ISSN 1533-9211

DOI: 10.5281/zenodo.10847726

APACHE SPARK AND HADOOP: A DETAILED COMPARISON OF THE TWO PROCESSING PARADIGMS

ENAS TAWFIQ NAFFAR [1], LAMA SADI AWAD [2] and

Dr. FAWAZ AHMAD ALZAGHOUL [3]

[1] Faculty of Information Technology, Philadelphia University, Jordan. Email: enaffar@philadelphia.edu.jo
[2, 3] King Abdullah II School of Information Technology, The University of Jordan Amman.
Email: [2]LMA9220481@ju.edu.jo, [3]fawaz@ju.edu.jo

Abstract

The exponential growth in data volume highlights the crucial need for efficient massive dataset processing, storage, and analysis. This paper examines the fundamental details of Apache Hadoop and Apache Spark, two popular large data processing frameworks. Their fundamental designs, data storage structures, processing techniques, fault tolerance systems, and general structural frameworks are all thoroughly examined. Apache Hadoop, an innovative framework, utilizes the Hadoop Distributed File System (HDFS) along with the MapReduce programming model to achieve distributed data storage and processing. On the other hand, Apache Spark, a more modern equivalent, uses in-memory processing and Resilient Distributed Datasets (RDDs) to improve performance. The comparative examination reveals the subtleties of each framework's data storage, processing models, and fault tolerance techniques. Spark can handle batch and real-time processing, which contrasts with Hadoop's conventional batch-oriented processing using MapReduce. The study investigates how these structural differences affect the system's overall efficiency, scalability, and simplicity of use. The results of this study add to our understanding of Hadoop and Spark's fundamental underpinnings. Organizations can choose a framework that best suits their unique data processing needs by providing information about their internal structures and processing techniques.

Keywords: Big Data, Hadoop, Spark, MapReduce, RDDs.

1. INTRODUCTION

The big data sector is facing a significant problem in organizing, analyzing, and extracting insights from enormous datasets due to the extraordinary spike in data output in this period. This paper explores the complex structural aspects of two major frameworks: Apache Spark and Apache Hadoop. Standing as cornerstones in the vast field of distributed data processing, these frameworks are vital in managing the "5V" that make up big data: volume, velocity, variety, veracity, and value [1]. Table 1 displays Big Data Features.

A deep comprehension of the architectural subtleties included in Hadoop and Spark is essential for navigating the complex routes found in big data environments. These frameworks are more than just technical fixes; they are critical decision points for companies trying to use distributed computing and deal with massive amounts of data. Moreover, they play a crucial role in facilitating the speed at which data is influx, controlling a variety of data formats, guaranteeing accuracy through fault tolerance systems, and finally producing value in the form of useful insights.

ISSN 1533-9211

DOI: 10.5281/zenodo.10847726

Table 1: Features of Big Data

Features	Description
Volume	The enormous volume of data that is produced and analyzed, which frequently exceeds the capability of conventional databases.
Velocity	The rate at which information is created, gathered, and analyzed in real-time or almost real-time is a critical factor for applications that require speed and agility.
Variety	Flexible processing and storage solutions are necessary due to the wide range of data kinds and formats, including unstructured, semi-structured, and structured data.
Veracity	The data's correctness and dependability, considering the noise, inconsistencies, and uncertainties seen in big datasets.
Value	The useful information and practical insights gained from big data analytics highlight how crucial it is to wring value out of the mountains of data.

2. APACHE HADOOP'S STRUCTURAL COMPONENTS AND APACHE SPARK'S FRAMEWORK ARCHITECTURE

This section provides a detailed discussion of the Hadoop MapReduce and Spark operating principles.

1. Apache Hadoop Structural Components

Hadoop is a distributed computing framework, offering an open-source solution crafted in Java. This robust framework facilitates the establishment of a reliable, fault-tolerant, scalable, and adaptable architecture tailored for large-scale batch processing of big data. Leveraging the combined computational and storage capacities of clusters, Hadoop is designed to enable distributed storage and parallel processing, seamlessly handling extensive volumes of data [2]. Its architecture makes it an optimal solution for addressing the demands of large-scale data processing with efficiency and effectiveness. Figure 1 illustrates the three layers that constitute the Hadoop architecture.

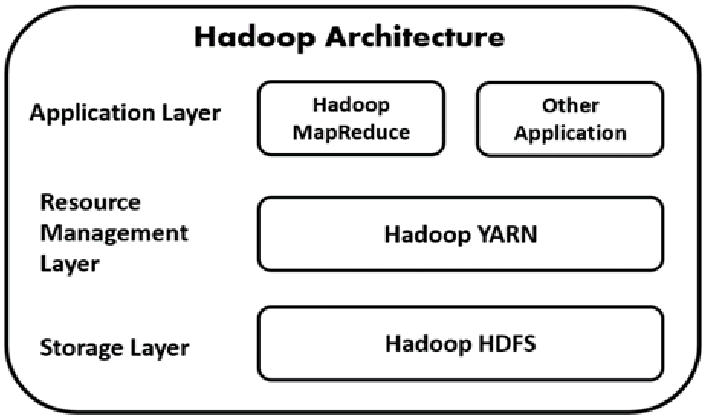


Fig 1: Hadoop architecture

ISSN 1533-9211

DOI: 10.5281/zenodo.10847726

A) Hadoop MapReduce (MR):

