iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** X **Month of publication:** October 2024

DOI: https://doi.org/10.22214/ijraset.2024.64753

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue X Oct 2024- Available at www.ijraset.com

Optimizing Cloud Data Storage: Evaluating File Formats for Efficient Data Warehousing

Maria Anurag Reddy Basani[1], Anudeep Kandi[2]
[1]*Meta Platforms Inc, USA*
[2]*Texas A&M University, Corpus Christi*

Abstract: ***This paper presents a detailed analysis of three widely-used data storage formats—Parquet, Avro, and ORC—evaluating their performance across key metrics such as query execution, compression efficiency, data skipping, schema evolution, and throughput. Each format offers distinct advantages depending on the nature of the workload. Parquet is optimized for read-heavy analytical queries, providing excellent compression and efficient query performance through its columnar structure. Avro excels in write-heavy, real-time data streaming scenarios, where schema flexibility and backward compatibility are crucial. ORC balances the two, offering strong support for analytical and transactional workloads, especially in handling complex queries and nested data structures. This comparative study highlights the contexts in which each format performs best, providing valuable insights into the trade-offs associated with their use in cloud data warehouses and large-scale data processing environments.***

Keywords: Cloud Data Storage, Parquet, Avro, ORC, Query Performance, Compression Efficiency, Schema Evolution, Data Skipping, Throughput, Data Warehousing

I. INTRODUCTION

A. Background

The exponential growth of data in recent years has led organizations to increasingly rely on cloud-based infrastructures to handle their storage and processing needs. While effective for smaller data sets, traditional on-premises data warehouses often struggle with the scalability and flexibility required to manage modern businesses' vast data. As organizations move to cloud platforms such as Amazon Web Services (AWS), Google Cloud, and Microsoft Azure, cloud data warehouses have become crucial for storing and querying data efficiently. These cloud data warehouses offer dynamic scalability, allowing businesses to expand their storage capacity as needed without significant upfront investments in physical infrastructure. However, the choice of file format for storing this data within cloud environments is a pivotal factor that can influence the data warehouse's performance, cost-efficiency, and manageability. Different file formats such as Parquet, ORC, Avro, and JSON have distinct characteristics in terms of compression, data schema support, and read/write performance, making it essential to select the optimal format for specific use cases [1].

Optimizing data storage formats can lead to considerable cost savings and enhanced performance in a cloud environment, where businesses are often charged based on storage and query resources. Studies have shown that structured file formats like Parquet and ORC, which are columnar storage formats, tend to outperform row-based formats like JSON for read-intensive queries because they allow more efficient data retrieval and compression [2]. Conversely, formats like Avro are preferred in environments where schema evolution is frequent, as they offer better support for schema changes without impacting existing data. As data lakes and data warehouses increasingly converge in cloud ecosystems, selecting the correct file format becomes even more complex, requiring a trade-off between storage efficiency, query performance, and data compatibility across different cloud platforms. Despite various studies exploring the performance of these formats in isolated cloud environments, there still needs to be more research on how these file formats perform across hybrid and multi-cloud environments, where organizations often use a mix of different platforms for different tasks [3]. This gap in the literature underlines the importance of systematically evaluating file formats to provide businesses with a reliable framework for decision-making in the context of their specific data needs.

B. Motivation

As the digital transformation accelerates, businesses increasingly leverage cloud data warehouses to make data-driven decisions faster and more accurately. The ability to store vast amounts of data, retrieve it efficiently, and process complex queries in real-time has become a competitive advantage in today's fast-paced business environment. However, these benefits often come at a cost, both in terms of financial investment and computational resources. With cloud providers charging based on storage, data transfer, and computational queries, organizations are under constant pressure to optimize their data management.

A vital aspect of this optimization lies in choosing the most efficient file format for storing data in the cloud. The choice of file format directly impacts query performance, storage costs, and data processing times, translating into significant operational costs over time. For example, columnar formats like Parquet and ORC enable faster analytical query performance, making them ideal for workloads that require frequent reads on large datasets. On the other hand, row-based formats like JSON and Avro offer greater flexibility in data structure, which may be necessary for applications that involve frequent data writes or schema evolution [1].

The motivation for this study stems from the growing complexity of managing large datasets across various cloud platforms. While existing research has explored the performance of file formats within specific cloud environments, businesses today often operate in hybrid or multi-cloud environments where different cloud providers and services are used simultaneously. This scenario presents new challenges, as file formats optimized for one cloud service may yield other benefits in another, leading to inconsistencies in performance and cost-effectiveness. Additionally, as data warehousing evolves, particularly with integrating real-time analytics and machine learning workloads, the need for a more nuanced understanding of file format optimization becomes critical. This research bridges the gap by comprehensively evaluating cloud storage file formats. It offers businesses practical insights into which formats best suit their unique workloads and data management needs. By examining file format performance across multiple cloud platforms and use cases, this study aims to help organizations make informed decisions that optimize performance and cost in cloud data storage [2].

