ESP Journal of Engineering & Technology Advancements (ESP-JETA)
ISSN: 2583-2646
Special Issue: International Conference on Cloud Security, Cyber governance and Global Impacts (ICCSCGI 2026)

Original Article

Optimizing Data Lakehouse Performance Using Adaptive Query Optimizers

Dr. Sandeep Nair[1], Dr. Ritu Malhotra[2]
[1]*Department of Information Technology, Advanced Computing Institute, India*
[2]*School of Cybersecurity and Digital Governance, National Technology University, India*

Abstract: *Data lakehouse architecture has emerged as a transformative solution in modern big data analytics by integrating the scalability and flexibility of data lakes with the reliability and performance capabilities of traditional data warehouses. Despite its advantages, data lakehouse environments face major performance challenges due to heterogeneous data formats, distributed storage systems, increasing query complexity, and dynamic workloads. Traditional static query optimization techniques are often insufficient in handling rapidly changing analytical demands, leading to increased query latency, inefficient resource utilization, and higher operational costs in cloud-based infrastructures. This research focuses on optimizing data lakehouse performance through the implementation of adaptive query optimizers capable of dynamically adjusting execution strategies during runtime.*

The proposed study introduces a Hybrid AI-Assisted Adaptive Query Optimization Framework designed to improve query execution efficiency in distributed lakehouse systems. The framework combines cost-based optimization techniques, runtime statistics collection, machine learning–driven execution planning, adaptive join reordering, and intelligent workload management to enhance system performance. By continuously monitoring workload patterns, system resources, and execution behaviors, the optimizer dynamically modifies query plans to achieve better scalability, lower latency, and improved throughput. The framework also integrates predictive analytics models that assist in selecting optimal execution paths based on historical query patterns and real-time system conditions. The research evaluates the effectiveness of the proposed framework using large-scale benchmark datasets and distributed analytics platforms such as Apache Spark, Databricks, and Trino. Performance metrics including query execution time, CPU utilization, memory efficiency, scalability, and resource consumption are analyzed to compare adaptive optimization approaches with conventional static optimization methods. Experimental findings demonstrate that adaptive query optimization significantly reduces execution delays, enhances workload balancing, and minimizes infrastructure costs in cloud-native data lakehouse environments.

Furthermore, this study highlights the importance of integrating artificial intelligence and autonomous optimization mechanisms into next-generation analytics systems. The proposed framework contributes to the advancement of intelligent big data processing by enabling real-time decision-making, automated resource allocation, and self-optimizing query execution strategies. The research concludes that adaptive query optimizers represent a critical technological advancement for improving the efficiency, scalability, and reliability of modern data lakehouse platforms, thereby supporting the growing demands of enterprise-scale analytics and real-time data processing applications.

Keywords: *Data Lakehouse, Adaptive Query Optimization, Big Data Analytics, Cost-Based Optimization, Machine Learning, Distributed Query Processing, Intelligent Workload Management, Cloud Computing, Apache Spark, Databricks, Trino, Runtime Query Execution, Query Performance Optimization, AI-Assisted Analytics, Real-Time Data Processing, Scalable Data Systems, Lakehouse Architecture, Distributed Computing, Resource Optimization, Autonomous Query Planning*

I. INTRODUCTION

The rapid growth of digital technologies, cloud computing, and large-scale enterprise applications has significantly increased the volume, variety, and velocity of data generated across industries. Organizations today rely heavily on advanced analytics systems to process massive datasets for decision-making, business intelligence, artificial intelligence, and real-time operational insights. Traditional data warehouses, while effective for structured analytical

processing, often struggle to handle unstructured and semi-structured data at scale. On the other hand, data lakes provide scalable and low-cost storage solutions but lack the transactional consistency, governance, and optimized query performance required for enterprise analytics. To address these limitations, the concept of the data lakehouse has emerged as a modern data management architecture that combines the flexibility of data lakes with the performance and reliability features of data warehouses.

Data lakehouse platforms enable organizations to store, process, and analyze diverse datasets within a unified environment. Technologies such as Apache Spark, Delta Lake, Apache Iceberg, and Databricks have contributed significantly to the adoption of lakehouse architectures in modern cloud-based ecosystems. These platforms support large-scale distributed processing, real-time analytics, machine learning workloads, and high-performance query execution. However, despite their advantages, data lakehouses face several critical performance challenges due to complex query workloads, heterogeneous data formats, distributed storage systems, and dynamic execution environments.

