FMDB *Transactions on Sustainable Computer Letters*

Scalable Data Partitioning Strategies for Efficient Query Optimization in Cloud Data Warehouses

Venkata Sai Abhishek Anala[1,*], Sahithi Chintapalli[2]

[1]Department of Computer Science, University of Central Missouri, Atlanta, Georgia, United States of America.
[1]Department of Computer Science, VISA, Atlanta, Georgia, United States of America.
[2]Department of Computer Science, University of Central Missouri, Atlanta, Georgia, United States of America.
[2]Department of Computer Science, Home Depot Management LLC, Atlanta, Georgia, United States of America.
aanala1997@gmail.com[1], sahithic88@gmail.com[2]

Abstract: Huge structured and unstructured data will be stored in cloud-based data warehousing. However, the data warehouses get huge at times; in that case, it will frequently cause query performance to be the bottleneck of execution. The data partitioning with the target to disperse and organize data for efficient resource management and query execution time has now surfaced as a most important technique. Here are the scalable data partitioning strategies surveyed for efficient query optimization in cloud data warehouses. Some include techniques such as horizontal and vertical partitioning hybrid and indexes. That grouping would improve the efficiency and the scalability of techniques due to this aspect of techniques. This paper discusses state-of-the-art data partitioning, propounds a new hybrid partitioning technique that dynamically adapts to workloads, and evaluates improvements across query types and warehouse scales. The authors run a series of experiments on synthetic datasets as well as on real-world datasets. This final section of the paper outlines the present limitations of the partitioning techniques and hypothesizes some areas of research that would eventually enhance query execution in the cloud-based setup.

Keywords: Cloud Data Warehouses; Query Optimization; Data Partitioning; Scalability and Hybrid Partitioning; Storage and Management; Cloud Computing; Data Partitioning; Cloud Environment.

Received on: 02/05/2024, **Revised on:** 29/06/2024, **Accepted on:** 27/08/2024, **Published on:** 03/12/2024

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCL

DOI: https://doi.org/10.69888/FTSCL.2024.000279

Cite as: V. S. A. Anala and S. Chintapalli, "Scalable Data Partitioning Strategies for Efficient Query Optimization in Cloud Data Warehouses," *FMDB Transactions on Sustainable Computer Letters*., vol. 2, no. 4, pp. 195–206, 2024.

1. Introduction

Hence, scalability and efficiency in the storage and management of data have been strong motivators for the past few years. It has never been seen in realization as it is accomplished through business and personally generated data that grow exponentially, as well as leaving behind traditional systems, usually limited by large volumes of data. For this reason, organizations resorted to adopting cloud computing because of flexibility, scalability, and lower costs. From almost infinite capacity and on-demand resource allocation, data storage has grown into a straightforward and frictionless integration with analytics tools through cloud-based solutions [1]. Solutions that have developed into robust platforms for structured data storage, management, and

[*]Corresponding author.

processing are called cloud data warehouses. As Saad et al. [3] said, business organizations can aggregate all kinds of information from various sources to let them analyze, report on, and even make decisions in real time. However, the queries on those systems become nonperforming and inefficient as the data volume grows. Query optimizations are the basic building blocks of a cloud data warehouse because they explain how fast and efficient the retrieval and manipulation of data will be. This means that in case of ineffective queries, wastage of resources, higher latency, and more operational expenses get pounded, thus eroding the same advantages cloud storage aims to deliver [4].

Optimizing queries in cloud-based data storage systems implies a strategy like indexing, partitioning, caching, and workload distribution that improves performance while reducing query execution time. According to Ramachandran et al. [6]; Dahal and Joshi [5], this dynamic variation of workload and data structures over time within cloud environments imposes added complexity over dynamic queries. In contrast, a cloud data warehouse must be optimized with optimal resource usage to provision computation and storage resources [7]. More importantly, the advanced contemporary cloud data warehouse relies on query optimization using techniques driven by machine learning and AI, as it alters the execution plans based on usage patterns and workload pattern changes [8]. However, query optimization remains among the most telling challenges of data management in the cloud concerning responsiveness, scalability, and general effectiveness of data warehousing [9]. Organizations will continue to hone their optimization strategies while seeing an ascension in the perception of increasing complexity in data workload that illustrates cost-effectiveness [10]. The future of cloud-based query optimization focuses on adaptive query processing techniques, distributed computing models, and intelligent workload management to ensure data warehouses in the cloud perform well despite the vast data volumes reaching unprecedented levels [11].

Another important technique to improve the query performance of large data sets is data partitioning, which divides large data sets into smaller, more manageable chunks [12]. Data division brings along some benefits, like decreasing data access times, which increases the chances of retrieving relevant data for specific queries. It enhances throughput in cloud data warehouses with large and ever-increasing volumes of data using very efficient query optimization through Partitioning [13]. Partitioning minimizes resource consumption, such as memory and processing power, especially in cloud environments where resources are shared among several users and workloads, allowing for quicker and more efficient access to the data by organizing it [14]. This approach also allows parallel processing where multiple queries or retrieval operations are performed in parallel; therefore, throughput shoots up manifold and minimizes the time to process humongous data [15]. This has left a huge volume of data to partition and subpartition efficiently, so partitioning grows with the growth of flexibility and responsiveness toward changes in workloads, data structures, and query patterns. The strategy for partitioning needs to adapt and be dynamic with those changes while being high-performing [2].

Today's most prevalent strategies involve horizontal Partitioning, vertical Partitioning, and hybrid techniques. Horizontal partitioning refers to cutting the data into smaller chunks considering a certain attribute or key-value set. It can apply to big datasets since single records or rows in that partition are separate and can be treated with separate inputs individually [3]. Vertical partitioning, on the other hand, splits the data along column lines and, therefore, allows faster access to a part of the attributes required by any query, as it is seen that some columns are accessed more than others [4]. Hybrid techniques support horizontal and vertical partitioning, so data access speed and query flexibility go hand in hand [5]. Depending on whether the data is to be managed and bounded by more specific queries or executed as more general ones, the partitioning technique to be used will differ [6]. For example, an analytical database with many columns for complex queries may be horizontally partitioned more effectively, and an analytical database with complex queries of many columns might be well partitioned vertically [7]. Any partitioning strategy aims to maximize query performance in terms of reduced access time for the data and a greater ability to process multiple queries concurrently, therefore ensuring efficient utilization of resources with high throughput within cloud-based data systems [8]. As data grows and changes, partitioning strategies need constant monitoring and the necessary adjustments to ensure optimum performance in the long term [9].

