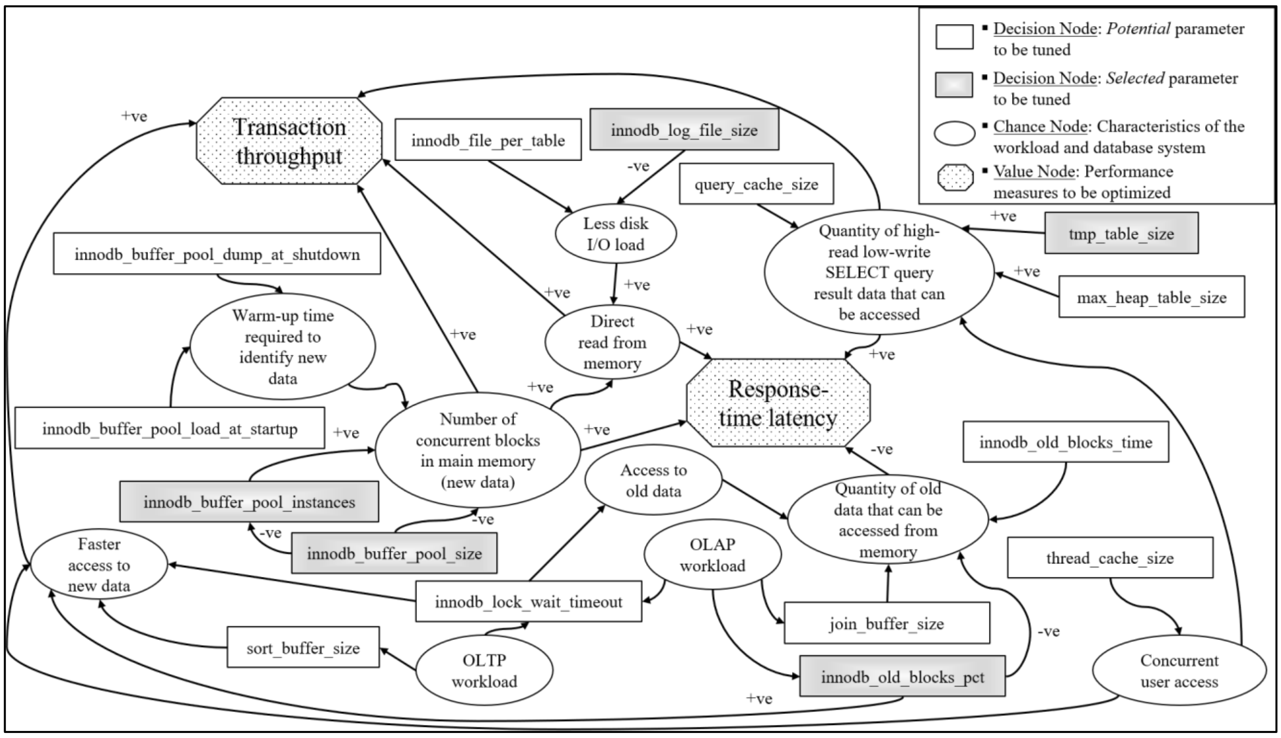
Figure 4 shows the graphical model in the form of a generic influence diagram.

3. Results

3.1. Probabilistic Graphical Model

A probabilistic graphical model based on a Directed Acyclic Graph (DAG) was used to represent the set of decision variables and their conditional dependencies. The uncertainties in the characteristic of the workload and the DDBMS parameter settings (input values) were represented as the probability distributions. This was in the form of a probability density function (pdf) for continuous probability distributions and a probability mass function (pmf) for discrete probability distributions. The probability distributions were obtained through the Monte

Carlo simulations and opinions from domain experts in the literature review.