One of the major challenges in data lakehouse systems is inefficient query processing caused by static query optimization techniques. Traditional query optimizers generate execution plans before runtime based on estimated statistics and predefined cost models. Although these methods perform adequately in stable environments, they often fail to adapt to dynamic workload changes, unpredictable data distributions, and varying system conditions commonly found in distributed analytics platforms. As a result, organizations experience increased query latency, resource contention, excessive cloud infrastructure costs, and reduced system scalability. These limitations highlight the need for more intelligent and adaptive optimization mechanisms capable of improving query execution efficiency in real time.

Adaptive query optimization has emerged as a promising solution for overcoming the limitations of static optimization methods. Adaptive query optimizers dynamically adjust execution strategies during runtime by utilizing real-time system statistics, workload monitoring, and execution feedback. Advanced optimization techniques such as adaptive join reordering, runtime partition pruning, workload-aware scheduling, and machine learning–driven execution planning enable distributed systems to optimize resource allocation and improve query performance automatically. The integration of artificial intelligence into query optimization further enhances the ability of analytics platforms to predict workload behavior, select optimal execution plans, and reduce computational overhead.

This research focuses on optimizing data lakehouse performance using adaptive query optimizers within distributed cloud environments. The study proposes a Hybrid AI-Assisted Adaptive Query Optimization Framework designed to improve query execution efficiency, scalability, and resource utilization in modern lakehouse architectures. The framework incorporates runtime statistics collection, machine learning–based decision models, intelligent workload management, and dynamic execution plan adaptation to address the growing demands of enterprise-scale analytics systems. By evaluating the proposed framework using real-world benchmark datasets and distributed query engines, this research aims to demonstrate how adaptive optimization techniques can significantly enhance the performance, reliability, and cost-efficiency of data lakehouse platforms. Ultimately, the study contributes to the advancement of intelligent, self-optimizing big data analytics systems capable of supporting next-generation real-time data processing applications.

II. BACKGROUND OF DATA LAKEHOUSE ARCHITECTURE

The evolution of data management systems has been driven by the increasing demand for scalable, flexible, and high-performance analytics platforms capable of handling massive volumes of structured, semi-structured, and unstructured data. Traditional data warehouses were initially designed to support business intelligence and analytical reporting by storing structured data in a highly organized format. These systems provided strong transactional consistency, optimized query processing, and efficient reporting capabilities. However, traditional warehouses often required expensive infrastructure and struggled to process rapidly growing datasets generated from social media platforms, Internet of Things (IoT) devices, cloud applications, and real-time streaming systems.

To overcome these limitations, organizations adopted data lakes, which provided low-cost and highly scalable storage environments capable of storing raw data in multiple formats. Data lakes allowed enterprises to store structured, semi-structured, and unstructured data without predefined schemas, enabling greater flexibility for

advanced analytics and machine learning applications. Despite their scalability advantages, data lakes introduced several operational challenges, including poor data governance, inconsistent metadata management, lack of transactional reliability, and slow query performance. The absence of optimized query engines and efficient indexing mechanisms often resulted in significant performance degradation during large-scale analytical operations.

The emergence of the data lakehouse architecture represents a significant advancement in modern data engineering by combining the strengths of traditional data warehouses and data lakes into a unified platform. A data lakehouse integrates scalable storage capabilities with advanced data management features such as ACID transactions, schema enforcement, metadata governance, and optimized analytical query processing. This architecture enables organizations to manage diverse data workloads within a single system while supporting real-time analytics, machine learning, and business intelligence applications simultaneously.

Modern lakehouse platforms rely heavily on distributed computing technologies and cloud-native infrastructures to support large-scale analytics processing. Technologies such as Apache Spark provide distributed query execution and parallel data processing capabilities, while storage frameworks like Delta Lake and Apache Iceberg enhance reliability through transaction management, schema evolution, and metadata optimization. In addition, platforms such as Databricks and Snowflake have accelerated the adoption of lakehouse architectures by offering cloud-based analytics services with improved scalability and performance.

Although data lakehouses provide several advantages, they also face critical performance challenges associated with distributed storage systems, dynamic workloads, and heterogeneous data formats. Query execution in lakehouse environments often involves complex joins, large-scale aggregations, and multi-stage distributed processing operations that can lead to increased latency and inefficient resource utilization. Traditional static query optimization methods are insufficient in handling continuously changing data patterns and workload conditions. As a result, adaptive query optimization techniques have become increasingly important in improving execution efficiency within modern lakehouse systems.

The growing importance of data lakehouse architectures reflects the need for intelligent, scalable, and cost-efficient analytics solutions capable of supporting enterprise-scale data processing. By integrating adaptive optimization strategies and artificial intelligence–driven query planning mechanisms, organizations can significantly improve performance, reduce operational costs, and enable real-time analytical decision-making in distributed cloud environments.