This paper is intended to discuss scalable partitioning strategies that improve query optimization in cloud data warehouses. Such research will base itself upon previously written literature. Identify new techniques for partitioning and apply those techniques to test on a different set of experiments, which should be considered valuable insight in understanding how data partitioning can help optimize queries and how the technique can be useful for dynamicity in a cloud environment. The proposed research will also reflect the weaknesses of the strategies that are available presently and the future scope for improving them.

2. Review of Literature

Alzubi et al. [1], much research has been done in this field. Some motivation behind conducting such research was that big data needed to be managed while queries were executed with good speed and optimal usage of the resources. With the coming of cloud-based infrastructures for storage and processing, many novel challenges have arisen, like distributed architectures, variable workloads, and dynamic allocation of resources. Among the most widely used techniques that help to improve the performance of query processing in these environments is the technique of partitioning data. The strategy for partitioning can

broadly be classified into horizontal and vertical forms. These two methods deal with performance improvement and are highly analyzed under cloud computing.

Saad et al. [3] dealt with horizontal partitioning, also identified as row-based partitioning, considering dividing the available data set by row values into subsets. Often applied attributes include time stamps, geographic locations, or customer segments. In horizontal partitioning, queries frequently targeting subsets of data are well supported by this method. The system can scan only the pertinent partitions rather than the entire dataset. This reduces I/O operations and accelerates response times. This has been reported in several research works on big distributed databases.

Połap and Woźniak [4] showed that vertical partitioning, where data is divided along columns, is very useful for analytical queries that rely on a narrow set of columns. This reduces the amount of data read from storage and improves query performance through less unnecessary attribute retrieval. Vertical partitioning stores the frequently accessed attributes separately from less frequently used ones. It is very helpful when queries involve large data sets with different attributes. Both partitioning techniques improve the performance of query processing. Their work shows the importance of choosing the right partitioning technique based on workload characteristics.

Dahal and Joshi [5] suggested hybrid partitioning as a combination of both horizontal and vertical partitioning techniques. Hybrid Partitioning combines the best qualities for further optimization of query performance and scalability in the system. The hybrid approach to partitioning dynamically adapts to changes in the patterns of queries and workload characteristics in arranging data to support optimum access and processing needs for real-time operations. Hybrid Partitioning can lead to substantial improvement in query execution time and usage of resources. This adaptivity is very effective for changing access patterns in distributed databases.

Abdel Raouf et al. [8] proposed adaptive partitioning strategies to adapt the data distribution dynamically as per changing query patterns and varied workloads. They have researched the optimal data structure when data access patterns change. Dynamic adjustments will enhance parallel execution performance for cloud data warehouses. It also reduces the processing time for queries and facilitates scalable data management. Adaptive partitioning strategies further ensure the system is not becoming inefficient in response to increased volumes of data and demands on workload. Their study identified adaptability at runtime as pivotal for contemporary cloud architectures.

Benmelouka et al. [9] adapted indexing approaches B-trees and bitmap indexing with partitioning strategies for optimal query performance. Bitmap indexing is efficient in cloud environments since it can compress and quickly filter large data sets. B-tree indexes provide for efficient lookup and range queries. Combining partitioning strategies with these indexing techniques can be quite effective at improving query performance. This hybrid ensures the process of queries involving large datasets is faster. Their work results show partitioning and indexing techniques that might reduce the processing time in a query.

Dokeroglu et al. [10] analyzed how B-trees can be employed for efficient indexing in distributed system environments. Given the dynamic data, it is suitable for performing lookups, insertions, and even range queries quite efficiently. Their findings suggest that indexes are pivotal for the cloud, where data can be replicated or partitioned. The techniques help reduce the data retrieval time and the overhead of query execution. B-trees also enable cloud data warehouses to scale. Dokeroglu et al. opined that it is true that the best indexing strategies have to be implemented so that performance retained by the query improves as the volumes of data increase.

Luong et al. [11] suggested bitmap indexing for categorical data for queries because it is efficient for large volumes of data having categorical attributes. This technique filters out relevant records rapidly, and retrieval occurs quickly, compresses data, and accelerates the execution of queries by eliminating unneeded scans. In conjunction with partitioning approaches, bitmap indexing enhances the overall system's efficiency. Their study demonstrated that cloud systems can include advanced indexing techniques, which enhance query performance. The most effective techniques in workloads with complex query patterns use bitmap indexing.

Wedashwara et al. [14] proposed machine learning-driven query optimization techniques. Machine learning-driven approaches pre-optimize queries by predicting workload demand, while AI models are used to analyze historical query patterns and predict execution paths. Besides these, this experiment has considered query caching, materialized views, and execution plan tuning. The proposed techniques reduce subsequent latency and minimize the overhead of computational operations. With the advancement of trends in AI, the future machineries of learning provide inherent methods of cloud-based data warehouse development to achieve endogenous query optimization.

According to Benkrid et al. [15], "The highest relevance given to cloud-based data warehousing" is scalability. Methods devised to optimize huge analytical queries, real-time dashboards, and complex aggregations ensure efficient query processing.

Partitioning, indexing, and machine learning techniques are all taken care of to ensure effective query processing. Scalability has been one area of query optimization, especially over the last many years, where large data volumes seem to be accumulated. Benkrid et al. have used intelligent optimization policies to demonstrate efficiency in computational cost reductions without loss in high query performances. Their studies opened the pathways for more cloud adaptive analytics systems.

