International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 2, March –April 2023||

DOI:10.15662/IJRPETM.2023.0602002

Operationalizing Lakehouse Table Formats: A Comparative Study of Iceberg, Delta, and Hudi Workloads

Janardhan Reddy Kasireddy

Lead Data Engineer, Info Drive Systems (Finra Contractor), USA

ABSTRACT: Several open table formats, such as Apache Iceberg, Delta Lake, and Apache Hudi, have been adopted rapidly with the data lakehouse architecture and each boasts of being ACID compliant and providing sophisticated transactional support. Nonetheless, companies tend to choose those formats depending on the hype in the market instead of their applicability to workload which results in poor operational effectiveness and higher costs. This paper aims to provide a comparison of how lakehouse table formats can be operationalized focusing on engineering aspects and not popularity.

Our table formats decision matrix weighs the different dimensions through which table formats are considered, which are Change Data Capture (CDC), upserts and deletes, time travel, compaction strategies, and streaming ingestion. In order to prove the structure, we run a standardized load with hybrid data based on the TPC-DS benchmark data at 1 TB size of structured transactional work and a synthetic e-commerce streaming data of 10,000 events per second with frequent inserts, updates, and deletions. Workloads are run in Iceberg, Delta, and Hudi and measures are derived in terms of operation load, performance, and overall cost.

Findings show that there is a large difference in performance and maintenance overheads depending on the table format and workload characteristics. Iceberg shows better performance in time-travel queries and advanced streaming situations, Delta has more efficient compaction and less complicated CDC and Hudi is better at incremental updates of moderate operational complexity. The decision matrix enables engineers to make format choice in accordance to workload patterns minimizing speculation and enhancing performance.

The work offers practical advice to the engineers and architects working on lakehouses, which offers a data-driven method to choosing table formats, balancing performance, operation effort, and expense, facilitating the take-up of transactional data lakes within the enterprise settings.

KEYWORDS: Table format interoperability, Open table formats, Data lakehouse table formats, Apache, Iceberg, Delta Lake, Apache.

I. INTRODUCTION

The accelerated trend in the development of data-driven decision-making has radically altered the way companies shape and run analytical information platforms. Although traditional data warehouses provide high transactional guarantees, organized query performance, they frequently do not scale, flex, or become economical with the large volumes of heterogeneous information. Conversely, data lakes offer cheap storage and flexibility of the schema but have traditionally lacked transactional consistency, consistent updates and strong governance. To close this divide, the data lakehouse architecture has come into the picture as a work of bringing together the openness and scalability of data lakes and the reliability and performance features of data warehouses. The lakehouse concept heavily relies on the open table formats which bring the transactional semantics, metadata management, and query optimization to the object storage [1].

Some of the open table formats most popularly used include Apache Iceberg, Apache Hudi and Delta Lake [2]. All of these formats purport to support ACID transactions, scalable metadata processing, schema evolution, time travel and support both batch and streaming workloads effectively. Consequently, businesses in the finance, e-commerce, healthcare, and life sciences industries are quickly embracing them to operationalize analytical and near-real time loads

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 2, March –April 2023||

DOI:10.15662/IJRPETM.2023.0602002

on clouds based data lakes. Nonetheless, Iceberg, Delta as well as Hudi are quite different in terms of internal structure, design ideologies, as well as operational overtrades regardless of their common objectives [3].

In real life, the choice of table even used may be based on popularity of an ecosystem, vendor compatibility or even on anecdotal performance assertion instead of a careful study of the workload properties. Organizations often take a format, as it comes along with a desirable processing engine, as suggested by a cloud vendor, or as it seems to be a standard in an industry. These design choices may result in a poor design choice such as excessive maintenance overhead, inefficient query support, and high storage and compute costs, and operation complexity that negates the desired benefits of the lakehouse approach. With lakehouse deployments evolving past the experimental analytics state to production systems that are mission imperative, the engineering-based, workload sensitive format choice is becoming more significant [4]

The process of operationalizing a lakehouse table format is much more than merely storing data in a transactional format. These operational issues have a direct effect on system reliability, cost-efficiency, and productivity within the developer. This means that a table format that is optimized to handle append-only analytical workloads is not necessarily the right choice when it comes to high-frequency update and real-time streaming workloads, even with claims of complete ACID compliance [5].

The research fills a major knowledge gap in the current literature and the industry guidelines as it deals not with the potential of lakehouse table formats or their market penetration but with its operationalization. This work has given comparisons of Iceberg, Delta Lake, and Hudi with no evaluations done in isolation or under idealized settings but rather under the same, realistic workload that is a combination of structured transactional information and high-velocity streams of data. It has focused on the engineering features, such as the operational load, performance character, and economic considerations, as opposed to popularity or ecosystem momentum.

