International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V73I1P105

Volume 73 Issue 1, 44-50, January 2025

Review Article

Mastering Big Data Formats: ORC, Parquet, Avro, Iceberg, and the Strategy of Selection

Srinivasa Rao Nelluri[1], Flavia Ann Albert Saldanha[2]

[1]*Independent Researcher, Charlotte, NC, USA.*
[2]*Independent Researcher, Cincinnati, OH, USA.*

[1]*Corresponding Author : srinivasa.r.nelluri@gmail.com*

Received: 19 November 2024 Revised: 26 December 2024 Accepted: 14 January 2025 Published: 30 January 2025

Abstract - *In today's times, when data volumes are massive, and the speed of data is continuous, managing and optimizing such extremely large and complex datasets can be a huge ordeal for organizations. Optimizing storage costs and maintaining performance and efficiency becomes key, especially when dealing with Big Data datasets. When it comes to Big Data, it becomes extremely important for data teams to have the right data format and framework strategy from the get-go to be able to design and develop robust, efficient and sustainable processes around this data. A poor choice with data processing file formats could potentially hurt operational and/or analytical consumption, thereby leading to a low return on investment of this data. This paper explores the characteristics, advantages, and limitations of several prominent file formats in big data ecosystems: ORC, Parquet, Avro, Iceberg, and others. Each format is evaluated based on key criteria, including storage efficiency, query performance, schema evolution, and compatibility across platforms and analytical engines. By analyzing these formats in practical scenarios, this paper provides a decision matrix to guide data engineers, architects, and analysts in selecting the most suitable format based on their unique workload and infrastructure requirements. This comparative analysis ultimately serves as a strategic resource for organizations to make informed, efficient, and scalable choices in their big data environments.*

Keywords - *Apache Kafka, Apache Hadoop Distributed File System (HDFS), Apache Flink, Delta Lake, Snowflake.*

1. Introduction

Big data is one of the most influential technologies of the modern era. However, in order to support the maturity of big data systems, the development and sustenance of heterogeneous environments are required. In today's data-driven world, the data comes in large volumes and diverse formats from several different channels, such as customer interactions, IoT devices, social media, etc., which has necessitated highly efficient storage and processing solutions within big data ecosystems. As organizations seek to leverage vast datasets for competitive advantage, they must consider how to store, query, and manage data at scale. The choice of file format [1] plays a central role in these efforts, impacting factors such as data storage efficiency, query performance, compatibility with analytics platforms, and scalability across large datasets. Inadequate or suboptimal file format selection can lead to increased storage costs, slower processing times, and difficulty managing evolving data structures. While JSON and CSV files are still common for storing data, they were never designed for the massive scale of big data and tend to eat up resources unnecessarily (for example, JSON files with nested data can be very CPU-intensive). They are in text format and, therefore, human-readable. However, they lack the efficiencies offered by binary options. So, as data has grown, file formats have evolved. File format impacts speed and performance and can be a key factor in determining whether you must wait an hour for an answer – or milliseconds. Several file formats have been developed to meet different demands in big data processing. Columnar formats like Optimized Row Columnar (ORC) and Parquet provide high compression and fast analytical processing, making them popular choices for data warehousing and business intelligence applications. Avro, a row-based format, offers efficient data serialization and schema evolution capabilities, which makes it ideal for streaming data and real-time pipelines. Recently, newer table formats like Iceberg have emerged to address the needs of data lakes, adding support for features like partition evolution and ACID transactions, which improve data management and query consistency on cloud-based platforms. This paper presents how to choose the right file format for the respective use case. Also, comparative analysis of ORC, Parquet, Avro, Iceberg, and other prominent file formats, focusing on their performance characteristics, use case suitability, and flexibility within various big data architectures. The analysis also includes a

decision matrix summarizing optimal scenarios for each format, helping organizations make informed choices that enhance both performance and cost-effectiveness in their big data solutions. This introduction to file formats highlights the critical role of format selection in building scalable, efficient, and future-proof data pipelines and storage solutions in the big data landscape.

2. Background and Context

2.1. Row-Based Formats

Row-based format storage organizes data by rows. Each row stores a complete record, and each record includes all the fields (or columns) of that record. Row-based storage is optimized for transaction-oriented operations where you often need to access and modify entire records, such as inserting new customer details or updating an order. There are different types of row-based big data formats, and one such format considered for comparative analysis and review is the Avro file format.

2.1.1. Avro File Format

Apache Avro [2] is used for compact binary format as a data serialization standard. Typically, it is used to store persistent data related to communication protocols and HDFS. One of the major advantages of using Apache Avro is high ingestion performance, which is attributed to fast and lightweight deserialization and serialization. It is important to mention that Avro does not have an internal index. However, the directory-based partitioning technique available in HDFS can be used for facilitating random access of data. Data compression algorithms supported by Apache Avro.

2.2. Column-Based Formats

Column-based formats organize data by columns. Each column is stored separately, allowing the system to read or write specific columns independently. This format is common in data warehouses like Google Big Query and Amazon Redshift. Column-based storage is optimized for read-heavy operations and analytical queries, where you typically need to scan and aggregate data across many rows but only a few columns. Columns with similar data types can be highly compressed, reducing storage costs and improving read performance. Aggregations and calculations are faster since only the required columns are read, and the data is already organized in a columnar format. Apache ORC and Apache Parquet are the most popular and widely used file formats for Big Data analytics, and they share many common concepts in their internal design and structure. In this section, we will present the main aspects of columnar file formats in general and their purpose in optimizing query execution:

2.2.1. Optimized Row Columnar (ORC) File Format

Apache Optimized Record Columnar (ORC) is a self-describing (includes metadata), a type-aware columnar file format designed initially for Hadoop workloads but is now used as a general-purpose storage format. It is optimized for large streaming reads and has many advantages over its predecessor, the ORC file format [3]. The metadata in ORC is stored at the end of the file (after the file footer) using Protocol Buffers, providing the ability to add new fields to the table schema without breaking readers.

2.2.2. Parquet File Format

Apache Parquet is an open-source columnar storage format using complex nested data structures. It is a general-purpose storage format that can be used or integrated with any data processing framework or engine. Parquet supports efficient compression [3] and encoding schemas on a per-column level. At the end of the file, metadata describing the file structure are stored. File metadata contains references to all of the column chunk metadata start locations to easily access them. Furthermore, it allows us to immediately filter out columns not needed by the query.

2.3. Tabular-Based Formats

A tabular-based format in big data refers to a structured way of storing large datasets where information is organized into rows and columns. Each row represents a single record, and each column represents a specific attribute or feature of that record. The structure makes it simple to interpret and access data. With well-defined columns, data analysis tools can quickly filter and retrieve specific information. Most data analysis platforms and tools can readily handle tabular data. A very popular type in this category of formats is Apache Iceberg. It addresses these challenges by providing a robust and reliable table format that supports rollbacks and restores previous states of your data. This ensures data integrity and consistency, making data management more efficient and less error-prone.

2.3.1. Iceberg File Format

Apache Iceberg is an open-source table format purpose-built for addressing the limitations of traditional data lake architectures. Designed with scalability and flexibility in mind, Iceberg provides a robust framework for managing datasets in modern data lakes. Its features, such as ACID transactions, schema evolution, and data versioning, have made it a game-changer for organizations seeking to optimize their analytical workloads. Unlike earlier table formats like Apache Hive or Apache Hudi, Iceberg prioritizes performance and consistency, ensuring that analytical queries yield accurate and reliable results, even at scale.

3. Materials and Methods

This study employs a comparative methodology to evaluate and analyze several big data file formats, specifically ORC, Parquet, Avro, and Iceberg. The analysis is based on technical documentation, experimental testing within a controlled environment, and real-world use case studies.

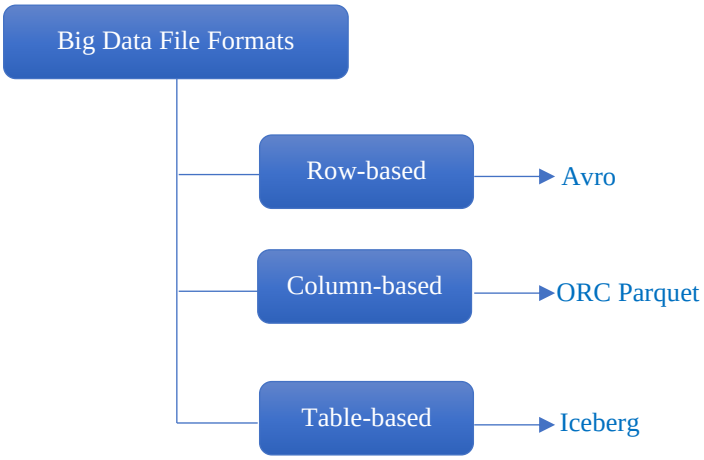

Fig. 1 Big data file format types

Each format is assessed based on predefined criteria, including storage efficiency, query performance, schema evolution capability, partitioning support, and compatibility with popular big data tools and platforms. The testing framework and methods are described below:

3.1. Data and Environment Setup

Data: A set of diverse datasets was selected to represent various real-world scenarios. These include structured, semi-structured, and unstructured data types, which simulate different data volumes and schema complexities typical in big data environments.

Environment: Tests were conducted using a distributed processing environment built on Apache Spark, Hive, and Hadoop clusters to ensure compatibility and performance under industry-standard conditions. For streaming and serialization tests, Apache Kafka was also integrated.

Infrastructure: All tests were run on cloud-based infrastructure using virtual machines configured with similar compute and storage specifications to maintain consistency across evaluations.

3.2. File Format Criteria

Each file format was evaluated based on the following key criteria:

Storage Efficiency and Compression: Compression ratio tests were conducted by converting the datasets into ORC, Parquet, Avro, and Iceberg formats and then measuring the storage footprint. Compression rates were calculated as a percentage reduction in storage size relative to the original data.

Query Performance: Performance was tested using common analytical queries (e.g., aggregation, filtering, sorting) to simulate real-world analytics workloads. The tests included:

Read/Write Speed: Measured by querying data [4] stored in each format with Spark SQL and Hive, assessing the time taken to perform both simple and complex queries.

Scalability: Tests were scaled across increasing data volumes to observe query performance under large dataset conditions.

Schema Evolution: Each format's ability to handle schema changes [5], such as adding or modifying columns, was evaluated. This included assessing how each format maintains compatibility with applications after schema modifications.

Partitioning and Data Pruning: Partitioning capabilities were tested by applying various partitioning strategies (e.g., based on date and region) and measuring query response time improvements. This helped identify formats that support efficient data pruning, which can significantly impact query performance.

Platform Compatibility: Compatibility was assessed by integrating each file format with big data processing tools (e.g., Spark, Hive, Flink, Kafka) and cloud storage services (e.g., AWS S3, Google Cloud Storage, Azure Data Lake). Compatibility metrics included ease of integration and support for features like ACID transactions and partition evolution.

3.3. Methods

Benchmarking: For each criterion, a benchmarking script was created using Python and Spark SQL to run identical queries on each format and record processing times and memory consumption. Data ingestion and transformation

times were also captured to measure each format's impact on end-to-end pipeline performance.

Compression and Storage Analysis: Compression tests involved [6] applying various encoding and compression techniques (e.g., Snappy, Zlib) to measure storage footprint and retrieval speeds across formats.

Schema Evolution Testing: Simulated schema changes were applied to datasets [7] stored in each format to test adaptability and backward compatibility. Avro and Iceberg, known for strong schema evolution support, were further tested in scenarios with frequently changing data structures.

Performance Metrics Collection: Performance data, including [8] CPU, memory usage, and I/O throughput, were collected during each test using monitoring tools. This data was analyzed to assess the resource efficiency of each file format under different workloads.

3.4. Data Analysis

The collected data was aggregated and analyzed to provide quantitative comparisons across formats. Average values for key metrics, such as query performance and storage efficiency, were calculated and presented as benchmark figures. Statistical analysis [9] was used to evaluate the significance of performance differences among the file formats under varying data volumes and query types. Results were visualized in bar charts and line graphs to highlight comparative performance and the trade-offs between formats.

3.5. Decision Matrix Development

Based on the results, a decision matrix was developed to summarize the optimal use cases for each file format. This matrix factors in data types, query patterns, workload requirements, and compatibility needs. The matrix [10] serves as a practical guide for data engineers and architects, helping them to select the most suitable file format for specific big data environments.

4. Results and Discussion

The comparative evaluation of ORC, Parquet, Avro, and Iceberg formats revealed notable differences in performance, storage efficiency, schema evolution capabilities, and compatibility with big data tools. These findings are presented in relation to each evaluation criterion and discussed with respect to typical use cases, highlighting where each format excels and where trade-offs exist.

4.1. Storage Efficiency and Compression

Results: ORC and Parquet, both columnar formats, demonstrated the highest compression ratios, reducing data storage by up to 70-80% on average for structured datasets. Avro, a row-based format, had a lower compression rate, averaging around 40-50%, while Iceberg's compression was similar to Parquet when used with columnar storage.

Discussion: The high compression achieved by ORC and Parquet makes them ideal for data warehousing, where storage efficiency and read performance are crucial [12]. Avro's lower compression is balanced by its lightweight storage and high write performance, making it more suitable for streaming and real-time processing. Iceberg, while capable of comparable compression, is primarily used for managing data lakes [13], where data versioning and complex partitioning provide additional value.

4.2. Query Performance

Results: Parquet and ORC significantly outperformed Avro in read-heavy analytical queries, particularly when handling aggregations and filtering on large datasets. Parquet showed a slight edge over ORC in Spark and Hive environments, with query times averaging 10-15% faster in complex analytical scenarios. Iceberg exhibited solid performance in scenarios that involved partitioning and time-travel queries, leveraging its metadata handling to improve query times.

Discussion: The columnar design of ORC and Parquet allows them to optimize for analytical workloads, where only relevant columns are accessed, reducing I/O overhead. Parquet's broad compatibility with tools like AWS Athena and Google BigQuery [14, 15] makes it particularly versatile for cross-platform analytics. Avro, with its row-based structure, is less efficient for these operations but ideal for fast sequential reads in streaming and ETL pipelines. Iceberg adds flexibility with its ability to handle complex partitioning, though its query performance depends on the underlying file format (e.g., Parquet or ORC).

4.3. Schema Evolution

Results: Avro and Iceberg exhibited the strongest schema evolution capabilities, handling additions and modifications to the schema without requiring data rewriting. Parquet and ORC offered limited schema evolution, mainly supporting the addition of new columns but not modifications or deletions.

Discussion: Avro's robust schema evolution makes it a preferred choice for Kafka-based streaming [16] and environments where schema changes are frequent. Iceberg further enhances schema evolution by supporting not only schema modifications but also partition evolution, which allows data engineers to adapt partitioning schemes as business needs evolve. This makes Iceberg particularly

valuable in dynamic, large-scale data lakes where schema and partitioning changes are common. The limited schema evolution capabilities in ORC and Parquet suit environments with stable schemas, such as data warehousing.

4.4. Partitioning and Data Pruning

Results: Iceberg's support for partition evolution and efficient data pruning made it superior in managing large, partitioned datasets in data lakes. Parquet also showed effective data pruning with predicate pushdown, especially when queries focused on specific partitions.

Discussion: Partitioning support is a critical factor in improving query performance on large datasets. Iceberg's advanced partition handling allows for more complex data pruning, which is especially beneficial in cloud data lakes where scalability is essential. Parquet also performs well with partitioned data, especially in environments where predicate filtering is common. ORC supports partitioning effectively within Hadoop-based ecosystems but lacks the partition flexibility seen in Iceberg. Avro does not support partitioning natively, limiting its use to scenarios that require efficient, partition-based data access.

4.5. Platform Compatibility

Results: Parquet proved to be the most widely compatible format, with support across Spark, Hive, AWS Athena, Google BigQuery, and other big data tools. [17] ORC is highly optimized for Hadoop ecosystems but has limited support in non-Hadoop tools. Avro is well-suited for streaming platforms like Kafka but has fewer integrations with analytics tools. Iceberg, as a table format rather than a file format, works with multiple engines (Spark, Flink, Trino) and supports multiple underlying formats.

Discussion: Parquet's compatibility across cloud platforms and analytics tools makes it highly versatile for multi-platform architectures, supporting data interoperability in diverse big data environments. ORC's compatibility is narrower, making it ideal for Hadoop-based infrastructures but less suited for cloud-native systems. Avro's strength lies in streaming applications, as it is designed for efficient serialization and is widely used in Kafka pipelines [18]. Iceberg's flexibility in data lakes allows organizations to standardize on a single format across different processing engines, making it a future-proof choice for organizations moving towards cloud-native and multi-engine analytics [19].

4.6. Summary of Key Findings

The results suggest that no single file format is optimal for all scenarios; instead, each format has distinct strengths that make it suitable for specific big data needs:

- ORC is optimal for Hadoop-based data warehousing and ETL processes requiring high compression and analytical performance.
- Parquet is versatile and suitable for cross-platform data lakes, especially in cloud-native analytics environments.
- Avro excels in real-time and streaming applications due to its schema evolution support and efficient row-based storage.
- Iceberg is designed for large-scale data lake management, supporting complex partitioning, time travel, and schema evolution.

5. Conclusion

This comparative analysis demonstrates that the choice of file format depends on the specific requirements of the big data application. Organizations should consider factors such as data processing patterns, platform compatibility, and long-term data management needs when selecting a format. As data lakes and cloud-native architectures evolve, newer formats like Iceberg that support rich metadata and flexible partitioning may become increasingly important. The decision matrix developed in this study provides a practical guide for data engineers, architects, and analysts to select the right file format for their unique workloads, ensuring efficiency, scalability, and cost-effectiveness in big data processing.

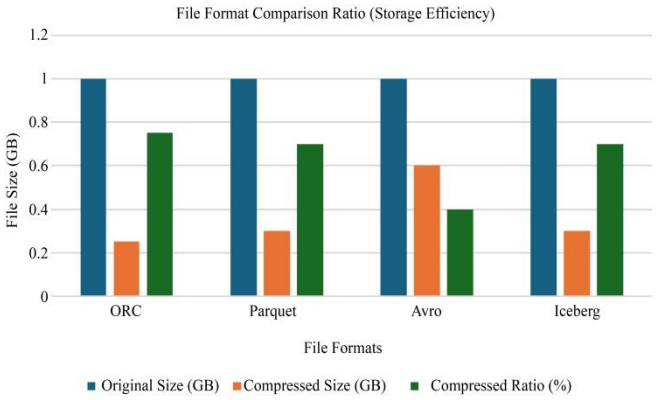

Fig. 2 Compression ratio (storage efficiency)

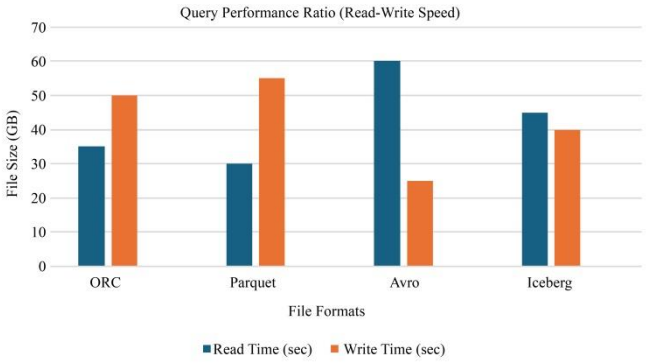

Fig. 3 Read-write speed compare analysis

Table 1. Decision matrix for big data file compare study

Criteria	ORC	Parquet	Avro	Iceberg
Storage Efficiency & Compression	High compression (70-80%) for structured datasets	High compression (70-80%) for structured datasets	Moderate compression (40-50%)	Comparable to Parquet with columnar storage
Best Use Case	Data warehousing, ETL processes	Data warehousing, cross-platform analytics	Real-time streaming, event logging	Data lakes, versioning, and partitioning
Query Performance	Excellent for analytical queries, but slower than Parquet in Spark/Hive	Best for cross-platform analytics, faster in Spark/Hive	Less efficient for analytical queries, optimized for sequential reads	Solid performance with partitioning and time-travel queries
Best Use Case	Analytical workloads, large datasets	Cross-platform analytics, cloud-native environments	Streaming, real-time data processing	Complex partitioning and time-travel queries in data lakes
Schema Evolution	Limited to adding columns	Limited to adding columns	Excellent schema evolution (additions, modifications)	Strong schema and partition evolution, ideal for dynamic environments
Best Use Case	Stable schemas, data warehousing	Stable schemas, cloud-native analytics	Real-time data streams, Kafka-based systems	Dynamic data lakes, evolving schemas and partitions
Partitioning & Data Pruning	Effective within Hadoop ecosystems, limited flexibility	Effective with predicate pushdown, supports partition pruning	No native support for partitioning	Advanced partition evolution and data pruning, ideal for large-scale data lakes
Best Use Case	Hadoop-based ecosystems with partitioned data	Partitioned data in cloud-native environments	Not suitable for partitioning scenarios	Large-scale data lakes requiring advanced partitioning and pruning
Platform Compatibility	Optimized for Hadoop-based systems	Highly compatible across cloud platforms and big data tools	Best for streaming platforms (e.g., Kafka)	Supports multiple engines (Spark, Flink, Trino) and multiple underlying formats
Best Use Case	Hadoop ecosystems	Multi-platform big data analytics, cloud-native environments	Streaming and real-time data pipelines	Multi-engine data lakes, cloud-native environments
Overall Strength	Ideal for Hadoop-based data warehousing	Versatile, cloud-native, cross-platform analytics	Best for real-time data streaming and schema evolution	Best for large-scale data lake management, schema evolution, and partitioning

References

[1] Seref Sagiroglu, and Duygu Sinanc, "Big Data: A Review," *2013 International Conference on Collaboration Technologies and Systems*, San Diego, CA, USA, pp. 42-47, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[2] Samiya Khan, and Mansaf Alam, "File Formats for Big Data Storage Systems," *International Journal of Engineering and Advanced Technology*, vol. 9, no. 1, pp. 1-7, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[3] Todor Ivanov, and Matteo Pergolesi, "The Impact of Columnar File Formats on SQL-on-Hadoop Engine Performance: A Study on ORC and Parquet," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 5, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] Vishal Naidu, "Performance Enhancement Using Appropriate File Formats in Big Data Hadoop Ecosystem," *International Research Journal of Engineering and Technology*, vol. 9, no. 1, pp. 1247-1251, 2022. [Google Scholar] [Publisher Link]

[5] Ibrar Yaqoob et al., "Big Data: From Beginning to Future," *International Journal of Information Management*, vol. 36, no. 6, pp. 1231-1247, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[6] Amanpreet Kaur Sandhu, "Big Data with Cloud Computing: Discussions and Challenges," *Big Data Mining and Analytics*, vol. 5, no. 1, pp. 32-40, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Akram Elomari, Larbi Hassouni, and Abderrahim Maizate, "The Main Characteristics of Five Distributed File Systems Required for Big Data: A Comparative Study," *Advances in Science, Technology and Engineering Systems Journal*, vol. 2, no. 4, pp. 78-91, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[8] Spyros Blanas et al., "Parallel Data Analysis Directly on Scientific File Formats," *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, Snowbird Utah USA, pp. 385-396, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[9] Software: Landmark Solution, Halliburton. [Online]. Available: https://www.halliburton.com/en/software

[10] Eileen McNulty, Understanding Big Data: The Seven Vs, 2014. [Online]. Available: https://dataconomy.com/2014/05/22/seven-vs-big-data/

[11] Thomas H. Davenport, and Jill Dyche, "Big Data in Big Companies," *International Institute for Analytics*, 2013. [Google Scholar]

[12] James Manyika et al., "Big Data: The Next Frontier for Innovation, Competition, and Productivity," *Mickensy Global Institute*, 2011.[Google Scholar] [Publisher Link]

[13] Avita Katal, Mohammad Wazid, and R.H. Goudar, "Big Data: Issues, Challenges, Tools and Good Practices," *2013 Sixth International Conference on Contemporary Computing*, Noida, India, pp. 404-409, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[14] Xin Luna Dong, and Divesh Srivastava, "Big Data Integration," *2013 IEEE 29th International Conference on Data Engineering*, Brisbane, QLD, Australia, pp. 1245-1248, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[15] Firat Tekiner, and John A. Keane, "Big Data Framework," *2013 IEEE International Conference on Systems, Man, and Cybernetics*, Manchester, UK, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[16] Chun-Wei Tsai et al., "Big Data Analytics: A Survey," *Journal of Big Data*, vol. 2, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[17] Uthayasankar Sivarajah et al., "Critical Analysis of Big Data Challenges and Analytical Methods," *Journal of Business Research*, vol. 70, pp. 263-286, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[18] Xiaolong Jin et al., "Significance and Challenges of Big Data Research," *Big Data Research*, vol. 2, no. 2, pp. 59-64, 2015. [CrossRef] [Google Scholar] [Publisher Link]