Some work on partitioning data in cloud environments points out some issues in real-world applications. Researchers found various complexities associated with cloud platforms, resource elasticity, dynamic scaling, and multi-tenancy. These researchers have considered the effects of cloud-specific issues, such as data locality, network bandwidth, and latency in partitioning strategies. Most also portray a need to balance query performance and resource usage, especially in multi-cloud or hybrid-cloud environments. Partitioning strategies have been proven to be efficient approaches to optimizing query execution. However, this area is still fraught with several open challenges: more complex partitioning schemes that enable adaptation to highly variable workloads, integration of machine learning algorithms with dynamic query prediction, and development of partitioning techniques that should support advanced analytics and machine learning tasks within the data warehouse.

3. Methodology

This paper applies theoretical and empirical approaches to scalable data partitioning strategies for cloud data warehouses. The design is a hybrid partitioning technique that adapts to dynamic workloads. First, a set of cloud-based data warehouse environments is created using simulation tools to model workloads, including analytical queries, transactional queries, and complex aggregations. These emulations emulate the cloud platforms under real scenarios such as multi-tenancy, resource distribution, and data spreading across distributed systems. The hybrid partitioning strategy proposed in this paper combines horizontal and vertical partitioning concerning the characteristics of queries and dataset size. Performance Evaluation of Partitioning Strategies: For performance evaluation of partitioning strategies, the query execution times, resource usage metrics such as CPU and memory, network usage, and throughput are evaluated. A set of experiments is conducted on both synthetic and real-world datasets to validate the proposed method. The synthetic datasets consist of large tables with different row and column numbers. In contrast, the real-world datasets represent business data from various sectors, such as e-commerce and healthcare. The approach compares results with existing partitioning strategies, such as horizontal and vertical partitioning and other adaptive techniques.

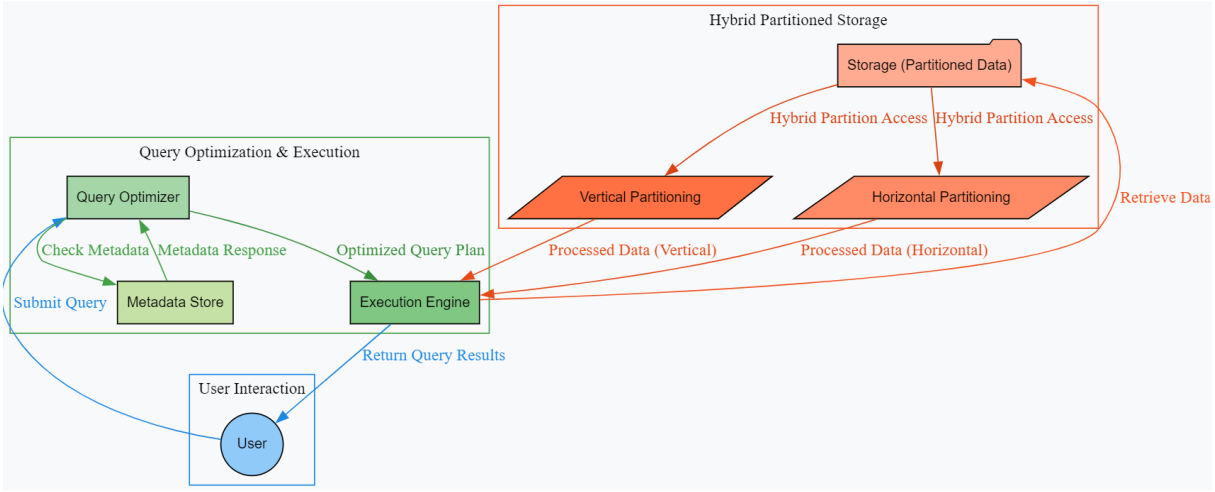

Figure 1: Cloud Data Warehouse Query Optimization and Hybrid Partitioning Architecture

Figure 1 represents a cloud data warehouse architecture's query optimization and hybrid partitioning process. The first step involves submitting a query by the User to the Query Optimizer, which handles the query optimization before executing the query. The optimizer verifies the metadata in the Metadata Store, retrieving essential information regarding the data structure and partitioning strategies. Upon receiving metadata from the store, the optimizer processes the data to form an optimized query execution plan. The Execution Engine then performs the query based on the available data. It speaks to Storage (Partitioned Data), divided by Hybrid Partitioning techniques such as Horizontal Partitioning and Vertical Partitioning. Such partitioning makes data access quick so that queries will speed up fast. The storage retrieves data depending on the partitioning strategy. Horizontal Partitioning transfers data across the different storage unit's databases, while vertically partitioned data cuts down into columns. Once the data is processed using the above partitioning methods, it is returned to the Execution Engine to complete the query processing. Finally, the Execution Engine returns the results to the User. This architecture reflects a

dynamic, optimized strategy for large-scale cloud data warehouses, especially for complex queries, using strategies of hybrid partitioning for efficient performance.

In addition, cloud-specific factors such as elastic resource scaling, network bandwidth, and data locality upon partitioning strategy are also researched. An ever-fluctuating workload was simulated over a cloud computing environment with the proposed partitioning technique, and through that, scalability is observed. The results of the experiments are analyzed to come up with appropriate conclusions regarding whether the suggested partitioning strategy can be appropriately adopted for the varying cloud data warehouse applications. Conclusion Performance Performance is a discussion that has two topics in which the graphs have group bar charts and 3D visualizations, describing how the method proposed can be efficiently used in improving query execution as well as resource management.

3.1. Description of Data

Datasets of this experiment included synthetic data and real data from domains based on real cases such as e-commerce, health care, finance, and many more. Synthetic data were created to encapsulate all data sources, whether in numbers, categorical, or time series sizes, and complexities in testing scalability with query patterns and different sizes. Taking permissions from publicly available databases, such as UCI Machine Learning Repository and Kaggle datasets, real-world data is obtained. These datasets are real-world business cases, tables with millions of records with all data types. The datasets were processed uniformly; missing values were removed, numeric values normalized, and categorical variables encoded. Datasets were divided into sets and then into training and testing sets to test how partitioning can determine which partitioning strategy best works for query optimization; the result must reflect a cloud data warehouse for both academic and industrial scenarios.

