World Journal of Advanced Research and Reviews

eISSN: 2581-9615 CODEN (USA): WJARAI

Cross Ref DOI: 10.30574/wjarr

Journal homepage: https://wjarr.com/

(Review Article)

A comparative analysis of big data processing paradigms: Mapreduce vs. apache spark

Sifat Ibtisum [1, *], Ehsan Bazgir [2], S M Atikur Rahman [3] and S. M. Saokat Hossain [4]

[1] *Department of Computer Science, Missouri University of Science and Technology, Missouri, USA.*
[2] *Department of Electrical Engineering, San Francisco Bay University, Fremont, CA 94539, USA.*
[3] *Department of Industrial, Manufacturing and Systems Engineering, University of Texas at El Paso, TX 79968, USA.*
[4] *Department of Computer Science, Jahangirnagar University, Dhaka, Bangladesh.*

World Journal of Advanced Research and Reviews, 2023, 20(01), 1089–1098

Publication history: Received on 16 September 2023; revised on 24 October 2023; accepted on 27 October 2023

Article DOI: https://doi.org/10.30574/wjarr.2023.20.1.2174

Abstract

The paper addresses a highly relevant and contemporary topic in the field of data processing. Big data is a crucial aspect of modern computing, and the choice of processing framework can significantly impact performance and efficiency. The technical revolution of big data has changed how organizations handle and value large databases. As data quantities expand quickly, effective and scalable data processing systems are essential. MapReduce and Apache Spark are two of the most popular large data processing techniques. This study compares these two frameworks to determine their merits, shortcomings, and applicability for big data applications. Nearly quintillion bytes of data are created daily. Approximately 90% of data was produced in the previous two years. At this stage, data comes from temperature sensors, social media, movies, photographs, transaction records (like banking records), mobile phone conversations, GPS signals, etc. In this article, all key big data technologies are introduced. This document compares all big data technologies and discusses their merits and downsides. Run trials using multiple data sets of varying sizes to validate and explain the study. Graphical depiction shows how one tool outperforms others for given data. Big Data is data generated by the rapid usage of the internet, sensors, and heavy machinery, with great volume, velocity, variety, and veracity. Numbers, photos, videos, and text are omnipresent in every sector. Due to the pace and amount of data generation, the computing system struggles to manage large data. Data is stored in a distributed architectural file system due to its size and complexity. Big distributed file systems, which must be fault-tolerant, adaptable, and scalable, make complicated data analysis dangerous and time-consuming. Big data collection is called 'datafication'. Big data is 'datafied' for productivity. Organisation alone does not make Big Data valuable; we must choose what we can do with it.

Keywords: SparkR; Spark Core; Apache Spark; MapReduce; Graph X

1. Introduction

The phrase "Big Data" refers to a compilation of data sets that are of such significant size and complexity that they pose challenges for processing using traditional data mining techniques and tools. The primary objective of big data analytics is to extract valuable insights from vast datasets and convert them into a comprehensible format for subsequent use. The primary components of big data encompass the activities of acquisition, organisation, retention, exploration, dissemination, transmission, examination, and representation.

In recent times, there has been a significant surge in the recognition of the significance of this particular domain. This is mostly due to its ability to provide organizations with valuable information and enhanced comprehension of both organised and unorganised data. Consequently, this may potentially result in improved decision-making processes that are based on a more comprehensive understanding of the subject matter. Within the realm of business, big data analytics

* Corresponding author: Sifat Ibtisum

refers to the systematic examination of extensive datasets, sometimes referred to as "big data," with the objective of revealing concealed patterns, unexplored relationships, market tendencies, client inclinations, and other pertinent business insights. The convergence of contemporary technological advancements and the latest progressions in data analytics algorithms and methodologies has facilitated the utilisation of big data analytics by organizations. Several significant challenges arise when attempting to effectively use big data analytics. These challenges encompass data quality, storage, visualisation, and processing. Several corporate examples of big data include social media material, mobile phone details, transactional data, health records, financial papers, Internet of things, and weather information. In order to extract relevant judgements from a large dataset, it is imperative to employ effective processing control, analytical capabilities, and talents [1-4]. The accurate selection of data within a bigger dataset is crucial for the analysis of big data, as it encompasses a significant volume of data. Various firms provide predictive analytics and data mining solutions for corporations, such as Predictive Analytics Suit, IBM SPSS Statistics, and Microsoft Dynamics CRM Analytics Foundation. The primary objective of software deployed on big data platforms and utilised for big data analytics is to facilitate the efficient analysis of vast datasets. Big Data analytics (BDA) is expected to have a significant influence on several industries, including banking, vehicles, healthcare, telecom, government, transportation, and travel.