▭ Decision Node: *Potential* parameter to be tuned
▨ Decision Node: *Selected* parameter to be tuned
◯ Chance Node: Characteristics of the workload and database system
▨ Value Node: Performance measures to be optimized
Transaction throughput
Response-time latency
innodb_file_per_table
innodb_log_file_size
query_cache_size
tmp_table_size
max_heap_table_size
Less disk I/O load
Direct read from memory
Quantity of high-read low-write SELECT query result data that can be accessed
innodb_buffer_pool_dump_at_shutdown
Warm-up time required to identify new data
innodb_buffer_pool_load_at_startup
Number of concurrent blocks in main memory (new data)
innodb_buffer_pool_instances
innodb_buffer_pool_size
Access to old data
Quantity of old data that can be accessed from memory
innodb_old_blocks_time
Faster access to new data
innodb_lock_wait_timeout
OLAP workload
join_buffer_size
thread_cache_size
sort_buffer_size
OLTP workload
innodb_old_blocks_pct
Concurrent user access
+ve
-ve

Figure 4. Graphical model representing the decision variables.

A random value from the probability distribution of each input was then sampled and the Monte Carlo simulation was applied to estimate the probability distributions of the outputs (desired levels of transaction throughput and response time latency). Using the random value from the probability distribution of each input, the simulation was repeated for 10,000 times in each experiment to obtain a precise estimation of the output distributions. A sample size of 10,000 was considered to be adequate, given the inherent uncertainty in the inputs. In other words, a higher precision in this case would be an aesthetic preference rather than a functional need.

3.2. Algorithm Design

The following is the pseudo-code of the designed algorithm with the order of O(n):

Algorithm 1: Performance tuning algorithm

```
        Input    Current state of DDBMS, $s_0$
        Output   The action, $a$, that leads to the most desirable state
1.  function perfTuner ()
2.  Identify the most probable next state of the workload from its probability distribution: $s_1$
    /*The chance nodes*/
3.  n = 1
4.  while $n <= 3$ do
5.      Select the n^th next combination of server configurations that can be performed under the conditions of the identified next state
        /*The decision nodes*/
6.      while within computational budget do
7.          Simulate the behaviour of the workload
            /*The chance nodes*/
8.          Update the probability of transitioning to the next state based on the Bayes' theorem (i.e. on the condition that the selected next actions have been performed)
9.          Update the discounted reward value of the next states in a Markov reward process
            /*The value nodes*/
10.     n = n + 1
11. Select the action, $a$, that has the highest probability of receiving the highest discounted value.
    /*The final decision node*/
12. return $a$
```

3.3. Empirical Algorithmics

The philosophical assumptions made in this work promote the revelation of truth through objective observations. It was through experiments based on the empirical algorithmics that the required objective observations were made.

A chronological 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 central location of the Perl script also promoted easier orchestration of the test bed during the manipulation of the algorithm. This benefit could also be translated to a live environment. The Perl script then implemented the algorithm that recommended the most appropriate configuration or sequence of configurations to implement. This was then implemented by editing the text-based configuration file on each node/member of the distributed database system cluster, as shown in Figure 5.

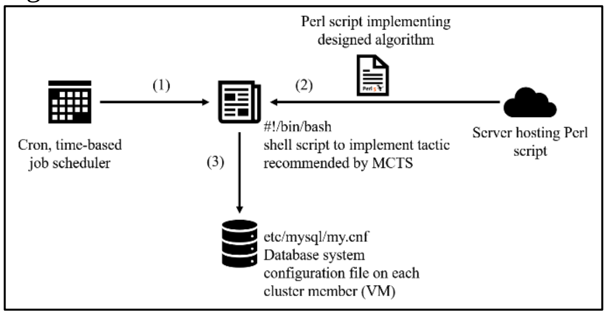

Perl script implementing designed algorithm
(1)
(2)
(3)
Cron, time-based job scheduler
#!/bin/bash
shell script to implement tactic recommended by MCTS
Server hosting Perl script
etc/mysql/my.cnf
Database system configuration file on each cluster member (VM)

Figure 5. Implementation architecture of the algorithm.

Each conducted experiment involved identification of the bottleneck parameters that had a significant impact on the system performance, and proactively reconfiguring them using the designed algorithm so that they could adapt to the current workload. The results presented in this work indicated that all 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, as proposed in the first alternative hypothesis. In addition to this, the response time latency was lower when the algorithm was running. Table 1 and table 2 present the results obtained from conducting the experiments.