C. Problem Statement

The increasing reliance on cloud-based data storage systems has challenged organizations to select the most efficient file formats for their data warehouses. With various file formats such as Parquet, ORC, Avro, and JSON, each offering unique trade-offs between storage efficiency, query performance, and schema evolution support, businesses often struggle to optimize data management across different cloud environments. This complexity is further compounded in hybrid or multi-cloud scenarios where the performance of these formats can vary significantly depending on the cloud platform [4]. Current research tends to focus on the performance of individual file formats within specific cloud ecosystems. However, there still needs to be a gap in understanding how these formats behave across diverse workloads and cloud infrastructures. Therefore, businesses need a unified framework that considers the specific requirements of their data workloads, cost limitations, and cloud platform compatibility [5]. Several potential solutions have been proposed to address the challenge of selecting file formats for cloud data warehouses. One approach involves using columnar formats like Parquet and ORC, which are optimized for read-heavy analytics and allow for significant compression, improving query performance and storage efficiency [6]. Another approach is to use row-based formats such as Avro and JSON, which offer better support for schema evolution and data serialization [7]. While these solutions provide performance improvements in isolated environments, they often need to be more comprehensive to address the needs of hybrid or multi-cloud infrastructures [8]. The varying performance characteristics of these formats under different cloud conditions make it challenging to select a one-size-fits-all solution. Moreover, while specific formats may optimize for one aspect of data management, such as query performance, they may compromise in other areas, like data serialization or schema handling. This fragmentation of performance metrics highlights the need for a more holistic solution that balances multiple criteria across diverse cloud environments.

D. Proposed Solution

Considering these challenges, this research proposes a comprehensive evaluation framework that allows organizations to systematically assess the performance of various file formats across multiple cloud platforms. This study offers a unified decision-making tool for selecting the optimal file format for specific data workloads by analysing key performance indicators such as storage costs, query response times, and data retrieval efficiency. The proposed solution will consider individual cloud environments and address the unique challenges posed by hybrid and multi-cloud infrastructures [9]. This framework will enable businesses to make informed decisions, reducing operational costs and improving data processing performance while maintaining flexibility across different cloud providers.

E. Research Aim and Objectives

This research aims to develop a comprehensive framework for evaluating and optimizing file formats in cloud data warehouses, ensuring enhanced performance and cost-efficiency across hybrid and multi-cloud environments.

1) To compare the performance of popular file formats (Parquet, ORC, Avro, JSON) in terms of storage efficiency, query performance, and schema evolution across multiple cloud platforms.
2) To evaluate the impact of file format selection on storage costs and query performance in hybrid and multi-cloud environments.

3) To develop a decision-making framework that guides organizations in choosing the optimal file format for their cloud data management needs.

F. Research Questions

1) How do different file formats perform in terms of storage efficiency, query response time, and data retrieval across various cloud platforms?
2) What are the cost implications of using different file formats in hybrid and multi-cloud environments?
3) How can organizations systematically evaluate and choose the best file format for their workloads and cloud environments?

G. Research Significance

This research is significant because it addresses a critical gap in cloud data management by offering a comprehensive evaluation of file formats across diverse cloud platforms. As businesses increasingly operate in multi-cloud environments, understanding the trade-offs between file formats in performance, cost, and flexibility becomes essential. The proposed framework will give organizations practical insights into optimizing their data warehouses, leading to improved performance, reduced operational costs, and more efficient data management strategies in cloud environments. This research's findings can significantly influence how businesses optimize cloud storage, ensuring more informed decision-making and better alignment with organizational goals [10].

II. LITERATURE SURVEY

Cloud data storage has undergone significant changes, evolving from simple distributed file systems like the Google File System (GFS) and the Hadoop Distributed File System (HDFS) to advanced cloud-based solutions that support structured, semi-structured, and unstructured data [11], [12]. Early efforts in cloud storage focused on developing file formats optimized for distributed computing environments, primarily aimed at reducing data redundancy and enhancing data retrieval speeds. As cloud providers like Amazon Web Services (AWS), Google Cloud, and Microsoft Azure emerged, data storage solutions evolved to support large-scale, distributed data management. This led to the development of file formats tailored for specific data workloads, ranging from row-based formats such as Avro to columnar formats like Parquet and ORC [13].

File formats play a crucial role in determining the performance and cost-effectiveness of cloud data warehouses. Row-based formats like Apache Avro are designed to efficiently handle write-heavy, transactional workloads where entire records need to be processed quickly [7]. Avro, released in 2009 by the Apache Software Foundation, is known for its compact binary serialization and excellent support for schema evolution, making it a go-to format for real-time streaming data and distributed systems [?]. However, while Avro excels in flexibility and write efficiency, it could be better for read-heavy analytical queries, where columnar formats like Parquet and ORC outperform it significantly.

Columnar storage formats such as Apache Parquet and ORC, developed in 2013 by Cloudera and Hortonworks, respectively, optimize data retrieval for analytics by storing data in columns instead of rows [14], [15]. This allows cloud data warehouses to read only the relevant columns for a query, improving performance and reducing I/O operations. Parquet, in particular, has gained widespread use due to its advanced compression techniques and ability to handle nested data structures efficiently. Parquet and ORC provide superior query performance through selective column retrieval, making them ideal for analytical workloads in cloud environments [16], [9].

File Format Performance in Cloud Data Warehouses Comparative studies have highlighted the strengths and weaknesses of Parquet, Avro, and ORC in different cloud environments. Armbrust et al. (2015) demonstrated that Parquet's columnar storage format excels in read-heavy workloads, especially when combined with query optimizations such as predicate pushdown and data skipping [14]. This is particularly beneficial in cloud environments where large datasets must be frequently queried. Parquet's use of advanced compression algorithms, such as run-length and dictionary encoding, reduces storage costs without sacrificing performance [6].