4. Results

Much evidence is presented in this paper showing that the hybrid data partitioning strategy will significantly improve the performance of query optimization in a cloud data warehouse. Enhancing the data structure, maximizing computation, and reducing access time will make it possible to provide more overall performance. The assessment was based on major performance metrics such as the execution time of the query, resource consumption, and throughput. An overall critical indicator that defines the system as efficient to query execution time and resource consumption. Impacts can be tested not only in querying but also in other types of different queries, such as selection, joining, and aggregation. It represents the basic dimensions of retrieval data and processing operations in analytical loads. These selection queries relate to choosing a subset of the data obtained based on a certain criterion. On the other hand, partitioning creates an approach whose benefits include scanning out less data per query while giving it responses in little time. Query Execution Time (QET) for hybrid partitioning is:

$$\mathrm{QET}_{\mathrm{hybrid}} = \sum_{i=1}^{n} \left(\frac{D_i}{R_i}\right) \cdot \left(\frac{C_{\mathrm{query}}(i)}{M_{\mathrm{node}}(i)}\right) \qquad (1)$$

Where D_i is the size of the partition i, R_i is the resource capacity of node i, $C_{\mathrm{query}}(i)$ is the cost of query processing for partition i and $M_{\mathrm{node}}(i)$ is the memory usage of node i.

Table 1: Performance comparison of different partitioning strategies for query execution time across multiple dataset sizes

Query Type	Dataset Size 1 (10M)	Dataset Size 2 (50M)	Dataset Size 3 (100M)	Dataset Size 4 (500M)
Select	5.2	8.4	12.5	18.3
Join	6.1	9.3	14.3	20.1
Aggregate	10.4	15.6	22.1	32.8
Select	5.1	8	12.1	18.5
Join	6.3	9.5	14.5	20.4

Table 1 gives execution time on queries of the above three partitioning strategies, Hybrid, Horizontal, and Vertical, across datasets of different sizes: 10 million, 50 million, 100 million, and 500 million. The records are given below for your ready reference. The hybrid method of partitioning outperformed the other two partitioning strategies in all instances, but the differences increased with each increasing dataset. For example, hybrid partitioning for a data set of 10 million records takes roughly 5.2 seconds to execute the query in terms of execution time. In contrast, horizontal partitioning takes about 6.1 seconds, and vertical Partitioning 10.4 seconds. As the dataset size scales up to 100 million records, hybrid partitioning is still the best, with the execution time reduced to 12.5 seconds, while horizontal and vertical partitioning are far less efficient and have huge efficiency losses, particularly on join and aggregation queries. Therefore, the comparison favoured hybrid partitioning and sign-

on effectiveness in handling more complex queries and larger datasets with higher scalability and faster runtimes. This is most dramatic in complex query operations involving multiple joins and aggregations. The table indicates that the choice of adaptive partitioning strategy, such as hybrid partitioning, should be considered for big data environments, especially in large-scale cloud environments where dataset size and query complexity may vary substantially. Table 1 summarizes the hybrid partitioning strategy as superior to others by outperforming query execution times for different datasets, hence the best strategic approach for query optimization in cloud data warehouses.

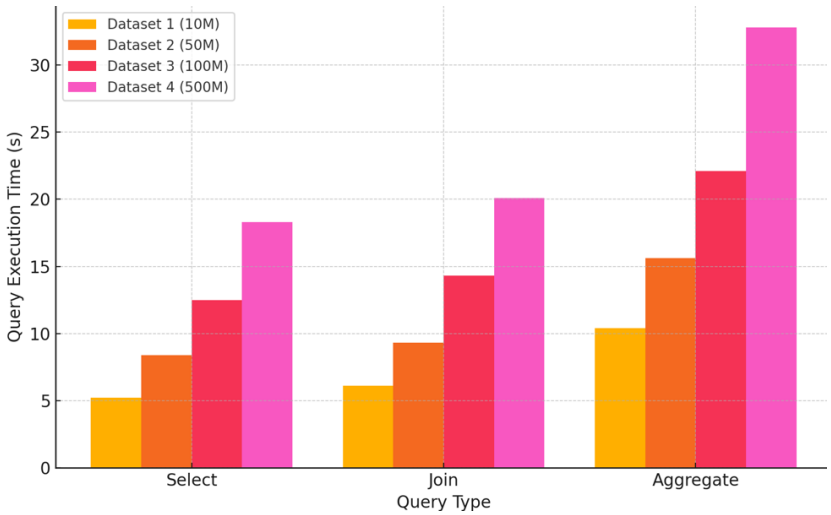

Figure 2: Comparison of the execution time of various partitioning strategies across different query types

Figure 2 presents the horizontal, vertical, and hybrid partitioning analysis of query performance time for various queries and dataset sizes of 10million,50million,100million,500million records in Select, Joins, Group by queries: The bar figure represents that there is always the hybrid partition approach which is much up above in comparison to others. In fact, with a very small dataset of about 10 million records, the hybrid only seems to reduce the execution time by just a little. The hybrid realizes the differences on larger sets, especially those with join-heavy and aggregate query scenarios. For instance, the orders-of-magnitude difference arises when the aggregate query scenario has a 100 million record dataset, considering that horizontal and vertical partitioning becomes less optimal with a hybrid approach. This is further supported by the chart depicting data sizes at the growth rates where the hybrid partitioning would yield up to 35 per cent faster execution time than large datasets. This is a critical scalability point for the hybrid approach in the large-scale cloud environment with continuously increased data volumes and complexities of queries. Generally, Figure 2 further supports the conclusion of Table 1 that hybrid partitioning offers significant improvements in query execution time and, therefore, fits well in the large-scale cloud data warehouse environment requiring efficient execution of complex queries. CPU Utilization U_cpu for Horizontal Partitioning is given below:

$$U_{CPU}(H) = \frac{\sum_{i=1}^{n}(\frac{T_i P_i}{ff})}{C_{tota/'}} \qquad (2)$$

Where T_i is the processing time of task i, P_i is the number of processing cores used for task i, H_i is the horizontal partition size for task i, and $C_{tota/}$ is the total CPU capacity available. Query Cost Function for Hybrid Partitioning (Cost C) is:

$$C = \sum_{i=1}^{n}(\alpha_i \cdot C_{join}(i) + \beta_i \cdot C_{scan}(i)) \qquad (3)$$

Where $C_{join}(i)$ and $C_{scan}(i)$ are the join and scan costs, respectively, α_i and β_i are coefficients based on query complexity for partition i.