According to a study conducted by IDC Digital Universe and released in 2011, it was reported that around 130 Exabytes of data were generated and stored in the year 2005. The aforementioned quantity had a significant increase to 1,227 Exabytes in the year 2010, and it was anticipated to exhibit a growth of 45.2% to reach 7,910 Exabytes by the year 2015. The use of this dataset has the potential to uncover valuable information that is now hidden inside it. In 2004, the company "Google" unveiled MapReduce, a technology that subsequently served as the basis for Hadoop and other comparable methodologies. The first objective of Hadoop was to create an index of the whole World Wide Web (WWW). Presently, several organizations are utilising the open-source Hadoop technology to efficiently process substantial amounts of data. The use of Big Data for the purpose of obtaining commercial value and gaining a competitive edge is a widespread global trend that is expected to persist and expand, hence presenting an array of connected opportunities [5-8]. According to a research study conducted by MGI (McKinsey Global Institute) and McKinsey's Business Technology Office, the effective use of extensive data sets is projected to become a vital element in the success and expansion of enterprises. It is expected to contribute to enhancing consumer benefits, boosting production, and fostering innovation. Leveraging Big Data involves making thoughtful choices in selecting an appropriate Big Data analytics platform and tools, which holds substantial significance for any organization. This document serves as a comprehensive guide to the field of Big Data. The provided content encompasses comprehensive information on big data, encompassing its defining features and classifications, the concept of the 5 Vs (Volume, Velocity, Variety, Veracity, and Value), diverse data formats, the significance of processing big data, the wide-ranging uses of big data, as well as an overview of the many analytical tools employed for dataset analysis in the context of big data. Furthermore, a series of experiments were conducted utilising diverse datasets in order to establish the superiority of certain tools over others. This study presents a comparative analysis of data interpretation in order to enhance comprehension of the subject matter. The objective of this article is to offer a short and comprehensive resource on the fundamental aspects of Apache Spark, with the aim of assisting individuals in bridging the gap and facilitating their initiation into the utilisation of Apache Spark, as well as their engagement with this dynamic project. Our primary focus is on the utilisation of Apache Spark to facilitate the execution of efficient large-scale ML, graph analysis, and stream processing tasks. Apache Spark utilises in-memory processing for data, whereas Hadoop MapReduce stores data on disc following a map or reduce operation.

2. Apache spark

Apache Spark is a robust platform for processing large volumes of data, known for its ability to seamlessly integrate hybrid frameworks. A hybrid architecture provides assistance for the concurrent use of batch and stream processing functionalities. Despite sharing many ideas with Hadoop's MapReduce engine, Spark demonstrates superior performance compared to the latter. For example, when comparing the performance of Spark and MapReduce in handling a batch processing workload, it has been seen that Spark can exhibit shorter processing times. This can be attributed to Spark's use of the "full in-memory computation" feature, which allows for computations to be performed entirely in memory. In contrast, MapReduce relies on the conventional approach of reading data from disc and writing results back to disc. Spark has the capability to operate alone or it may be integrated with Hadoop in order to supplant the MapReduce framework.

The primary benefit of the Spark batch processing model in comparison to MapReduce is its ability to perform computations in-memory. The interaction between Spark and the disc is limited to two specific tasks: the initial loading of data into memory and the subsequent storage of results back into memory. All intermediate findings are handled in memory. The utilisation of in-memory processing in Spark results in a notable increase in speed compared to its rival batch processing framework, Hadoop. In addition, the implementation of holistic optimisation in Spark further enhances its computational efficiency by allowing for the analysis of a whole set of tasks in advance. The accomplishment of this task involves the generation of Directed Acyclic Graphs (DAGs).

The Spark Stream Processing Model encompasses the capability of performing stream processing tasks alongside batch processing operations by utilising micro-batches. Micro-batching involves the grouping of data streams into tiny batches, which are then processed as regular tasks by the Spark batch engine. While the micro-batching technique demonstrates effectiveness, it may nevertheless result in performance disparities compared to authentic stream processing frameworks.

The process of executing a Spark application has five fundamental components: a driver program, a cluster manager, workers, executors, and tasks. A driver program refers to an application that utilizes Spark as a library and establishes a high-level control flow for the intended computation. The worker in a Spark application is responsible for providing CPU, memory, and storage resources. On the other hand, an executor refers to a Java Virtual Machine (JVM) process that Spark generates on each worker specifically for that application. A job refers to a collection of calculations, such as a data processing method, that are executed by Spark on a cluster in order to get results for the driver application. A Spark application has the capability to initiate and execute many tasks. The Spark framework divides a computational job into a directed acyclic graph (DAG) consisting of stages, with each stage including a set of jobs. A task refers to the most basic and indivisible item of work that is dispatched by Spark to an executor. The primary means through which the driver application interacts with Spark is through a SparkContext, which serves as the principal entry point for accessing Spark functions. A SparkContext is a representation of a connection established with a computational cluster [7].

2.1. RDD

The Spark core framework is constructed based on the abstraction of Resilient Distributed Datasets (RDDs) [8]. A RDD is a collection of records that is partitioned and immutable. RDDs offer fault-tolerant and parallel data structures that enable users to explicitly store data on disk or in memory. Users may also exercise control over the partitioning of data and manipulate it using a diverse range of operators [9]. Efficient data sharing between calculations is a crucial necessity for many workloads. An RDD can be generated through the utilization of either external data sources or existing RDDs.

RDD, which stands for Resilient Distributed Datasets, serves as a fault-tolerant distributed memory abstraction. It does this by avoiding data replication and instead maintaining a graph of activities, known as an RDD's lineage (as seen in Figure 1), that were utilized in its construction. The system is capable of effectively recalculating missing data in the event of a failure. In order to ensure consistency between iterations, it is possible to exert control over the partitions of a Resilient Distributed Dataset (RDD). This may be achieved through the utilization of Spark core, which enables the co-partitioning of RDDs and the co-scheduling of jobs, hence minimizing the need for data migration. In order to prevent redundant processing, it is necessary for the application to explicitly cache Resilient Distributed Datasets (RDDs) when they are expected to be utilized many times.

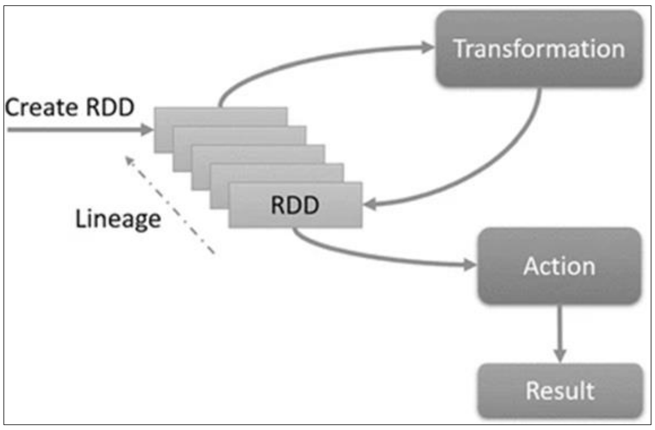

Figure 1 Evaluation of RDD (image adapted from: http://www.slideshare.net/GirishKhanzode/apache-spark-core)

2.2. Spark SQL

Spark SQL, previously referred to as Shark, Spark SQL is a distributed computational platform designed to operate on both structured and semi-structured datasets. The platform enables the use of analytical and interactive applications for real-time and past data, sourced from diverse formats like JSON, Parquet, and Hive tables.

Spark Streaming is a framework that facilitates the real-time processing of streaming data. To facilitate streaming analysis, Spark streaming improves the rapid scheduling functionality of Apache Spark by partitioning data into smaller batches. The last step involves the implementation of a process referred to as transformation, which is executed on the aforementioned subsets of data that may be readily acquired from live streams and other data sources, including but not limited to Twitter, Apache Kafka, IoT sensors, and Amazon Kinesis.

MLlib is a software library that offers efficient algorithms characterised by their superior performance and rapid execution. It facilitates the utilisation and expansion of machine learning techniques by providing a user-friendly interface and the ability to handle large-scale data. There exist several machine learning methods, including regression, classification, clustering, and linear algebra. Additionally, it offers a library that caters to lower-level machine learning primitives, such as the generic gradient descent optimisation technique. Additionally, it offers several functionalities including model assessment and data importation. This functionality is applicable in programming languages such as Java, Scala, and Python.

2.3. GraphX

GraphX is optimised for distributed computation, rendering it well-suited for managing graphs of considerable magnitude that surpass the memory capacity of a single machine. By capitalising on the distributed characteristics of Spark, it efficiently executes operations on graph data across a cluster of processors.

GraphX offers programming interfaces (APIs) that facilitate the construction of directed graphs from distributed data collections. It enables the representation of data as vertices and edges, which can then be used to generate graphs. In addition to transportation and social networks, these graphs have the capability of simulating a vast array of real-world systems. This facilitates a wide range of graph operations, such as filtering, mapping, joining, and aggregating, which enable the execution of intricate transformations on graph data. By utilising these operations, one can modify attributes of nodes and edges, exclude nodes or edges according to particular criteria, or generate completely new graphs from pre-existing ones. This offers an extensive assortment of pre-installed libraries and algorithms designed to facilitate routine graph-related endeavours. These consist of algorithms designed to determine the shortest paths, compute connected components, and conduct graph analytics, among other tasks. By utilising these algorithms, users are not required to implement them manually.

2.4. Spark Core

The Spark core serves as the foundation for several features inside Apache Spark. The platform offers a wide array of application programming interfaces (APIs) and software tools for several programming languages, including Scala, Java, and Python. These APIs are designed to enhance the efficiency and convenience of software development processes. The implementation of in-memory processing in Spark core is aimed at enhancing computational performance and addressing the limitations associated with MapReduce.

2.5. SparkR

SparkR is a software package designed for the R programming language, which facilitates the use of Spark's capabilities within the R shell. The Data Frame serves as the core data structure for data processing in the R programming language. In a similar vein, the SparkR Data Frame functions as the fundamental unit inside the SparkR framework. The tool has the capability to execute a range of operations on extensive datasets, including but not limited to selection, filtering, and aggregation.

3. Comparison Between Hadoop and Spark

Hadoop [12] a well-recognised and valuable open-source software system that facilitates distributed storage, allowing for the storage of substantial volumes of enormous datasets across clusters. The architecture of the system allows for seamless expansion from a singular server to a multitude of nodes. Hadoop is capable of parallel processing of extensive datasets, leading to efficient and prompt outcomes. Hadoop consists of two primary components, namely the Hadoop Distributed File System (HDFS) and MapReduce.

The Hadoop Distributed File System (HDFS) partitions files into smaller units known as blocks and distributes them among several nodes for storage [13]. Hadoop Distributed File System (HDFS) consists of two types of nodes: data-nodes, also known as worker nodes, and name-nodes, sometimes referred to as master nodes [14, 15]. All actions, namely deletion, reading, and writing, are dependent on the utilisation of these two distinct sorts of nodes. The workflow of the Hadoop Distributed File System (HDFS) can be described as follows: Firstly, the name-node requests access permission. If the acceptance of the proposal is granted, the file name will be transformed into a comprehensive compilation of HDFS block identifiers. This compilation will encompass the files themselves, as well as the specific data-nodes responsible for storing the blocks associated with each respective file. Subsequently, the list of identification (ID) will be transmitted back to the client, enabling users to perform subsequent activities based on the received information.

The computing system known as MapReduce [16] encompasses two fundamental functions, namely Mappers and Reducers. The mappers will execute the map function to process the files and convert them into novel key-value pairs [17]. Subsequently, the newly generated key-value pairs are allocated to distinct partitions and arranged in ascending order according to their respective keys. The inclusion of a combiner is discretionary and may be identified as a local reduction operation that facilitates the pre-reduction of values with identical keys, hence mitigating the burden on input/output operations. Ultimately, the intermediate key-value pairs will be partitioned into distinct segments and subsequently transmitted to a reducer. The implementation of MapReduce necessitates the inclusion of a single operation known as shuffle. The term "shuffle" refers to the process of transporting the data produced by the mapper to the appropriate reducer. Upon completion of the shuffle process, the reducer initiates a series of copy threads known as Fetchers, which are responsible for retrieving the output files of the map job over the HTTP protocol [18-19]. The subsequent stage involves the consolidation of the generated output into distinct final files, which are subsequently identified as the input data for the reduction. Subsequently, the reducer undertakes the processing of the data in accordance with the reduced function, subsequently returning the output to the Hadoop Distributed File System (HDFS).

4. MapReduce

The programming paradigm and parallel processing framework known as MapReduce has significantly transformed the methodologies employed in large-scale data processing and data mining. The concept was initially developed by Google during the early 2000s and subsequently gained popularity through its adoption by the Hadoop open-source project. MapReduce is a key idea within the field of big data analytics and has significant importance in the execution of data mining operations. The MapReduce framework is a data processing approach that involves the partitioning of a data mining operation into two distinct phases: Map and Reduce. The system is specifically engineered to manage distributed and parallel computing over a cluster of standard, affordable hardware components.

The following is a concise exposition of the operational principles behind the MapReduce computational framework:

During the map step, the input data is partitioned into smaller segments called splits. The splits are processed individually by a group of worker nodes, which are computers inside the cluster. The data received by each worker node is subjected to a "map" function, resulting in the generation of a collection of key-value pairs. The result of the map phase consists of a set of intermediate key-value pairs [10, 11].

Following the completion of the map phase, the intermediate key-value pairs undergo a process of shuffling and sorting based on their respective keys. Ensuring the aggregation of all values corresponding to a certain key in close proximity is a critical measure, facilitating their seamless transition to the subsequent stage.

During the reduction phase, a separate group of worker nodes is responsible for taking the sorted and grouped key-value pairs and applying a "reduce" function to them. The reduction function is responsible for processing all values associated with a certain key and generating a collection of output values.

The outcome of the reduction step is commonly saved in a distributed file system or utilised for subsequent analysis, reporting, or other data-related activities. MapReduce is well recognised for its inherent simplicity and scalability, rendering it highly appropriate for the efficient processing of extensive datasets over expansive clusters of standard hardware. The abstraction provided by this approach simplifies several aspects of distributed computing, including data partitioning, distribution, and fault tolerance. Consequently, developers may direct their attention on crafting map and reduce functions that are specifically designed to meet their data processing requirements [12].

Although MapReduce has provided a fundamental framework for the processing of large-scale data, it possesses several limitations, namely in relation to iterative and interactive processing. This is where contemporary frameworks such as Apache Spark become relevant, since they provide enhanced performance and a more adaptable programming paradigm for large-scale data processing. However, MapReduce continues to be a fundamental idea and is employed in specific scenarios and contexts where the emphasis is on simplicity and dependability.

5. Spark vs MapReduce: Which is faster?

Big data analytics is currently a very active and dynamic field of study, with many challenges and a pressing need for ground-breaking discoveries with broad implications across numerous industries. To meet the computational requirements of large-scale data analysis, it is crucial to have an efficient framework for the design, implementation, and administration of the required pipelines and algorithms. Apache Spark has evolved into a comprehensive engine for large-scale data analysis across a variety of use cases. In the field of data science and engineering, a novel methodology has been proposed whereby a variety of data challenges can be effectively addressed by combining a single processing engine with general-purpose programming languages. Due to its powerful programming style, which facilitates rapid and scalable processing, Apache Spark has acquired widespread use in both academic and industrial contexts. The aforementioned project has emerged as the most actively pursued open-source initiative in the field of big data and has gained considerable prominence within the Apache Software Foundation as one of its most vigorously pursued initiatives. In the domain of big data, acquiring credible references is essential for maximising Apache Spark's benefits and making significant contributions to its development. The official programming guide is the primary and most up-to-date reference for Apache Spark, containing comprehensive usage information. In addition, a number of published articles demonstrate the practical application of Apache Spark in addressing big data-related challenges. Moreover, Databricks, the company founded by the creators of Apache Spark, has created a collection of illustrative applications to demonstrate the diverse uses of Apache Spark for various workloads. The official Databricks and Spark Hub blogs, which offer a comprehensive compilation of Spark-related news, events, tools, and other pertinent content, are additional reliable sources. Notwithstanding, the rapid adoption and development of Apache Spark, as well as the expanding scholarly investigation of its application in big data analytics, make it difficult for novices to comprehend its expansive growth and research endeavours. Regarding current scholastic knowledge, there is a lack of a comprehensive synthesis regarding the implementation of Apache Spark in the field of big data analytics. According to theoretical analysis, it is anticipated that Spark will exhibit superior performance compared to Hadoop MapReduce. Spark offers graph processing functionality and integrates the MLlib machine learning library in addition to its primary data processing capabilities. Spark's extraordinary performance capabilities allow it to perform both real-time and bulk processing operations. Hadoop MapReduce is ideally suited for bulk processing applications. If the client desires a real-time solution, an alternative platform, such as Impala or Apache Storm, must be utilised. For graph processing purposes, Apache Graph can be employed. Apache Mahout was formerly used by the MapReduce framework for machine learning; however, Spark has since superseded it. The Spark framework requires a substantial quantity of memory. Spark, similar to conventional databases, stores computational tasks in memory for cache purposes until they are expressly removed. Spark's performance may decrease when executed on Hadoop YARN alongside other resource-intensive services or when working with datasets that exceed the available memory capacity. MapReduce, on the other hand, terminates its processes upon the completion of a task, allowing it to coexist with other services with minimal performance differences. Spark is a framework for high-performance distributed computation that is renowned for its speed and efficiency. Apache Spark performs significantly better than Hadoop, with a 100-fold increase in efficiency when operating in memory and a 10-fold increase when operating on disc. Spark enables the storage of transitory data by decreasing the frequency of disc read/write cycles and utilising in-memory storage. MapReduce's reliance on disc I/O for receiving and writing data diminishes processing performance due to the inherent latency of these operations. In this investigation, an experiment will be conducted to demonstrate that Spark is faster than MapReduce.

