

Big Data File Formats: Evolution, Performance, and the Rise of Columnar Storage

Sainath Muvva

Abstract

The unprecedented growth in data volume has necessitated more optimal storage mechanisms. This paper traces the evolution of big data file formats, from the well-known CSV/text to sophisticated serialization standards such as Avro, Parquet, and ORC. We analyze the constraints of conventional row-based formats and explore the emergence of columnar storage structures designed for high-performance analytics. This paper investigates how contemporary formats tackle key challenges in big data ecosystems, including data compression, schema adaptability, and query optimization, particularly in distributed computing environments. We examine the growing popularity of Avro, Parquet, and ORC within platforms like Hadoop and Spark, evaluating their influence on storage efficiency and data access speeds.

Keywords: Big data, CSV, Avro, Parquet, ORC, data compression, schema flexibility, distributed computing, Hadoop, Spark

Introduction

The exponential growth of digital information in recent years has created unprecedented challenges in managing, analyzing, and storing vast amounts of data. Traditional file formats, once adequate for organizing information, are now struggling to meet the demands of modern big data ecosystems. This limitation is particularly apparent in distributed computing environments like Hadoop and Spark, where high performance, scalability, and flexibility are crucial. As a result, the field of big data has seen a continuous evolution of file formats to address these emerging requirements.

This study explores the development trajectory of big data file formats, starting from the basic but widely used CSV to more sophisticated solutions such as Avro, Parquet, and ORC. These advanced formats were developed to address the shortcomings of earlier approaches, including poor storage efficiency, lack of native schema support, and inefficient performance in distributed environments. We pay special attention to the emergence of columnar storage formats, which have become increasingly popular due to their ability to significantly improve data access speeds and optimize analytical processing, particularly for large-scale datasets.

Our study delves into a comparative analysis of these modern serialization standards, evaluating their respective strengths and limitations within the context of big data ecosystems. We explore the complexities involved in selecting the most suitable file format for various applications, taking into account crucial factors such as data compression efficiency, query execution speed, and interoperability with diverse data processing frameworks. Furthermore, we investigate emerging trends in file format development, including the integration of transactional capabilities and adaptation to cloud-centric architectures, which are set to reshape the future of big data storage solutions. Through this

comprehensive exploration of big data file format evolution, we aim to provide actionable insights for data professionals and organizations striving to optimize their data management and analytical processes in an increasingly data-driven world.

When selecting a file format for large-scale data operations, several critical aspects must be evaluated. These considerations are influenced by the data's characteristics, storage requirements, and query patterns. Key factors to assess include:

- **Data Representation:** The choice between human-readable text formats and efficient binary encodings depends on the need for direct data access versus optimized performance.
- **Data Structure Support:** Effective formats should accommodate both simple data types and complex nested structures. Binary encoding can significantly reduce storage footprint[1].
- **Schema Management:** Some formats enforce data schemas, ensuring consistency and providing crucial metadata. The ability to evolve schemas is vital for long-term data management, particularly with dynamic datasets.
- **Storage Organization:** Row-oriented storage suits transactional systems, while column-oriented formats excel in analytical workloads by minimizing unnecessary I/O operations.
- **Distributed Processing Capability:** Formats that support data splitting facilitate parallel processing in distributed environments, enhancing performance for large-scale computations.
- **Data Compression:** Efficient compression reduces storage costs and accelerates data transfer. Column-based formats often achieve superior compression, especially for sparse datasets [1].
- **Processing Model Compatibility:** Different formats may be optimized for batch processing, real-time streaming, or both, influencing their suitability for specific use cases.
- **Technology Ecosystem Alignment:** The chosen format should integrate well with the existing data infrastructure and have robust community support.

By carefully weighing these factors, data professionals can identify the most appropriate file format to address their specific requirements in data processing, storage efficiency, and analytical performance. This decision is crucial for optimizing big data operations and ensuring scalability as data volumes continue to grow.

CSV/Text

CSV (Comma-Separated Values) is a popular format for tabular data storage, prized for its simplicity and widespread compatibility. It organizes data into comma-separated rows, facilitating easy data exchange across diverse systems. However, CSV's straightforward nature becomes a limitation in big data environments, where it struggles with complex data structures and scalability. Despite its enduring popularity, these shortcomings make CSV less suitable for sophisticated, large-scale data processing needs in modern analytics ecosystems.

Pros:

- **Simplicity:** Easy to read, create, and edit without specialized tools.
- **Universal compatibility:** Supported by a wide range of software and systems.

- Human-readable: Can be opened and understood with basic text editors.
- Lightweight: Small file size for basic data.
- Easy data exchange: Straightforward to import/export between different applications.

Cons:

- Limited data types: No native support for complex data structures or data type specification.
- No schema enforcement: Lacks built-in data validation or schema definition.
- Parsing issues: Can be problematic with special characters or multi-line fields.
- No compression: Files can become large for big datasets.
- Not splittable: Challenges in parallel processing for big data environments.
- No schema evolution: Difficult to change structure over time without manual updates.

Use Case: small datasets, Simple data Exchange, and Config files

Json

JSON (JavaScript Object Notation) is a versatile, human-readable data format that excels in representing structured information. It supports a variety of data types and nested structures, making it ideal for data exchange in web applications. Due to its simplicity and flexibility, JSON has become a standard choice for APIs, configuration files, and NoSQL databases.

Pros:

- Simplicity: Easy to read and write for both humans and machines
- Flexibility: Supports various data types and nested structures
- Wide adoption: Extensively used in web applications, APIs, and NoSQL databases
- Language-agnostic: Compatible with numerous programming languages
- Schema flexibility: Allows for easy addition of new fields without breaking existing applications

Cons:

- Performance limitations: Less efficient for large-scale data processing compared to binary formats
- Lack of native splitting: Challenges in parallel processing for big data environments
- No built-in indexing: Can be slower to read and process large datasets
- Storage inefficiency: Verbose nature can lead to larger file sizes compared to optimized formats
- Limited optimization features: Lacks built-in compression or columnar storage capabilities

Use Case: NoSQL storage, small datasets, API responses

Avro

Within the Apache Hadoop ecosystem, Avro stands out as a sophisticated data serialization system. This row-oriented format excels in efficiently packaging and storing information. Avro's distinctive feature lies in its dual approach: it employs a compact binary format for data storage while utilizing JSON for schema representation. This innovative pairing yields multiple benefits, including optimized storage

utilization, flexible schema adaptation, and seamless compatibility with a wide array of big data platforms. Such characteristics position Avro as a powerful tool for managing complex data structures in modern, large-scale data environments [2]. Avro works well with Kafka for efficient message serialization

Pros:

1. Efficient Serialization: Optimizes data storage and transfer in large-scale systems.
2. Flexibility: Supports schema evolution, crucial for dynamic data environments.
3. Distributed Processing: Enables parallel processing through data splitting capabilities.
4. Compression: Offers robust compression, reducing storage and transfer costs.
5. Versatility: Suitable for both batch and streaming applications.
6. Self-Describing: JSON schema accompanies data, ensuring consistent interpretation.

Cons:

1. Readability: Binary format challenges direct human interpretation.
2. Language Support: Not as universally integrated as some alternatives.

Use Case: Evolving Data Schema, Write Intensive applications, both batch and real time streaming

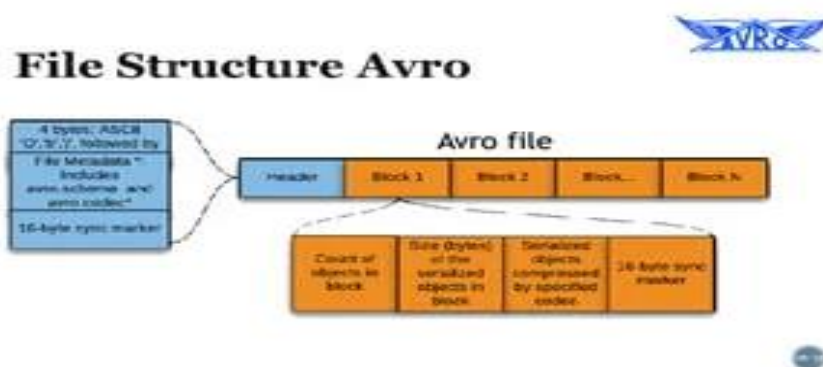


Fig 1. Avro File Structure [3]

ORC

ORC (Optimized Row Columnar) is an advanced data storage format designed for efficient handling of large-scale information. It organizes data in columns rather than rows, enabling rapid access and analysis of specific data types. ORC's innovative compression techniques significantly reduce storage requirements while maintaining quick read and write capabilities. Tailored for big data environments, ORC excels in processing massive datasets, packaging information into "stripes" that include both data and metadata. This format acts as a digital librarian, streamlining data storage, retrieval, and analysis in modern big data and analytics ecosystems.

Pros:

- Efficient Compression: Significantly reduces data size, lowering storage costs and improving I/O performance.

- OLAP Optimization: Excels in analytical workloads, enabling fast retrieval of specific columns from large datasets.
- Performance: Offers superior read/write speeds, particularly beneficial for data warehousing and big data systems.
- Batch Processing: Well-suited for large-scale data processing in distributed environments like Hadoop.
- Query Efficiency: Columnar structure accelerates complex analytical queries.

Cons:

- Limited Real-Time Updates: Doesn't support appending data to existing files, requiring full file recreation for updates.
- No Schema Evolution: Lacks support for schema changes without complete data rewrite.
- Tool Compatibility: Not universally supported across all big data tools, potentially limiting adoption.
- Static Data Focus: Better suited for analytics on relatively stable datasets rather than frequently changing data.
- Learning Curve: May require adaptation for teams more familiar with traditional row-based formats.

Use Case: Hadoop, Hive, Business Intelligence, OLAP workloads

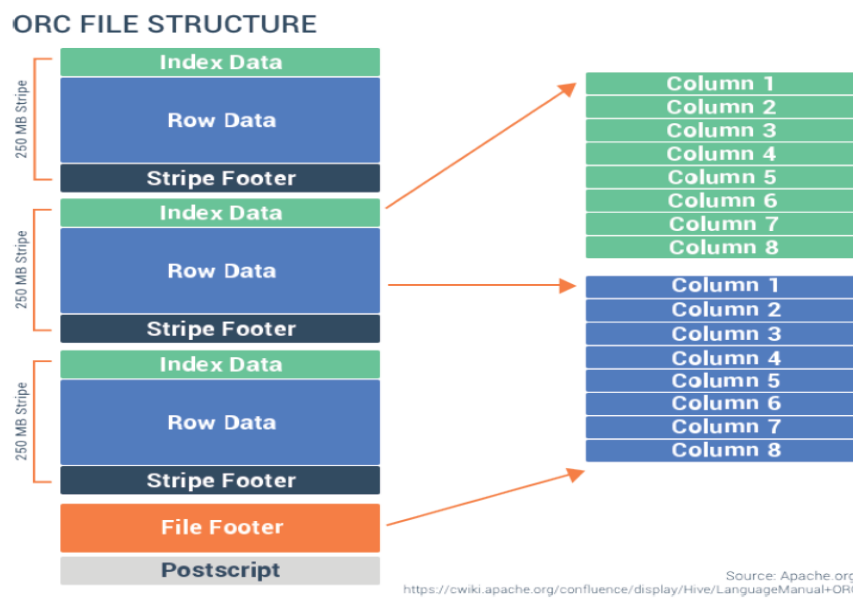


Fig 2. ORC File Structure

Parquet

Apache Parquet is a columnar storage format optimized for big data analytics. It efficiently compresses and encodes complex data structures, significantly reducing storage requirements. Parquet's design

accelerates analytical queries by allowing rapid access to specific data columns. It integrates seamlessly with the Hadoop ecosystem and other big data platforms. With embedded metadata and support for nested data types, Parquet offers a powerful solution for high-performance, large-scale data processing and analysis.

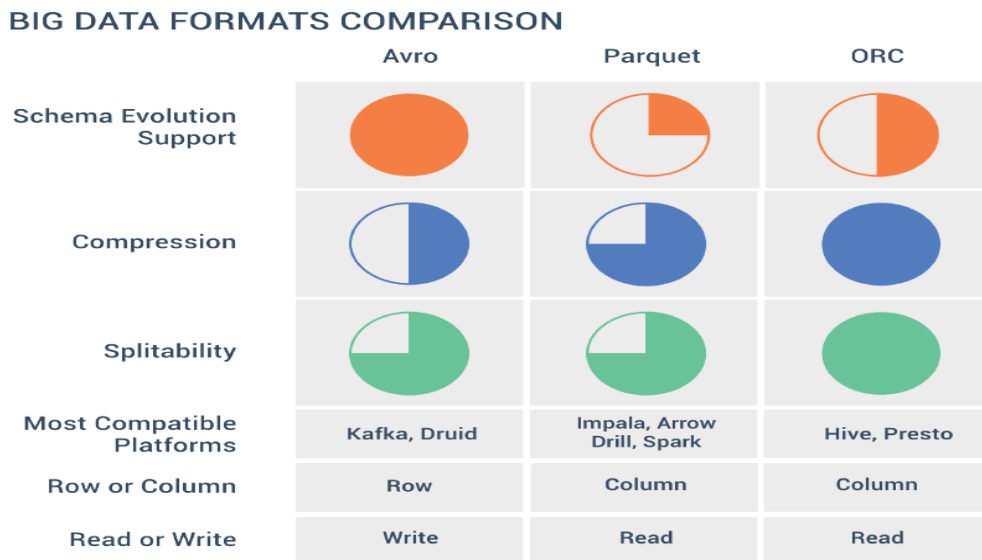
Pros:

- Efficient Columnar Storage: Enables better compression and faster analytical queries.
- Splittable Files: Supports parallel processing in distributed systems, enhancing performance.
- Schema Evolution: Allows data structure modifications without breaking compatibility.
- OLAP Optimization: Excels in complex analytical workloads.
- 5. Storage Efficiency: High compression ratios reduce storage costs.

Cons:

- Not Human-Readable: Binary format requires specialized tools for interpretation.
- Limited Real-Time Updates: Immutable files make frequent data updates challenging.
- Write-Once Model: Optimized for scenarios with infrequent writes and many reads.
- Complexity: May require additional expertise compared to simpler formats.
- Overhead for Small Datasets: Benefits may be less apparent for smaller data volumes.

Use Case: Large scale data analytics, Business Intelligence, Read Intensive Applications, best suited for Apache Spark



Source: Nexla analysis, April 2018

Fig 3. Comparison of ORC, Avro, Parquet [6]

Challenges

The implementation of advanced file formats in big data landscapes presents a multifaceted challenge for organizations. Key issues include identifying the most suitable format from a diverse range of

options, each offering distinct benefits. A critical concern is achieving a balance between analytical efficiency and the ability to handle varied data structures and evolving schemas. Technical teams often grapple with maintaining data consistency across different formats while ensuring seamless integration with an array of big data platforms. Another significant hurdle is optimizing data compression without compromising processing speed. Many organizations struggle to adapt traditionally batch-focused formats for near real-time analysis, a growing requirement in today's fast-paced data environments. Furthermore, the specialized knowledge required to effectively leverage these sophisticated formats often necessitates significant investment in team training and skill development. Successfully navigating these challenges is crucial for organizations aiming to fully harness the power of modern file formats in their big data infrastructures.

Conclusion

The evolution of specialized file formats in big data has transformed data storage and processing, progressing from basic structures to highly optimized solutions. These modern formats have revolutionized how organizations manage vast data volumes, significantly enhancing query efficiency, storage optimization, and system performance. This enables faster, more accurate insights and agile decision-making. However, the variety of available formats presents challenges, as there's no one-size-fits-all solution. The optimal choice depends on specific organizational needs, data characteristics, and analytical requirements, considering factors like query patterns and storage constraints. As big data continues to evolve, we expect further innovations in file format technology, particularly in real-time processing, schema adaptability, and performance optimization. Organizations that stay informed about these advancements and strategically select formats aligned with their needs will gain a competitive edge in the increasingly data-driven business landscape.

References

1. Aida Ngom, "Comparison of different file formats in Big Data", <https://www.adaltas.com/en/2020/07/23/benchmark-study-of-different-file-format/#consideration-to-be-taken-into-account-for-choosing-a-file-format>
2. Kapil Sreedharan, "Apache Avro Demystified", <https://kapilsreed.medium.com/apache-avro-demystified-66d80426c752>
3. AlirexaAlikhani, "Apache avro and overview hadooptools", <https://www.slideshare.net/slideshow/apache-avro-and-overview-hadoop-tools/99369894>
4. Rohan Karanjawala, "All You Need To Know About ORC File Structure In Depth", <https://www.linkedin.com/pulse/all-you-need-know-orc-file-structure-depth-rohan-karanjawala/>
5. Yacine El-Jarrari, "Data Factory: the advantages of a denormalised Apache Parquet storage", <https://blog.atinternet.com/en/data-factory-the-advantages-of-a-denormalised-apache-parquet-storage/>
6. Alex Woodie "Big Data File Formats Demystified", <https://www.datanami.com/2018/05/16/big-data-file-formats-demystified/>