4. Discussion

This work confirms that the capacity of probabilistic reasoning to handle uncertainty can be combined with the capacity of Markov chains to map the stochastic environmental phenomena to ideal self-optimization actions. This temporal precedence finding is consistent with that of [13], 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.

Table 1. Empirical algorithmics results for transaction throughput.

	Average transaction throughput (transactions per s)	Maximum & minimum transaction throughput (transactions per s)
Default	4,958.46 (σ = 350.74)	5,298.36 and 2,935.70
Low	5,272.92 (σ = 210.94)	5,648.19 and 4,809.30
Medium	5,413.56 (σ = 303.06)	5,912.09 and 4,721.93
High	5,784.91 (σ = 435.78)	6,074.99 and 3,665.23
Adaptive	5,812.75 (σ = 249.11)	6,454.89 and 5,379.13

Table 2. Empirical algorithmic results for response time latency.

	Average response time latency (microseconds)	Maximum & minimum response time latency (microseconds)
Default	3.50 (σ = 0.61)	6.00 and 3.00
Low	3.04 (σ = 0.19)	4.00 and 3.00
Medium	3.02 (σ = 0.14)	4.00 and 3.00
High	3.07 (σ = 0.38)	5.00 and 3.00
Adaptive	2.56 (σ = 0.50)	3.00 and 2.00

A comparison of the study by [13] with a previous study by [14] accords with our initial observation that although humans are better at understanding an 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 [15], which championed the concept of self-management in computing in order to automate the previously unachievable tasks or tasks that were performed in a sub-standard manner. The research is, however, keen to caution that this does not imply automation in order to replace the database administrators. To the contrary, the findings in this work propose the use of automation to enable human beings to free their minds from mundane tasks in order to concentrate on the previously unachievable tasks. This corroborates the findings from a study by the authors in [16], who investigated the history and future of workplace automation.

The findings of the current work seem to contradict the findings by [17], which applies vertical and horizontal partitioning of data to promote scalability. The results of the current study also seem to contradict the findings by [18], which applied the creation of cost-driven indices to promote scalability. Although indices 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.

Businesses rely on the 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 that are, in turn, used to implement appropriate actions.

The role of IT, therefore, goes beyond the 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 shows, the 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 is required to be maintained at an acceptable level, hence, the importance of automated performance tuning in database systems.

Unplanned downtime remains a constant threat to businesses. An antidote to minimize unplanned downtime and maximize the time when the database system performance is at an optimum level is to conduct preventive maintenance, as shown in figure 6.

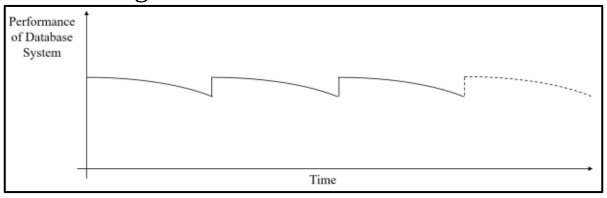


Figure 6. Preventive maintenance over time.

However, one of the biggest challenges with preventive maintenance is to determine when to do the maintenance. The current work implies that performing multiple Monte Carlo simulations can take advantage of probabilistic reasoning to estimate the Mean-Time-To-Failure (MTTF). As shown in figure 7, there are multiple possible estimates of the duration of time when the database system performance will be at an unacceptable level. The higher the number of Monte Carlo simulations using a mathematical model based on the Bayes' theory (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 7.

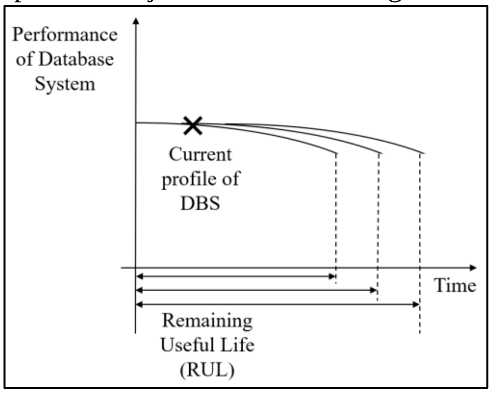


Figure 7. Trajectories of remaining useful life over time.