III. PERFORMANCE CHALLENGES IN DISTRIBUTED DATA LAKEHOUSES

A. Increasing Query Execution Latency

Modern data lakehouse platforms process extremely large datasets distributed across multiple storage nodes and cloud environments. As the size and complexity of analytical workloads increase, query execution latency becomes a significant challenge. Complex SQL operations such as joins, aggregations, filtering, and nested queries require substantial computational power and data movement across distributed systems. Traditional query optimization techniques often fail to generate efficient execution plans for dynamic workloads, resulting in slow response times and reduced analytical efficiency. High query latency negatively impacts real-time business intelligence applications, machine learning pipelines, and operational analytics systems that depend on fast and accurate data processing.

B. Inefficient Resource Allocation Mechanisms

Distributed lakehouse environments rely heavily on cloud-based computational resources including CPU, memory, storage, and network bandwidth. Inefficient allocation of these resources can lead to poor system performance, increased infrastructure costs, and reduced scalability. Static query optimizers typically allocate resources based on predefined assumptions rather than real-time workload conditions. Consequently, some processing nodes become overloaded while others remain underutilized. This imbalance creates bottlenecks during query execution and limits the overall efficiency of distributed analytics systems. Intelligent workload-aware resource management has become essential for improving performance and minimizing operational expenses.

C. Managing Heterogeneous Data Formats

Data lakehouses store information in multiple formats such as Parquet, ORC, JSON, CSV, and Avro to support diverse analytical applications. Although this flexibility enhances scalability and interoperability, it also introduces

performance challenges related to schema inconsistencies, metadata management, and varying storage structures. Different file formats require different query execution strategies and access methods. Traditional optimization approaches often struggle to adapt execution plans efficiently for heterogeneous datasets, leading to slower processing speeds and inefficient storage access. Effective optimization mechanisms must dynamically analyze data structures and select the most suitable execution techniques for each workload.

D. Handling Dynamic Workload Variability

Modern enterprise analytics environments experience continuously changing workloads due to varying user demands, streaming data sources, and concurrent query execution. Workload variability significantly affects query performance because static optimization techniques cannot adapt quickly to sudden changes in system conditions. During peak workloads, processing engines may experience resource contention, increased scheduling delays, and reduced throughput. Adaptive query optimization techniques are necessary to monitor runtime statistics, predict workload patterns, and dynamically modify execution plans according to current system performance. Efficient handling of workload variability is critical for maintaining consistent performance in cloud-native lakehouse systems.

E. Optimizing Distributed Query Processing

Distributed query processing involves dividing analytical tasks across multiple computing nodes to improve scalability and parallel execution. However, distributed processing introduces challenges such as network communication overhead, data shuffling, synchronization delays, and partition imbalance. Queries that require large-scale data movement between nodes often experience performance degradation and increased execution time. Optimizing distributed query processing requires intelligent strategies such as adaptive join reordering, dynamic partition pruning, and workload-aware scheduling. These techniques help reduce unnecessary data transfer and improve execution efficiency across distributed infrastructures.

F. Enhancing Real-Time Analytics Performance

Organizations increasingly depend on real-time analytics for applications such as fraud detection, recommendation systems, predictive maintenance, and customer behavior analysis. Real-time analytical systems require low-latency query execution and rapid processing of streaming data. Traditional query optimizers are not designed to support continuously changing data streams and real-time decision-making requirements effectively. Adaptive query optimization frameworks that utilize machine learning models, runtime monitoring, and automated execution plan adaptation can significantly improve the responsiveness and scalability of real-time analytics platforms operating within data lakehouse environments.

IV. COMPARATIVE ANALYSIS OF EXISTING QUERY OPTIMIZATION SYSTEMS

Modern data lakehouse platforms use different query optimization techniques to improve analytical performance, scalability, and resource efficiency. Several distributed query engines and cloud-native analytics platforms have introduced adaptive optimization capabilities to address the limitations of traditional static query processing methods. These systems differ in architecture, execution strategies, workload management techniques, and support for real-time optimization. Understanding the strengths and limitations of existing optimization systems is essential for designing more intelligent and adaptive query optimization frameworks for modern lakehouse environments.

Traditional query optimization approaches primarily rely on rule-based and cost-based optimization models. Rule-based optimizers generate execution plans using predefined optimization rules, while cost-based optimizers estimate execution costs based on statistics such as data size, cardinality, and available resources. Although these methods improve performance in stable environments, they are less effective in dynamic distributed systems where workloads continuously change during runtime. To address these challenges, modern platforms integrate adaptive query execution, runtime statistics collection, and machine learning–based optimization techniques.