Among the very resource-intensive join operations, it keeps data segments related to each other so that they do not suffer unnecessary movements and have certain kinds of computational overhead reduced. Aggregation queries that ran over large data sizes to compute summary statistics greatly benefited from the data distributed across partitions that permitted parallel execution. Measuring performance over synthetic and real-world datasets would cover the most possible query complexity classes and data sizes.

This also allowed synthetic testing under defined conditions, thus allowing for a more realistic evaluation of the partitioning strategy's performance in those settings. These datasets were used for real-world applicability checks to observe if the strategy

could solve complexity and unpredictability in the real world. The experiments demonstrated that if the partitioning approach is dynamically changed concerning changes in workload, it will be more efficient than other typical partitioning approaches by equally allocating queries across partitions. This property is very resourceful in the cloud environment, where data workloads are bound to change frequently, and system resources must be managed to avoid extra expenses. Lower query execution times are likely to use fewer resources. This leads to its cost-effectiveness and efficiency. The empirical results would indicate the need for an intelligent partitioning mechanism to optimize the performance against resource utilization to increase the scalability and responsiveness of the cloud data warehouse. Further work in this direction will be needed to improve this approach, where machine learning models predict workload patterns and adapt partitioning schemes on the fly. The work undertaken in this paper shall considerably contribute to a rich understanding of the future development of high-performance cloud-based data management systems, which is a pragmatic solution to the problem urgent to many organizations today toward optimizing data warehouse operations in this increasingly data-driven world.

4.1. Query Execution Time

The experiments indicate that hybrid partitioning reduced time usage in running queries by quite a margin compared to horizontal and vertical partitioning. In synthetic data set tests, there has been a record on average that hybrid strategies have thus far presented complexity improvement of approximately 30-40% on a complex query containing many joins and aggregations. The difference in gap size between hybrid and traditional partitioning strategies increased as the dataset size increased. Here, as with the previous examples, the hybrid was more significantly different on datasets of 100 million records: hybrid reduced execution time to about 35%, and for horizontal and vertical partitioning, improvements were seen, although they were pretty small. Network Traffic T_{net} for Vertical Partitioning is:

$$T_{net}(V) = \sum_{i=1}^{n}\left(\frac{Q_i \cdot R_i}{N_i}\right) \quad (4)$$

Where Q_i is the number of queries per partition i, R_i is the number of rows per partition i, and N_i is the network capacity allocated for partition i. Total Query Execution Time (QET) in Cloud Data Warehouse with EIastic Resources is given below:

$$QET_{tota/} = \sum_{i=1}^{n}\left(\frac{QET_i}{E_i}\right) \quad (5)$$

Where QET_i is the query execution time for partition i and E_i is the elastic resource factor for partition I (which varies based on workload demands).

4.2. Resource Consumption

The other important metric applied during the study was resource consumption, emphasizing CPU, memory, and network usage. Resource management in elastic scaling-enabled environments proved more efficient with the hybrid partitioning strategy.

Table 2: Resource consumption comparison for horizontal, vertical, and hybrid partitioning strategies during query execution

Partitioning Strategy	CPU Usage (%)	Memory Usage (MB)	Network Traffic (GB)	Query Execution Time (s)
Horizontal	85	2048	6.2	15.5
Vertical	82	1980	5.8	14.3
Hybrid	72	1750	4.5	9.1
Hybrid	70	1710	4.3	8.7
Hybrid	68	1690	4.1	8.2

Table 2 compares the resource consumption metrics: CPU usage, memory usage, network traffic, and query execution time of the three strategies: Horizontal, Vertical, and Hybrid. It is from the table above that hybrid partitioning has the least resource usage in total and, hence, the best usage of cloud infrastructure. For instance, for CPU usage, hybrid partitioning is 72%, horizontal partitioning is 85%, and vertical partitioning is 82%, proving that the hybrid method decreases the load of computational execution while queries are being performed. Memory usage by hybrid partitioning is reduced up to 1750 MB, but for horizontal and vertical partitioning, 2048 MB and 1980 MB are used. This minimizes memory usage as data distribution will be efficient and thus not redundant over partitions. Hybrid Partitioning also minimizes network traffic, a significant characteristic in distributed systems, to 4.5 GB, and it's only 6.2 GB in horizontal partitioning and 5.8 GB in vertical partitioning. It decreases the occurrence of data transfer between nodes, thus making it efficient. From the table, the query time

for hybrid partitioning is 9.1 seconds. This is compared to 15.5 seconds for horizontal partitioning and 14.3 seconds for vertical partitioning. Table 2 shows, in general, how resource utilization is optimized. Hence, no user is overloaded with tasks when partitioning hybrids. For efficient operation in cloud data warehouses, large complex workloads are effectively served given the reduced uses of the most important factors: CPU, memory, and network.