5. Conclusions and Recommendations

Numerous opportunities exist to extend this work further in multi-disciplinary research works. Further research works could analyze 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 exploration of the potential benefits of adaptive control in Software Defined Everything (SDx). The work, therefore, recommends the scholars to apply a multi-disciplinary approach because it combines expertise from various fields. This can lead to creative high-impact research works. However, a multi-disciplinary perspective should be approached with caution because of the lack of the potential meaningful evaluation from the team. The domain-specific concepts tend to be accepted without question or rejected without constructive criticism in multi-disciplinary research works.

One of the surprising results obtained in this work was the fact that the *"tmp_table_size"* configuration parameter affected OLTP workloads instead of OLAP workloads. It is interesting to conduct further research works to obtain an explanation for this odd phenomenon. In addition, future research works 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 the mechanical, electrical, and other physical engineering states of the server's hardware when modelling and defining the system profile. The use of Kalman filters to perform this modelling, as opposed to the Bayes' theory-based models, should also be explored further.

6. References

[1] D. G. Sullivan, "Using probabilistic reasoning to University, automate software tuning (Doctoral Dissertation) ", Harvard Cambridge, Massachusetts, 2003.

[2] T. Vodopive, S. Samothrakis and B. Šter, "On Monte Carlo Tree Search and Reinforcement Learning, *Journal of Artificial Intelligence Research*", vol. 60, pp. 881–936, 2017, doi: 10.1613/jair.5507, 2007.

[3] T. Li, Z. Chunqiu, Y. Jiang, W. Zhou, L. Tang, Z. Liu, & Y. Huang. , "Data-Driven Techniques in Computing System Management", *ACM Computing Surveys*, vol. 50, no. 3, pp. 45–88, 2017.

[4] A. O. Omondi, I. A. Lukandu and G. W. Wanyembi , "A Selection Variation for Improved Throughput and Accuracy of Monte Carlo Tree Search Algorithms", *IJCIT*, vol. 7, no. 6, pp. 286–294, Jul. 2018.

[5] A. Bousdekis, B. Magoutas, D. Apostolou and G. , "Mentzas A proactive decision making framework for condition-based maintenance", *Industrial Management & Data Systems*, vol. 115, no. 7, pp. 1225–1250, 2015, doi: 10.1108/imds-03-2015-0071, 2015.

[6] U. K. Chajewska, D. Koller and R. Parr, R, "Making rational decisions using adaptive utility elicitation", *in American Association for Artificial Intelligence*, 2000, pp. 363–369, 2000.

[7] G. Su, T. Chen, Y. Feng, D. S. Rosenblum and P. S. Thiagaranj, "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, 2016.

[8] B. Hjørland, "Empiricism, rationalism and positivism in library and information science", *Journal of documentation*, vol. 61, no. 1, pp. 130–155, 2005, doi: 10.1108/00220410510578050.

[9] T. Lemke, "Varieties of materialism, "*BioSocieties*, vol. 10, no. 4, pp. 490–495, 2015, doi: 10.1057/biosoc.2015.41.

[10] P. Sanders, "Algorithm engineering: An attempt at a definition, in Efficient Algorithms: Lecture Notes in Computer Science", S. Albers, H. Alt, and S. Naher, Eds. Heidelberg, Berlin: Springer-Verlag, 2009, pp. 321–340.

[11] R. Kitchin, "Thinking critically about and researching algorithms", *Information, Communication & Society*, vol. 20, no. 1, pp. 14–29, 2017, doi: 10.1080/1369118X.2016.1154087.

[12] F. Desprez, G. Fox, E. Jeannot, K. Keahey, M. Kozuch, D. Margery, et al. , " Supporting experimental computer science", Argonne National Laboratory, Rocquencourt, France, Technical Memo 362, 2012.

