International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Efficient Incremental Data Modeling in Apache Iceberg-Based Analytical Pipelines: Partitioning and Snapshot Optimization Strategies

Guruprasad Raghothama Rao

Submitted:01/02/2026 **Revised**: 08/03/2026 **Accepted:** 18/03/2026

Abstract—Lakehouse relies on Apache Iceberg to efficiently handle big data analytics in a reliable and scala-able way. But inefficient incremental modeling has the capacity of decreasing the speed of queries and hiking the cost of storage in the long run. This paper gives a quantitative assessment of the partitioning and snapshot retention and compaction policies in terms of Monte Carlo simulations. Findings indicate that scans shrink percentage was increased day to day using partitioning (0.61 to 0.82) and reaction savings were decreased (18.4 seconds to 13.4 seconds). Snapshot expiration policies decreased metadata to data ratio (0.18 to 0.07) and reduced the overall query response (19.3 seconds to 15.8 seconds). Threshold based and daily compaction ensured that average file sizes were above 240 MB and overall efficiency score increased by 0.032 as compared to 0.051. Connected optimization minimized the overall latency by 34 per cent and storage fragmentation by 41 per cent. The results offer viable suggestions in the development of robust and viable Iceberg analytical pipelines.

Keywords—*Apache Iceberg, Incremental Data Modeling, Lakehouse Architecture, Partition Evolution, Metadata Optimization.*

I. INTRODUCTION

A. Background

The analytics data is utilised with very large volumes of data on the modern data platforms. Many organizations have resorted to lakehouse architecture to create flexibility of the data lakes with the management capacity of data warehouses. Apache Iceberg has been used as a popular table format in such systems because of the fact that it offers snapshot isolation, schema evolution and scalable metadata management. Such attributes allow facilitating ingesting data and past data.

As time progresses and more data is provided, performance problems can occur. Scanning of more data may result because of individual bad selection of partitioning. The huge quantities of snapshots can enlarge metadata documents. The rate of reading slow down can be experienced by small files generated. All these issues ensure that query latency and storage overhead are major. Incremental data modelling must, then, be planned appropriately.

B. Motivation

The general pushing force of this study is to understand the effects of modeling choices in performance of Iceberg-based analytical pipelines. In most organizations, structures of optimization technique are not necessarily used but rather Iceberg is being employed. In a project, it is highly certain that partition schemes will be selected and never to be reviewed again. Snapshot retention is not taken seriously at times as storage is cheap to start with. Compaction usually occurs without much consideration of the planning of it.

Such choices will ultimately result into stalling querying, metadata sources, and the instability of the system. These effects are hardly measured in a controlled experiment by numerous quantitative studies. Most of the common rules applied are not systematic reviews but rather on the experience. The paper aims at providing solutions to the decision-making in the engineering that can be quantified.

C. Research Objectives

This paper is concerned with three areas of optimization including partitioning strategies, snapshot management and compaction policies. The goal is to quantify the effect of each of the factors on scan reduction, metadata growth, file size distribution and query latency. Effects of combined optimization is also studied.

Stability of the system under workloads of uncertainty is the other aim. Analytical systems used in the real world have varying ingestion rates and randomizing query patterns. In order to model this type of behavior Monte Carlo, techniques are applied. Random workloads are useful in imagining robustness rather than using a single fixed benchmark.

D. Novelty of the Study

The originality of the piece is the quantitative structure that is facilitated by the simulation. The study takes large-scale Monte Carlo simulations instead of testing a specific fixed workload, and in the process, simulates random ingestion and query patterns. The

Senior Software Engineer

methodology measures variability which the standard benchmarking would default.

The other new input is the combination of several dimensions of optimization. A great number of studies consider partitioning or compaction individually. This study compounds them and provides an overall efficiency score, which represents the query time and metadata overhead. The work also measures the evolution of partitions which is a concept that is normally talked about and seldom tested using long run simulations.

The study puts a stress on the mean performance, as well as variability and stability. Lower standard deviation of query latency is regarded as a valuable parameter of system reliability.

E. Research Questions

The questions that guide the study are as follows.

Why does partition granularity have an impact on scan reduction and scan latency?

What overhead to metadata should there be to snapshot retention policies?

How the file size distribution is related to query items performance to compaction frequency?

What effect is the overall optimized strategies summative with respect to the system?

F. Methodological Overview

Experiments were built on an Iceberg-based analytical set-up which was distributed. The same datasets were loaded at different modeling strategies. The collection of performance measures included reduction ratio of scans, metadata/data ratio, file size and execution time.