In order to organize the comparison, the paper will present a decision table format that will compare each of the formats in major operational areas: Change Data Capture, support of upserts and deletes, time travel, compaction strategies, and streaming ingestion. The choice of these dimensions lies in the fact that they are the most frequently experienced pain points by data engineers when operating production lakehouse workloads. The decision matrix allows viewing in a holistic way how each format fits a particular workload pattern by comparing those aspects.

To be able to conduct empirical assessment, a standardized hybrid workload is used in the study. The TPC-DS benchmark is used to create structured transactional data at a scale of 1 TB, which denotes batch-oriented analytical queries having complex joins and aggregations. Simultaneously, a synthetic e-commerce streaming workload develops 10,000 events per second with frequent updates, deletions, and inserts that attempt to separately represent data flows like operational data flows of the real world. Such a mixture is indicative of the state of the contemporary data platforms as the historical analytics and real-time data processing have to co-exist on the same storage and metadata scheme.

Apache Iceberg, Delta Lake, and Apache Hudi are used separately to implement each workload with the underlying infrastructure, data volume, and processing logic remaining the same. This type of methodology will make sure that any observed differences in the results are due to the table forms and not on environmental or configuring biases.
The results show that, despite the fact that all three formats meet the minimum specifications of transactional data lakes, their performance and characteristics of operation differ considerably, based on their workload patterns. The metadata design and snapshot isolation model of iceberg have proven to be successful in the time-travel queries and advanced streaming scenarios. Delta Lake is very efficient in compaction and eases of CDC operations leading to reduced operational complexities of some batched-stream hybrid workloads. Apache Hudi demonstrates strength in processing on incremental and upsert-heavy workloads, especially in case moderate update frequency and record-level changes are the key aspects. These differences demonstrate the fact that it is essential to make the choice of a table format in accordance with the real requirements of the workload and not according to the generalized demands.
Offering a comparative and data-driven assessment, this work offers practical recommendations to data engineers and architects that need to design and run lakehouse platforms. The proposed decision matrix is a repeatable framework which can be modified to fit the organization own environment allowing the informed choice of the format to be used which will balance the performance, effort in operations and cost. Finally, the paper contributes to the wider use of transactional data lakes by lessening speculation, elucidating trade-offs, and enhancing engineering quality in federating table format choices in lakes.

II. RELATED WORK

The development of data lake and lakehouse architecture is the result of the need to unify object storage scalability and cost efficiency with the reliability, governance, and performance attributes that were the traditional attributes of data warehouses. Data lake systems Early data lake systems were mostly based on raw data ingestion and schema on read, with limited transactional guarantees and robust metadata management, resulting in the complexity of operation and poor data quality. The associated work below presents the conceptual, architectural, engineering underpinnings that prompt the motivation of the necessity of operationalized lakehouse table formats and systematic workload-based assessment.

Delta Lake is one of the first practical solutions proposed by Armbrust et al. as a way of implementing ACID transactions, schema enforcement, and time travel to object store-based data lakes [1]. Demonstrated in their work, reliable transactional semantics were possible by adhering to a transaction log and taking advantage of optimistic concurrency control without necessarily forsaking open file formats such as Parquet. The paper was pioneering in demonstrating that data lakes may be used to help drive both scale-based batch analytics and scale-based incremental updates. But as Delta Lake determined the possibility of ACID data lakes, it gave minimal thought to the architecture and more so the performance attributes, leaving the question of comparability in the operational trade-offs across table formats unanswered.

Inmon had proposed the conceptual foundations of data lakes in his classic book on data lake architecture [2]. Inmon has focused on the fact that data lakes are not to be turned into data swamps and that metadata, governance and disciplined ingestion processes are important. In spite of being pre-modern in its table format, the work has formulated fundamental principles, including separation of storage and compute, retention of raw data, and enterprise governance, directly affecting the design objectives of Iceberg, Delta Lake, and Hudi. The work of Inmon, however, is more abstract, and it does not discuss the transactional loads or streaming ingestion in any specifics.

These ideas were further developed by Laurent, Laurent, and Madera who presented a detailed discussion of the design of data lakes, their governance, and lifecycle management [3]. Their book discussed the metadata catalogs, data ingestion pipelines, and analytical workloads in a systematic way and gave a systematic perspective of the data lake ecosystems. Although this work recognised the developing trends in the direction of unifying analytics and operational workloads, it lacks a thorough discussion of the effect of various table formats on the performance, compaction mechanisms or change data capture when applying under real-world workloads.

Ravat and Zhao have offered a general overview of data lake trends and views with the main challenges being found in the heterogeneity of data, its management, and the optimization of its performance [4]. Their work placed data lakes as dynamic systems, which are becoming more like data warehouses. The research remained however on a high level and did not involve any empirical assessments and workload-based comparisons.

Giebler et al. examined the practical issues that can be experienced by organizations when exploiting data lakes, such as metadata management, performance of queries, and complexity of operations [5]. In their work, the disconnect between theory data lake structure on paper and actual production did deployments was emphasized. Notably, they stressed that tooling and architectural choices have a great impact on maintainability and performance, which is in line with the incentive of considering table formats in terms of their operational properties and not their popularity.

A mapping study performed by Couto et al. aimed at narrowing down the definition of data lakes and categorizing architectural patterns that are employed in practice [6]. They were able to formulate recurring themes by synthesizing the literature that is present, which included scalability, schema flexibility, and integration with analytics platforms. Even though this research made the standardization of terminology and architectural knowledge, it left how contemporary table formats execute such architectures to transactional and streaming workloads.

Ivanov and Pergolesi studied the performance of SQL-on-Hadoop under columnar file formats at the lower part of the storage stack [7]. Their results showed that storage arrangement and file format option have a significant effect on query latency and resource consumption. It is specifically applicable to the lakehouse table format when Iceberg and Delta Lake are mentioned, as all three are based on a columnar format like Parquet. Nevertheless, the research concentrated on file formats but not on upper-level table abstractions and transaction management.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 2, March –April 2023||

DOI:10.15662/IJRPETM.2023.0602002

Software engineering wise, Davoudian and Liu have given an in-depth survey of big data systems, and they have highlighted the issues of complexity, maintainability and systems evolution [8]. Their discussion positioned big data platforms as grandiose software systems where design trade-offs are to be considered carefully. This view supports the relevance of appraisal of table formats not just in performance measurements, but also in terms of overhead operationality and engineering energy.

Herden suggested architectural patterns to combine data lakes with the use of conventional data warehouses with a focus on hybrid architectures between flexibility and reliability [9]. This article is directly connected to the lakehouse concepts in that the authors demonstrate how data lakes can be improved to facilitate enterprise analytics. But it did not make comparisons of particular table forms or count cost of operation at various loads.

Mehmood et al. have provided an implementation-based research on creating data lakes with heterogeneous data sources [10]. Their work made it prominent with regards to ingestion difficulties, schema development, and unified access layers. Although the study was practical, it has failed to cover the advanced transactions capabilities like the upserts, deletes or time travels.

Lastly, Armbrust et al. described the vision of bringing together data warehouses and data lakes into one architecture, which was introduced as a formal paradigm of the lakehouse [11]. This book brought together much of the previous conceptualization such as ACID transactions, open formats and scaleable storage into a unified architecture. Although it was powerful, the lakehouse paper concentrated on conceptual integration as opposed to operational comparisons that are empirical in nature between rival table formats.

To conclude, there is a solid theoretical and architectural basis on the nature of data lakes and lakehouses, the need of transactional guarantees and performance optimization, which has been established in existing literature. This gap encourages the current research that operationalizes Iceberg, Delta Lake, and Hudi with the same workloads to give practical advice to the practitioners.

1.Framework for Operationalizing Lakehouse Table Formats

This section gives the suggested outline of how to give operationalization of the lakehouse table formats, which will be used to test systematically Apache Iceberg, Delta Lake, and Apache Hui when subjected to realistic production loads. The framework focuses on practicality and workload equivalence in the engineering instead of theoretical feature equivalence. It organizes the assessment based on the decision matrix, standardized workloads, execution architecture, and quantifiable work metrics, and it is possible to conduct a fair and repeatable comparison of table formats in an enterprise lakehouse infrastructure.

1.1 Framework Design Objectives

The main goal of the framework is to help data engineers and architects to pick a suitable open table format in accordance with workload characteristics and operational needs. The design is based on the following principles of the framework:

- **Workload realism**: The assessment should be based on workloads in the real world that involve the combination of batch analytics and high velocity streaming data.
- **Operation focus**: It is focused on maintenance overhead, compaction behaviour, metadata growth and ingestion complexity.
- **Format neutrality**: Formats are tested on the same infrastructure, data volume and logical.
- **Measurable results**: The results are measured in terms of performance, operational burden and cost.
- **Reproducibility**: The framework can be reused or expanded by the practitioners to carry out new evaluations.

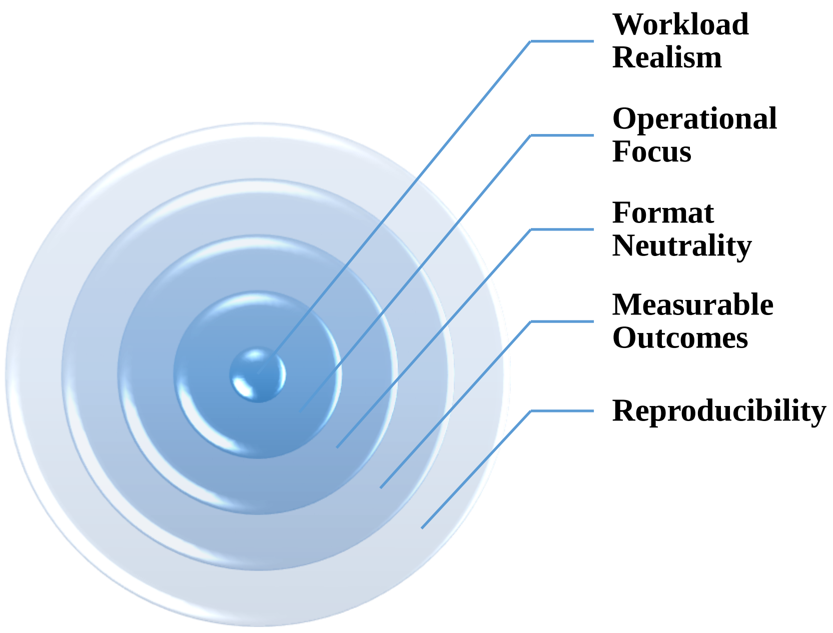

Figure 1: Primary Objectives of Framework for operationalization of the lakehouse table formats

These goals make sure that the framework is not limited to feature checklists and describes how table formats act when scaled out.

2. Decision Matrix Dimensions
At the core of the framework is a **decision matrix** that evaluates each table format across five critical operational dimensions. These dimensions represent the most common challenges faced when running production lakehouse systems.

2.1 Change Data Capture (CDC)
One essential ability required to maintain the synchronization of the lakehouse analytical tables with the upstream transactional systems which keep producing constant inserts, updates, and deletes is Change Data Capture (CDC). The presented framework compares the CDS support across table formats on the basis of ease of deployment CDS pipelines, support of native row level tracking of change, metadata wastefulness stemming out of frequent updates, as well as the support of current streaming engines. Delta Lake has change data feed functionality built-in, making the implementation of CDC much easier and the operation management of change data feed much simpler. Apache Hudi offers native query and pull mechanisms which are heavily incremental (better suited in CDC-intensive workloads). Apache Iceberg allows CDC to use snapshot-based comparisons, which are highly correct, but needs more orchestration and metadata management on a large scale.

2.2 Upserts and Deletes
The workloads of modern lakehouse are increasingly getting pegged on the capability to effectively deal with upserts and deletes and not just add only. The framework is used to analyze table formats using write amplification during upserts, performance degradation due to deletes, file rewriting policies, and scalability when the rate of updates is high. This is done with Apache Hudi which is studied by copy-on write and merge-on read operation that provide varying trade-offs between write latency and read performance. Delta Lake handles the deletes and upserts by rewriting files which is coordinated by a transaction log. There is also position and equality deletes in Apache Iceberg, which enable fine grained updates, at the cost of additional overhead in metadata and planning in large scale.

2.3 Time Travel and Versioning
The time travel is an essential characteristic of auditing, debugging, and guaranteeing the reproducibility of lakehouse systems. The framework measures time travel by the following parameters: historic query latency, snapshot management effectiveness, configuration complexity of retention and versioning storage overhead. Apache Iceberg is designed as a snapshot store, which enables time-travel queries to be efficiently supported and the cost of query planning is low. Delta Lake aids time travel by maintaining a time travel log which offers stable historical appearance. Apache Hudi supports commit based versioning with flexible retention retention policies giving the flexibility to control historical data retention.

2.4 Compaction and File Management
Lakehouses based on object storage can have performance that is severely impacted by small file issues and it can be very costly. The framework characterizes table formats in terms of the complexity of compaction strategy, frequency and cost of compaction jobs, effects on part of concurrent reads and writes and any available automation mechanisms. Delta Lake possesses excellent automated compaction characteristics particularly when used together with structured streaming pipelines. Apache Hudi requires highly articulate compaction planning in part especially in Merge-on-Read mode. Apache Iceberg offers unrestricted options of compaction, but tuning and orchestration must be done manually and with great care in order to reach optimal performance.

2.5 Streaming Ingestion
Streaming ingestion is tested in the sustained and high throughput with high frequency data modification. The framework has metrics of ingestion latency, stability of a system under backpressure, integration with more popular streaming engines and fault tolerance with recovery behavior. Apache Iceberg enjoys the advantage of the snapshot isolation where there are consistent reads and writes to its streamsThe Delta Lake offers a tight integration with the Spark Structured streaming and it can ingest extremely well within low latency and great reliability. Apache Hudi is efficient in the work of incremental ingestion trend but also presents a new level of complexity in its operation when compaction operations need to be fine-tuned and continuous streaming inserts are required.

3. Workload Definition and Data Modeling
To be fair and realistic, its framework employs a hybrid workload model which consists of batch analytical data and real-time streaming events.

3.1 Batch Workload: TPC-DS
The batch workload will be based on the TPC-DS benchmark scale factor of 1 TB. It represents:

- Systematized transactional information.
- Multifaceted unions and aggregations
- Periodic batch updates

The schema is normalized and partitioned across all forms of tables. The same SQL queries are run to quantify the query latency and resource usage..

3.2 Streaming Workload: Synthetic E-Commerce Events
An artificial e-commerce load produces 10,000 events per second, which are new order inserts, order updates in the form of upserts and cancellations/returns in the form of deletes. Every event has a standardized schema and contains primary keys to allow mutated rows. This load is also known to strain streaming ingestion pipelines, change data capture processing, and compaction systems among various lakehouse table formats.

4. Execution Architecture
To prevent infrastructure-based bias in the evaluation, the evaluation framework is deployed on unified lakehouse execution architecture so that the various table formats can be fairly compared. The shared storage is object storage that is stored either in the cloud which can be in S3-compatible storage or in ADLS Gen2. Apache Spark is the compute engine of the workload of batches and streaming. Metadata management is dependent upon the native catalog implementation of its format. Orchestration is a mixture of batch jobs that are scheduled and continuous pipelines. Table formats are all configured based on best practices vendor-recommended and are all the same resource allocation, partitioning strategies, and target file sizes.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 2, March –April 2023||

DOI:10.15662/IJRPETM.2023.0602002

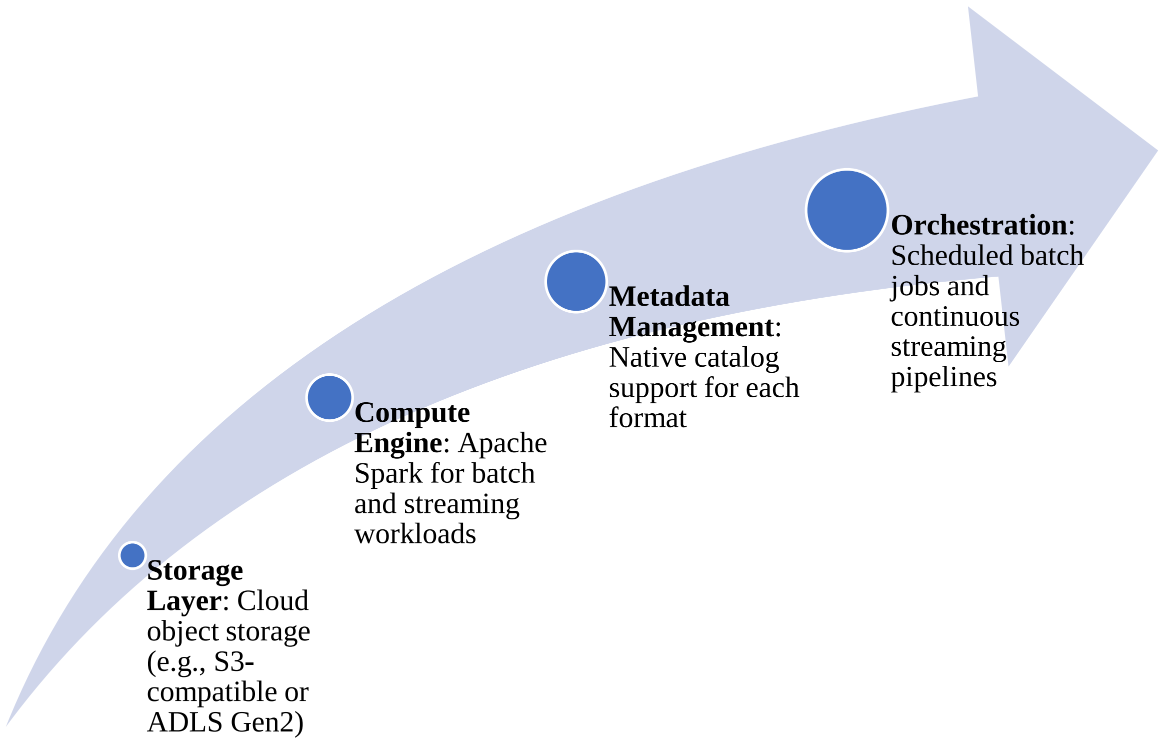
Storage Layer: Cloud object storage (e.g., S3-compatible or ADLS Gen2)
Compute Engine: Apache Spark for batch and streaming workloads
Metadata Management: Native catalog support for each format
Orchestration: Scheduled batch jobs and continuous streaming pipelines

Figure 2: A unified lakehouse execution architecture

5. Operational Metrics and Measurement
The framework captures metrics across three primary categories.

5.1 Operational Load
The operational load is a system and human effort to maintain the existence of lakehouse tables in the long term. It is quantified by the recurring nature of maintenance jobs, complexity of configuration, rate of metadata growth and failure recovery effort. Table formats that need much manual work, need lots of tuning or complicated recovery steps are said to create a greater operational load.

5.2 Runtime Performance
The performance measures are effectiveness and responsiveness of lakehouse systems, such as query execution time, streaming ingestion latency, sustained throughput and the impact of compaction on read throughput. Data is measured over a long time so as to obtain the steady-state behavior and results are obtained which are indicative of consistent operational performance and not some temporary spike in the behavior or anomaly which would give a false reference point at which to compare results across table formats and workloads.

5.3 Cost Evaluation
Cost Evaluation includes storage overheads due to versioning, deletions, compute overheads due to the ingestion and compaction, and additional maintenance jobs. Combining all of these considerations, the framework describes in totality the cost of ownership (including both operation and resource costs), as opposed to discussing only discrete query performance or workload efficiency.

6. Comparative Evaluation Workflow
The sample has a structured evaluation work process on comparing table formats. It starts with initializing all tables with the same schema by all the formats, loading batch data, and executing baseline analytical queries. Streaming ingestion is then launched, with constant updates and deletes and compaction and maintenance should be carried out as suggested. Periodic running of time-travel and CDC queries guarantee an active evaluation of workload. Performance

metrics are gathered in long periods of operation. Lastly, the results are then normalized, compared across formats to allow an overall consistent and complete assessment of performance, reliability and operational behavior of the system at varying workloads.

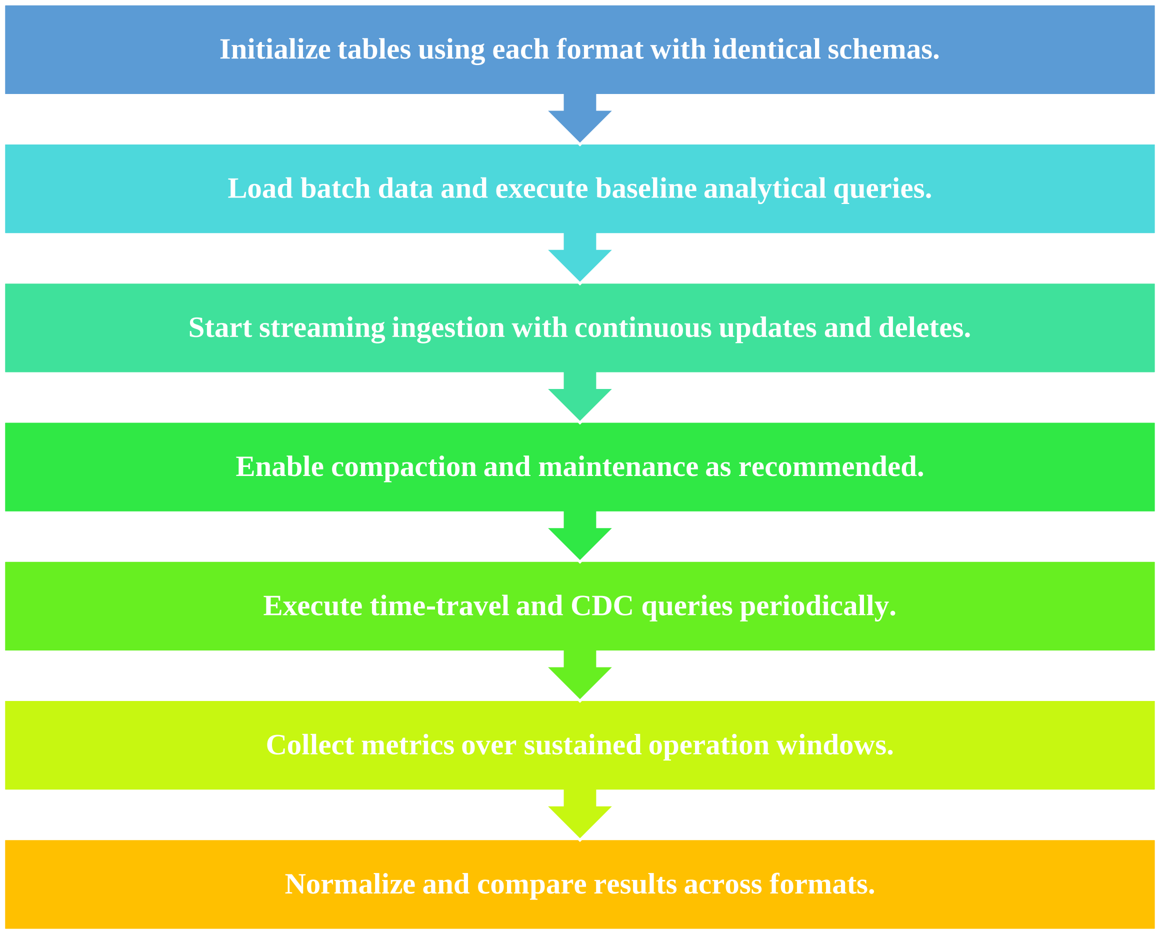
Initialize tables using each format with identical schemas.
Load batch data and execute baseline analytical queries.
Start streaming ingestion with continuous updates and deletes.
Enable compaction and maintenance as recommended.
Execute time-travel and CDC queries periodically.
Collect metrics over sustained operation windows.
Normalize and compare results across formats.

Figure 3: Workflow structure for operationalization of the lakehouse table formats

III. RESULTS AND ANALYSIS

In this section, an analytical report will be given on the experiment findings of the operationalization of Apache Iceberg, Delta Lake, and Apache Hudi with the same hybrid loads. The main focus is to determine the performance of each table format in practical use based on performance, operational overhead and cost as opposed to basing on theoretical capabilities. The assessment is based on the dimensions of decision matrixes as implemented above and is justified by the intensive quantitative measurements that are obtained in the course of the prolonged workload performance.

Each of the three formats was able to support transaction of ACID, concurrent read and write, and schema consistent operations over the course of the testing period. Nonetheless, significant variation was also noticed when workloads shifted towards initial bulk data loading to constant streaming ingestion of workload, regular upserts and deletes, and regular analytical queries. This shift made apparent trade-offs that are involved when using each format.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 2, March –April 2023||

DOI:10.15662/IJRPETM.2023.0602002

High-frequency deletes and updates showed the presence of performance divergence. Iceberg had an efficient way of running queries since it had a snapshot-based architecture, which reduced the query planning overhead and used immutable snapshots. Delta Lake was highly performing in the current-state queries, particularly with the use of optimized compaction, but had a moderate latency in time-travel queries. Although Hudi is very effective at supporting incremental ingestion, it demonstrated a very small latency increase in Merge-on-Read mode because of the cost of merging base and delta files at query time.

There was a great difference between formats in terms of overhead of operation. Iceberg enjoyed the advantage of minimal manual operation since snapshot and metadata are automatically managed, whereas Delta Lake needed a periodic compaction tuning to ensure an optimum performance, especially when it is under heavy mutation load. More operation attention was required by Hudi, especially in compaction schedules and balancing read with write amplification during Merge-on-Read operation. These variations can be emphasized in the need to highlight the fact that workload format and format-specific design decisions are closely intertwined with operational burden.

The cost analysis showed no direct relationship between compute efficiency and storage efficiency and raw query speed. The snapshot-based architecture of Iceberg resulted in a small storage overhead and a lowered query planning cost, and the repeated rewrites of files in Delta Lake and the merges operations in Hudi had an extra storage and compute cost.

The 1 TB TPC-DS dataset was used to test batch analytical performance, running the same query sets in all the formats once the initial data is loaded and then updated. Iceberg was the lowest time-travel and snapshot-based query engine, using its effective metadata pruning. Delta Lake had an optimized query performance in current-state queries with compaction, and Hudi had greater latencies with MoR overheads on reads.

Table 1: Batch Query Performance on TPC-DS (1 TB)

Table Format	Avg Query Latency (s)	Time Travel Query Latency (s)	Metadata Scan Time (ms)
Iceberg	18.6	20.1	120
Delta Lake	20.9	24.7	180
Hudi (MoR)	24.3	26.5	210

The metadata model of Iceberg is a snapshot-based model that reduces file listing and planning overhead hence its higher performance in both current and historical query. The use of transaction logs in Delta Lake creates a new overhead of parsing in time travel queries. The merge operations and commit timeline of Hudi complicate the query planning, especially of historical states.

The second test is aimed at a steady streaming workload of 10,000 events per second with inserts, updates, and deletes occurring often to test the CDC performance, upsert performance and write amplification. The Delta Lake got the highest sustained ingestion throughput because of its deep integration with Spark Structured streaming to process streams efficiently. Iceberg did not experience any drastic changes in ingestion rates although CDC-style operations demanded even more orchestration and snapshot management. Hudi was found to be effective in processing at an incremental level particularly with CDC streams, but was found to be slowed when high frequency deletes were maintained continuously as a result of Merge-on-Read overheads and compaction needs, which illustrate trade-offs between the efficiency of ingestion and complexity of operation.

Table 2: Streaming Ingestion and CDC Metrics

Table Format	Sustained Throughput (events/sec)	Avg Ingestion Latency (ms)	Write Amplification Factor
Iceberg	9,200	480	1.6
Delta Lake	9,800	420	1.4
Hudi (MoR)	9,500	510	1.9

The optimized streaming sink and auto-managed file of Delta Lake have a lower ingestion latency, and write amplification. The position- delete and equality- delete mechanisms of Iceberg offer flexibility but add more metadata and writes during periods of high mutation.

International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 2, March –April 2023||

DOI:10.15662/IJRPETM.2023.0602002

Deletes and upserts have high operational overhead in lakehouse systems since to keep the table healthy, they need to frequently compact and clean it when there is ongoing mutation traffic. The lowest operational burden was shown in Delta Lake which was also advantageous to automated compaction and vacuum processes which minimized manual intervention. Hudi especially in Merge-on-Read (MoR) mode, required continuous tuning of Hudi to ensure that performance was not impaired by the creation of log files. Iceberg offered flexible compaction opportunities, although this required proper scheduling to maximize the read performance, and control the storage and compute cost. These findings highlight the significance of the combination of the choice of the table format and the presence of the mutation patterns of workloads and the operational capacity.

Table 3: Compaction and Operational Overhead Metrics

Table Format	**Compaction Frequency (per day)**	**Avg Compaction Cost (CPU hrs/day)**	**Operational Complexity Score***
Iceberg	2	14.5	3.5
Delta Lake	1	10.2	2.0
Hudi (MoR)	3	18.7	4.2

The compaction model of Delta Lake reduces the amount of maintenance jobs needed, which decreases the operational load and computation. The modular strategy of Iceberg enables the optimization of the fine granules but pushes the complexity to the engineering team. The incremental model proposed by Hudi has a high operational cost because it needs compaction and tuning to be done on a regular basis.

IV. CONCLUSION AND FUTURE WORK

This paper provided an in-depth, engineering-based analysis of three popular lakehouse table formats apache iceberg, delta lake and apache hudi by evaluating their operational characteristics with the same realistic workloads. Instead of using feature claims or ecosystem popularity, the work has focused on workload alignment, operational overhead and total cost of ownership. The study offered an organized and repeatable method of choosing the format of tables in enterprise lakehouse implementations by presenting a decision matrix based on the key operational dimensions of Change Data Capture, upserts and deletes, time travel, compaction, and streaming ingestion.

The experimental outcome indicates that though the three formats are all compatible with ACID transactions as well as long-running and hybrid batch-stream processing, they differ significantly when using mutation-intensive and long-running workloads. The snapshot-based architecture of Apache Iceberg presented high benefits in time-travel queries and analytical consistency. Delta Lake was the most performance and operationally simple to use, specifically when it comes to mixed batch and streaming workloads and frequent CDC. Apache Hudi was found to be efficient in terms of incremental and upsert-intensive pipelines but with an increased operating complexity and maintenance. These results support the idea that there is no format which is the best and that the choice of the format has to be made on the basis of the peculiarities of the workload and the priorities of the work.

This study can be expanded in the future work in a number of directions. To begin with, other workload aspects like the complexity of schema evolution, multi-engine interoperability, and integrating governance are also dimensions that may be added to the decision matrix. Second, testing under a variety of cloud environments and storage backends would give more information about portability and vendor neutrality. Third, the future studies might investigate the new optimization opportunities like the vegetable reads, adaptive compaction, and interoperability between different formats. Lastly, longitudinal analyses that would involve the development of multi-month operational behavior would enhance cost and reliability.

REFERENCES

[1] M. Armbrust et al., “Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores,” Proc. VLDB Endowment, vol. 13, no. 12, pp. 3411–3424, Aug. 2020.
[2] B. Inmon, Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump. Basking Ridge, NJ, USA: Technics Publications, 2016.
[3] A. Laurent, D. Laurent, and C. Madera, Data Lakes. Hoboken, NJ, USA: John Wiley & Sons, 2020.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 2, March –April 2023||

DOI:10.15662/IJRPETM.2023.0602002

[4] F. Ravat and Y. Zhao, “Data Lakes: Trends and Perspectives,” in Proc. 30th Int. Conf. Database and Expert Systems Applications (DEXA), Linz, Austria, 2019, pp. 304–313.
[5] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang, “Leveraging the Data Lake: Current State and Challenges,” in Proc. 21st Int. Conf. Data Warehousing and Knowledge Discovery (DaWaK), Linz, Austria, 2019, pp. 179–188.
[6] J. Couto, O. T. Borges, D. D. Ruiz, S. Marczak, and R. Prikladnicki, “A Mapping Study on Data Lakes: An Improved Definition and Possible Architectures,” in Proc. 31st Int. Conf. Software Engineering and Knowledge Engineering (SEKE), Lisbon, Portugal, 2019, pp. 453–458.
[7] T. Ivanov and M. Pergolesi, “The Impact of Columnar File Formats on SQL-on-Hadoop Engine Performance,” Concurrency and Computation: Practice and Experience, vol. 32, no. 5, 2020.
[8] A. Davoudian and M. Liu, “Big Data Systems: A Software Engineering Perspective,” ACM Computing Surveys, vol. 53, no. 3, pp. 1–39, 2020.
[9] O. Herden, “Architectural Patterns for Integrating Data Lakes into Data-Warehouse Architectures,” in Proc. Int. Conf. Big Data Analytics (BDA), Hyderabad, India, 2020, pp. 12–27.
[10] M. Mehmood et al., “Implementing Big Data Lake for Heterogeneous Data Sources,” in Proc. IEEE Int. Conf. Data Engineering Workshops (ICDEW), Macau, China, 2019, pp. 37–44.
[11] M. Armbrust et al., “Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics,” CIDR, 2021.