Apache Spark introduced Adaptive Query Execution (AQE), which dynamically modifies query plans during execution based on runtime statistics. AQE improves performance by optimizing join strategies, reducing shuffle operations, and dynamically adjusting partition sizes. This adaptive capability significantly improves distributed query processing efficiency in large-scale analytics workloads.

Databricks enhances query optimization through its Photon execution engine and intelligent workload management features. The platform combines vectorized execution, predictive caching, and cloud-native resource scaling to improve analytics performance. Databricks also integrates machine learning support for workload prediction and automated optimization in enterprise-scale environments.

Snowflake uses an automatic optimization architecture that separates storage and compute resources to improve scalability and concurrency. Snowflake automatically manages indexing, partitioning, and query optimization without requiring extensive manual configuration. Its cloud-native architecture supports elastic scaling and efficient resource allocation for complex analytical workloads.

Trino and Presto focus on high-performance distributed SQL query execution across multiple data sources. These systems optimize federated query processing and support interactive analytics for heterogeneous datasets. However, they may experience limitations in adaptive runtime optimization compared to AI-assisted query optimization frameworks.

The following table presents a comparative analysis of major query optimization systems used in modern data lakehouse environments.

Platform / System	Optimization Technique	Key Features	Advantages	Limitations
Apache Spark AQE	Adaptive Query Execution	Runtime plan optimization, dynamic partitioning, adaptive joins	Improved distributed processing efficiency	High memory consumption
Databricks Photon	AI-Assisted Optimization	Predictive caching, vectorized execution, workload management	High-speed analytics performance	Platform dependency
Snowflake	Automatic Cost-Based Optimization	Elastic scaling, automated indexing, cloud-native optimization	Strong scalability and concurrency	Higher operational cost
Trino	Distributed SQL Optimization	Federated query processing, interactive analytics	Multi-source query support	Limited adaptive optimization
Presto	Parallel Distributed Processing	High-speed SQL execution across clusters	Low-latency query execution	Resource management complexity
Traditional RDBMS Optimizers	Static Cost-Based Optimization	Predefined execution plans	Stable performance in static workloads	Poor adaptability to dynamic environments

A. Adaptive Query Optimization Techniques

Adaptive query optimization techniques are designed to improve query execution performance in distributed data lakehouse environments by dynamically modifying execution strategies during runtime. Unlike traditional static optimization methods, adaptive optimization continuously monitors system conditions, workload behavior, and execution statistics to generate more efficient query plans. These techniques play a critical role in reducing query latency, improving scalability, optimizing resource utilization, and supporting real-time analytics applications in cloud-native data platforms.

B. Runtime Statistics Collection

Runtime statistics collection is one of the fundamental components of adaptive query optimization. Traditional query optimizers rely on precomputed metadata and estimated statistics, which may become outdated in dynamic environments. Adaptive systems continuously gather real-time information such as data distribution, partition sizes, CPU utilization, memory consumption, and network traffic during query execution. These runtime statistics allow the system to detect performance bottlenecks and modify execution plans dynamically. Accurate runtime monitoring

significantly improves decision-making processes in distributed analytics systems and enhances overall query execution efficiency.

C. Dynamic Performance Monitoring

Dynamic performance monitoring enables query engines to track execution behavior continuously throughout the processing lifecycle. Monitoring mechanisms analyze system-level metrics including resource usage, task completion time, data shuffling operations, and node performance. By identifying overloaded nodes and inefficient execution stages, the optimizer can redistribute workloads and improve processing balance across distributed infrastructures.

D. Execution Feedback Mechanisms

Execution feedback mechanisms allow adaptive optimizers to learn from previous query executions. Historical performance data helps the optimizer identify recurring workload patterns and improve future execution strategies. Feedback-driven optimization enhances prediction accuracy and contributes to more intelligent resource management in cloud-based analytics environments.

E. Adaptive Join Optimization

Join operations are among the most resource-intensive tasks in distributed query processing. Adaptive join optimization techniques dynamically select the most efficient join strategy during runtime based on current workload conditions and data characteristics. Instead of relying on static assumptions, adaptive systems evaluate factors such as dataset size, memory availability, and network bandwidth before selecting execution methods.

F. Dynamic Join Reordering

Dynamic join reordering rearranges the sequence of join operations during query execution to minimize computational overhead and data transfer costs. By optimizing join order based on runtime statistics, the system can reduce intermediate dataset sizes and improve execution speed.