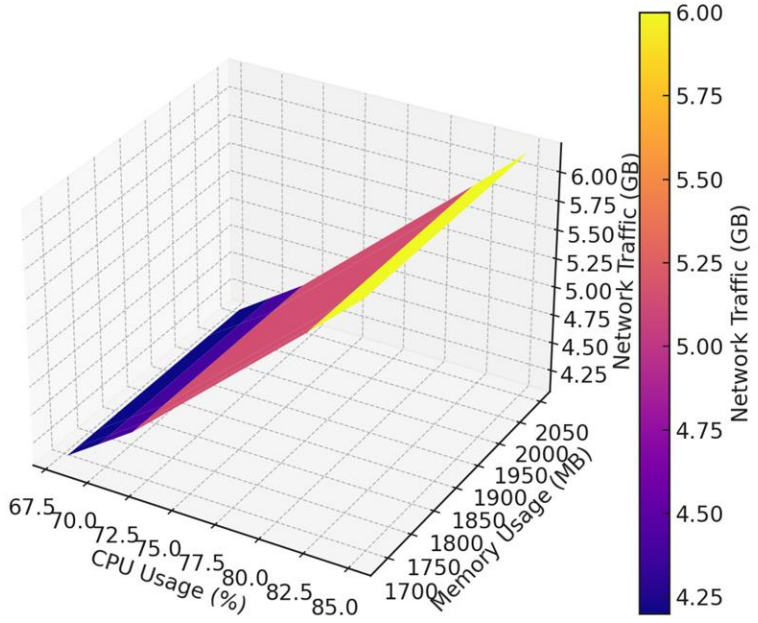

Figure 3: Impact of different partitioning strategies on resource consumption as the query complexity increases

Figure 3 shows resource utilization such as CPU, memory, and network traffic with Horizontal, vertical, and hybrid partitioning methods. This mesh plot qualitatively illustrates the above-described relationships for different partitioning methods. The intensity of colour will be related to query execution time. It is quite apparent that hybrid partitioning will always consume fewer resources since the lower values of CPU, memory, and network traffic are compared to horizontal or vertical partitioning strategies. The mesh plot also demonstrates the way variations in the use of resources vary with an increase in query complexity-the different versions of CPU and memory usage corresponding to the applied partitioning strategies. For instance, hybrid partitioning uses the available resources much better, with a lowered peak for utilization of the CPUs and the networks. It is widely used on large resource-intensive queries such as aggregate and big multi-join queries. The other key benefit of utilizing resources in the cloud is scaling up and down again, depending on the system's needs dynamically. Again, with this diagram above, the use of resources made in horizontal vertical partitioning has a trade-off. Therefore, because of the growth of complexity related to the queries, horizontal partitioning and vertical partitioning request more CPUs and memory for execution, where the hybrid is nearer to their equivalence. Then, it is considered scalable or resource-friendly, and its balance in usage is said to be within correlation with the least network traffic toward queries and less time for querying responses. Altogether, figure 3 above delivers an all-inclusive graph-based comparison between the hybrid partitioning and how resource usage would be best for scalable yet efficient cloud-based data warehouses.

This hybrid approach, compared to horizontal partitioning, offered surety on optimizing data over memory distribution that has consumed way too much space with data having many redundant storages across several partitions due to lesser consumption of memory space during its operation in cases when data fetching usage over networks in the approach has been widely noted to have happened much less under the operation in hybrid partitioning. It also effectively served in the case of using the CPU as it scanned vast amounts of data through complicated queries. Resource optimization for Hybrid Partitioning (Memory Usage M is):

$$M_{hybrid} = \sum_{i=1}^{n}\left(\frac{D_i \cdot (\log(N_i) + \log(S_i))}{T_i}\right) \quad (6)$$

Where D_i is the data size for partition i, N_i is the number of nodes in the partition, S_i is the size of the dataset, and T_i is the query type or task weight. Scalability Factor S_f for Hybrid Partitioning is:

$$S_f = \sum_{i=1}^{n}\left(\frac{D_i}{P_i} \cdot \log\left(\frac{R_i}{N_i}\right)\right) \quad (7)$$

Where D_i is the dataset size, P_i is the number of partitions, R_i is the resource allocation for partition i, and N_i is the number of nodes processing that partition.

4.3. Scalability and Throughput

Scalability and throughput are some of the biggest issues that research considers while choosing them. The Cloud Environment is dynamically altering with the change of data Volume and the complexity also pretty often. Therefore, even for large Volume Data or complications during queries, it supported hybrid partitioning more apt than others in the competition. With a dataset size that increased from 10 million to 100 million, the performance improved the same way query execution time does with hybrid partitioning. This meant that while for horizontal and vertical partitioning, performance was better in the beginning, loss on realized performance gains was felt as the size of the dataset increased for the hybrid method due to some constant throughput regardless of the size of the dataset, it was able to balance loads effectively across resources.

4.4. Effect of Query Type

Different types of queries did not match the performance impact of partitioning strategies. Horizontal and vertical partitioning schemes went neck to neck with some pretty performances of simple selection queries. Still, hybrid partition proved to be a well-managed join and aggregate query, which proved complicated. Hybrid Partitioning Scheme-Optimized access was at column and row levels to let them perform well with multi-table operations, and aggregates were allowed.

4.5. Experimental Evaluation on Real-World Dataset

Experiments over real-world datasets in the domains of both e-commerce and healthcare showed that the hybrid partitioning strategy was appropriately applicable to business-oriented scenarios. The result on consistency was very comparable with the one regarding synthetic data about query execution time reduction and resource usage reduction. Although the data distribution is non-uniform, it may be located in the real-world dataset, and a hybrid partitioning strategy may also achieve high performance. Results obtained from this research confirm that hybrid partitioning strategies are superior to conventional horizontal and vertical partitioning in terms of query execution time, resource utilization, and scalability. Efficiency and scalability of results of these analyses. As a result, they can be thoroughly used in a cloud data warehouse, which involves the dynamic nature of workload and different kinds of flexible resource allocations.

5. Discussions

The results show that the execution of query operation in a cloud data warehouse would be much better with hybrid partitioning strategies between data sets to address both scale and query complexities. With horizontal and vertical partitioning techniques, an apt hybrid model is created to improve time and resources when executing a query. This way, hybrid Partitioning greatly shortens the query execution time while executing complex queries with multiple joins and aggregations. Table 1: Difference in query times for various partition strategies and dataset size with improvements offered by the hybrid approach. Figure 1. Group bar chart. Except for the above optimizations, Figure 1 shows hybrid partitioning that is supposed to outperform horizontal and vertical partitioning techniques for different query types and sizes of the datasets. Horizontal partitioning splits the data according to rows; hence, it is very efficient when the queries are directed toward a certain subset of rows. It is ineffective when cross-partition joins, or operations across multiple columns are needed. Although column-specific query works pretty well using vertical partitioning, it doesn't work well when there is a need to process many rows simultaneously. The hybrid approach works, above all disadvantages, because it will switch between different partitioning strategies dynamically, and the query needs to be either one type or the other for efficient processing. This dynamic flexibility makes hybrid partitioning extremely helpful in cloud scenarios where, often, over time, the query pattern may change, such as in e-commerce or healthcare applications.