Big data processing frameworks like Apache Spark and MapReduce are increasingly vital in healthcare due to the exponential growth of medical data. These technologies offer efficient and scalable ways to manage, analyze, and derive valuable insights from large and complex healthcare datasets. In healthcare, the volume of data is staggering, encompassing electronic health records (EHRs), medical imaging, genomics, patient monitoring, and more. MapReduce and Apache Spark excel in processing such data through parallelism, distributing tasks across clusters of machines for expedited processing.

MapReduce, pioneered by Google, is well-suited for batch processing tasks. It segments data into smaller chunks, processes them in parallel, and then aggregates the results. It's instrumental in tasks like analyzing patient records, identifying disease patterns, or processing claims data [20-22]. On the other hand, Apache Spark, with its in-memory processing capabilities, is ideal for iterative and interactive workloads. It outperforms MapReduce for machine learning, real-time data analytics, and graph processing. In healthcare, this means faster analysis of medical images, real-time patient monitoring, and rapid genome sequencing. Both frameworks offer healthcare professionals the tools to uncover insights, enhance patient care, streamline operations, and support research endeavors. They are pivotal in leveraging big data to drive innovation and improve healthcare outcomes. However, it's essential to consider the specific use case and data characteristics when choosing between MapReduce and Apache Spark in healthcare applications [23, 24].

6. Experimental Results

The fifteen queries were executed in Hive on both MapReduce and Spark platforms, and the execution times were recorded. The time taken for processing user queries to retrieve records is documented in both Table 1 and Table 2.

Table 1 Execution time taken by Hive with Spark

Query	**Time in sec**
QR_1	1.93
QR_2	1.63
QR_3	1.50
QR_4	1.89
QR_5	2.04
QR_6	4.48
QR_7	1.88
QR_8	2.59
QR_9	3.74
QR_{10}	4.63
QR_{11}	8.73
QR_{12}	6.38
QR_{13}	3.55
QR_{14}	0.98
QR_{15}	2.32

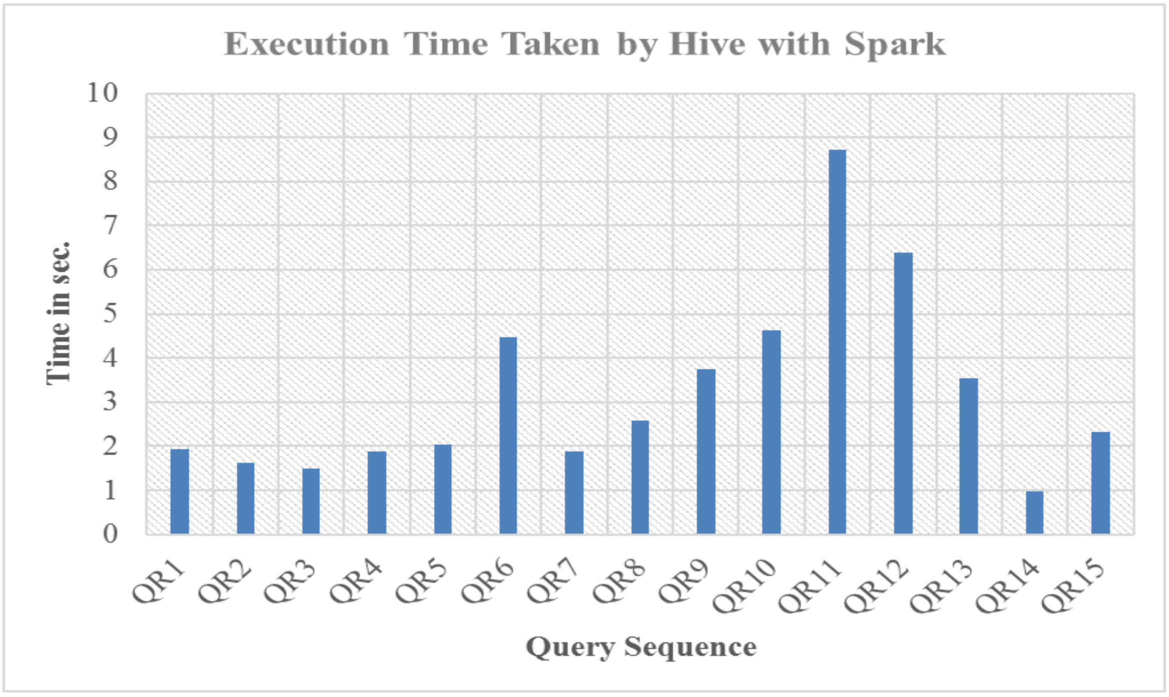

Figure 1 Execution Time Taken by Hive with Spark

Table 2 Execution time taken by Hive with MapReduce

Query	Time in sec
QR_1	9.68
QR_2	6.94
QR_3	7.59
QR_4	13.89
QR_5	10.92
QR_6	6.68
QR_7	3.32
QR_8	4.18
QR_9	4.73
QR_{10}	7.92
QR_{11}	11.92
QR_{12}	7.01
QR_{13}	3.04
QR_{14}	1.94
QR_{15}	4.62

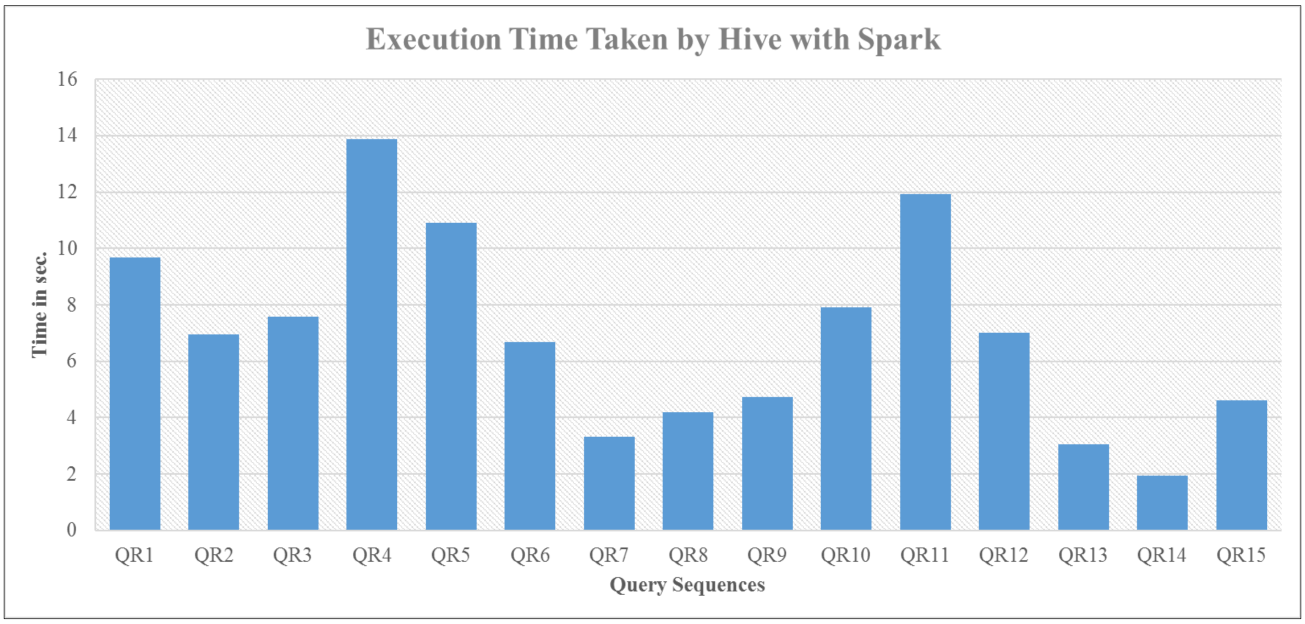

Figure 2 Execution Time Taken by Hive with Spark

It is quite apparent that the performance of Spark in executing queries is considerably faster than that of MapReduce. This is due to the fact that Spark carries out processing in memory, while MapReduce requires reading from and writing to disk. Consequently, there is a disparity in processing speed. Figs. 1 and 2 provide a more comprehensive illustration of the discrepancy in execution speed between Spark and MapReduce [25].

7. Conclusion

The use of big data and related technologies can yield substantial gains for a company. However, the widespread use of these technologies has made it more challenging for businesses to exert tight control over huge, disparate data sets in preparation for additional analysis and research. The application of Big Data has numerous results. It helps by giving businesses a vast potential to use against rivals and to fuel rapid expansion. Big Data may be used effectively to increase throughput, modernize processes, and boost efficiency across whole organizations or economies, but only if certain conditions are met. In order to reap the benefits of Big Data, it is essential to understand how to assure the intelligent usage, administration, and re-use of Data Sources, such as public government data, inside and beyond the country. Choosing the most appropriate method for data filtering and/or analysis is essential. Apache Spark and MapReduce are useful for efficient analytical processing. We have compared Hive's efficiency against that of Spark and MapReduce in this study.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] A. Botta, W D Donato, V. Persico and A. Pescapé, Integration of cloud computing and internet of things: a survey, Future Generation Computer Systems, vol. 56, pp.684-700, 2016.

[2] Ibtisum, S. (2020). A Comparative Study on Different Big Data Tools

[3] A. Kankanhalli, J. Hahn, S. Tan, and G. Gao, Big Data And Analytics In Healthcare: Introduction to the Special Section$. Information Systems, Frontiers, 18(2), pp.233-235, 2016.

[4] N. Stoianov, M. Urueña, M. Niemiec, P. Machnik, and G. Maestro, Integrated security infrastructures for law enforcement agencies, Multimedia Tools and Applications, vol.74, no. 12, pp.4453-4468, 2015.

[5] M. Sharma and J. Kaur, A Comparative Study of Big Data Processing: Hadoop vs. Spark, 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2019, pp. 1073-1077.

[6] S. K. Sahu, M. M. Jacintha and A. P. Singh, Comparative study of tools for big data analytics: An analytical study, 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 2017, pp. 37-41, doi: 10.1109/CCAA.2017.8229827.

[7] Salloum, S., Dautov, R., Chen, X. et al. Big data analytics on Apache Spark. Int J Data Sci Anal 1, 145–164 (2016). https://doi.org/10.1007/s41060-016-0027-9

[8] Zaharia, M., Chowdhury, M., Das, T., Dave, A.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. NSDI’12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation pp. 2–2. doi:10.1111/j.1095-8649.2005.00662.x (2012)

[9] Zaharia, M., Chowdhury, M., Das, T., Dave, A.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. NSDI’12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation pp. 2–2. doi:10.1111/j.1095-8649.2005.00662.x (2012)

[10] C. Ranger et al. Evaluating mapreduce for multi-core and multiprocessor systems. In Proceedings of the 2007 IEEE HPCA, pages 13–24, 2007.

[11] Yoo, R. M., Romano, A.K. and Kozyrakis, C. 2009. Phoenix Rebirth: Scalable MapReduce on a Large-Scale Shared-Memory System. Proceedings of the 2009 IEEE International Symposium on Workload Characterization, pp. 198-207.

[12] Maitrey, Seema, and C. K. Jha. MapReduce: simplified data analysis of big data. Procedia Computer Science 57 (2015): 563-571.

[13] Landset S, Khoshgoftaar TM, Richter AN, Hasanin T. A survey of open source tools for machine learning with big data in the hadoop ecosystem. J Big Data. 2015; 2(1):24.

[14] HiBench Benchmark Suite. https://github.com/intel-hadoop/HiBench. Accessed 06 October 2023.

[15] Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. In: 2010 IEEE 26th symposium on mass storage systems and technologies (MSST). New York: IEEE; 2010. p. 1–10.

[16] Luo M, Yokota H. Comparing hadoop and fat-btree based access method for small file i/o applications. In: International conference on web-age information management. Berlin: Springer; 2010. p. 182–93.

[17] Taylor RC. An overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics. BMC Bioinform. 2010; 11:1.

[18] Vohra D. Practical Hadoop ecosystem: a definitive guide to hadoop-related frameworks and tools. California: Apress; 2016.

[19] Lee K-H, Lee Y-J, Choi H, Chung YD, Moon B. Parallel data processing with mapreduce: a survey. AcM sIGMoD record. 2012;40(4):11–20.

[20] S M Atikur Rahman, Sifat Ibtisum, Ehsan Bazgir and Tumpa Barai. The Significance of Machine Learning in Clinical Disease Diagnosis: A Review. International Journal of Computer Applications 185(36): 10-17, October 2023. https://doi.org/10.5120/ijca2023923147

[21] S M Atikur Rahman, Sifat Ibtisum, Priya Podder and S. M. Saokat Hossain. Progression and Challenges of IoT in Healthcare: A Short Review. International Journal of Computer Applications 185(37): 9-15, October 2023. https://doi.org/10.5120/ijca2023923168

[22] Sarker, B., Sharif, N. B., Rahman, M. A. & Parvez, A. S. (2023). AI, IoMT and Blockchain in Healthcare. Journal of Trends in Computer Science and Smart Technology, 5(1), 30-50. https://doi.org/10.36548/jtcsst.2023.1.003

[23] Podder, P., Bharati, S., Rahman, M. A., & Kose, U. (2021). Transfer learning for classification of brain tumor. In Deep learning for biomedical applications (pp. 315-328). CRC Press.

[24] Bharati, S., Robel, M. R. A., Rahman, M. A., Podder, P., & Gandhi, N. (2021). Comparative performance exploration and prediction of fibrosis, malign lymph, metastases, normal lymphogram using machine learning method. In Innovations in Bio-Inspired Computing and Applications: Proceedings of the 10th International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2019) held in Gunupur, Odisha, India during December 16-18, 2019 10 (pp. 66-77). Springer International Publishing.

[25] Ahmed, N., Barczak, A.L.C., Susnjak, T. et al. A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench. J Big Data 7, 110 (2020). https://doi.org/10.1186/s40537-020-00388-5