G. Broadcast Join Selection

Broadcast join selection is used when one dataset is significantly smaller than another. The optimizer broadcasts the smaller dataset across all processing nodes to minimize network communication and improve parallel execution performance. Adaptive selection mechanisms determine whether broadcast joins are beneficial under current workload conditions.

H. Machine Learning–Driven Query Optimization

Machine learning has become increasingly important in modern adaptive query optimization systems. AI-based optimization models analyze historical workloads, execution patterns, and system behavior to predict optimal execution strategies automatically. Machine learning techniques improve optimization accuracy and support autonomous decision-making in large-scale distributed environments.

V. PROPOSED HYBRID ADAPTIVE OPTIMIZATION FRAMEWORK

The proposed Hybrid AI-Assisted Adaptive Optimization Framework is designed to improve query execution performance, scalability, and resource efficiency in modern data lakehouse environments. The framework combines adaptive query optimization techniques, machine learning algorithms, runtime statistics collection, and intelligent workload management to overcome the limitations of traditional static optimization systems. The primary objective of the framework is to dynamically optimize query execution plans during runtime based on changing workload conditions, resource availability, and data distribution patterns.

Modern data lakehouse platforms process large-scale analytical workloads across distributed cloud infrastructures. Traditional query optimizers generate execution plans before runtime using predefined cost models and estimated metadata. However, these static optimization techniques often fail to adapt effectively to dynamic system conditions, leading to high query latency, inefficient resource allocation, and poor scalability. The proposed hybrid framework addresses these issues by continuously monitoring execution behavior and automatically modifying query strategies in real time.

The architecture of the proposed framework consists of several intelligent components working together to improve distributed query processing. The Query Analyzer module parses incoming SQL queries and identifies

workload complexity, data access patterns, and execution requirements. The Runtime Statistics Collector continuously gathers system metrics such as CPU utilization, memory usage, network traffic, and partition distribution during execution. This real-time information is then processed by the AI Optimization Engine, which uses machine learning models to predict optimal execution strategies and resource allocation decisions.

Another critical component of the framework is the Adaptive Execution Manager, which dynamically adjusts execution plans during runtime. The manager performs adaptive join reordering, partition pruning, workload balancing, and execution plan switching to improve query performance under varying workload conditions. In addition, the Intelligent Resource Scheduler allocates cloud resources dynamically according to workload demand and system availability, reducing infrastructure costs while maintaining scalability.

The framework also integrates query caching and materialized view optimization mechanisms to reduce repetitive computation and improve response time for frequently executed analytical queries. By combining AI-driven prediction models with adaptive runtime optimization techniques, the proposed framework enhances both operational efficiency and analytical performance in distributed data lakehouse systems.

The following table summarizes the major components of the proposed framework and their functions.

Framework Component	Function	Key Features	Performance Benefits
Query Analyzer	Analyzes incoming queries and workload complexity	SQL parsing, workload classification	Improved execution planning
Runtime Statistics Collector	Collects real-time execution metrics	CPU monitoring, memory tracking, partition analysis	Better runtime decision-making
AI Optimization Engine	Predicts optimal query execution strategies	Machine learning models, workload prediction	Reduced query latency
Adaptive Execution Manager	Dynamically modifies execution plans	Join reordering, partition pruning, plan switching	Improved processing efficiency
Intelligent Resource Scheduler	Allocates computational resources dynamically	Workload-aware scheduling, auto-scaling	Enhanced scalability and cost efficiency
Query Caching Module	Stores frequently accessed query results	Intelligent caching, result reuse	Faster query response time
Materialized View Manager	Maintains precomputed analytical results	Dynamic materialized views	Reduced repetitive computation
Distributed Monitoring System	Tracks system-wide performance	Node monitoring, execution feedback	Improved system reliability

VI. EXPERIMENTAL METHODOLOGY

The experimental methodology of this research is designed to evaluate the effectiveness of the proposed Hybrid AI-Assisted Adaptive Optimization Framework in improving the performance of distributed data lakehouse systems. The evaluation focuses on measuring query execution efficiency, scalability, resource utilization, and workload adaptability under different analytical conditions. A systematic experimental setup is essential to compare the proposed adaptive optimization framework with traditional static query optimization techniques and existing adaptive systems.

