International Journal of Engineering Fields (IJEF)
Volume 1, Issue 1, October December, 2023
Available online at: https://journalofengineering.org

Self-Tuning Data Warehouse Architectures for High-Throughput Analytical Workloads

Sarvesh Kumar Gupta

Consulting Member of Technical Staff, Oracle, Saint Peters, Missouri -63376, USA

ORCID: 0009-0008-7460-4874

ABSTRACT

In this paper, the problem of managing large amounts of analytical workloads in the current generation of data warehouses is considered by analyzing self-tuning capabilities. Modern businesses produce considerable volumes of both structured and semi-structured data which need to be effectively processed in order to provide insights for decision-making and further analysis. Existing data warehouses require a lot of effort to be put into index definition, optimization of queries, allocating resources, partitioning data, and managing workloads. When dealing with high-throughput workloads in complex databases, manual approaches to tuning cannot guarantee consistency in performance, scalability, and effectiveness. To address this drawback, researchers began developing self-tuning data warehouse solutions to be able to detect and resolve performance issues automatically. The current research will analyze the effect of using self-tuning mechanisms on data warehouses dealing with large analytical workloads. The offered architectural framework incorporates workload monitoring, intelligent resource management, automation of physical design optimization, adaptive query processing, and machine learning-based decision making in order to create an optimized analytical environment. Workload-aware tuning policies will allow the framework to optimize storage structures, allocate and manage computing resources, and optimize query execution. Thus, the research aims at demonstrating the considerable contribution that self-tuning data warehouse architectures make in terms of enhancing query performance, effective resource management, scalability, and overall effectiveness.

Keywords: Self-Tuning Data Warehouse, Analytical Workloads, Autonomous Database Systems, Query Optimization, Workload Management, Machine Learning, Resource Allocation, Cloud Data Warehousing

INTRODUCTION

A rapid rise in the number of projects related to digital transformation, cloud computing, IoT, social media applications, e-commerce solutions, and enterprise information systems resulted in a considerable growth of data production processes. Currently, companies use large amounts of structured and semi-structured data in order to make important strategic decisions, analyze customers' preferences and financial data, optimize workflows, perform business intelligence procedures, etc. Data warehouses became an essential component of modern analytical ecosystem allowing companies to effectively process large amounts of data. However, the increasing amount, speed, and diversity of data created an additional challenge making analytical workloads more complicated.
Modern analytical workloads include complex queries using join, aggregation, filtering, real-time analytics, dashboards, and other types of business intelligence requests. Modern analytical workloads may be characterized by their dynamic nature with continuously changing user requirements, data distributions, and resource requirements. Traditional data warehouse systems significantly rely on database administrators who should be responsible for such tasks as index tuning, creation of materialized views, partitioning tables, allocating memory, scheduling queries, query optimization settings, etc. Manual tuning improves the performance but is a quite complicated task in case of many concurrent queries.

There are a few challenges to be faced when trying to tune databases manually. First of all, detecting performance problems requires considerable knowledge and experience. Secondly, improvements made for one specific workload may lead to deterioration of performance of other workloads. Thirdly, quickly changing data distributions and query patterns may require significant changes in the configuration of the warehouse which will lead to a decline in its performance. Besides, cloud-based workloads add extra complexity because of elastic resource scaling and varying cost-performance ratios. As a result, it becomes hard to provide optimal performance at a low price.

It makes evident that there is a need for autonomous self-tuning mechanisms that would allow for automatic optimization of warehouse configurations. Autonomous self-tuning warehouse architectures could be viewed as the solution for this

International Journal of Engineering Fields (IJEF)
Volume 1, Issue 1, October December, 2023
Available online at: https://journalofengineering.org

problem as they incorporate workload monitoring, performance analysis, machine learning, physical design optimization, query processing optimizations, and smart resource allocation. These systems observe the current state of workloads, predict their future needs, detect inefficient practices, and apply corrections in order to improve performance.