Hadoop framework is built on the programming style and processing engine known as MapReduce. Google first presented it, and Hadoop later adopted and used it to manage the distributed processing of huge datasets. The MapReduce approach divides work into three primary stages: the Map phase, Shuffle and Sort, and the Reduce phase. This makes processing data in parallel across a Hadoop cluster easier [3]. The MapReduce work steps are depicted in Figure 2.

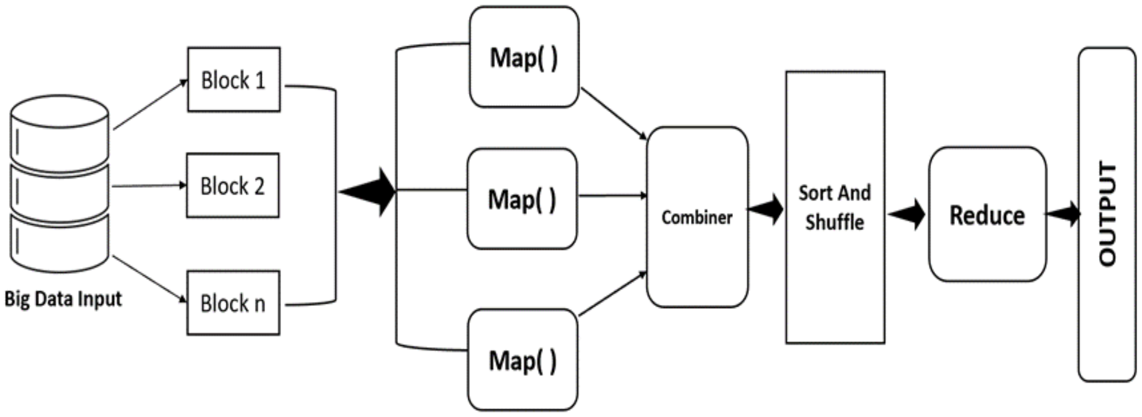


Fig 2: MapReduce in Hadoop

1. Map Phase:

In the Map phase, the incoming data is split up into smaller pieces, which are then processed independently by several simultaneous Map activities. The input data is transformed into a collection of intermediate key-value pairs for each Map job by applying a user-defined function known as the "Map" function. Following that, the key-value pairs produced by the Map jobs are sorted and classified according to their keys.[4]

The main objective of the Map phase is to extract pertinent information that may be utilized in further processing and to divide the incoming data into digestible chunks.

2. Shuffle and Sort:

Following the completion of the Map phase, the framework sorts, and shuffles data. This entails grouping all values related to a certain key and rearranging the intermediate key-value pairs according to their keys. To prepare the data for the ensuing Reduce phase, this step is essential.

3. Reduce Phase:

After the Map phase, the grouped key-value pairs are processed in the Reduce phase by several simultaneous Reduce jobs. To compile, condense, or handle the intermediate data further, each Reduce job employs a user-defined "Reduce" function. The last collection of key-value pairs, or the processed result, is what comes out of the Reduce step.

ISSN 1533-9211

DOI: 10.5281/zenodo.10847726

Consolidating the data retrieved during the Map phase and producing the result in a comprehensible manner for the specified calculation are made possible by the Reduce phase of the system.

B) Hadoop YARN:

Several tasks are executed using Hadoop. Each of them requires resources to finish the tasks at hand concurrently and to manage these resources effectively, Hadoop has a crucial component called YARN. The YARN Architecture of Hadoop is depicted in Figure 3.

The Resource Manager is the primary component in the Hadoop YARN architecture that oversees efficiently managing and allocating resources throughout the cluster. It decides how to distribute resources to various programs after receiving resource requests from them. A Node Manager is responsible for managing the resources on each node in the Hadoop cluster and informing the Resource Manager about the node's availability and health. Simultaneously, for each application in the cluster, an Application Master negotiates resources with the Resource Manager and works with Node Managers to carry out and oversee operations, overseeing the application's lifecycle with attention to detail [5].

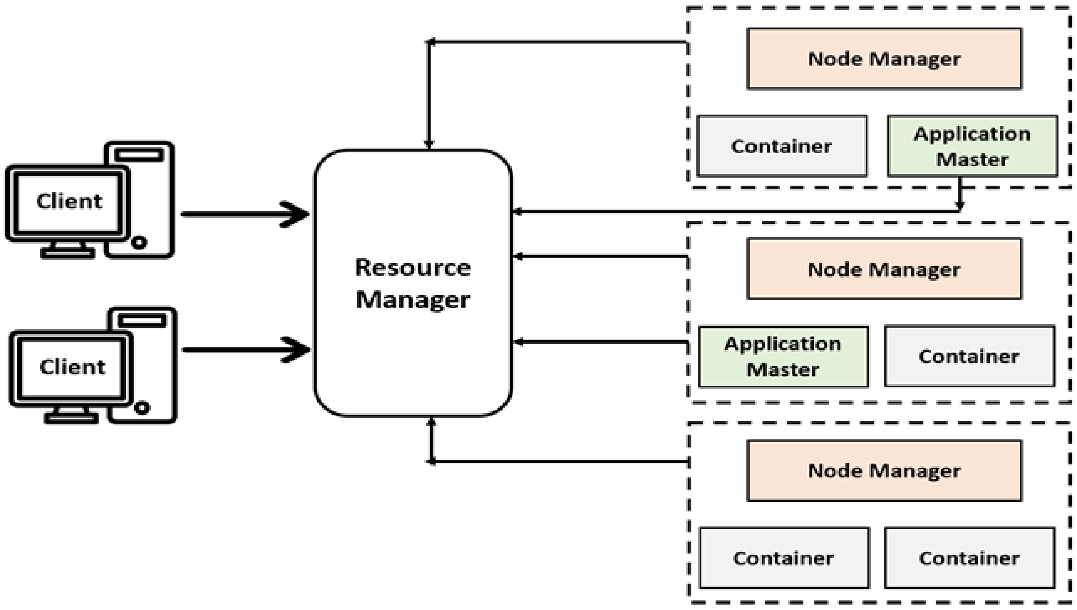


Fig 3: Apache Hadoop YARN Architecture

C) Hadoop Distributed File System (HDFS):

The Hadoop file system, known as HDFS, divides data into several smaller parts and distributes each component among several nodes since it is not feasible to store a significant quantity of data on a single node. HDFS Design is shown in Figure 4. Large amounts of data may be streamed to user applications and stored on the HDFS. Thousands of computers in a big cluster handle a user application activity in addition to hosting directly connected storage. The

ISSN 1533-9211

DOI: 10.5281/zenodo.10847726

NameNodes and DataNodes that function as masters and workers respectively make up Hadoop's distributed file system. Keeping track of file blocks and directory structure, the NameNode is an essential part that handles namespace operations and metadata management. HDFS filenames are converted into lists of block IDs and DataNodes to enable effective data access.

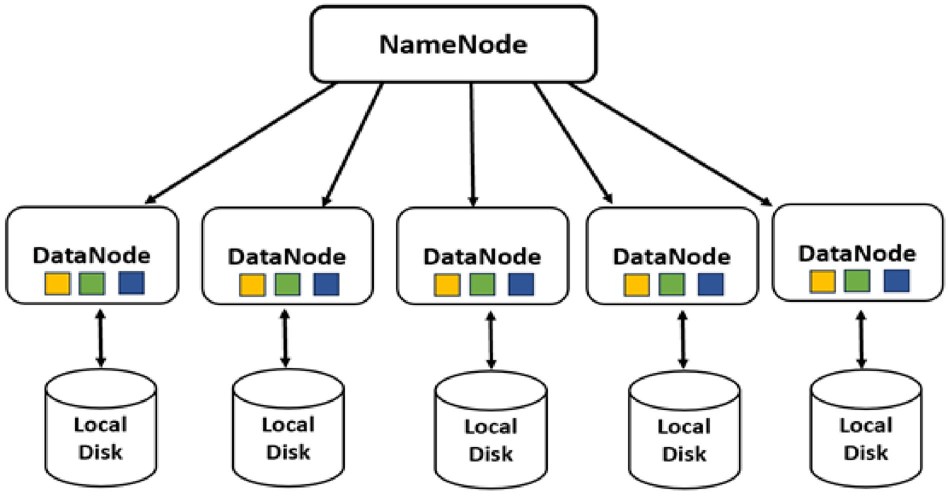


Fig 4: Hadoop Distributed File System Design

2. Apache Spark: Structural Components

Spark is a distributed general-purpose computing framework, that provides an open-source solution for large-scale data processing with language-integrated APIs in Scala, Python, Java, and R. Spark has upper-level libraries for various purposes, including machine learning, graph analysis, streaming, and structured data processing [15] as depicted in Figure 5. One of the most powerful features of Spark is its capability to process data on disk and in memory, making its performance outstands. In addition to its performance, Spark's flexibility, ease of use, and compatibility with different cluster managers contribute to its popularity in big data processing. Figure 6 illustrates the architecture of Spark.

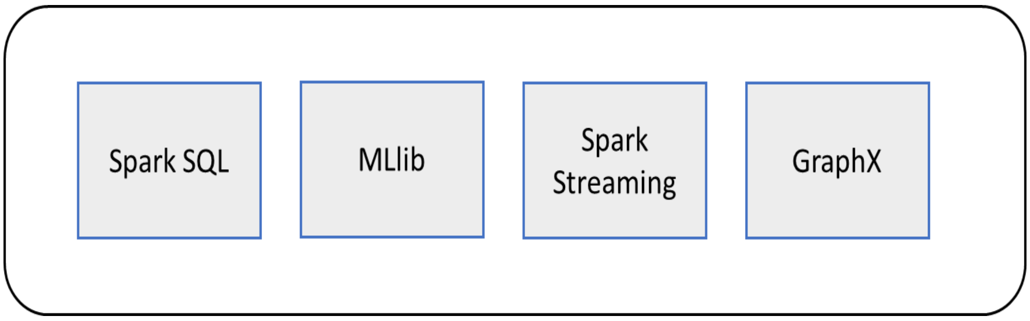


Fig 5: Upper-level libraries in Spark

ISSN 1533-9211

DOI: 10.5281/zenodo.10847726

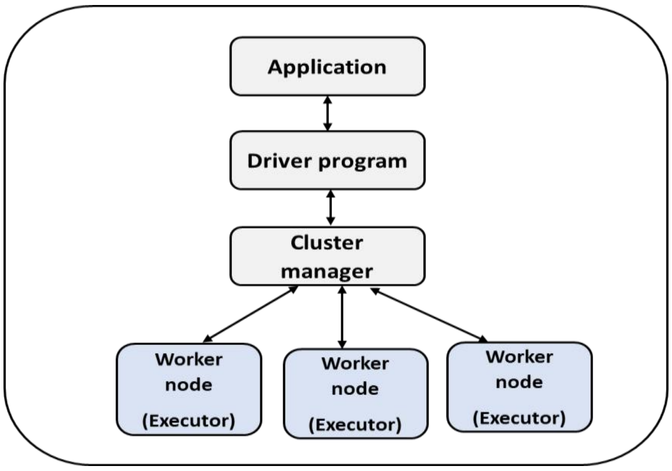


Fig 6: Spark's high-level architecture

A) Application:

A Spark application is a user program, that is typically written in languages like Scala, Java, Python, or R, that specifies a sequence of data processing tasks to be performed using the Spark framework.

B) Driver Program:

A driver program is the main control process for a Spark application that utilizes Spark as a library and contains the main function where *Spark context* is created. The Spark context not only manages the execution of tasks but also creates and manipulates distributed datasets on the cluster which are known as Resilient Distributed Datasets (RDDs). Additionally, it has the Directed Acyclic Graph (DAG) scheduler, as depicted in Figure 7. The Spark context serves as a gateway connecting to the Spark cluster [17]; it coordinates the execution of tasks, communicates, and coordinates with the cluster manager, and collects results from worker nodes.

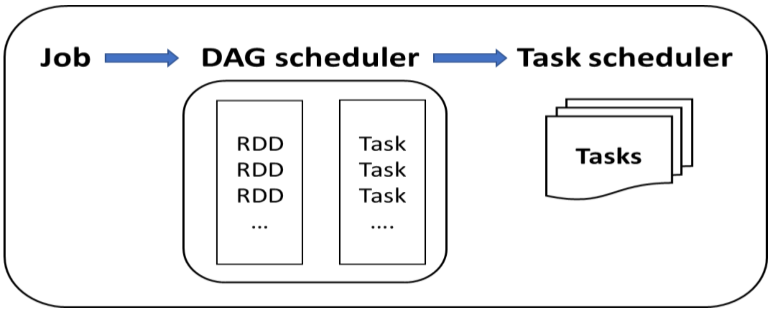


Fig 7: Spark context

A ***job*** is a set of computations, consisting of multiple tasks, that Spark performs on a cluster to get results to the driver program. Multiple jobs can be initiated by a Spark application. In Spark, a job is split into a directed acyclic graph (DAG) of stages where each stage is a set of tasks. A ***task*** is the smallest unit of work sent to an executor [15].

ISSN 1533-9211

DOI: 10.5281/zenodo.10847726

A ***DAG***, which corresponds to the logical execution plan in Spark, is typically a graph, whose nodes represent the RDDs and edges represent transformations among RDDs. A ***DAG scheduler*** manages the scheduling and execution of tasks based on the logical execution plan represented by the DAG.

RDDs are a fundamental component of Spark, representing an immutable collection of elements that can be operated on in parallel. RDDs offer two primary types of operations: transformations like map, filter, ReduceByKey, and GroupByKey [18], which create a new RDD from an existing one, and actions that trigger the computation and return results to the driver program like gather, count, and Count ByKey [18].

The ***task Scheduler*** in Spark is responsible for making decisions about which tasks to run, where to run them, and when to run them, based on the available resources that have been allocated by the cluster manager.

C) Cluster manager:

The cluster manager is responsible for managing and allocating the resources (CPU and memory) to applications based on their requirements. It also adopts several techniques to ensure fault tolerance. Spark currently supports different cluster managers [16] as seen in Figure 8:

- ***Standalone***: A simple resource manager included with Spark as its built-in cluster manager. It is well suited to development and testing for small-scale deployments.
- ***Apache Mesos***: A general open-source resource manager, that provides an abstraction of resources, which facilitates their dynamic allocation to applications. It is efficient in large-scale deployments.
- ***Hadoop YARN***: A resource manager in Hadoop, that can be used with Spark, to deploy and manage Spark applications in a Hadoop environment. This integration allows Spark to benefit from YARN capabilities.
- ***Kubernetes***: An open-source orchestration service, that serves as a powerful resource manager for Spark applications. Kubernetes allocates the resources dynamically based on the specified resource requirements and facilitates the integration of Spark into cloud-native architectures and large-scale deployments.

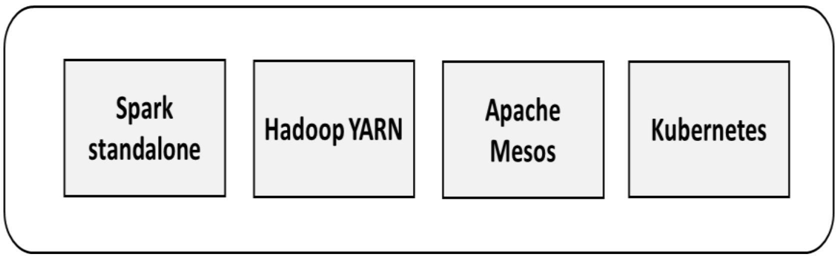


Fig 8: Spark cluster manager

D) Worker nodes:

The worker nodes, as depicted in Figure 9, sometimes called the “salve nodes”, are responsible for executing the tasks that have been assigned to them by the cluster manager and the task scheduler. Each worker node runs an executor, which is a process that executes the task and returns the results of the execution to the Spark context.

The number of worker nodes depends on the requirements and the needs of the application. Increasing the number of worker nodes contributes to handling a huge amount of data and increasing parallel processing.

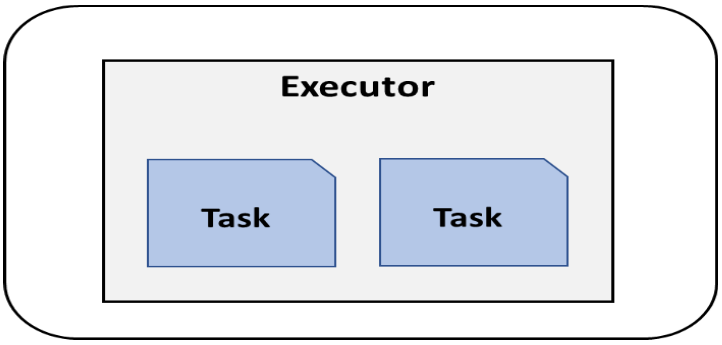


Fig 9: A worker node

3. RELATED WORK

Numerous comparative studies that address Spark and Hadoop technologies along with associated topics have been published. Table 2 presents a selection of these Comparatives along with the primary subjects they studied.

Table 2: Comparative Studies of Hadoop VS Spark

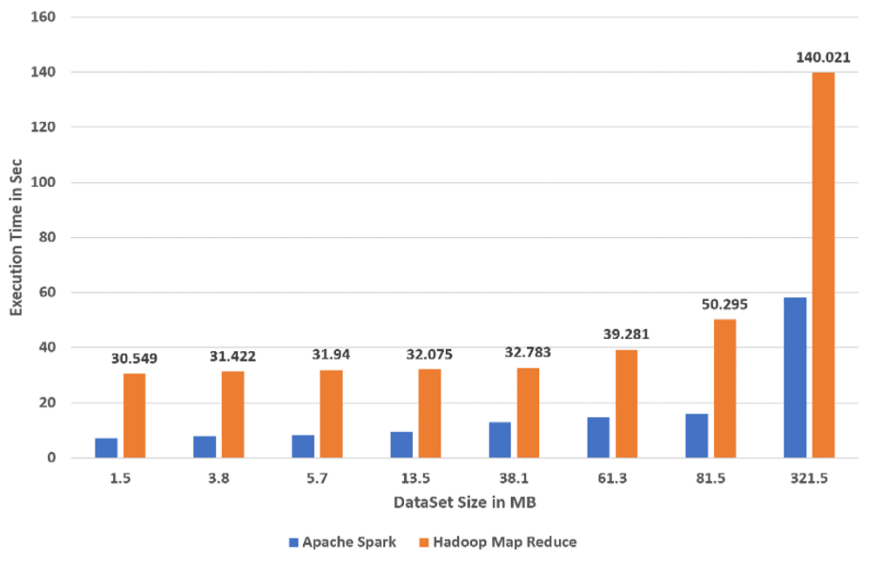
Shwet Ketu1 et al. [6]	The effectiveness of Hadoop MapReduce and Spark has been observed through experimental studies to determine their applicability under various distributed computing environment restrictions.
Muhammad Ali &Khurshed Iqbal [7]	A comprehensive review of the benefits and drawbacks of two well-known technologies, Spark and Hadoop, about financial data management.
S. Alkatheri et al. [8]	The research evaluated several frameworks using performance parameters, including processing time, CPU use, latency, throughput, execution duration, job performance, scalability, and fault tolerance. The proposal is that businesses look closely at these performance metrics before deciding a framework.
Khadija AZIZ et al. [12]	The analysis of Twitter data using Spark performed better than Hadoop MapReduce, according to experimental results. Spark might evaluate data in just a few seconds, consider it as an excellent memory processing tool that enables real-time processing of streaming data.
Y. Benlachmi and M. L. Hasnaoui [18]	This research investigates the techniques of the Spark model as an alternative to Hadoop MapReduce for efficient analysis of big data in HDFS.

The Seybold REPORT
ISSN 1533-9211
DOI: 10.5281/zenodo.10847726

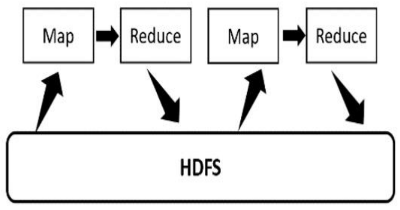
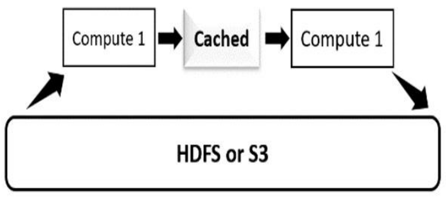
IV. A COMPARISON BETWEEN HADOOP AND SPARK

Based on the many properties of these two technologies, highlighted in Table 3, a comparison between Spark and Hadoop is performed.

Table 3: Comparison between Apache Hadoop and Spark

1. Performance	While reading and writing rely on disk operations in Hadoop, processing is done both in-memory and on-disk in Spark. The speed of processing is influenced by variations in processing technology. Since the processing for Spark is primarily done in-memory, it often requires less time than Hadoop MapReduce, which depends more on disk operations.
2. Execution Time for similar datasets	Spark provides quicker execution speeds than Hadoop MapReduce. Figure 10 shows the comparison between different dataset sizes executed in both Hadoop and Spark.  **Fig 10: Execution time comparison of Apache Spark and Hadoop**
3. Fault Tolerance	Hadoop uses a method of duplicating data over several nodes to provide strong fault tolerance. To do this, it splits up each file into chunks and duplicates it on many machines. Conversely, Spark has more fine-grained mechanisms; it makes use of Resilient Distributed Datasets (RDDs) lineage information, where lost partitions can be recovered without the need to recompute the whole dataset. Spark allows a more efficient recovery by only re-executing the affected computation. Spark also supports optional checkpoints.
4. Data Processing	Using a sequential process, Hadoop processes data in batches. Reading data from the cluster, performing operations, and writing the outcomes back to the cluster are all part of the MapReduce paradigm. With Spark, on the other hand, batch and real-time processing are smoothly integrated. The cluster's data is read by it, real-time operations are carried out, and the treated data is quickly returned to the cluster.
5. Cost	Hadoop is well known for being inexpensive because it uses open-source software and commodity hardware, which lowers infrastructure costs. Although Spark is more effective for some workloads, its dependency on in-memory processing and possible quicker development times might result in greater expenses.
6. Data Storage	Because Hadoop can store enormous amounts of data, it was employed for data archiving. With Hadoop, a user's data may be stored for many years, as well as for a

The Seybold Report
ISSN 1533-9211
DOI: 10.5281/zenodo.10847726

	dozen years in both original and archived form. Unlike Hadoop, Spark is not primarily designed for long-term storage; rather, it is focused on quick and effective data processing. To maximize their big data operations, organizations frequently combine Spark for iterative or real-time processing with Hadoop for long-term storage.[9]
7. Accessing External data sources	Hadoop primarily accesses data stored in its HDFS, however, it can be integrated with external storage systems like Hive and HBase. Spark can integrate well with various external storage systems, like HDFS, Cassandra, HBase, Amazon S3, and other sources.
8. Usability	Without an interactive mode, Hadoop is a sophisticated system that requires a significant amount of coding work to accomplish complex data processing jobs by requiring the usage of low-level APIs. Alternatively, Spark provides easy-to-use programming language APIs, making the development process more transparent and efficient. Because of its interactive mode, which offers intermediate feedback during queries and operations, Spark is easier to use and more responsive than the more complex MapReduce model.
9. Scalability	Hadoop is very scalable; it can accommodate many nodes in the cluster without any problems. Notably, Yahoo used a large 42,000-node Hadoop cluster to illustrate its scalability. By contrast, the biggest verified Spark cluster consisted of 8,000 nodes. However, it is anticipated that cluster sizes will increase to accommodate changing throughput needs as big data continues to rise [10].
10. Security	Secure user access is ensured via authentication techniques like Kerberos, which are supported by both Hadoop and Spark. In addition, Access Control Lists are used by Hadoop for authorization. Spark has its authorization systems. Furthermore, both systems offer data encryption for the security of data while it is in transit and at rest [11].
11. Language Support	Hadoop primarily supports Java for MapReduce programs, but it has a feature called Hadoop streaming which allows the support of different languages like Python, Perl and Ruby, while the primary language for Spark is Scala. Spark has APIs for other languages, including Java, Python, and R.
12. Iterative Computation	Iterative methods in standard Hadoop MapReduce suffer from costs since the processing mechanism is disk-based. Performance bottlenecks might arise since each loop includes writing interim findings to disk. Spark uses its in-memory processing power to excel in iterative calculation. It eliminates the requirement for repetitive reading from and writing to disk by caching intermediate data in memory. Because of this, Spark is far more effective than Hadoop MapReduce for iterative algorithms.   **Fig 11: Iterative workload handling in Hadoop Fig. 12: Iterative workload handling in Spark**
13. Machine Learning	Hadoop is capable of processing huge amounts of data; however, it does not have built-in machine learning libraries, but it can be used with another software which is Mahout for processing data. Mahout includes clustering, classification, and batch-based collaborative filtering. But, Spark has a dedicated machine learning library called MLlib, which is divided into two packages: spark.mllib and spark.ml. The two

ISSN 1533-9211

DOI: 10.5281/zenodo.10847726

	packages have a variety of machine learning tasks such as featurization, transformations, model training, model evaluation, and optimization [15]. Additionally, Spark's memory processing makes it more suitable for machine learning tasks.
14. Job scheduling and resource allocation	Hadoop provides control over resource allocation and job scheduling, however, resource allocation in Hadoop is mainly static; meaning that resources are allocated at the beginning of the execution of the job and it is unlikely to change during execution. Spark, on the other hand, offers more advanced scheduling and optimization features, and has a dynamic resource allocation.
15. Applications	Hadoop can be used in systems that require scalable storage and batch processing of data, while Spark is best adopted in streaming data processing to offer real-time analytics, machine learning, and interactive data exploration.

5. REAL-WORLD IMPLEMENTATIONS: SPARK AND HADOOP

Organizations are looking for strong techniques to store, handle, and effectively analyze large amounts of data within the ever-changing big data technology landscape. In this section, two paradigm-shifting examples—Netflix utilizing Apache Spark and Facebook utilizing Hadoop are displayed.

- **Facebook's Innovative Hadoop Utilization**

Facebook, one of the biggest social media networks in the world, uses Hadoop to highlight its unmatched scalability and dependability. The platform's ability to successfully use Hadoop is evidence of the technology's flexibility in responding to the ever-increasing demands of a data-centric environment. Using Hadoop, Facebook has demonstrated how the technology can be a key component in managing and extracting value from massive datasets in high-traffic, real-world applications. Facebook was able to overcome the obstacles presented by the exponential development of user-generated data [13].

- **Netflix's Utilization of Apache Spark**

Providing millions of consumers with individualized content recommendations was a problem for Netflix, a worldwide streaming network. With its capacity to process data in memory, scale, and provide real-time analytics, Apache Spark was found to be the answer.

Apache Spark's powerful characteristics enable Netflix's recommendation engine to be implemented. The streaming behemoth improves user experience by assuring quick and effective data handling by utilizing Spark's in-memory processing capabilities. Furthermore, by instantly analyzing user interactions with Spark Streaming, Netflix can provide its broad user base with timely and appropriate content recommendations in real time [14].

6. SPARK AND HADOOP INTEGRATION'S IMPORTANT POTENTIAL

In the ever-developing field of big data, the way businesses manage enormous datasets has been completely transformed by the convergence of Spark and Hadoop, which has become a strategic need. Massive volumes of data may be stored in an unmatched way that is both scalable and fault-tolerant thanks to Hadoop's distributed file system (HDFS). However, the

ISSN 1533-9211

DOI: 10.5281/zenodo.10847726

speed and efficiency of its conventional MapReduce processing paradigm were constrained. To enhance Hadoop, Spark is a potent in-memory computing engine that offers data analytics with hitherto unattainable speed and capacity.

Organizations may make use of both the powerful storage capabilities of HDFS and the blazingly quick processing power of Spark by combining Spark with Hadoop. Through this connection, organizations will be able to stay competitive in the big data age, get insights, and make educated choices by creating a consistent and efficient data processing environment.

Spark and HDFS work together seamlessly to provide optimal resource usage, which allows activities to be carried out directly on the nodes that host the data. This optimizes the possibility for parallel processing while minimizing data transport costs, resulting in notable speed gains. Moreover, Spark's compatibility with the Hadoop ecosystem, which includes varied data formats, promotes efficient interchange and improves processing and storage efficiency [19].

YARN, which is the resource manager in Hadoop, can be used as a cluster manager for both Hadoop and Spark. This leads to enhanced resource utilization and sharing of resources across multiple applications. As mentioned previously, Spark can run on YARN to allow Spark applications to coexist with Hadoop jobs in the same cluster. However, we should pay attention to the fact that when Spark is running on YARN with other shared services, performance might degrade.

When integrating Hadoop and Spark, data processing pipelines should be designed to leverage the strengths of both Hadoop and Spark. For instance, Spark may be used for iterative machine learning tasks and real time analysis while Hadoop may be used for batch processing. The benefits of combining Hadoop and Spark go much beyond improved operational effectiveness. The paradigm of data processing is radically altered by this integration, opening new avenues for advanced data-driven applications, machine learning, and real-time analytics.

7. CONCLUSION

The rise of big data has necessitated parallel and distributed processing, along with advanced data analysis. Hadoop and Spark are two prominent frameworks that are capable of handling vast amounts of data. In this research, we examine the two frameworks, highlight their main features, and compare them in terms of their architectures and other properties, including performance, execution time, fault tolerance, data processing, cost, data storage, access to external data sources, usability, scalability, security, language support, iterative computation, machine learning, job scheduling, and applications.

Hadoop and Spark should not be considered as two mutually exclusive alternatives, but rather complementary of each other. Hadoop outstands in batch processing and scalable storage capabilities, while Spark’s strengths lie in its in-memory processing, real-time analysis, and machine learning. Choosing between them depends on the requirements of the system, integrating them is yet another option that leverages the advantages of the two frameworks.

ISSN 1533-9211

DOI: 10.5281/zenodo.10847726

References

1) Urvashi Gupta, Rohit Sharma,(2023) 1st Edition ,eBook ISBN 9781003452591.

2) Hoger K. Omar1, Alaa Khalil Jumaa2, June (2022). Distributed big data analysis using Spark parallel data processing. ISSN: 2302-9285, DOI: 10.11591/eei. v11i3.3187

3) R Rawat1 and R Yadav2, (2021). Big Data: Big Data Analysis, Issues and Challenges and Technologies DOI 10.1088/1757-899X/1022/1/012014

4) Akaash Vishal Hazarika, G Jagadeesh Sai Raghu Ram, (2017). Performance Comparison of Hadoop and Spark Engine. 978-1-5090-3243-3/17/$31.00 ©2017 IEEE.

5) Samadi, Y., Zbakh, M. and Tadonki, (2018). Performance comparison between Hadoop and Spark frameworks using HiBench benchmarks. Concurrency and Computation: Practice and Experience, 30(12), p.e4367.

6) Shwet Ketu1, Pramod Kumar Mishra1, Sonali Agarwal2, (2020). Performance Analysis of Distributed Computing Frameworks for Big Data Analytics: Hadoop Vs Spark, DOI: 10.13053/CyS-24-2-3401

7) Muhammad Ali & Khurshed Iqbal, (2022). The Role of Apache Hadoop and Spark in Revolutionizing Financial Data Management and Analysis: A Comparative Study

8) S. Alkatheri, S. Abbas and M. A. Siddiqui, (2019) "A Comparative Study of Big Data Frameworks," International Journal of Computer Science and Information Security, vol. XVII, no. 1, pp. 66-73, 2019.

9) Yong Cao, (2023). Big Data Real-Time Processing Architecture Based on Hadoop. https://doi.org/10.1117/12.2667514.

10) Ni Nyoman Putri Utam, Heri Wijayato, (2023). Top-K Query for Large Dataset of Restaurant Review Based on Hadoop MapReduce Framework. https://doi.org/10.1051/e3sconf/202346502033

11) Yusuf Perwej, (2019). The Hadoop Security in Big Data: A Technological Viewpoint and Analysis. DOI: https://doi.org/10.26438/ijsrcse/v7i3.114

12) Khadija AZIZ, Dounia ZAIDOUNI, Mostafa BELLAFKIH, (2018). Real-Time Data Analysis Using Spark and Hadoop. 978-1-5386-4225-2/18/$31.00 ©2018 IEEE

13) Md. Shohel Rana, Md Altab Hossin, S M Hasan Mahmud, Hosney Jahan, Md. Anwar Hossen,(2018). A New Method to Handle Facebook Users in the Distributed Database System. DOI: 10.1109/ICSESS.2018.8663823

14) Pooja Choudhary, Kanwal Garg, (2021). Comparative analysis of Spark and Hadoop through Imputation of Data on Big Datasets. **DOI:** 10.1109/IBSSC53889.2021.9673461

15) Salman Salloum, Ruslan Dautov, Xiaojun Chen, Patrick Xiaogang Peng, Joshua Zhexue Huang, Big data analytics on Apache Spark, Int J Data Sci Anal (2016) 1:145–164 DOI 10.1007/s41060-016-0027-9

16) Cluster Mode Overview - Spark 3.5.0 Documentation (apache.org)

17) Eman Shaikh, Iman Mohiuddin, Yasmeen Alufaisan, Irum Nahvi, Apache Spark: A Big Data Processing Engine, 2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)

18) Y. Benlachmi and M. L. Hasnaoui, "Big data and Spark: Comparison with Hadoop," *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4),* London, UK, 2020, pp. 811-817, doi: 10.1109/WorldS450073.2020.9210353. https://techvidvan.com/tutorials/hadoop-spark-integration/