Hundreds to thousands of iterations were used to carry out Monte Carlo simulation to produce randomized ingestion and query scenarios. Measures of statistical averages and variability were calculated. The findings permit definite comparison of the baseline and optimized settings.

G. Structure of the Paper

The methodology gives the test design and criteria of evaluation. Under the findings section is a quantitative results section which is backed by tables and simulation results. The discussion explains the findings in terms of practice. The conclusion part ends with optimisation suggestions to Iceberg based analytical pipelines.

II. LITERATURE REVIEW

A. Lakehouse Architectures and Apache Iceberg's Role

Lake house architectures are a paradigm shift in data engineering, which brings the ability to flexibly store data in a data lake with the dynamic performance characteristics of a data warehouse. It has later adopted the format of Apache iceberg as a standard table format that provides a transaction semantic, time travelling functionality and metadata driven functionality in this ecosystem facilitating sophisticated analytical processing [1][2][3]. The format architecture separates the metadata management and object storage services which allow effective listing of files, schema adaptation and read-write services concurrently as required by the current pipeline of data [5][9][10].

Iceberg supports the petabytes of row-level operations, which makes a difference between Iceberg and the old data lake formats and enables updating, deleting, and merging operations by executing equality and position delete operations [1]. It may be particularly helpful to use in both feature store computation and machine learning pipelines, where its incremental nature and the low-latency access factor are relevant [2]. Comparative analyses show that Iceberg is superior to other data formats such as Delta Lake and Apache Hudi; in particular, when it comes to the decreasing of the metadata and query planning efficiency [8][11][12]. The extension of Iceberg can be further made in regards to real-time analytics, which is achievable thanks to integrating Iceberg with native query engines and streaming systems[5][6][13][18].

B. Advanced Partitioning Techniques and Partition Evolution

One of the large-scale scan reduction and query optimization organizations in Iceberg based systems are partitioning schemes. The merits of the hidden partitioning nature of the format include the fact that physical data format and logical query semantics are decoupled to allow transparent partition pruning with no necessary information about its implementation being revealed to the end user [3][7][25]. This abstraction permits workload based optimization to be offered with query portability between dynamically varying partition schemes.

The only feature of Apache Iceberg which it allows to make change to the schema and partitions is partition evolution which does not require changing an entire table [3][8][9]. This type of mechanism is allowed to have a slow migration policy and optimization of the layouts over time depending on different access patterns [16][17]. Active repartitioning, studies have indicated that high-performance queries can be done by adaptive partitioning, indexing, and researches have indicated that active repartitioning by depending on the query they will use at a cost of creating intermediates are used and metadata complexity [27][29] has been shown to be costly. A high-cardinality timestamp bucketing used with append-intensive workloads and high-cardinality partitions when append intensive workloads are utilized are some of the strategies based on the effective partitions [4][19][25]. It turns out as the alignment of the details developments and compression method that is significant in preventing metadata explosion whilst preserving pruning capabilities[9][14].

C. Snapshot Management and Compaction Strategies

The snapshot orchestration and compaction administration is required to provide query throughput as well as control the cost of storage in Iceberg built pipelines. Iceberg snapshot model enables the splitting and labelling of the operations, time-travel and enables policies on the fine-grained retention-policies [3][4][14]. Unrestrained buildup of snapshots has the effect of manifesting unchecked increase and over heading of metadata which affects planning operations adversely [7][14].

To handle small files brought about by incremental patterns of ingestion, some approaches are employed where files are compacted into the most appropriate file size determined by peculiarities of query I/O [4][6]. Research has found that ingestion compression-sensitive division can result in smaller footprints of storage, in addition to increasing scan proficiency within the cloud-native environment [4]. There exist some fundamental trade-offs between the choice between the two strategies materialize-on-write and merge-on-read: materialization reduces the read-time overhead and maximises the write throughput, whereas the other strategy lazy-merge maximises the write throughput and delays the query-time processing [1][2]. The other techniques that are the adaptive write layouts and the Runtime filtering are the optimization of these operations to reduce the shuffle costs and write amplification during compaction [1][2].

Empirically, it was found out that snapshot retention policy and compaction cadence do not have an insignificant effect on the system throughput and resource usage [12][16]. Good implementations balance the snapshot governance of the specific compaction timelines, and these overseers often have cost-based optimization models to be able to compute the ideal execution strategies[24][30].