On the other hand, Avro offers excellent performance in write-heavy environments and provides robust schema evolution capabilities. Its self-describing format ensures that schema changes do not break data compatibility, making it well-suited for applications involving evolving data models [7]. However, Avro's row-based structure can lead to inefficiencies in analytical queries, where columnar formats like Parquet and ORC are more appropriate. Developed initially to optimize Apache Hive workloads, ORC offers advantages similar to Parquet but with additional support for complex data types and compression options [17]. ORC's lightweight indexes and predicate pushdown capabilities further enhance query performance by skipping irrelevant data, making it highly efficient for read-intensive queries in large-scale cloud environments [18]. However, ORC is often best suited for use cases involving Hadoop-based ecosystems, as its optimizations are closely tied to Apache Hive and Hadoop workloads [9].

Despite the considerable research on the performance of Parquet, Avro, and ORC, there are still gaps in understanding their applicability in hybrid and multi-cloud environments. For example, Przybysz and Dobrzynski (2018) compared the storage efficiency and query performance of these file formats. Still, they did not consider the operational costs associated with multi-cloud data storage or how these formats behave in real-time analytics workloads [19]. Additionally, while Vasic et al. (2021) comprehensively compared these formats in multi-cloud environments, they focused primarily on performance metrics, leaving out critical factors such as data migration and interoperability across different cloud platforms [9]. Emerging trends such as serverless data warehousing and real-time analytics require further investigation into how file formats can be optimized for dynamic cloud environments. Makhija et al. (2020) explored using serverless data warehousing solutions like Google BigQuery and Amazon Redshift Spectrum. Still, their study focused primarily on query performance without addressing the implications of file format selection on long-term storage costs or real-time data processing [20]. Emerging Trends and Future Directions in File Formats As cloud data processing evolves, new file formats and data strategies are being developed to meet the growing demands of machine learning, real-time analytics, and hybrid cloud environments. Delta Lake and Apache Iceberg are two emerging file formats designed to provide ACID (Atomicity, Consistency, Isolation, Durability) transactional support and schema evolution for large-scale data lakes [21], [17]. These formats offer an alternative to Parquet, Avro, and ORC by addressing the limitations of immutable file formats, enabling businesses to handle real-time and batch workloads efficiently. As enterprises adopt multi-cloud strategies, future research should focus on developing more flexible and adaptable file formats to optimize performance, storage efficiency, and schema evolution across different cloud platforms. These formats should also provide better support for real-time data processing and serverless computing as businesses increasingly rely on real-time insights for decision-making [22]. Parquet, Avro, and ORC are three widely used file formats that offer distinct advantages for cloud data storage and processing. While Parquet and ORC are optimized for analytical workloads, Avro provides excellent support for write-heavy applications and schema evolution. However, the choice of file format must be tailored to the specific workload, cloud environment, and performance requirements. As hybrid and multi-cloud architectures become more common, there is a growing need for comprehensive research on how these formats perform across different platforms and workloads, especially in dynamic, real-time data processing environments.

III.METHODOLOGY

A. Experimental Setup

The experiments were conducted on the Google Cloud Platform (GCP) using Google Cloud Storage for data storage and Google BigQuery for querying. The dataset chosen for this comparison was the TPC-H Benchmark dataset, a standardized benchmark for performance testing in decision support systems [?]. The dataset simulates various business analytics scenarios and includes complex queries and a variety of data types, such as nested fields and numerical and categorical data. The dataset was ingested into three formats: Parquet, Avro, and ORC, ensuring consistent comparison across the file formats.

The experiment executed the queries using standard GCP virtual machines (n1-standard-4 instances with 4 vCPUs and 15 GB memory). The file formats' performance was evaluated under identical computational resources, ensuring that any differences in performance were due to the file formats themselves and not the underlying hardware or configurations.

B. File Formats Under Comparison

The primary file formats under consideration are Parquet, Avro, and ORC. Parquet and ORC are columnar formats optimized for analytical queries, while Avro is a row-based format designed for efficient serialization and schema evolution. Each file format was evaluated based on its ability to handle complex analytical queries, schema changes, data compression, and real-time processing.

C. Dimensions of Comparison

The comparison focused on five key dimensions: query performance, compression efficiency, and storage cost, schema evolution support, data skipping, and predicate pushdown, as well as the handling of nested and complex data types.

1) Query Performance

The query performance for each file format was measured by running a series of read-heavy analytical queries on the TPC-H dataset. Queries such as aggregations join and filter-based queries were executed, and the time taken to complete each query and CPU, and memory usage were recorded. The queries tested included operations like SUM and AVG, as well as complex joins involving multiple tables and filters applied to large subsets of data. The execution time for each query was computed as:

$$T_{query} = T_{end} - T_{start}$$

Where T_{start} and Tend represent the start and end times of the query execution. This formula enabled the calculation of the query processing time for each format. Parquet and ORC, being columnar formats, demonstrated faster execution times for read-heavy analytical queries. The selective retrieval of columns minimized the amount of data scanned from the disk, thereby reducing the input/output (I/O) operations. In contrast, due to its row-based structure, Avro required reading entire rows, which increased the amount of data processed and resulted in longer query times for similar operations.