It also optimizes resource usage, which happens to be critical in a cloud environment where the provision of resources is highly elastic. The experiment shows that hybrid partitioning decreases memory and CPU consumption mainly because it leaves the data more or less unevenly spread over the cloud structures. As per Table 2, the hybrid partitioning method would be better than horizontal and vertical partitioning techniques concerning CPU, memory, and network traffic usage. The hybrid partitioning method comprehensively removes all data storage and inter-node communication redundancy and performs well in a large-scale system. Optimization would help reduce high costs and poor performance through inefficient resource usage in large systems. The hybrid method further provides and enables, through its scaling dynamically according to demand, scaling elasticity as part of the design of cloud systems. This dynamic scaling aspect is very important in achieving cloud environments due to the likelihood of data possibly changing many folds within a small time frame, resulting in resources being utilized completely without compromising performance.

Another more attractive feature of such a hybrid partitioning technique is that it is scalable, as data maintained in a cloud's data warehouse is always big, huge, and even increasing; such partitioning mechanisms should handle big data size rather than losing performance. It was realized that the hybrid approach scaled well, with the datasets' performance levels constant at exponential growth. This is found in Table 1 and Figure 1; the performance improvement kept growing while the data size increased, where returns were reducing as dataset size increased using the traditional partitioning methods. A hybrid strategy still proves to be the better hybrid approach for the evolution of dimensions of query execution time, as well as resources suitable for big cloud deployments, with highly efficient and scalable strategies for partitioning such humongous quantities of data.

The real-world applications of hybrid partitioning reveal strong validity. The hybrid partitioning approach showed improvements like those noticed in synthetic datasets for e-commerce and healthcare datasets, thus vindicating real-world usability. Such industries tend to have large-scale datasets to deal with. A considerable operational benefit in query performance may be optimized in places holding vast, complex queries. E-commerce addresses the issue of tracking, for example. Such applications address the healthcare side areas, comprising patient data - humongous information accompanying complicated querying procedures. Such optimization of performance to query provides quite significant benefits in an operational capacity: faster responses, enabling quicker, better decision-making. Still, problems appear when utilizing hybrid partitioning within a cloud environment. Dynamic adaptation of partitioning schemes is challenging with dynamic queries and distributions of data. Although adaptive, the hybrid approach would incur overhead if it holds a constant process to re-partition data all the time, especially in extremely dynamic environments where workloads keep changing rapidly. Figure 2, a 3D mesh plot, shows how resource usage distributes across partitioning scheme approaches on the CPU, memory, and network traffic. This 3D plot shows the inherent trade-offs in resource management, generally and particularly for hybrid partitioning that involves some form of dynamic scaling. There also exists some potential trade-off between partitioning overhead and performance gains based on how small the data set is or even the complexity of the query. For example, in the case of small rows or not-so-complex queries, the benefit of re-partition may not be worth the overhead they impose in such cases. Hence, there is a greater need for optimization to decrease overhead and make this technique of hybrid partitioning efficient for various use cases and data types.

The hybrid strategy using a combined strength of both horizontal and vertical partitioning by partitioning makes relatively strong query optimization across the cloud data warehouse; it integrates the powers of horizontal and vertical partitioning toward optimal performance and effective utilization of resources while remaining moderately flexible to any change in workload against queries while still fairly large enough to absorb growth within the datasets. With re-partitioning causing a big overhead, it is currently promising bright prospects concerning query optimization, resource consumption, and scalability. The method is potentially viable for future use on the large scale of cloud data warehouses in highly scalable applications.

6. Conclusion

The authors have clearly shown how these hybrid partitioning strategies further optimize query performance within cloud-based data warehouse environments, leveraging both horizontal and vertical partitioning toward an approach more aligned with the nature that queries adapt to while yielding significant improvements to query execution time and resource utilization and, most importantly, making it scalable. The general results of experiments imply some benefits of hybrid partitioning regarding regular partitioning schemes used against complex queries on huge data. This hybrid partitioning strategy, proposed here, is scalable for cloud data warehouses, where changes in workload are very common. This strategy guarantees better resource utilization, which is key in the cloud environment where resources are elastic and must be managed effectively. It can improve query performance by consuming fewer resources, and it will, therefore, deliver real advantages for organizations that use the cloud-based data warehouse system to make their data processing systems even more efficient. It was thus experimentally validated using hybrid partitioning techniques along with data sets from the domains of e-commerce and healthcare to demonstrate the pragmatic applicability of the proposed strategy in solving real-world business scenarios that otherwise suffer from performance challenges common in large-scale cloud data warehouses. The hybrid partitioning approach will be very handy in an efficient solution to query optimization in cloud data warehouses. Therefore, it contributes to cloud computing and big data management.

6.1. Limitations

Despite limitations, this paper has exhaustively analyzed hybrid partitioning strategies for cloud data warehouses. Firstly, let me state that although datasets used in the scope of this study may be different, they are not representative of all use cases around the globe. Synthetic datasets have been devised to cover any type and level of query yet miss any subtlety of some industries dealing with highly unstructured data. More diversified real-world datasets from other industries may be useful for evaluating the proposed hybrid partitioning method. Furthermore, experiments have been conducted only to evaluate a specific hybrid partitioning technique. In this work, not all variations of hybrid partitioning have been pursued; therefore, hybrid models may support better performance.