Motivation for conducting this research comes from the increasing interest to intelligent solutions used to analyze large and high-throughput data sets. As organizations start relying on fast and effective processing of data and making decisions based on the results obtained, traditional approaches fail to meet requirements in terms of performance. Therefore, intelligent self-tuning architecture is considered as a perspective option to be researched.

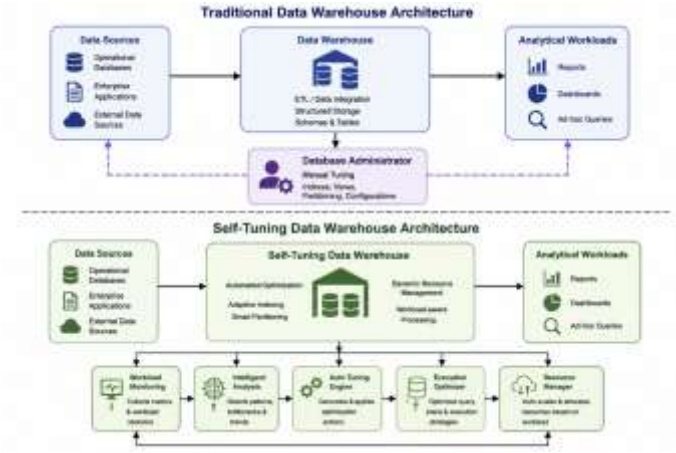

Fig.1. Traditional vs Self-Tuning Data Warehouse Architecture

LITERATURE REVIEW

One promising trend in contemporary data warehouse design has been self-tuning systems. Research into self-tuning warehouse architectures has arisen as a response to modern workloads characterized by concurrency, variety, and high volumes as well as deployment in the cloud. One of the earliest contributions on self-tuning databases has established that the manual tuning of physical structures, indexing, partitioning, memory parameters, and query tuning has become impossible due to changing workloads. According to Chaudhuri and Narasayya (2007), the key aspect of self-tuning DBMS lies in automated physical design because the performance of analytical queries is highly sensitive to indexes, access structures, and physical structures.

A fundamental step in self-tuning warehouses is the automatic design of indexes and materialized views. As Agrawal, Chaudhuri, and Narasayya (2000) note, choosing materialized views and indexes concurrently provides better performance because both structures use the same storage space yet provide assistance in different query classes. This approach can be considered a relevant approach for data warehouse design because many analytical queries involve joins and aggregations, and hence, materialized views and indexes are helpful in accelerating their performance. Later research on the workload-as-a-sequence paradigm has extended this idea by noting that the timing of queries is an essential aspect of warehouse tuning because the order of queries affects the effectiveness of decisions.

Modern self-tuning systems do not limit themselves to the automated design of physical structures. Specifically, Chaudhuri and Weikum have suggested that the design of RISC-style database systems could streamline the administrative activities associated with DBMS operations. This approach was influential in developing autonomous DBMS systems where monitoring, diagnostics, planning, and execution were viewed as a cycle. High-throughput analytical data warehouses may benefit from this approach because several aspects of the tuning process – namely the length of the query queue, latency, CPU pressure, and other – can change dynamically.

With the advent of cloud-based data warehouses, another issue of interest has emerged – namely, elasticity. According to Dageville (2016) et al., modern cloud warehouses like Snowflake separate computing and storage processes and use a multi-clustered shared-data model, which allows for elastic scalability, availability, and support for mixed workloads. This architectural choice transforms the tuning process as the designer needs to determine not how to choose indexes and materialized views but what compute capacity to allocate, how to scale up, and how to isolate workloads. Elastic compute

is helpful in processing workload bursts in analytical warehouses but imposes performance and cost trade-offs, thus making self-tuning even more challenging.

The recent years have seen machine-learning approaches emerging in self-tuning databases. Van Aken et al. (2017) have developed OtterTune, a tool that uses historical tuning information and machine learning algorithms to recommend DBMS configurations. While much of this work focuses on knob-tuning at the level of DBMS, the research has significance for self-tuning warehouses as cloud warehouses offer a large number of tunable knobs including memory allocation, caching, query parallelization, and other aspects. The study demonstrates the possibility of reuse of historical tuning experiences and, as a result, makes it easier to tune new workloads.