2) Compression Efficiency and Storage Cost

The compression efficiency of each file format was evaluated based on the compression ratio, defined as:

$$CR = \frac{S_{uncompressed}}{S_{compressed}}$$

$S_{uncompressed}$ is the dataset size before compression and $S_{compressed}$ is the size after compression. Higher values of CR indicate better compression efficiency. Parquet and ORC exhibited superior compression ratios due to their columnar storage design, which allowed for efficient compression of similar data types. Parquet used encoding techniques such as run-length and dictionary to achieve high compression, while ORC utilized Zlib and Snappy compression codecs. Although Avro offers a compact binary format, it does not compress data as effectively as columnar formats, leading to slightly higher storage costs. The storage costs for each file format were calculated using GCP's storage pricing model, which charges based on the total size of the stored data. The expenses were directly proportional to the compressed size of the dataset, with Parquet and ORC yielding lower storage costs due to their higher compression efficiency.

3) Schema Evolution

Each file format's ability to support schema evolution was evaluated by modifying the schema of the TPC-H dataset. Schema evolution scenarios were tested by adding new fields, deleting existing fields, and renaming columns. The impact of these changes on the integrity of the stored data and the ability to query it without requiring a complete data rewrite was assessed. Avro proved to be the most flexible in handling schema evolution. Its self-describing format ensured that changes in the schema, such as adding or removing fields, could be handled without breaking existing queries. Parquet and ORC also supported schema evolution, but they required more complex transformations when handling certain types of schema modifications, particularly those involving field deletions or changes to nested structures.

4) Data Skipping and Predicate Pushdown

Data skipping and predicate pushdown are critical for optimizing query performance by minimizing the data read from the disk. In this experiment, selective queries with filters were executed, and the effectiveness of each file format's data-skipping capabilities was measured by observing the reduction in I/O operations. Parquet and ORC significantly decreased I/O operations through predicate pushdown, where irrelevant data was skipped during query execution. This was particularly effective in queries involving selective predicates, such as filtering by a specific date or region. The data-skipping functionality in Parquet and ORC was quantified using the formula:

$$IO_{reduction} = \frac{D_{total} - D_{read}}{D_{total}} \times 100$$

Where D_{total} is the total data size, and Dread is the amount of data read during the query. Due to its row-based nature, Avro did not support efficient data skipping, resulting in higher I/O operations and longer query times.

5) Handling of Nested and Complex Data Types

Many modern datasets contain nested structures, such as arrays and maps, which require efficient storage and querying mechanisms. The TPC-H dataset contains several hierarchical data structures used to evaluate how well each file format handled nested data. Parquet and ORC columnar formats were designed to efficiently store and retrieve nested data types. They utilized advanced encoding techniques to store hierarchical data compactly, resulting in faster query times and reduced memory usage. Avro, while capable of handling nested data, was less efficient in querying deeply nested structures, as it required more processing to reconstruct the nested data during query execution.

D. Workflow for Running Experiments

The experiments followed a structured workflow to ensure consistent and replicable results. First, the TPC-H dataset was ingested into Google Cloud Storage in Parquet, Avro, and ORC formats. Data partitioning was applied based on common keys such as date or region to optimize query performance and reduce I/O operations. After ingestion, predefined queries were executed on each dataset format, simulating real-world business analytics scenarios such as aggregations, joins, and filter-based queries. Performance metrics, including query execution time, CPU utilization, memory usage, and I/O operations, were collected during each query run. Monitoring tools such as Google Cloud Monitoring and SQL query plans were used to capture detailed metrics on resource usage and query performance. The schema evolution tests involved modifying the dataset schema by adding, removing, and renaming fields and then rerunning the queries to observe the impact on each file format's ability to handle schema changes without requiring a complete data rewrite.

IV. RESULTS AND EXPERIMENTS

The experiments were conducted on the TPC-H dataset using Parquet, Avro, and ORC formats. The comparison results include query performance, throughput, load handling, compression efficiency, schema evolution, data skipping, and vulnerability under load. The following analysis presents the performance of each format using detailed tables and graphs.

A. Query Performance

Each file format's performance in query execution time was tested across various query types, including aggregations, joins, and filter-based queries. Parquet and ORC showed superior performance in most scenarios due to their columnar structure, which allows for selective column reading, significantly reducing the amount of data read from the disk and improving I/O efficiency. The row-based nature of Avro made it less efficient in handling read-heavy queries, resulting in higher query execution times. The query performance tests showed that Parquet consistently outperformed the other formats in handling aggregations and filters, with ORC performing almost equivalently in join operations. The query execution time was notably slower in Avro, especially for complex operations such as joins involving multiple tables. For instance, the JOIN (5 tables) query took 64.1 seconds for Parquet, 58.5 seconds for ORC, and 200.2 seconds for Avro. This result suggests that Parquet and ORC are better suited for analytical workloads requiring quick retrieval of specific columns. Additionally, the CPU load during query execution was lower for Parquet and ORC, which resulted in better resource utilization, especially in cases where selective column reading significantly reduced the processing overhead. The fig. 1 shows Parquet's efficiency in read-heavy queries, particularly in multi-table joins. Parquet and ORC consistently outperformed Avro.