[13] D. Van Aken, A. Pavlo, G. J. Gordon and B. Zhang, "Automatic Database Management System Tuning through Large-Scale Machine Learning", *presented at the ACM International Conference on Management of Data*, Chicago, IL, USA, 2017, pp. 1009–1024, doi: 10.1145/3035918.3064029.

[14] S. W. Cheng and D. Garlan, "Stitch: A language for architecture-based self-adaptation", *Journal of Systems and Software*, vol. 85, no. 12, pp. 2860–2875, 2012.

[15] J. O. Kephart and D. M. Chess, "The vision of autonomic computing", *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[16] H. D. Autor, "Why are there still so many jobs? The history and future of workplace automation", *The Journal of Economic Perspectives*, vol. 29, no. 3, pp. 3–30, 2015, doi: 10.1257/jep.29.3.3.

[17] J. W. Kim, S. H. Cho and I. M. Kim, "Workload-Based column partitioning to efficiently process data warehouse query", *International Journal of Applied Engineering Research*, vol. 11, no. 2, pp. 917–921, 2016.

[18] S. Chaudhuri and V. Narasayya, "Self-Tuning Database Systems: A Decade of Progress", *in Proceedings of the 33rd international conference on Very Large Databases*, 2007, pp. 3–14.

الگوی استدلال احتمالی و زنجیره مارکوف به معنای بهبود عملکرد تصمیمات تنظیم تحت عدم قطعیت

Allan Odhiambo Omondi*، Ismail Ateya Lukandu و Gregory Wanyembi

گروه فناوری اطلاعات، دانشگاه استراتور، نایروبی، کنیا.

ارسال ۲۰۱۹/۰۹/۱۷؛ بازنگری ۲۰۲۰/۰۱/۰۸؛ پذیرش ۲۰۲۰/۰۳/۱۷

چکیده:

شرایط متغیر محیطی و پدیده‌های زمان اجرا به توسعه دهندگان سیستم‌های اطلاعاتی پیچیده کسب و کار نیاز دارد تا پارامترهای پیکربندی را در معرض دید مدیران سیستم قرار دهند. این امر به آنها امکان می‌دهد تا با تنظیم پارامترهای پیکربندی گلوگاه در پاسخ به تغییرات فعلی یا پیش بینی تغییرات آینده برای حفظ عملکرد سیستم در سطح مطلوب، مداخله کنند. با این حال، این مداخلات تنظیم عملکرد دستی به دلیل خستگی، سطح مختلف تخصص و اعتماد بیش از حد به پیش بینی‌های نادرست از وضعیت‌های آینده سیستم اطلاعات کسب و کار مستعد خطا و فقدان استاندارد هستند. هدف از این کار تحقیقاتی این است که چگونه ظرفیت استدلال احتمالی برای کنترل عدم اطمینان را می‌توان با ظرفیت زنجیره‌های مارکوف برای ترسیم پدیده‌های تصادفی محیطی به اقدامات ایده آل بهینه‌سازی ترکیب کرد. این کار با استفاده از یک طرح تحقیق تجربی مقایسه‌ای انجام می-شود که شامل جمع آوری اطلاعات کمی از طریق شبیه‌سازی انواع الگوریتم‌های مختلف است. این کار نتایجی را فراهم می‌کند، که نشان می‌دهد استفاده از الگوریتم در یک سیستم پایگاه داده توزیع شده عملکرد تصمیمات تنظیم را تحت عدم اطمینان بهبود می‌بخشد. این پیشرفت به صورت کمی با تأخیر زمان پاسخ ۲۷٪ کمتر از متوسط و عملکرد معامله ۱۷٪ بیشتر از متوسط اندازه‌گیری می‌شود.

کلمات کلیدی: نظریه بانک اطلاعات، تنظیم خودکار، نظریه تصمیم گیری، قضیه بیز، یادگیری تقویت، شبیه سازی مونت کارلو، محاسبات خودمختار.