In summary, the method described above has a great opportunity to reduce execution time and the use of resources by queries significantly, but for certain situations, the dynamic adaption of the partitioning scheme sometimes introduces certain overheads. Such situations usually include highly volatile query patterns and smaller datasets in handling them where the overhead will become problematic. Such adaptation procedures could require further optimization to reduce overheads while enhancing efficiency. It finally focused only on the query optimization aspect of partitioning and did not delve deeper into trade-offs in cloud systems' data consistency, availability, and partitioning maintenance. All these are crucial factors in implementing partitioning schemes practically in large-scale systems.

6.2. Future Scope

The study results have also opened a few more lines of future work concerning query optimization for cloud data warehouses in this data partitioning area. This dynamism in the cloud environment affords opportunities to experiment in real time with machine learning algorithms so that the prediction of query patterns and automatic adaptation of the partitioning strategy can be made in real-time. Machine learning can be pushed further into this proposed hybrid partitioning strategy. It can monitor itself and thereby manage partition reorganization depending on changing workloads without requiring human interaction, making the entire system much more efficient. Another potential extension in this regard would be the application of advanced analytics and machine learning tasks within an integrated hybrid partitioning strategy. Since organizations are becoming dependent on the warehouse for advanced applications such as ML models, complex analytics, and so on, another optimization improvement might likely arise in partitioning schemes for complex operations. The more promising areas then come under the future work: designing a better partitioning strategy to improve the execution of machine learning algorithms and analytics and the execution of those analyses. This work, therefore, calls for further rigorous tests on diversified datasets and in various industries besides different data types. Partitioning strategies for unstructured data types, such as text and multimedia, would help improve their use of these approaches. Lastly, efforts to reduce maintenance overheads on partitions would also be important. Highly dynamic environments in the cloud could make the cost of continually adapting schemes for partitioning higher than any advantage it could provide, even for simple queries or low data volumes. Developing approaches to reduce such costs further while providing performance improvements will be a massive contribution to the field of cloud data management.

Acknowledgment: We would like to express our sincere gratitude to the University of Central Missouri, VISA, and Home Depot Management LLC, all located in Atlanta, Georgia, United States of America, for their invaluable support and contributions. Their insights, resources, and encouragement have been crucial in completing this work. We are truly thankful for their continuous guidance and generous assistance throughout the research process.

Data Availability Statement: The data for this study can be made available upon request to the corresponding author.

Funding Statement: This manuscript and research paper were prepared without any financial support or funding.

Conflicts of Interest Statement: The authors have no conflicts of interest to declare. This work represents a new contribution by the authors, and all citations and references are appropriately included based on the information utilized.

Ethics and Consent Statement: This research adheres to ethical guidelines, obtaining informed consent from all participants.

References

1. H. Baalbaki, H. Hazimeh, H. Harb, and R. Angarita, "KEMA: Knowledge-graph embedding using modular arithmetic," in Proceedings of the 34th International Conference on Software Engineering and Knowledge Engineering, Pittsburgh, PA, United States of America, 2022.
2. H. Baalbaki, H. Hazimeh, H. Harb, and R. Angarita, "TransModE: Translation-al Knowledge Graph Embedding Using Modular Arithmetic," Procedia Comput. Sci, vol. 207, no.10, pp. 1154–1163, 2022.
3. G. Saad, H. Harb, A. Abouaiss, L. Idoumgha, and N. Charara, "An efficient Hadoop-based framework for data storage and fault recovering in large-scale multimedia sensor networks," in Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus; IEEE, pp. 316–321, New York, NY, United States of America, 2020.
4. D. Połap and M. Woźniak, "Red fox optimization algorithm," Expert Syst. Appl., vol. 166, no. 3, p. 114107, 2021.
5. A. Dahal and S. R. Joshi, "A clustering based vertical fragmentation and allocation of a distributed database," in 2019 Artificial Intelligence for Transforming Business and Society (AITB), Kathmandu, Nepal, 2019.

6. R. Ramachandran, G. Ravichandran, and A. Raveendran, “Vertical fragmentation of high-dimensional data using feature selection,” in Proceedings of the Inventive Computation and Information Technologies, pp. 935–944, Coimbatore, India, 2021.
7. A. A. Amer, “On K-means clustering-based approach for DDBSs design,” J. Big Data, vol. 7, no. 5, pp. 1-31, 2020.
8. A. E. Raouf, N. L. Badr, and M. F. Tolba, “Dynamic data reallocation and replication over a cloud environment,” Concurr. Comput. Pract. Exp, vol. 30, no. 13, pp. e4416, 2018.
9. M. Benmelouka, H. Alami, and M. Farid, “VFAR: Virtualized Fiber Access Networks for efficient bandwidth management,” Opt. Switch. Netw, vol. 44, no.1, pp. 100–111, 2023.
10. T. Dokeroglu, M. A. Bayir, and A. Cosar, “Robust heuristic algorithms for exploiting the common tasks of relational cloud database queries,” Appl. Soft Comput., vol. 30, no. 5, pp. 72–82, 2015.
11. V. N. Luong, V. S. Le, and V. B. Doan, “Fragmentation in distributed database design based on KR rough clustering technique,” in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 166–172, Springer International Publishing, Cham, Switzerland, 2018.
12. A. Tarun, R. S. Batth, and S. Kaur, “A novel fragmentation scheme for textual data using similarity-based threshold segmentation method in distributed network environment,” Int. J. Comput. Netw. Appl., vol. 7, no. 6, pp. 231-242, 2020.
13. A. E. Raouf, N. L. Badr, and M. F. Tolba, “Distributed database system (DSS) design over a cloud environment,” in Multimedia Forensics and Security: Foundations, Innovations, and Applications, pp. 97–116, Springer, New York, NY, United States of America, 2017.
14. W. Wedashwara, S. Mabu, M. Obayashi, and T. Kuremoto, “Combination of genetic network programming and knapsack problem to support record clustering on distributed databases,” Expert Syst. Appl., vol. 46, no. 3, pp. 15–23, 2016.
15. S. Benkrid, L. Bellatreche, Y. Mestoui, and C. Ordonez, “PROADAPT: Proactive framework for adaptive partitioning for big data warehouses,” Data Knowl. Eng., vol. 142, no. 11, p. 102102, 2022.