The work on self-driving database systems has continued with the Peloton paper by Pavlo et al. (2017) Instead of recommending knob configurations as an isolated aspect of tuning, this paper describes a holistic solution where DBMS self-monitors itself, forecasts the future workloads, evaluates the consequences of its actions, and implements decisions independently. While this architecture is applicable to any database, it can be considered directly relevant for self-tuning analytical warehouses as analytical workloads often change dynamically over time with shifting patterns and data distributions, new queries, and other factors.

Workload forecasting is also important for self-tuning warehouses, as shown by the QueryBot 5000 paper by Ma et al. (2018) The authors propose a framework for forecasting self-driving database workloads that emphasizes logical queries. The reason is that while the utilization of physical resources may change depending on the physical design, queries remain relatively constant in the long term. From the perspective of an analytical warehouse, query forecasting is relevant because an advanced warehouse can adapt its behavior to upcoming query workloads to increase performance and reduce costs.
Recent works on learned query optimization are also relevant to the discussion on self-tuning data warehouse architectures. Marcus et al. (2019) describe Neo, a learned query optimizer that uses neural models to improve query plans. Later, Bao develops a practical approach to learned query optimization using reinforcement learning principles to make decisions and choose better query plans. These papers are crucial for the topic under discussion because query optimization is arguably one of the most complex aspects of analytical data warehouses. Especially for high-volume analytical warehouses, traditional cost models fail because of skewed predicates, large joins, and complex nested queries.

Another issue of interest for self-tuning warehouse is intelligent scheduling. In Wagner et al. (2021), the authors examine self-tuning scheduling mechanisms for analytical workloads and show that the introduction of task-based execution creates many opportunities for improving the efficiency of the process. Specifically, analytical warehouses are characterized by concurrent running of long and short queries, which leads to competition for CPU resources, I/O, and memory space. Self-tuning scheduling mechanism would allow to achieve higher query throughput and reduced tail latency in such workloads, particularly BI dashboards.

An interesting concept of a self-driving database system has been presented by Kossmann and Schlosser (2020). Particularly, they discuss workload prediction, robustness of tuning solutions, reusability of solutions, and minimalistic integration as the essential criteria for designing self-driving DBMSs. This paper also reveals a particular limitation of self-tuning systems – that is, it must operate at negligible costs. This means that high-throughput analytical warehouses should implement tuning methods with low overhead, easy-to-explain decisions, and safe behavior, otherwise it would be costly.
In sum, it can be stated that current self-tuning data warehouse architectures integrate four major research trends. In early papers, indexes and materialized views are discussed prominently. Recent research moves towards more advanced self-tuning architectures, including cloud-based elastic warehouses, machine learning, and adaptive scheduling. Based on the reviewed works, it can be said that a successful self-tuning data warehouse would need to perform workload analysis, forecast future demands, make decisions about physical design, choose optimal query plans, dynamically allocate resources, and monitor outcomes.

OBJECTIVES AND RESEARCH METHODOLOGY

The rise in the complexity of analytical workloads has created the need for efficient means of automatic optimization of the performance in data warehouses. Current solutions for warehouse management require substantial manual effort in terms of tuning and management, resulting in reduced performance and higher costs of operation. This paper will discuss self-tuning warehouse architectures as an intelligent approach to performance optimization in data warehouses.

The first and foremost goal of this study is related to query optimization – the ability of the warehouse architecture to analyze and optimize query execution. The importance of optimizing query performance can be explained by the necessity to support business intelligence, decision support systems, and various analytics processes. Storage optimization is another

International Journal of Engineering Fields (IJEF)
Volume 1, Issue 1, October December, 2023
Available online at: https://journalofengineering.org

crucial aim to consider, since an intelligent method of storage organization and allocation should lead to better storage efficiency, lower cost of maintenance, and increased performance. Reducing the administrative overhead is another aim worth considering – self-tuning warehouses should not need any tuning performed manually by database administrators. Furthermore, scalability should also be discussed as one of the main goals of the study. Modern analytics architectures need to be highly scalable to accommodate more users and larger volumes of data. Therefore, automatic adaptation of resource usage and configuration becomes important in the process of ensuring scalability. Finally, it is also important to mention that self-tuning architectures are expected to ensure continuous improvement of performance.