The experimental environment is developed using a distributed cloud-based infrastructure to simulate real-world enterprise analytics workloads. The system architecture consists of multiple computational nodes connected through a high-speed distributed processing network. Each node is configured with multi-core processors, high-capacity memory, and scalable storage resources to support large-scale data processing operations. Cloud-native deployment technologies such as containerized clusters and distributed orchestration platforms are utilized to improve scalability and workload management efficiency. The experimental platform integrates distributed analytics technologies including Apache Spark, Kubernetes, and Docker to support adaptive query execution and cloud resource scaling.

To ensure accurate performance evaluation, the experiments use benchmark datasets commonly employed in big data analytics research. Large-scale structured and semi-structured datasets are generated to simulate enterprise data lakehouse environments containing transactional records, streaming analytics data, and machine learning workloads. Benchmarking frameworks such as TPC-DS and synthetic workload generators are used to create complex analytical queries involving joins, aggregations, filtering operations, and nested subqueries. These datasets help evaluate the framework's ability to process heterogeneous data formats efficiently under varying workload conditions.

The performance evaluation process focuses on several critical metrics. Query execution time is measured to determine how quickly the system processes analytical workloads under adaptive optimization conditions. Throughput analysis evaluates the number of queries processed successfully within a given period, reflecting the scalability and concurrency capabilities of the framework. Resource utilization metrics including CPU consumption, memory usage, network bandwidth, and storage access efficiency are analyzed to measure workload balancing and infrastructure optimization. In addition, latency reduction and system response time are examined to assess the framework's effectiveness in supporting real-time analytics applications.

The experimental procedure involves executing identical query workloads across multiple optimization environments, including traditional static optimization systems, existing adaptive query optimizers, and the proposed AI-assisted adaptive framework. Comparative analysis is performed by recording execution statistics under different workload intensities and cluster configurations. Runtime monitoring tools continuously collect system-level performance data during execution to identify bottlenecks, workload distribution patterns, and resource allocation behavior.

Furthermore, machine learning models integrated within the proposed framework are evaluated based on prediction accuracy, adaptive decision-making efficiency, and execution plan optimization performance. The experiments analyze how effectively the AI-based optimizer predicts workload patterns and dynamically adjusts execution strategies in response to changing system conditions. This evaluation helps determine the framework's capability to support autonomous query optimization in large-scale distributed environments.

The overall experimental methodology provides a comprehensive evaluation of the proposed framework's ability to improve data lakehouse performance. By analyzing execution efficiency, scalability, resource optimization, and workload adaptability, the research demonstrates the practical advantages of integrating adaptive query optimization and artificial intelligence into modern distributed analytics systems.

VII. RESULTS AND PERFORMANCE ANALYSIS

The experimental evaluation of the proposed Hybrid AI-Assisted Adaptive Optimization Framework demonstrates significant improvements in query execution efficiency, scalability, and resource utilization within distributed data lakehouse environments. The results obtained from the experimental setup indicate that adaptive query optimization techniques combined with machine learning–driven decision-making provide substantial advantages over traditional static optimization methods. The analysis focuses on performance metrics such as query execution time, throughput, system scalability, latency reduction, and infrastructure resource consumption.

One of the most significant improvements observed during the experiments is the reduction in query execution latency. The proposed framework dynamically adjusted execution plans during runtime based on workload conditions and real-time system statistics. Compared to traditional static query optimizers, the adaptive framework achieved faster processing speeds for complex analytical workloads involving joins, aggregations, and large-scale distributed operations. Runtime optimization techniques such as adaptive join reordering, dynamic partition pruning, and workload-aware scheduling contributed significantly to reducing unnecessary data movement and computational overhead.

The experiments also demonstrated improvements in throughput and concurrent query processing. Under high workload conditions, traditional optimization systems experienced resource contention, increased scheduling delays, and reduced execution efficiency. In contrast, the proposed adaptive framework distributed workloads more effectively across computational nodes using intelligent resource scheduling mechanisms. This balanced resource allocation improved parallel processing efficiency and increased the number of queries processed successfully within a

given time period. The results confirm that adaptive optimization techniques are highly effective in supporting enterprise-scale analytics workloads with multiple concurrent users.

Resource utilization analysis further highlights the effectiveness of the proposed framework. The integration of runtime monitoring and machine learning–based optimization enabled more efficient allocation of CPU, memory, storage, and network resources. The framework continuously analyzed workload behavior and adjusted computational resource distribution dynamically according to processing demands. As a result, the system reduced unnecessary resource consumption and improved overall infrastructure efficiency. Cloud auto-scaling mechanisms also minimized operational costs by allocating additional resources only when required during peak workloads.

Another important outcome of the study is the improvement in scalability across distributed environments. The proposed framework maintained stable performance as dataset size and workload complexity increased. Traditional optimization systems often experienced performance degradation when processing large-scale distributed queries due to inefficient execution planning and network communication overhead. However, the adaptive framework successfully optimized distributed query execution by reducing shuffle operations, balancing partition distribution, and improving node-level workload coordination. These capabilities enabled the system to process large datasets efficiently without significant increases in execution latency.

The machine learning components integrated into the framework also produced positive results. Predictive workload analysis and intelligent execution planning improved optimization accuracy and reduced the need for manual performance tuning. The AI optimization engine successfully identified workload patterns and selected appropriate execution strategies based on historical execution data and runtime statistics. This autonomous decision-making capability improved system responsiveness and enabled continuous optimization in dynamic cloud environments.

Comparative analysis between the proposed framework and existing optimization systems demonstrated that AI-assisted adaptive optimization provides superior performance for distributed data lakehouse applications. While systems such as Apache Spark and Trino offer adaptive processing capabilities, the proposed framework achieved better workload adaptability and resource efficiency through deeper integration of machine learning and runtime intelligence.

the experimental results confirm that adaptive query optimization significantly enhances the performance, scalability, and reliability of modern data lakehouse architectures. The proposed Hybrid AI-Assisted Adaptive Optimization Framework successfully addresses major challenges associated with distributed analytics systems, including high query latency, inefficient resource utilization, and workload variability. These findings demonstrate the potential of intelligent adaptive optimization techniques in supporting next-generation enterprise analytics and real-time data processing applications.

VIII. CONCLUSION

The rapid growth of big data analytics, cloud computing, and distributed processing technologies has significantly increased the importance of efficient and scalable data management systems. Data lakehouse architecture has emerged as a powerful solution that combines the flexibility of data lakes with the structured analytical capabilities of traditional data warehouses. Despite its advantages, modern data lakehouse environments continue to face major performance challenges related to query latency, resource utilization, workload variability, and distributed processing complexity. Traditional static query optimization techniques are no longer sufficient for handling dynamic enterprise analytics workloads and real-time data processing requirements.

This research explored the role of adaptive query optimization in improving the performance of distributed data lakehouse systems. The study analyzed existing optimization approaches, identified key performance bottlenecks, and examined the limitations of traditional cost-based and rule-based optimization techniques. To address these challenges, a Hybrid AI-Assisted Adaptive Optimization Framework was proposed that integrates runtime statistics collection, machine learning–based decision-making, adaptive execution planning, intelligent workload management, and cloud auto-scaling mechanisms. The proposed framework was designed to dynamically modify query execution strategies during runtime according to changing system conditions and workload patterns.

The experimental evaluation demonstrated that adaptive query optimization significantly improves query execution efficiency, scalability, and resource management in distributed analytics environments. Runtime optimization techniques such as adaptive join reordering, partition pruning, workload-aware scheduling, and intelligent resource allocation successfully reduced query execution latency and minimized infrastructure overhead. The integration of machine learning algorithms further enhanced the system's ability to predict workload behavior, automate optimization decisions, and continuously improve execution performance through execution feedback mechanisms.

The results of the study confirmed that the proposed framework outperformed traditional static optimization systems in several critical performance areas, including throughput, concurrent query processing, resource utilization, and distributed scalability. The framework also demonstrated strong adaptability in handling heterogeneous data formats, large-scale datasets, and dynamic cloud workloads commonly found in enterprise lakehouse environments. By combining artificial intelligence with adaptive query execution techniques, the proposed system provided a more intelligent, autonomous, and cost-efficient approach to distributed analytics optimization.

Furthermore, the research highlighted the growing importance of AI-driven optimization strategies in next-generation data processing systems. As organizations increasingly rely on real-time analytics, machine learning applications, and cloud-native infrastructures, adaptive optimization mechanisms will become essential for maintaining high-performance analytics environments. The findings of this study contribute to the advancement of intelligent data lakehouse technologies by providing a scalable framework capable of supporting modern enterprise data processing requirements.

adaptive query optimization represents a critical advancement in modern distributed analytics systems. The proposed Hybrid AI-Assisted Adaptive Optimization Framework offers an effective solution for improving the performance, scalability, and reliability of data lakehouse platforms. Future research can further enhance this framework by integrating deep learning techniques, autonomous workload prediction models, real-time streaming optimization, and energy-efficient resource management approaches to support the evolving demands of cloud-based big data analytics.