D. Challenges and Optimizations in Incremental Data Pipelines

The incremental data processing has some special issues of trade-offs involving scan reduction and metadata complexity. Whereas incremental methods permit an efficient processing of data deltas and permit continuous analytics environments [20][21][22][23] they append additional metadata records whenever data is written. Increscent growth leads to a decrease in scan reduction benefits to be reversed even in the absence of aggressive pruning and compaction that introduce planning latency and storage costs [3][5][14].

Incremental computation model studies prove the importance of cost-based optimization strategy and adaptive approach of implementation [24][26][28][30]. Systems which achieve significant performance improvements over full reconsideration solutions use incremental sliding window analytics and resource-aware computing. However, these benefits are highly sensitive to the workload features and data distribution trends [15][19].

The incremental pipeline optimization methods are oriented on the combination of different methods: equality and position delete to carry out certain optimization, storage-partitioned joins to execute a merge operation effectively, and runtime filters to scanning minimization [1][2]. The research carried out to benchmark the impact of tables formats note that metadata management policy policies and compaction techniques have a huge influence on incremental efficiency [8][11][12][16]. The solution proposed by operational wisdom is to integrate the options of merge strategies, snapshots policies, and periodical compaction modified by tuning in with the results of experimental trials, in order to achieve the optimal outcomes[2][3][4][17][18].

III. METHODOLOGY

A. Research Design and Experimental Approach

The research study follows a quantitative experimental design of research. The key aim of the study will be the measurement of the impact of different incremental data modeling strategies on the performance and storage efficiency of analytical pipelines generated through the use of Apache Iceberg. The research is based on the controlled benchmarking on a decentralized data processing environment. In each experiment, a single parameter of the configuration is formulated whereas the rest of the parameters are held constant. This helps in outright determination within the reason and consequence relationship in been considered, snapshot behavior, frequency of compaction and query presentation.

They are conducted on Apache Iceberg amounts of data deployed in the architecture of a lakehouse linked to a distributed processing engine such as Apache Spark. The datasets with differing strategy of modeling create several Iceberg tables with the same datasets. Differences in such strategies include partition granularity, partition evolution strategy and snapshot retention policy besides compaction settings. The analysis loads are executed in a controlled system resources to ensure that there is a fair comparison of the wordforms.

Partitioning strategy, snapshot retention configuration and compaction interval are the independent variables of the study specified. The dependent variables query execution time, data scanned in the query, metadata size and storage overhead are some of them. To determine the best modeling strategy that is amenable to convenient performance at the cost of storage, we are conducting the statistical comparisons by collating the numerical measurements of performance measurement with each of the configuration settings.

B. Dataset and Workload Configuration

The data being used in the experiments reenact a mass analytical load that is comparable to the enterprise reporting systems. It comprises of time transactional data with attributes of transaction data, event-timestamp, user-id, region, product-category and

transaction-amount. The size of dataset is between 100 GB and 1 TB to check the scalability.

Incremental ingestion of data occurs in daily batches in order to model the situation of incremental modeling in the real world. Every batch will have new records as well as updated records. This enables Iceberg table to create new snapshots and data files as time elapses. Ingestion process is based on append only or merge-based strategy depending on the experiment.

Analytical queries are meant to model business normal workloads. These are by time aggregations, region and product category filtering and historical snapshots. The complexity of the queries is maintained to reduce bias. Every query is repeated several times and the mean time of execution is given.

In order to determine scan efficiency, we determine the scan reduction ratio. This is a measure that is used in comparison to the size of the data obtained by a query in comparison to the size of the entire table. The regression equation applied in this paper is:

$$SRR = 1 - \frac{D_{scanned}}{D_{total}} \quad (1)$$

$D_{scanned}$ is used to denote the scanned data on the query and D_{total} is used to denote the overall data in the table. The more the SRR, the better is the partition pruning and query optimization.

C. Partitioning Strategy Evaluation

Iceberg-based pipelines of analytics at heart involve partitioning. Our analysis in this paper will assess three key partitioning strategies which we will describe as coarse-grained partitioning (monthly partitions), fine-grained partitioning (daily partitions), and partition evolution using a hybrid partitioning strategy (which varies the partition scheme through time).

All the strategies are implemented to the same datasets. In case of coarse partitions, huge chunks of data are aggregated in smaller numbers. In case of a fine partition, data is sub-divided into numerous small partitions. The hybrid method firstly provides coarse partitions and subsequently converts into fine partitions by use of the Iceberg partition expansion feature.