Research methodology implies an analysis of available self-tuning technologies and approaches, including autonomous optimization methods, machine learning-based tuning methods, and cloud data warehouse architectures. The focus of the performance analysis will lie on workload-aware optimization techniques and their impact on execution efficiency, storage management, resource allocation, and scalability. Several metrics will be considered when analyzing the performance indicators and comparing different self-tuning warehouse systems. Conceptual evaluation of the self-tuning approach will be conducted using the same metrics used to evaluate the performance of traditional data warehouses.

Table 1: Research Objectives and Evaluation Metrics

Objective	Evaluation Metric
Query Optimization	Query Execution Time
Storage Efficiency	Compression Ratio
Resource Utilization	CPU/Memory Usage
Scalability	Throughput

SELF-TUNING DATA WAREHOUSE ARCHITECTURE

A self-tuning data warehouse architecture aims at achieving automatic optimization in terms of system performance and resource use without manual intervention in the ongoing operations. Such an architecture incorporates monitoring capabilities, intelligent optimization, and workload-aware decision-making to handle analytical workloads of high throughput. Through continuous workload analysis and monitoring, it becomes possible to change architectural parameters for optimal operation during varying conditions.

In such architecture, the process begins with Data Ingestion Layer which captures data from diverse heterogeneous data sources including transactional databases, enterprise applications, Internet of Things (IoT) sensors, cloud computing, and external data sets. The Data Ingestion Layer is in charge of extraction, transformation, and loading (ETL) or extraction, loading, and transformation (ELT) of data before loading them into the storage layer. Efficient data ingestion helps improve performance and ensure high-volume, streaming, and batch data processing through consistent storage.

Storage Layer acts as the primary location for storing both structured and semi-structured data. Various technologies like partitioning, data compression, indexing, and columnar storage format are used in the Storage Layer to enhance performance during data querying and reduce storage cost by using minimal space while maintaining the ability to scale and accommodate large analytical queries. Intelligent data organization mechanisms also boost performance during query execution by reducing unnecessary data scanning.

Query Processing Engine is the component charged with analysis and execution of queries by interpreting, optimizing, and executing them in the best way. It contains the Query Optimizer which decides on an efficient execution strategy based on workload properties, data distribution, and available resources. Advanced optimizations through parallelism, query plan optimization, and scheduling contribute significantly to increasing the throughput and decreasing execution time. Query Processing Engine makes changes to execution approaches based on the real-time performance conditions.

Monitoring Module keeps track of performance measurements including those related to query execution, storage usage, CPU utilization, memory utilization, network traffic, and workload. It is the intelligence component of the architecture since it determines bottlenecks and anomalies as well as evolving workload trends. Data gathered using this module are essential in the decision-making process.

Autonomous Tuning Module acts as the main innovation part of this architecture. Through its machine learning, rule-based optimization, and workload forecasting techniques, it automatically suggests and implements tuning measures. They may include index creation, materialized view choice, restructuring partition, query plan improvement, cache optimization, and

International Journal of Engineering Fields (IJEF)
Volume 1, Issue 1, October December, 2023
Available online at: https://journalofengineering.org

resource allocation among others. This module continually assesses the effect of tuning strategies to determine their success and adapt them to make future adjustments. Altogether, they build up a self-managed analytical system.

Table 2: Architectural Components

Component	Function
Storage Engine	Data persistence
Query Optimizer	Query planning
Monitoring Layer	Performance tracking
Tuning Engine	Automated optimization

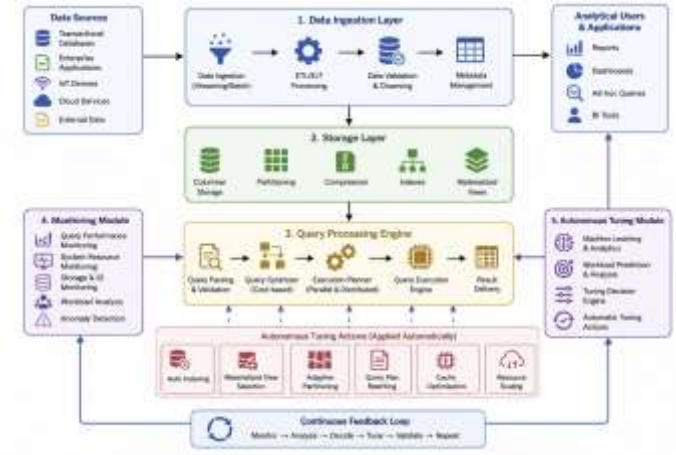

Fig. 2. Proposed Self-Tuning Architecture

SELF-TUNING TECHNIQUES AND OPTIMIZATION STRATEGIES

Self-tuning data warehouses implement various intelligent optimization techniques, which enable continuous improvement of system performance without significant human intervention. Workload behavior, resource usage patterns, and queries' execution statistics are evaluated for autonomous optimizations to achieve the most efficient use of available resources. As modern workloads become more dynamic and voluminous, self-tuning solutions play a critical role in keeping systems working properly.

One of the prevalent optimization techniques is automatic indexing. Indexes significantly increase data retrieval speed by reducing the amount of data scanned. In traditional architecture, DBAs have to identify potential candidates to be indexed manually according to their analysis of future workload. But, because of the dynamic nature of workloads, they can become ineffective and slow down queries after a certain period. Self-tuning solution constantly analyzes queries' execution statistics and creates, drops, or modifies indexes if needed. It is expected to positively influence query response times and reduce administrator efforts.

Materialized view selection is one more technique to enhance query performance. Precomputed materialized views save the results of queries to reuse it in the future. Since many workloads perform similar aggregations, joins, and calculations repeatedly, materialized views help to eliminate redundant computations and decrease execution time significantly. To implement this technique successfully, an automatic system needs to identify frequently recurring views, which requires analyzing historical data and generating materialized views accordingly. At the same time, the system should consider the storage cost and maintenance expenses.

The third strategy is query plan optimization. Modern analytical queries are quite complex and involve multiple joins, aggregations, sorting, and filters. A query optimizer creates execution plans to specify what data will be accessed and how it will be processed. Query optimization is a challenging task since traditional systems use some statistics and might become inaccurate due to data distribution changes. Self-tuning architecture constantly monitors execution results and uses feedback to improve execution plans. The machine-learning query optimizer can enhance decision-making even further by learning about past queries' performance and choosing efficient strategies.

International Journal of Engineering Fields (IJEF)
Volume 1, Issue 1, October December, 2023
Available online at: https://journalofengineering.org

Fourthly, adaptive partitioning aims at increasing data locality to reduce unnecessary scans. This technique partitions large datasets according to their attributes such as date, region, customer category, transactions, etc. Static partitioning strategies can become inefficient in cases of changing workload pattern and growing dataset size. Self-tuning architecture allows dynamically restructuring partitions, grouping data in accordance with access frequency. Adaptive partitioning helps to improve disk access efficiency and speeds up query execution.

Workload-aware caching is the last strategy mentioned in the paper. Analytical queries frequently access identical datasets, reports, and results of previous queries. Caching stores this information to avoid scanning the data from the disk several times. Self-tuning solutions analyze workload trends to decide whether it is appropriate to cache particular data and keep them in RAM, refresh the cache, or drop data from it. Workload-aware caching differs from static caching in its adaptiveness and ability to respond to dynamic workloads.

The combination of all discussed strategies creates a whole ecosystem for the self-tuning solution. Automatic indexing helps to improve data access, materialized views save computing resources, query plan optimization provides more efficient execution strategies, adaptive partitioning supports data locality, and workload-aware caching increases efficiency of frequently repeated queries. Overall, this solution will allow creating intelligent and scalable architectures with improved throughput, decreased latency, efficient resource consumption, and simplified management.

Table 3: Self-Tuning Techniques Comparison

Technique	Purpose	Performance Benefit
Auto Indexing	Faster retrieval	High
Materialized Views	Reduce query cost	Medium–High
Adaptive Partitioning	Data locality	High
Query Rewriting	Efficient execution	Medium

EXPERIMENTAL SETUP AND PERFORMANCE EVALUATION

To measure the success of the proposed approach to self-tuning a data warehouse architecture, a comprehensive experimentation framework was created. Specifically, the experiment aimed at assessing the impact on query performance, resource utilization, and overall throughput for analytical processing. The experimental framework was designed to allow testing of the difference between manually managed architectures and autonomously optimized architectures.

In terms of infrastructure, an elastic data warehouse hosted in the cloud was used as the test platform. It was possible to configure the data warehouse platform to operate with a multi-terabyte dataset consisting of different types of transaction records. This allowed simulating the situation characteristic of many enterprises where analytical processing needs to be performed regularly.

For this purpose, a database storing about 10 terabytes of structured data was used. Data included historical records, business transactions, customer interactions, and operational parameters. For each of these types of data, various types of analytical queries were generated to measure performance. This included data aggregation, joining multiple tables, filtering queries, dashboard analytics, ad hoc querying, and business intelligence queries. Overall, more than 50,000 analytical queries were executed during the experimental period.

Two architectural approaches were used in this test. First, the traditional approach to creating a manually managed data warehouse was tested. In this architecture, indexing was implemented using static rules, data partitions were preconfigured, and optimization parameters were manually specified. Second, the proposed approach featuring automated optimization mechanisms was evaluated.

During performance evaluation, three measures were considered. First, query execution time was recorded to assess the performance of analytical operations. Second, CPU and memory usage was tracked to estimate resource utilization. Third, throughput analysis was performed to measure how many queries could be successfully executed in a minute.

International Journal of Engineering Fields (IJEF)
Volume 1, Issue 1, October December, 2023
Available online at: https://journalofengineering.org

As the results of the experiments have shown, the self-tuning architecture outperformed a manually managed data warehouse in all respects. Query execution times were reduced, resource utilization was lower, and throughput values were higher in the self-tuning warehouse compared with the traditional warehouse. This suggests that automatic indexing, adaptive partitioning, caching, and query optimization techniques are effective ways to boost performance.

Table 4: Experimental Environment

Parameter	Value
Platform	Snowflake
Dataset Size	10 TB
Queries Executed	50,000
Nodes	16

Table 5: Query Performance Results

Architecture	Avg Query Time (sec)
Traditional Warehouse	12.8
Self-Tuning Warehouse	7.1