IX. REFERENCES

[1] Armbrust, M., Ghodsi, A., Xin, R., Zaharia, M., et al. "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics." *CIDR Conference Proceedings*, 2021.
[2] Zaharia, M., Xin, R. S., Wendell, P., Das, T., et al. "Apache Spark: A Unified Engine for Big Data Processing." *Communications of the ACM*, vol. 59, no. 11, 2016, pp. 56–65.
[3] Chaudhuri, S. "An Overview of Query Optimization in Relational Systems." *Proceedings of the ACM Symposium on Principles of Database Systems*, 1998.
[4] Graefe, G. "Query Evaluation Techniques for Large Databases." *ACM Computing Surveys*, vol. 25, no. 2, 1993, pp. 73–169.
[5] Neumann, T. "Efficiently Compiling Efficient Query Plans for Modern Hardware." *Proceedings of the VLDB Endowment*, vol. 4, no. 9, 2011, pp. 539–550.
[6] Abadi, D. J., Boncz, P., and Harizopoulos, S. "Column-Oriented Database Systems." *Proceedings of the VLDB Endowment*, vol. 2, no. 2, 2009, pp. 1664–1665.
[7] Stonebraker, M., Abadi, D., DeWitt, D., et al. "MapReduce and Parallel DBMSs: Friends or Foes?" *Communications of the ACM*, vol. 53, no. 1, 2010, pp. 64–71.
[8] Dean, J., and Ghemawat, S. "MapReduce: Simplified Data Processing on Large Clusters." *Communications of the ACM*, vol. 51, no. 1, 2008, pp. 107–113.
[9] Verbitski, A., Gupta, A., Saha, D., et al. "Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases." *Proceedings of the ACM SIGMOD Conference*, 2017.
[10] Curino, C., Jones, E. P. C., Madden, S., and Balakrishnan, H. "Workload-Aware Database Monitoring and Consolidation." *Proceedings of the ACM SIGMOD Conference*, 2011.
[11] Ding, B., and König, A. C. "Runtime Adaptive Query Optimization for Dynamic Workloads." *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 11, 2016.
[12] Li, Y., Patel, J., and Chandler, J. "Adaptive Optimization Techniques in Distributed Data Systems." *IEEE Big Data Conference Proceedings*, 2019.
[13] Pavlo, A., Paulson, E., Rasin, A., et al. "A Comparison of Approaches to Large-Scale Data Analysis." *Proceedings of the ACM SIGMOD Conference*, 2009.
[14] Vohra, D. *Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools.* Apress Publishing, 2016.

[15] Kleppmann, M. *Designing Data-Intensive Applications*. O'Reilly Media, 2017.
[16] Karau, H., Warren, R. *High Performance Spark: Best Practices for Scaling and Optimizing Apache Spark*. O'Reilly Media, 2017.
[17] Akidau, T., Bradshaw, R., Chambers, C., et al. *Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing*. O'Reilly Media, 2018.
[18] Apache Spark Documentation. "Adaptive Query Execution in Spark SQL." Apache Software Foundation, 2024.
[19] Delta Lake Documentation. "Delta Lake Transaction Log Protocol." Linux Foundation Projects, 2024.
[20] Apache Iceberg Documentation. "Apache Iceberg Table Format Specification." Apache Software Foundation, 2024.
[21] Databricks Whitepaper. "The Data Lakehouse Paradigm." Databricks Inc., 2022.
[22] Snowflake Technical Documentation. "Snowflake Query Processing and Optimization." Snowflake Inc., 2023.
[23] Trino Documentation. "Distributed SQL Query Optimization." Trino Software Foundation, 2024.
[24] Presto Documentation. "Presto Query Optimizer Architecture." Presto Foundation, 2023.
[25] Elmore, A. J., Das, S., Agrawal, D., and El Abbadi, A. "Zeppelin: Flexible and Scalable Query Processing for Cloud Data Analytics." *IEEE Cloud Computing Journal*, vol. 5, no. 4, 2018.
[26] Shvachko, K., Kuang, H., Radia, S., and Chansler, R. "The Hadoop Distributed File System." *IEEE Symposium on Mass Storage Systems and Technologies*, 2010.
[27] Chen, T., and Guestrin, C. "XGBoost: A Scalable Tree Boosting System." *Proceedings of the ACM SIGKDD Conference*, 2016.
[28] Sutton, R. S., and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 2018.
[29] Kumar, A., and Patel, M. "Machine Learning–Driven Query Optimization in Cloud Data Warehouses." *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 2, 2022.
[30] Singh, R., and Sharma, P. "Adaptive Resource Scheduling for Distributed Big Data Analytics." *Journal of Cloud Computing and Distributed Systems*, vol. 11, no. 3, 2023.