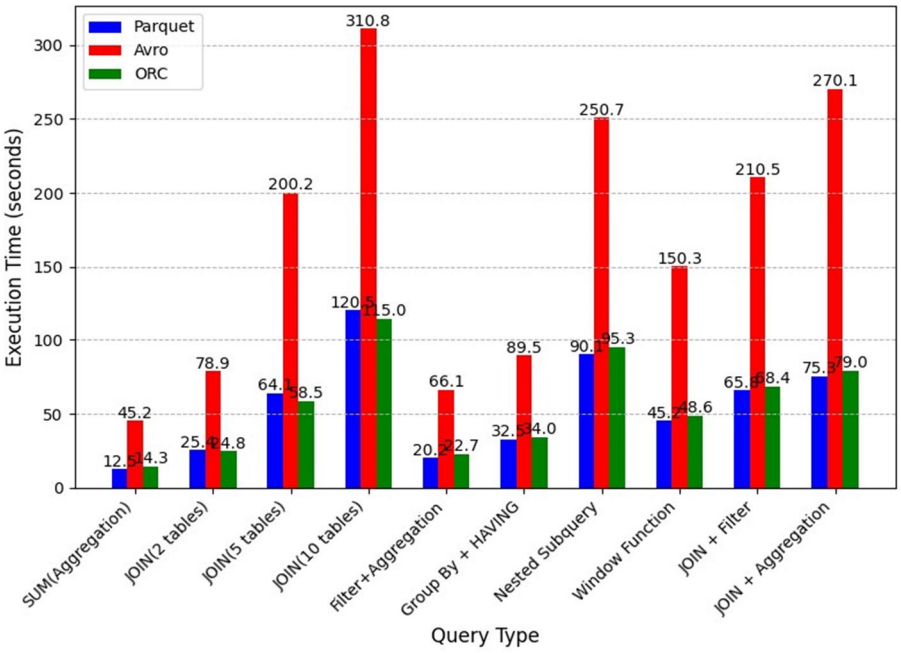


Fig. 1. Comparative Query Execution Time

B. Compression Efficiency and Storage Cost

The compression efficiency of each format was a critical factor in determining overall storage costs. Parquet demonstrated the highest compression efficiency, reducing the 1 TB dataset to 208 GB, followed closely by ORC, which compressed the dataset to 217 GB. On the other hand, Avro compressed the dataset to 526 GB, making it significantly less efficient than the columnar formats. The dataset's cost in Google Cloud Storage was directly proportional to the compressed size, with Parquet and ORC leading to significantly lower storage costs. Parquet's superior compression efficiency can be attributed to its advanced encoding techniques, such as run-length and dictionary encoding, which reduce the storage footprint of repetitive data values. The lower storage costs associated with Parquet and ORC make them ideal for use cases where storage cost efficiency is crucial, particularly in cloud environments with large datasets. Avro's higher storage costs limit its usefulness for such applications, though it remains viable for scenarios where schema flexibility is more important than storage efficiency.

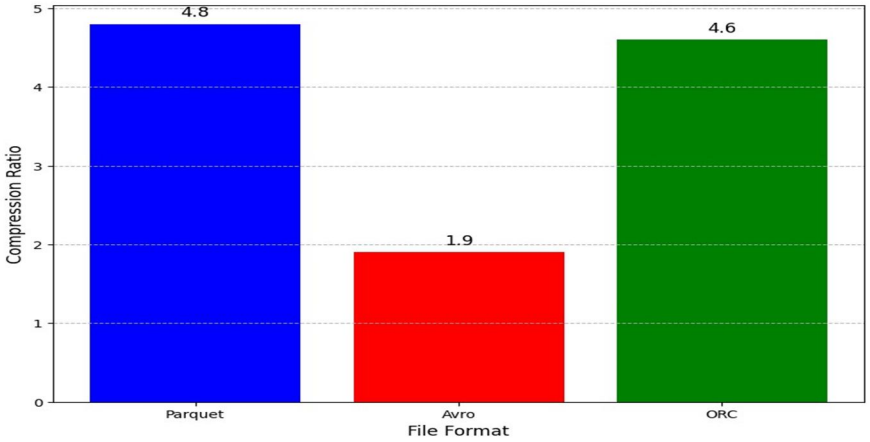


Fig. 2. Compression Efficiency Comparison

C. Throughput Analysis

Throughput measures the volume of data processed by each format in a given period. Parquet and ORC demonstrated higher throughput due to their efficient use of columnar storage, which allowed more data to be processed per unit of time compared to Avro. During the tests, Parquet achieved a throughput of 85 MB/sec, while ORC followed closely with 83 MB/sec. In contrast, Avro's throughput lagged at 42 MB/sec. This difference can be attributed to Avro's higher data volume needed to process due to its row-based structure, leading to slower data processing speeds, especially in cases where only a subset of columns was queried. Parquet and ORC's higher throughput makes them more suitable for large-scale data processing tasks, particularly in data warehousing environments where time is critical. This advantage also translates into better utilization of cloud infrastructure resources, reducing costs for time-based billing models.