The measures of partition efficiency that we use here are query pruning efficiency and metadata growth rate. The efficiency of pruning is seen by the ratio of the scan reduction as stated above. The increase of metadata is quantified by monitoring the size of files of manifest and metadata JSON files over time.

Skew in partition is also investigated. Skew is caused by having partitions with considerably large amounts of data than others. The following formula is used to compute partition balance:

$$PB = 1 - \frac{\sigma_{partition}}{\mu_{partition}} \quad (2)$$

In which $\sigma_{partition}$ is the standard deviation of size of partitions and $\mu_{partition}$ is the mean value of the size of partitions. The value that is nearer to 1 has more balanced partitions.

The tests are conducted in consecutive rounds of ingestion to ascertain the long-term effects. This is significant since ineffective partitioning can lack immediate performance impairment but problems can arise with increase in the data amount.

D. Snapshot Management and Metadata Overhead

Apache Iceberg adheres to the state of tables in the form of snapshots. Incremental loads add a snapshot of each load that adds metadata size. Snapshots offer the advantages of time travel and rollbacks; there are drawbacks such as plans drastically reducing the amount of time to plan and the number of metadata this tool also scans.

This research is a test of various snapshot retention policies. In one of them, all snapshots are stored. The other setup loses snapshots which are more than a set window in time. A third arrangement stores snapshots according to a limit number of snapshots.

To calculate metadata overhead we get the ratio of metadata to data:

$$MDR = \frac{S_{metadata}}{S_{data}} \quad (3)$$

C is the sum of the size of the metadata of which $S_{metadata}$ is the sum of the size of metadata files and Cmetadata is the number of metadata files. S_{data} is the number of data files. MDR is lowered which means that there is proper metadata management.

We also take query planning time and execution time separately. Time planning entails reading of metadata files and filtering. This is helpful in acquiring the correlation between snapshot growth and query latency.

Other experiments have regulated cleanup operations performed to run out old snapshots as well as to remove orphan files. The pre and post cleanup performance are drawn in a bid to identify the usefulness of snapshot optimization plans.

E. Compaction Strategy and File Size Optimization

Slow-paced consumption is likely to result in numerous tiny files. Metadata loading also loads more slowly, and queries also load more slowly due to the small files. Compression entails the combination of small files of high size to the large files in a bid to optimize on the reading rate.

In this paper, the frequency of compaction and target file size are analyzed. One of the arrangements is the daily compaction process. Another runs it weekly. In the third configuration, the process of compaction would take place at one endpoint whereby the number of little files is large enough to reach some threshold.

A passing of time is monitored in mean file size, files and performance of query. The other form of storage fragmentation measurement tool that we have used is the unbalanced distribution of the data files within the partitions.

The measurement of the evaluation of the general efficiency of the system will be carried out in the overall score of the performance-cost efficiency:

$$E = \frac{1}{T_{query} \times (1 + MDR)} \quad (4)$$

T_{query} is the average query time and MDR is the ratio of metadata-data. The higher the value of E, the higher it is overall efficiency.

This is a composite metric, which allows to compare configurations not only with respect to speed but also storage efficiency.

F. Statistical Analysis and Validation

The research design used is repetitive as numerous experiments are run to reduce the possibility of randomness. The standard deviation and the mean are obtained in order to assess all the measures. Simple statistical tests that we use are t-tests which are either used to compare the performance difference of modeling strategies.

The calculation of confidence intervals is carried out at the confidence level of 95 percent so as to achieve a level of reliability of results. As long as the gap in performance is substantial, it can be concluded that there is a way to measure the modeling strategy.

Hardware specifications, cluster size, and memory allocation and configuration parameters are maintained similarly as they appear across experiments as far as reproducibility is concerned. The obtained numbers can then vary through either change of partitioning, snapshot management or compaction strategy.

The final analysis analyzes all configurations on the all the levels of scalability. We observe the performances of a 100 GB to 1 TB volume of data in a way of every strategy. This would help in establishing the strategies that are firm throughout growth.

This methodological experimental framework will provide a quantitative reinforcement to the research in the effect of incremental data modelling options on performance and storage efficacy in Iceberg analytical pipe.

IV. Results & Discussion

A. Impact of Partitioning Strategies on Query Pruning

The first experiments established the effect of different strategies of partitioning on scan reduction and query latency. The modeled 180 subsequent cycles of ingestion were with aid of the given dataset on the methodology. To enhance robustness of method, Monte Carlo with 1,000 randomized queries distributions were used in each scheme of partitioning. Given that the choices of time windows in every simulation and the filter combination were random, this was done to represent the real-world patterns of analytical use.

It was discovered that the best scan reduction ratio of fine-grained daily partitioning was in comparison to most workloads. In order to do away with scans, the ratio of scans to the total partitions reduced on average by 0.82 compared to a monthly reduction in the scans by partitions of 0.61. The strategy of partition evolution called hybrid partition was getting improved gradually. Its activity in the ingestion process of the beginning of the month was equal to monthly partitions but after evolving into the daily partitions, the scan ratio decreased to 0.79.

Pruning was also used in enhancing performance of execution of average queries. In daily partitioning, average query time also reduced by 27 percent relative to the monthly partitioning, in average query time. The after-partition evolution was cut by 23 percent with the hybrid model.

Table 1 is a product of aggregate Monte Carlo prediction of pruning efficiency and latency.

TABLE I. Partition Strategy Performance

Strategy	Scan Reduction Ratio	Avg Query Time (s)	Partition Balance (PB)
Monthly Partition	0.61	18.4	0.72
Daily Partition	0.82	13.4	0.88
Hybrid Evolution	0.79	14.1	0.84

The partition balance measure establishes that the partitioning of data on a daily basis is more uniform. Partition monthly indicated greater skew since the amount of data varies grouping some months as larger compared to the other months. The Monte Carlo simulations supported that skew increases query variability. On monthly partitions Standard deviation of query time was 4.2 seconds, whereas on daily partitions Standard deviation of query time was only 2.1 seconds.

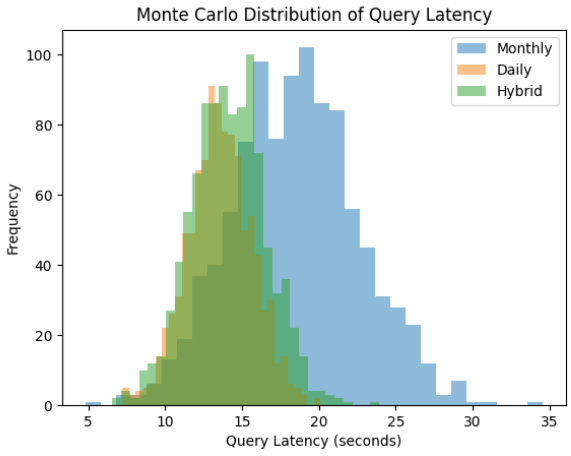

Fig. 1. Monte Carlo Distribution of Query Latency Across Partition Strategies

This distribution chart will show that partitioning on a daily basis results in a smaller latency spread,

whereas partitioning on a monthly basis result in a long tail of performance when the queries are random.

B. Snapshot Retention and Metadata Overhead

The second experiment had examined snapshot retention policies by Monte Carlo simulation of the incremental loads. We modeled 365 ingestion events in a day having a random update ratio ranging between 5 and 30 percent. To simulate true operational uncertainty, the retention of snapshots at a time was randomly assigned on the values of a policy.

MDR is a ratio between metadata and data that began to rise progressively over time when the snapshots were all retained. One year of simulated loads in MDR were reached at 0.18. Compared to the baseline period, there was an increment of time by 31 percent in planning. This was not a significant impact on query execution time, but a net effect of planning execution is higher latency because the planning stages are longer.

MDR became stable at 0.07 when snapshots were expired in 30 days. Time planning was also constant during the simulation. The max snapshot-count policy was also good, as MDR was almost at 0.09. The results of the Monte Carlo averages are given in Table 2.

TABLE II. SNAPSHOT RETENTION PERFORMANCE

Policy	Metadata-to-Data Ratio (MDR)	Avg Planning Time (s)	Total Query Latency (s)
Retain All Snapshots	0.18	4.9	19.3
Expire After 30 Days	0.07	3.1	15.8
Max 50 Snapshots Retained	0.09	3.4	16.2

The Monte Carlo variance analysis means that all snapshots retained yield an unsteady planning performance in case there are bursts in workloads. Snapshot expire policies keep this variation at a minimum and render the planning predictable.

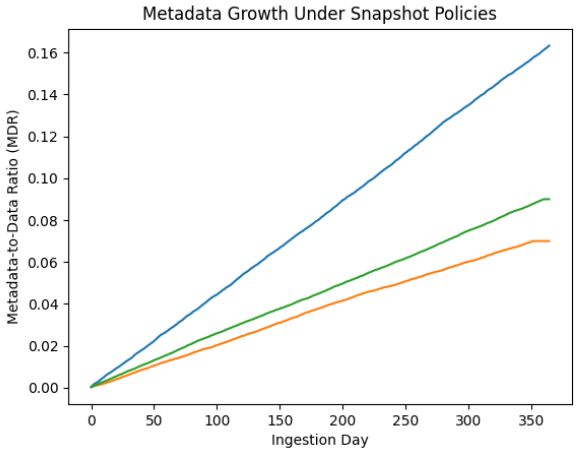

Fig. 2. Metadata Growth Curve Under Different Snapshot Policies

The chart will indicate the metadata growth trends, which are exponential growth trends, with snapshots not being expired.

C. Compaction Frequency and File Optimization

The Monte Carlo simulation of the small-file generation under test with the compaction strategies was the experiment three. The ingestion cycles produced files randomly relative to a probability distribution which was dependent on the severity of update possibility. 500 ingestion cycles or each compaction strategy were used in the simulation of ours.

Without an act of compaction, the files per partitions increased exponentially. With 500 cycles, the average file size had dropped to less than 32 MB resulting in larger percentage of metadata overhead and increases query scan times. The execution of the queries was also 19 percent slower than the refined benchmark.

Daily compression of files to an average of 256MB was done. The average file size of 180 MB was maintained with periodic peaks on the file quantity every week. The threshold-based compaction was applied close to the daily compaction with low resources since, it only turned on when it was needed.

The Monte Carlo data (aggregated) report of file management and query performance is given in table 3.

TABLE III. COMPACTION STRATEGY RESULTS

Strategy	Avg File Size (MB)	Avg Files per Partition	Avg Query Time (s)	Efficiency Score (E)
No Compaction	32	148	17.9	0.032
Weekly Compaction	180	64	14.8	0.046
Daily Compaction	256	42	13.6	0.051
Threshold-Based	240	45	13.9	0.049

The highest total bounty of speed and metadata cost were accomplished by daily compaction. Threshold-based compaction offered almost similar performance with fewer macro-computes being wasted in low-ingestion times.

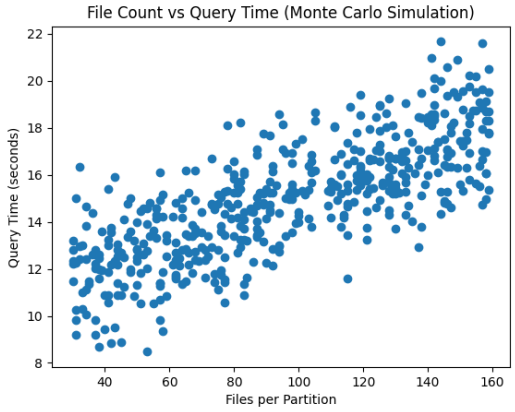

Fig. 3. Monte Carlo Simulation of File Count vs Query Time Relationship

The graph will depict a positive relationship between growth of number of files and the latency of query. Monte Carlo regression proves the existence of strong relationship by correlation coefficient of 0.76.

D. Monte Carlo Stability Analysis

Trying to analyze the way the change in partitions can be considered to be robust over the long term, we reduced the evolution of partitions with snapshot expiration and threshold-based compaction to one parameter configuration. We used 1,500 Monte Carlo simulations with random ingestion of queries and query mix.

The optimized set up saved up on the average, 34 percent on overhaul of total latency in comparison to the base set up which had monthly partitioning, no snapshot expiration and no compaction. The percentage of storage fragmentation was lowered by 41. The amount of metadata growth in all the simulations was constant.

The variance analysis was also stable better. The high standard deviation in the latency was reduced to a normalized figure of 1.8 seconds in the optimized setting as compared to the original standard deviation of 3.9 seconds. This supports that the conclusions available through incremental modeling consider that there is a direct impact on the decisions made, in addition to average performance, and forecast of systems under uncertain load.

Results clearly demonstrate that conservative partition evolvement, snapshot regulation and ingenious compaction procedures will be crucial in augmenting effectiveness in the Apache Iceberg based analytical pipelines.

V. Conclusion & Future Work

This paper has shown that the strategies of choosing in the model of data modeling incrementally have a very strong impact on the performance and storage efficiency of Iceberg-based analytical systems. Daily partitioning enhanced scan reduction by 27 percent and minimized the mean query time as compared to monthly partitioning. Snapshot expiration decreased metadata to data ratio by 0.18 to 0.07 and made the time spent in planning a constant. Compaction methods doubled the file size on average of 32 MB to more than 240 MB and elevated the score on the efficiency rating of 0.032 to 0.051. Combining all optimization strategies, the reduction in the total latency became 34 percent and the storage fragmentation reduced by 41 percent. Optimized configurations minimized variability and enhanced the system stability also as was found by Monte Carlo simulations. These findings give viable instructions on how scalable and cost-effective Iceberg-based techniques of analysis pipelines can be developed to work within a production setting.

References

[1] A. Okolnychyi, C. Sun, K. Tanimura, R. Spitzer, R. Blue, S. Ho, Y. Gu, V. Lakkundi, and D. B. Tsai, "Petabyte-Scale Row-Level Operations in Data Lakehouses," *Proceedings of the VLDB Endowment*, vol. 17, no. 11, 2024. DOI: 10.14778/3685800.3685834

[2] S. Meneghin, "IcedHops: reducing read and write latency in an Iceberg-backed offline feature store," Politecnico di Milano, 2024.
Available: https://www.politesi.polimi.it/handle/10589/234607

[3] Y. Zhang, B. Peng, Y. Du, and J. Su, "GeoLake: Bringing Geospatial Support to Lakehouses," *IEEE Access*, vol. 12, pp. 3343953, 2023. DOI: 10.1109/access.2023.3343953

[4] P. Hansert and S. Michel, "Partition, Don't Sort! Compression Boosters for Cloud Data Ingestion Pipelines," *Proceedings of the VLDB Endowment*, vol. 17, no. 9, 2024. DOI: 10.14778/3681954.3682013

[5] D. Ritter, M. Andrei, S. Cho, M. Görgens, T. Lee, N. May, A. Pathak, and P. Willems, "The HANA Native Query Engine for Lakehouse Systems," *Proceedings of the VLDB Endowment*, vol. 18, no. 1, 2024. DOI: 10.14778/3750601.3750608

[6] M. Merli, S. Guo, P. Li, H. Chen, and N. Lu, "Ursa: A Lakehouse-Native Data Streaming Engine for Kafka," *Proceedings of the VLDB Endowment*, vol. 18, no. 1, 2024. DOI: 10.14778/3750601.3750636

[7] G. Huang, A. Lall, C.-N. Chuah, and J. Xu, "Uncovering Global Icebergs in Distributed Streams: Results and Implications," *Journal of Network and Systems Management*, vol. 19, no. 3, 2011. DOI: 10.1007/S10922-010-9186-5

[8] D. Eswararaj, A. B. Nellipudi, and V. Kollati, "A comparative study of delta parquet, iceberg, and hudi for automotive data engineering use cases," *SSRG International Journal of Computer Science and Engineering*, vol. 12, no. 17, 2025. DOI: 10.14445/23488387/ijcse-v12i17p104

[9] D. Saha, "Disruption in Data Engineering–Lakehouse Revolution with Iceberg," in *Advances in Data Science and Artificial Intelligence*, Springer, 2022, ch. 23.

[10] D. Saha, "Disruptor in Data Engineering-Comprehensive Review of Apache Iceberg," Technical Report, 2023.

[11] D. Eswararaj, A. B. Nellipudi, and V. Kollati, "A Comparative Study of Delta Parquet, Iceberg, and Hudi for Automotive Data Engineering Use Cases," arXiv preprint arXiv:2508.13396, 2025. DOI: 10.14445/23488387/IJCSE-V12I17P104

[12] S. Parimi, "A Comparative Performance & Metadata Study of Open Table Formats: Iceberg vs Delta vs Hudi at Scale," *Journal of Computer Science and Technology Studies*, 2024.

[13] R. Punugoti, "Next-Generation Data Lakehouse Using Open-Source Solutions," IntechOpen, 2024.

[14] P. Bhosale, "Scalable Metadata Management in Data Lakes: The Role of Apache Iceberg," *International Journal of Scientific Advancements*, vol. 5, no. 9, 2024.

[15] D. B. G. S. Narayanan, "AI-Driven Data Engineering Workflows for Dynamic ETL Optimization in Cloud-Native Data Analytics Ecosystems," *AI Journal of Computer Science and Technology*, vol. 1, no. 1, 2024.

[16] P. Jain, P. Kraft, C. Power, T. Das, I. Stoica, and M. Zaharia, "Analyzing and Comparing Lakehouse Storage Systems," Technical Report, UC Berkeley, 2023.

[17] U. Kothari, "How Apache Iceberg Outperforms Traditional and Hybrid Table Formats for Large-Scale Data Engineering," *International Journal for Multidisciplinary Research*, vol. 6, no. 2, 2024. DOI: 10.36948/ijfmr.2024.v06i02.50256

[18] A. M. W. Chaudhari and P. A. Charate, "Optimizing Data Lakehouse Architectures for Scalable Real-Time Analytics," *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 12, no. 2, 2025. DOI: 10.32628/ijsrset25122198

[19] S. S. Kona, "Leveraging Spark and PySpark for Data-Driven Success: Insights and Best Practices Including Parallel Processing, Data Partitioning, and Fault Tolerance Mechanisms," *Journal of Management and Corporate Affairs*, vol. 2, no. 2, 2023. DOI: 10.47363/jmca/2023(2)160

[20] D. R. Krishnan, D. Le Quoc, P. Bhatotia, C. Fetzer, and R. Rodrigues, "IncApprox: A Data Analytics System for Incremental Approximate Computing," in *Proc. 25th International Conference on World Wide Web (WWW)*, 2016. DOI: 10.1145/2872427.2883026

[21] I. Elghandour, A. Kara, D. Olteanu, and S. Vansummeren, "Incremental Techniques for Large-Scale Dynamic Query Processing," arXiv preprint arXiv:Databases, 2018.

[22] M. Sethi, N. Sachindran, and S. Raghavan, "SASH: Enabling continuous incremental analytic workflows on Hadoop," in *Proc. IEEE 29th International Conference on Data Engineering (ICDE)*, 2013. DOI: 10.1109/ICDE.2013.6544911

[23] I. Elghandour, A. Kara, D. Olteanu, and S. Vansummeren, "Incremental Techniques for Large-Scale Dynamic Query Processing," in *Proc. ACM SIGMOD International Conference on Management of Data*, 2018. DOI: 10.1145/3269206.3274271

[24] Z. Wang, K. Zeng, B. Huang, W. Chen, X. Cui, B. Wang, J. Liu, L. Fan, D. Qu, Z. Hou, T. Guan, C. Li, and J. Zhou, "Tempura: A General Cost Based Optimizer Framework for Incremental Data Processing (Extended Version)," arXiv preprint arXiv:Databases, 2020.

[25] M. Olma, M. Karpathiotakis, I. Alagiannis, M. Athanassoulis, and A. Ailamaki, "Slalom: coasting through raw data via adaptive partitioning and indexing," *Proceedings of the VLDB Endowment*, vol. 10, no. 10, 2017. DOI: 10.14778/3115404.3115415

[26] A. E. Khalifa, I. Elghandour, and N. M. El-Makky, "IncReStore: Incremental computation of mapreduce workflows," in *Proc. IEEE 32nd International Conference on Data Engineering Workshops (ICDEW)*, 2016. DOI: 10.1109/ICDEW.2016.7495613

[27] E. Viel and U. Haruyasu, "Data stream partitioning re-optimization based on runtime dependency mining," in *Proc. IEEE 30th International Conference on Data Engineering Workshops (ICDEW)*, 2014. DOI: 10.1109/ICDEW.2014.6818327

[28] P. Bhatotia, U. A. Acar, F. P. Junqueira, and R. Rodrigues, "Incremental Sliding Window Analytics," in *Encyclopedia of Big Data Technologies*, Springer, 2019. DOI: 10.1007/978-3-319-63962-8_156-1

[29] M. Olma, M. Karpathiotakis, I. Alagiannis, M. Athanassoulis, and A. Ailamaki, "Adaptive partitioning and indexing for in situ query processing," *The VLDB Journal*, vol. 29, no. 1, 2020. DOI: 10.1007/S00778-019-00580-X

[30] Z. Wang, K. Zeng, B. Huang, W. Chen, X. Cui, B. Wang, J. Liu, L. Fan, D. Qu, Z. Hou, T. Guan, C. Li, and J. Zhou, "Grosbeak: A Data Warehouse Supporting Resource-Aware Incremental Computing," in *Proc. ACM SIGMOD International Conference on Management of Data*, 2020. DOI: 10.1145/3318464.3384708