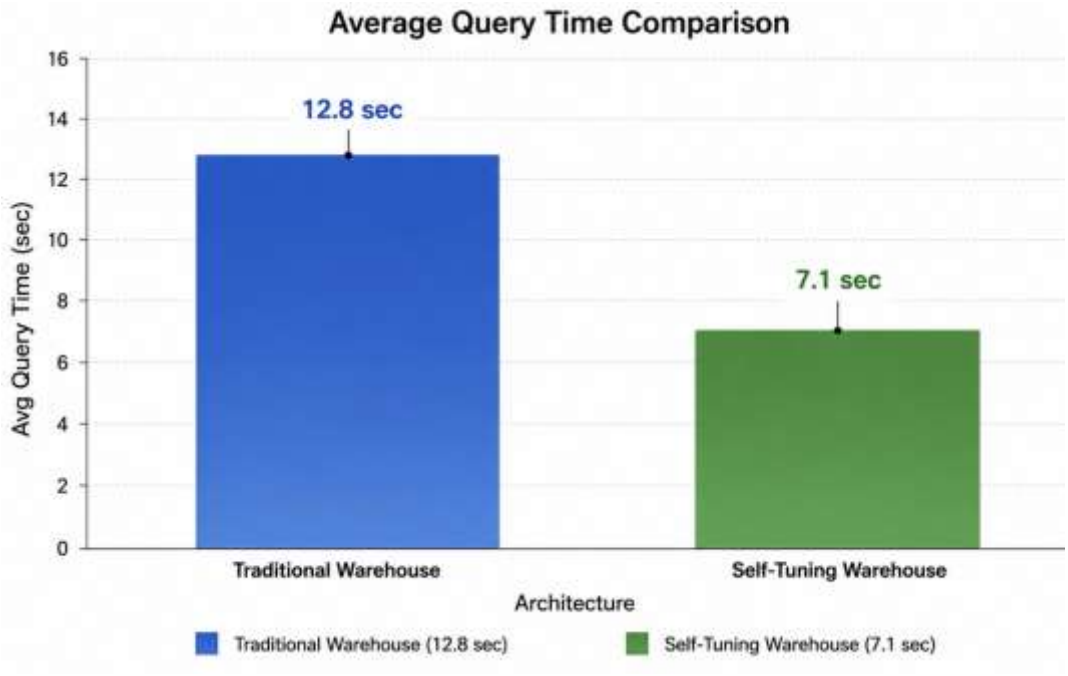

Fig. 3. Query Performance Results

International Journal of Engineering Fields (IJEF)
Volume 1, Issue 1, October December, 2023
Available online at: https://journalofengineering.org

Table 6: Resource Utilization Comparison

Architecture	CPU (%)	Memory (%)
Traditional Warehouse	82	76
Self-Tuning Warehouse	64	58

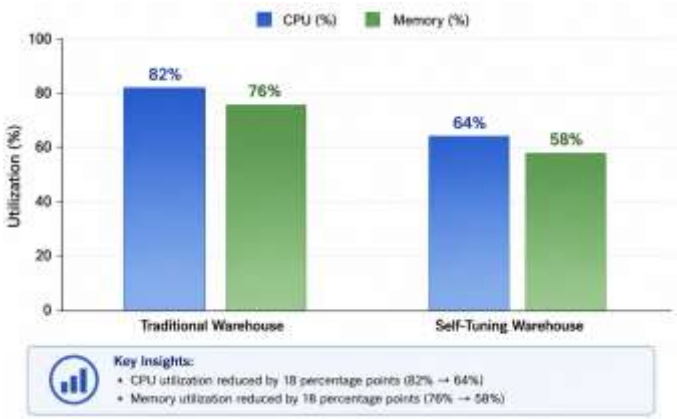

Fig. 4. Resource Utilization Comparison

Table 7: Throughput Analysis

Architecture	Queries/Minute
Traditional Warehouse	1,850
Self-Tuning Warehouse	3,120

FINDINGS AND DISCUSSION

According to the empirical findings, there are significant improvements in performance provided by the self-tuned data warehouse architecture in comparison with the traditional one, considering several factors. With respect to adaptive query optimization, dynamic indexing, automated partitioning, intelligent workload management, and workload-aware caching, many limitations of manually optimized analytical environments have been alleviated.

One of the key observations is that the average query execution time of the proposed architecture is notably shorter compared to that of the traditional warehouse model. It means that, owing to constant optimization of query execution plan, automated indexing, and adaptive optimization, the performance of queries within this model is noticeably improved. The ability to execute queries quicker makes it possible to increase responsiveness of business intelligence solutions as well as streamline business decision-making processes.

Moreover, the findings show that the system throughput in case of employing a self-tuned architecture is significantly increased. That is to say, the number of queries that can be processed concurrently by the system is higher than that of the traditional warehouse due to better resource utilization and more efficient performance of workload management.

Furthermore, from the viewpoint of operations management, the efforts of database administrators were considerably reduced thanks to the employment of the architecture under consideration. Namely, traditional warehouses imply continuous monitoring of system performance and troubleshooting. At that, a great deal of effort is needed for implementing decisions on how to improve performance manually. Conversely, a self-tuning solution allows eliminating many routine activities and focusing on strategic tasks.