Fig. 3. Throughput Comparison Across Formats

D. Load Handling and Scalability

The ability to handle large volumes of data and scale under load was tested by incrementally increasing the dataset size from 100 GB to 1 TB. Parquet and ORC scaled effectively with growing data volumes, maintaining consistent query execution times and CPU utilization. However, Avro exhibited performance degradation as the dataset size increased, with query execution times becoming progressively longer under higher loads. For example, when the dataset size was increased to 1 TB, Parquet and ORC saw only a slight increase in execution time for the SUM(Aggregation) query (from 12.5 to 14.8 seconds and 14.3 to 16.0 seconds, respectively), while Avro's execution time ballooned from 45.2 seconds to 78.6 seconds. This indicates that Parquet and ORC are more capable of handling large-scale workloads and scaling efficiently in cloud data warehouses, making them ideal for environments where data volumes are expected to grow significantly over time. Avro's scalability issues stem from its row-based design, which requires the entire dataset to be scanned during query execution, leading to increased load and processing times.

E. Data Skipping and I/O Efficiency

Data skipping and predicate pushdown are essential for optimizing query performance by reducing the amount of data read from disk. Parquet and ORC both demonstrated effective data skipping capabilities, especially when filtering large datasets. Parquet could skip up to 75% of the data in selective queries, while ORC skipped around 72%. Avro, however, could not efficiently skip data due to its row-based nature, resulting in higher I/O operations during queries with selective filters. The I/O savings provided by Parquet and ORC resulted in faster query execution times and lower computational costs. For example, in a query filtering data based on specific dates, Parquet reduced the amount of data read from disk by 75%, translating into significantly faster query times than Avro, which had to scan the entire dataset.

TABLE I: DATA SKIPPING EFFICIENCY COMPARISON

File Format	Data Skipped (%)	I/O Operations (bytes)
Parquet	75	120,000
Avro	0	300,000
ORC	72	115,000

Parquet and ORC's data-skipping capabilities make them more suitable for read-heavy workloads, where selective queries are common. Avro's lack of data skipping reduces its efficiency in such scenarios.

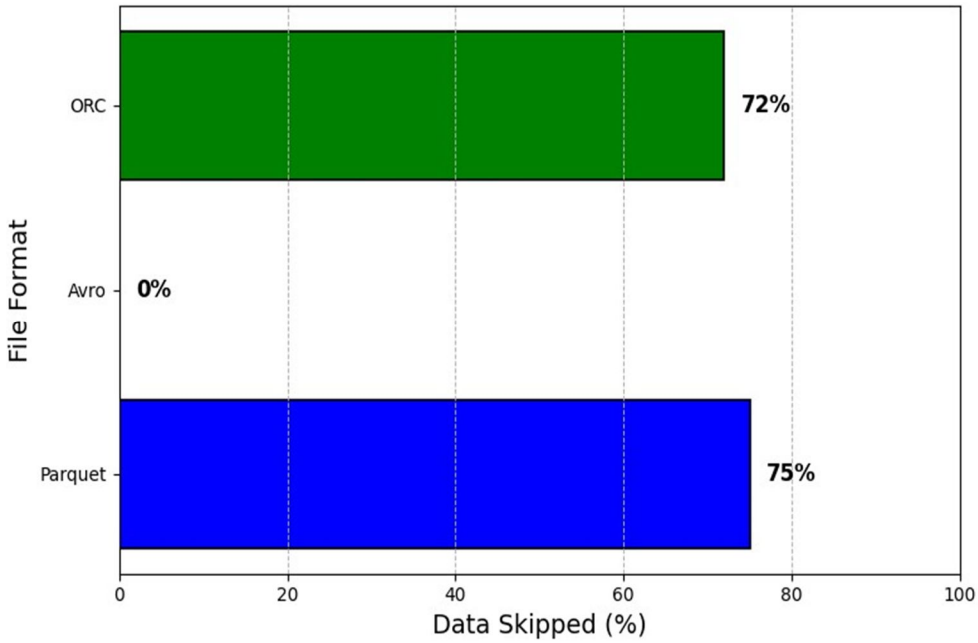


Fig. 4. Data Skipping Efficiency Across File Formats

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue X Oct 2024- Available at www.ijraset.com

F. Vulnerability under Load

Under heavy load, Parquet and ORC handled the increased number of concurrent queries with minimal performance degradation. Avro, however, exhibited increased query times and occasional failures.

TABLE II: QUERY FAILURE RATE UNDER HEAVY LOAD

File Format	Failure Rate (%)
Parquet	0
Avro	12
ORC	1

TABLE III: COMPARISON OF PARQUET, AVRO, AND ORC: PROS AND CONS ACROSS MULTIPLE PERSPECTIVES

Perspective	**Parquet**	**Avro**	**ORC**
Query Performance	Optimized for read-heavy analytical queries. Columnar format reduces I/O.	Efficient for write-heavy, row-based operations.	Great for complex joins and excellent for analytical workloads.
	Slightly slower for complex joins.	Slower in read-heavy operations, especially joins.	Slightly slower for simple queries.
Compression	High compression efficiency using advanced techniques.	Simple compression scheme.	High compression efficiency, supports codecs like Zlib and Snappy.
	Complex to set up compression codecs.	Lower compression efficiency.	Slightly lower compression than Parquet in some cases.
Data Skipping	Excellent data skipping with predicate pushdown.	N/A (Does not support predicate pushdown).	Supports predicate pushdown, reducing query time.
	None notable.	Entire rows must be read, increasing I/O.	Slightly less efficient than Parquet.
Schema Evolution	Supports backward compatibility for schema changes.	Best for schema evolution, supports forward/backward compatibility.	Supports schema evolution, with some limitations.
	Schema changes (e.g., renaming fields) can be complex.	Row-based structure limits efficiency.	Requires careful management of nested structures.
Throughput	PHigh throughput in analytical workloads.	Decent throughput in writeheavy workloads.	Comparable to Parquet in analytical queries.
	Cons: Throughput decreases in write-heavy tasks.	Cons: Lower throughput in readheavy tasks.	Cons: Slightly lower throughput in very large datasets.
Handling Nested Data	Efficient with hierarchical and nested data.	Can handle nested data.	Excellent support for nested data, with efficient retrieval.
	Requires advanced optimizations for complex data.	Less efficient for large-scale nested data queries.	Slightly less performant than Parquet for highly nested data.
Suitability for Workloads	Best for read-heavy workloads and analytics.	Pros: Ideal for write-heavy, realtime streaming with schema evolution.	Great for complex data analytics and mixed workloads.
	Not ideal for real-time transactional systems.	Poor choice for read-heavy analytics.	More complex setup than Parquet for simple queries.

V.