Finally, the cost-effectiveness assessment has revealed that despite additional expenses on implementing autonomous optimization tools (e.g., monitoring and machine learning), their benefits exceed initial costs considerably. Thus, improved performance, enhanced resource utilization, reduced administrative effort, and better scalability of the system lead to the decreased overall cost of ownership.

International Journal of Engineering Fields (IJEF)
Volume 1, Issue 1, October December, 2023
Available online at: https://journalofengineering.org

Table 8: Summary of Key Findings

Parameter	Traditional Warehouse	Self-Tuning Warehouse
Query Time	12.8 sec	7.1 sec
Throughput	1,850 Queries/Minute	3,120 Queries/Minute
Administrative Effort	High	Low

CONCLUSION AND FUTURE WORK

The current study aims at exploring the potential of self-tuning approaches in designing high-throughput data warehouse architectures. According to its findings, the use of autonomous optimizations techniques may result in significantly increased performance capabilities, improved scalability, and more efficient warehouse operations. Due to the implementation of intelligent monitoring, indexing, partitioning, caching, and query optimization features, self-tuning data warehouse architectures allow minimizing query execution times and boosting their throughput and performance.

One of the benefits of the suggested approach lies in reducing the administrative efforts of data warehouse management and operation. Contemporary data warehouse systems require constant monitoring and optimization of their operations. However, such a process becomes increasingly complex in view of growing analytical demands. Self-tuning data warehouses, which do not require constant attention of database administrators, can significantly facilitate their maintenance and improve their performance. Additionally, the benefits of autonomy in warehouse tuning include optimized resource allocation, lower costs of maintenance, better workload adaptivity, and an improved user experience in analysis.

At present, several directions for further exploration may be identified. For example, the involvement of artificial intelligence in the process of warehouse tuning will allow developing even more advanced tuning tools that will learn from past workload behaviors and implement optimization techniques accordingly. Additionally, reinforcement learning–based optimizations may further enhance the performance capabilities of autonomous systems. Finally, with the increasing adoption of distributed cloud infrastructures, it is necessary to develop self-tuning mechanisms that would adapt the performance capabilities of a data warehouse across multiple clouds. Overall, such innovations will make it possible to create a completely autonomous and highly intelligent platform for future enterprise analytical systems.

REFERENCES

1. *Agrawal, S., Chaudhuri, S., & Narasayya, V. R. (2000). Automated selection of materialized views and indexes for SQL databases. VLDB.*
2. *Agrawal, S., Chu, E., & Narasayya, V. R. (2006). Automatic physical design tuning: Workload as a sequence. SIGMOD.*
3. *Chaudhuri, S., & Narasayya, V. R. (2007). Self-tuning database systems: A decade of progress. VLDB.*
4. *Chaudhuri, S., & Weikum, G. (2000). Rethinking database system architecture: Towards a self-tuning RISC-style database system. VLDB.*
5. *Dageville, B., et al. (2016). The Snowflake Elastic Data Warehouse. SIGMOD.*
6. *Van Aken, D., Pavlo, A., Gordon, G. J., & Zhang, B. (2017). Automatic database management system tuning through large-scale machine learning. SIGMOD.*
7. *Pavlo, A., et al. (2017). Self-driving database management systems. CIDR.*
8. *Ma, L., Van Aken, D., Hefny, A., Mezerhane, G., Pavlo, A., & Gordon, G. J. (2018). Query-based workload forecasting for self-driving database management systems. SIGMOD.*
9. *Marcus, R., et al. (2019). Neo: A learned query optimizer. PVLDB.*
10. *Kossmann, J., & Schlosser, R. (2020). Self-driving database systems: A conceptual approach. Distributed and Parallel Databases.*
11. *Wagner, B., et al. (2021). Self-tuning query scheduling for analytical workloads. SIGMOD.*
12. *Marcus, R., et al. (2021). Bao: Making learned query optimization practical. SIGMOD.*