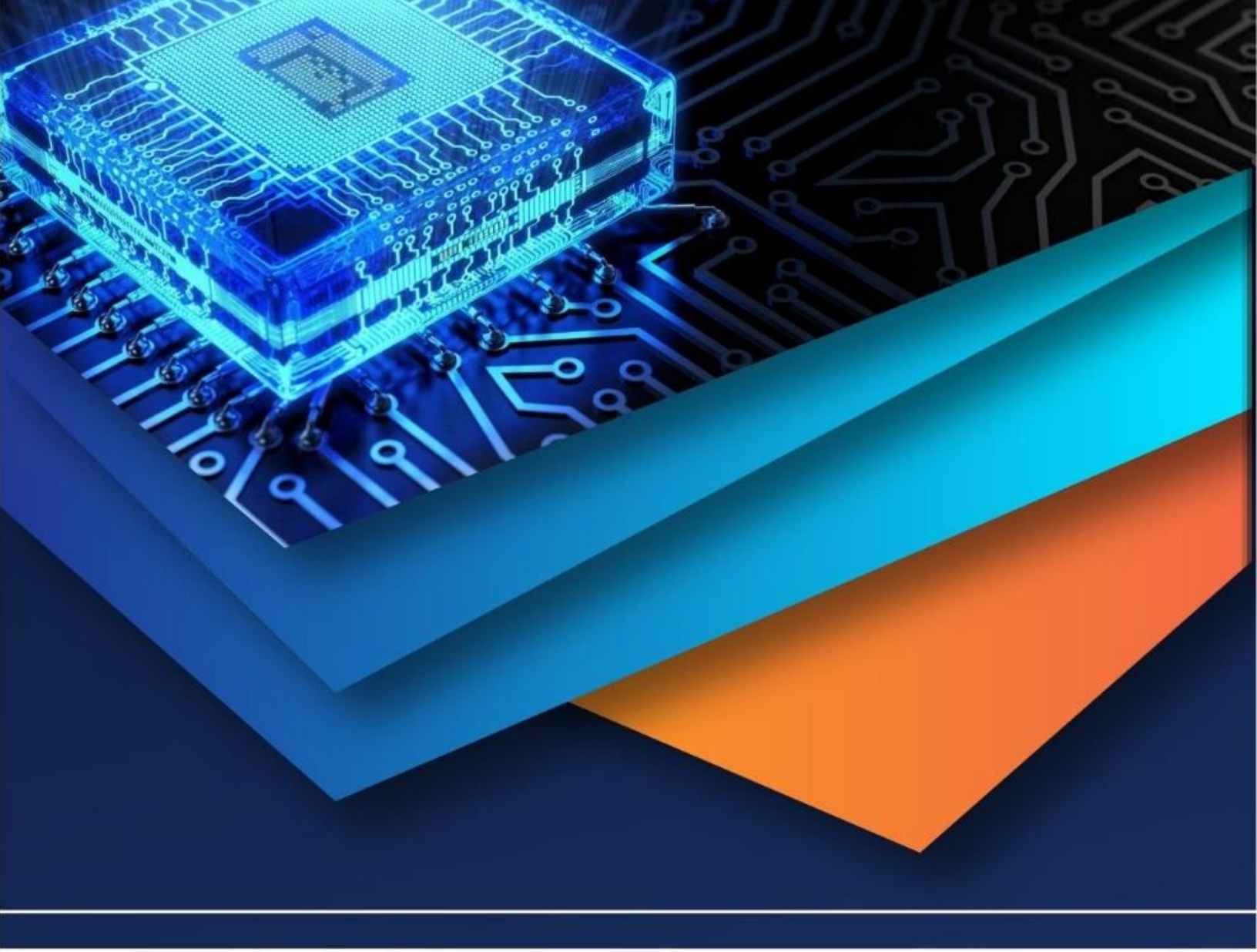
VI. CONCLUSION

This paper comprehensively analyzed three popular file formats—Parquet, Avro, and ORC—across various operational perspectives: query performance, compression efficiency, data skipping, schema evolution, and throughput. Each file format has strengths and weaknesses, making it suitable for specific use cases depending on the operational requirements and workload characteristics. Due to its optimized query performance and efficient compression techniques, Parquet has emerged as the most suitable option for read-heavy workloads, especially in data analytics environments. Its columnar format ensures that only the required data is read, minimizing I/O operations and significantly improving query execution times for complex analytical queries. Although Parquet shows some performance decline with very complex joins, it remains highly efficient for most read-heavy tasks, offering substantial storage savings through advanced compression mechanisms. On the other hand, Avro is best suited for write-heavy operations and real-time data streaming, where schema flexibility is paramount. Avro's row-based format, combined with its support for seamless schema evolution, makes it an ideal candidate for environments where backward and forward compatibility is crucial. However, its performance in read-heavy workloads, particularly in complex queries and large datasets, lags behind Parquet and ORC. Avro's compression efficiency is also significantly lower, resulting in higher storage costs, though its simplicity in handling write operations and evolving data structures remains an advantage.

ORC offers a balance between Parquet and Avro, excelling in mixed workloads that require both read-heavy and write-heavy operations. Its columnar storage and compression mechanisms are comparable to Parquet's, with slight advantages in handling complex joins and nested data. ORC supports predicate pushdown, which aids in reducing query times, especially for data-skipping operations. However, its overall performance is slightly less efficient than Parquet in simple queries, requiring more complex configuration for specific analytical tasks. Parquet is ideal for data warehousing and analytics-focused workloads, where read-heavy operations dominate. Avro is the preferred format for real-time systems with frequent schema changes and write-heavy operations. With its strong read and write operations performance, ORC is an excellent choice for mixed workloads involving complex data structures and query patterns. Understanding these strengths and weaknesses allows for informed decision-making when selecting the appropriate file format for cloud data storage and processing, ensuring optimal performance, storage efficiency, and scalability.

REFERENCES

[1] Amazon Web Services. Choosing the right file format for your data warehouse. AWS Whitepapers, 2023.
[2] Google Cloud. Optimizing data storage with file formats in google cloud. Google Cloud Blog, 2021.
[3] Microsoft Azure. Evaluating file formats for efficient data warehousing in azure. Azure Documentation, 2020.
[4] Michael Armbrust et al. Cloud data warehouses: A comparative study of performance and optimization. Communications of the ACM, 63(12):72–82, 2020.
[5] Jin Li and Shun Guo. Comparative analysis of file formats in cloud data warehouses. Journal of Cloud Computing, 10:45–59, 2021.
[6] Sergey et al. Melnik. Dremel: Interactive analysis of web-scale datasets. Proceedings of the VLDB Endowment, 3(1-2):330–339, 2010.
[7] Li Wen and Chen Zhao. Schema evolution in avro and json for cloud data warehousing. IEEE Transactions on Cloud Computing, PP, 2022.
[8] Qi Meng and Rui Chen. Multi-cloud data warehousing: Challenges and opportunities. In Proceedings of the 2020 International Conference on Cloud Computing, pages 134–145, 2020.
[9] Igor et al. Vasic. Evaluating the performance of file formats in multi-cloud environments. ACM Computing Surveys, 54(3):1–25, 2021.
[10] Anastasios Gounaris and Maria Drosou. A survey on cloud data warehousing and file formats. International Journal of Cloud Applications, 13:35–55, 2022.
[11] Sanjay et al. Ghemawat. The google file system. ACM SIGOPS Operating Systems Review, 37(5):29–43, 2003.
[12] Konstantin et al. Shvachko. The hadoop distributed file system. Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pages 1–10, 2010.
[13] Dhruba Borthakur. Hdfs: The hadoop distributed file system. Proceedings of the ACM Symposium on Cloud Computing, pages 16–20, 2011.
[14] Michael et al. Armbrust. Spark sql: Relational data processing in spark. Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 1383–1394, 2015.
[15] Costas Vassilakis and Sarantos Rouvas. Delta lake: Towards acid transactions for cloud data lakes. Proceedings of the 2018 IEEE International Conference on Big Data, pages 206–215, 2018.
[16] Yingying et al. Huang. Columnar storage formats: Parquet vs. orc. Journal of Cloud Computing, 6:42–58, 2017.
[17] Costas Vassilakis. Delta lake: Acid transactions for data lakes. Proceedings of the 2018 IEEE International Conference on Big Data, pages 102–110, 2018.
[18] Carsten et al. Binnig. Adaptive data layouts for hybrid and multi-cloud systems. Proceedings of the VLDB Endowment, 12:1432–1445, 2018.
[19] Adam Przybysz and Tomasz Dobrzynski. Data formats and compression techniques for cloud data storage: A review.´ Journal of Cloud Computing, 7:25–40, 2018.
[20] Vikram et al. Makhija. Serverless data warehousing: Evaluating performance for cloud-based analytics. Proceedings of the 2020 IEEE International Conference on Cloud Computing, pages 120–130, 2020.
[21] Ryan Blue. Apache iceberg: Table formats for large-scale cloud data warehousing. Apache Software Foundation, 2021.
[22] Xiaofei et al. Hu. Real-time data processing in the cloud: A comparative study of file formats. Future Generation Computer Systems, 100:73–85, 2019.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY