An In-Depth Comparative Study of Distributed Data Processing Frameworks: Apache Spark, Apache Flink, And Hadoop Mapreduce

Ms.S. Abikayil Aarthi [1], Ms.M.Dhaslima Shafreen [2], Ms.P.Narmatha[3], Ms.B.JayaPriya[4]
[1, 2, 3, 4] Dept of CSE
[1, 2, 3, 4] Kings College of Engineering, Punalkulam, Pudhukottai.

Abstract- *Data processing has become a crucial task as the volume and data complexity continue to grow. Distributed data processing frameworks are the solution to this issue. The paper-present comparative study of popular frameworks like Apache Spark, Apache Flint, and Hadoop Map Reduce. This is done by expressing and comparing the performance, scalability, fault tolerance, and architecture of each framework to help businesses make informed decisions about their choice of framework. The paper begins with a literature review and methodology, followed by a comprehensive comparative analysis. Apache Spark benefits from in-memory processing providing high performance and Apache Flintlocks on low-latency stream processing. Hadoop Map Reduce offers fault tolerance and scalability for batch-processing tasks. The analysis concludes that both Spark and Flink outperform Map Reduce in terms of performance and scalability. Overall, this study highlights the strengths and limitations of each framework and recommends Spark as the current best option due to its maturity, market share, and community support. However, we acknowledge Flink's innovative concepts and future development possibilities. The choice of the framework should be based on specific use cases and requirements for performance, fault tolerance, and real-time processing.*

I. INTRODUCTION

Processing of large amounts of data has become a valuable aspect of industry and research, necessitating the use of efficient and scalable frameworks to handle these tasks. As the volume and complexity of information continue to grow, efficient and scalable frameworks are required to handle the processing tasks. The frameworks are intended to ease computing. It involves transforming raw data into meaningful information. And also, consists of operations like data cleaning, visualization, analysis, and aggregation. Traditional computational power of one machine [13].

However, the distributed nature of the frameworks allows horizontal scaling by introducing more machines to process the data overcoming the aforementioned limitation [13]that careful planning of resource allocation algorithms can result in major performance and cost benefits. Fast data processing capabilities of frameworks like Spark are helpful from an operational standpoint and can potentially save individuals in the healthcare industry [6]. Research on predictive analytics highlights the continuing value of MapReduce [12], especially for non-time-sensitive applications that need precise predictions from big datasets. Effectiveness of Spark framework in managing complicated data operations is demonstrated by the performance evaluation of query processing jobs [2], which directly affects company insight and decisions.

While prior studies have explored various distributed data processing structures, this article offers a novel contribution by conducting an in-depth comparative study of three widely used frameworks: Apache Spark, Apache Flink, and Hadoop Map Reduce. Through a rigorous analysis of the architecture, performance, scalability, and fault tolerance of each framework. The author's aim to provide valuable insights that can guide businesses in making informed decisions about their choice of data processing platform. By synthesizing existing literature and conducting comprehensive comparative analyses, the research offers a unique perspective on the strengths and limitations of each framework, thereby contributing to the advancement of knowledge in the field of distributed data processing. The scientific contribution of the study is that it assesses these frameworks, highlights the pros and cons of each of them, and makes the necessary recommendations for businesses working with the large number of processing technologies.

The paper starts with a description of the methodology used in the research, particularly literature review and comparative analysis. It then follows with a comparison analysis where Spark, Flink, and Hadoop MapReduce are compared through four criteria mentioned above. The paper finishes with a discussion where the results of the comparative analysis are discussed.

Methods and Materials.

The methodology used in this research is a comprehensive literature review and comparative analysis of the three technologies that will be disscussed. The initial step in this study involves a literature review of scholarly articles, research papers, and technical reports that are specifically focusing on Apache Spark, Apache Flink, and Hadoop MapReduce. The literature review's goal is to establish a solid theoretical foundation and gather a deep understanding of the topic such as architectural design, features, strengths, limitations, and their applications in different scenarios. The comparative analysis is conducted in order to evaluate, compare three frameworks across multiple aspects that include architecture, scalability, performance, and fault tolerance. Based on the literature review findings, technical documentation, white papers, and case studies provided by the framework developers and user communities comparative analysis is carried out.

The details of each aspect are explained below.

Performance: Performance is a key point for any software because it affects the costs of running the program. The performance of each framework is evaluated the factors like execution speed, throughput, latency, and resource utilization. Benchmarks, experiments, and various empirical studies conducted by researchers and industry experts are considered to derive meaningful insights.

Scalability: The motivation for using distributed data processing frameworks is to be able to scale them, so it is important to see how this is implemented in each framework. The scalability criteria in our research were assessed by studying the frameworks' ability to handle large volumes of data and support an increasing number of nodes in a distributed cluster. We analyzed the frameworks' scalability features, limitations, and real-world use cases to evaluate their capacity to scale effectively.

Fault Tolerance: One of the main criteria for modern systems is resilience, so it is important to see the differences and similarities between the frameworks. The fault tolerance capabilities of the frameworks are assessed by analyzing their mechanisms for handling failures, fault recovery strategies, and data reliability guarantees. The literature review provides insights into the fault tolerance mechanisms employed by each framework.

Architecture:

The motivation behind the analysis of architectural details is understanding the differences of implementation that could affect the workflow of the framework. The architectural decisions implemented in the frameworks might give advantages or disadvantages to specific use cases which will be determined further.

Literature Review.

Apache Spark: Prior research has extensively explored the capabilities and performance of distributed data processing structures. Apache Spark offers batch and stream processing, machine learning, and graph processing. The studies have demonstrated Spark's ability to process large-scale datasets efficiently and its support for real-time data analysis [13]. The first figure below (Figure 1) represents Apache Spark's master/worker architecture. It means that a driver programinteracts with a single cluster manager coordinator that manages several workers in which executors run. The executor can run on the same machine also called a horizontal cluster or on separate servers which is an example of vertical clustering
[18].

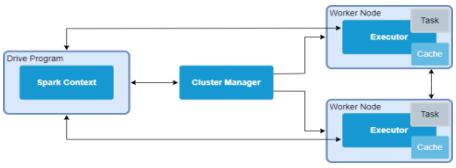

Fig. 1. Apache Spark Architecture.

Apache Flink:

On the other hand, Apache Flink emphasizes streaming data processing and provides robust fault tolerance and low-latency processing capabilities. Several comparative studies have evaluated Flink's performance and highlighted its advantages in handling continuous data streams and its support for event time processing [4]. Authors of official documentation [17] provides the chart of Apache Flink architecture (Figure 2), where Job Manager, Resource Manager, Dispatcher, and

Task Managers builds Apache Flink's architecture. And at the same time, task scheduling, failure recovery, and checkpoint synchronization are all handled by the Job Manager. The Resource Manager controls resource supply and allocation in the Flink cluster, when a job is submitted, the dispatcher offers a REST interface and launches a Job Master. Every Job Master oversees the execution of a particular Job Graph, which stands in for a Flink task [17]. The quantity of task slots defines the amount of concurrency, while task

managers handle task execution, data stream buffering, and exchange. Provided design makes possible for Apache Flink to perform distributed operations, manage resources, and process data effectively.

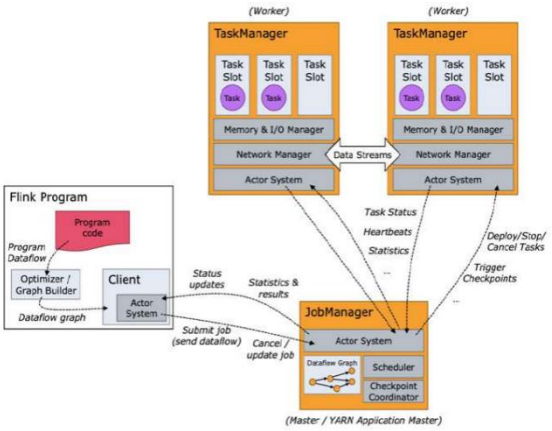

Fig. 2. Apache Flink Architecture.

Hadoop MapReduce:

Hadoop MapReduce is a framework that works with distributed data processing and has been extensively studied. It offers fault tolerance, scalability, and simplicity for batch processing tasks. Different studies have explored its strengths in processing large-scale batch workloads and its integration with the Hadoop ecosystem [1 1]. The Hadoop MapReduce architecture (Figure 3) includes a lot of necessary elements. For example, the Job Parts responsible for task scheduling, monitoring, and job execution. Also, Job Parts assigns tasks to the Task Trackers, who then perform, reduce, and report on these activities. While reducing activities carry out aggregation and create the final result, map tasks analyze inserted data, and gives intermediate key-value pairs by replicating data across numerous nodes, the Hadoop Distributed File System (HDFS) provides fault tolerance. Data from retrieved from HDFS use input format, which divides it into splits for mapping activities [5]. The final result of reduction operations is written back to HDFS using output. In Hadoop MapReduce, this design offers parallel processing, fault tolerance, and effective data storage. HDFS divides files into smaller chunks that are dispersed among nodes, which are divided into name and data nodes and hold responsibility for file operations. Mappers and Reducers used in Map-reduce and Hadoop computing framework, to process data. Key-value pairs produced by mappers are sorted, and to minimize input and output they also may be pre-processed by a "combiner." The large-scale data processing then efficiently managed by reducers which combine and process these pairs before returning the results to HDFS.

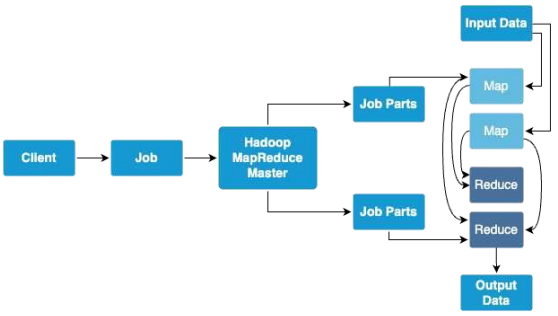

Fig. 3. Hadoop MapReduce Architecture

The aim and objectives of the study: The primary aim of the research is to compare these three distributed data processing frameworks: Apache Spark, Apache Flink, and Hadoop MapReduce. The research contrasts them on such parameters as architecture, performance, scalability, and fault tolerance. The goal is to help organizations make informed decisions about the best data processing platform for their specific requirements.

Objectives:

1. To evaluate performance of each framework with factors like execution speed, throughput, latency, and resource utilization.
2. To assess scalability of each framework, specifically their ability to handle large volumes of data on increased number of nodes in a distributed cluster.
3. To examine fault tolerance capabilities of each framework by analyzing their mechanisms for handling failures.
4. To analyze architectural differences of each framework to determine how they influence their performance, and suitability for various use cases.

Results and Discussion:

Performance:

Spark is known for its in-memory processing capabilities, which enable it to perform significantly faster than MapReduce for iterative and interactive workloads. By keeping data in memory, Spark minimizes disk I/O (Input/Output) and allows for efficient data sharing across multiple computations [8]. It also provides various high-level APIs, such as Resilient Distributed Datasets (RDDs) and DataFrames, which optimize query execution and improve overall performance [18].R offers effective distributed processing by enabling distributed information to be handled in parallel across a cluster, RDDs are intended to improve performance [20].RDDs' capacity to optimize data processing contributes to their performance advantages. RDDs take advantage ofin-memory computing to provide faster access to

data than disk-based activities. They provide data locality, which lowers network costs by allowing data to be processed on the same node where it is stored [20]. The paper [9] focuses on a critical component of big data processing with Apache Spark:maximizing data distribution and dependability using block size and replication factor setups. The authors investigate the influence of various factors on application performance, revealing how fine-tuning might result in more efficient data processing [9]

Recent studies also show ongoing improvements in Apache Spark's in-memory processing capabilities. Author [15] explains Spark ecosystem, highlighting its superior computing power over Hadoop. At the same time, created PokéMem, an addition for Spark that mitigates out-of-memory exceptions and reduces garbage collection overheads by managing untracked memory consumers in Spark's environment. PokéMem's approach to optimizing memory usage boosted Spark's efficiency and execution speed, and it also shows the framework's ongoing development to meet modern data processing demands [15].

Flink's main focus is low latency stream processing. Its streaming engine can handle large volumes of data which is used in fast and responsive analytics applications [17]. According to the research, Flink and Spark are the two most energy-efficient technologies, while Hadoop is the least. This is particularly relevant for applications where energy consumption is a critical consideration. Furthermore, MapReduce usually shows higher latency because of its disk-based operations and lack of in-memory computing [5]. Summing up, Apache Spark and Flink outperform Hadoop MapReduce in terms of performance. Some studies have shown that Apache Spark is able to run almost 100 times faster than Hadoop MapReduce [8].

Scalability: Spark's scalability is primarily attributed to its resilient distributed datasets and directed acyclic graph (DAG) execution model. The Apache Spark execution framework relies significantly on DAGs, which enable effective and reduced data processing workflows. DAGs are a logical execution plan that shows the steps taken to work with distributed data [7]. RDDs allow Spark to efficiently distribute data across a cluster, enabling parallel processing [20]. Additionally, Spark's DAG execution model optimizes task scheduling and minimizes data shuffling, resulting in improved scalability [14]. Flink's scalability stems from its fine-grained dataflow model, which enables efficient parallel processing of data streams [10]. Flink's distributed runtime architecture, coupled with its support for pipelined processing and stateful computations, makes it highly scalable [4]. Flink's ability to handle high throughput streaming data allows it effectively to scale out to larger clusters.MapReduce's scalability is limited by its batch-oriented, two-stage processing model,which involves map and reduce tasks [19]. While MapReduce effectively scales up to handle large datasets, it faces challenges with smaller, more frequent tasks due to the overhead of launching and managing individual map and reduce jobs [5]. Additionally,Map-reduce relies heavily on disk I/O, which can become a bottleneck when processing large volumes of data. In conclusion, Spark and Flink demonstrate superior scalability compared to MapReduce.

Fault Tolerance: Spark employs a fault tolerant mechanism called Resilient Distributed Datasets (RDDs) to enable fault tolerance [18]. RDDs store data inpartitions across the cluster, allowing for the recomputationof lost partitions in case of failures. Spark also supports lineage, a directed acyclic graph of transformations applied to RDDs, which facilitates the recovery of lost data by re-executing transformations on available partitions [14]. Flink, on the other hand, employs a different approach called exactly once processing semantics [17]. It uses a distributed snapshotting technique to capture the state of the processing pipeline at regular intervals. In the event of failure, Flink can roll back to the latest consistent snapshot and resume processing from there, ensuring exactly once semantics [10]. MapReduce offers fault tolerance through replication [11]. It replicates the input data across multiple nodes, ensuring that the failure of single node does not lead to data loss. If a node fails during computation, the MapReduce framework automatically reschedules the failed tasks on other available nodes.Another important point here is the use of specialized tools for fault tolerance.The Fault Tolerant Real-Time Cloud (FTRTC) project is a substantial advancement.

The primary objective is to establish cloud computing infrastructures that can support robust real-time applications, similar to those used in Industry 4.0. This programme is significant as it aims to establish a formalized approach to designing real-time cloud applications that can effectively handle different levels of fault tolerance throughout distributed execution on the cloud [1]. Another innovative approach uses machine learning to enhance fault tolerance mechanisms in the cloud. This model is based on existing knowledge to predict fault instances, so it can improve the efficiency of task allocation in cloud servers [3]. These advancements show a shift towards more intelligent fault tolerance mechanisms, not only bound to the data-processing frameworks' algorithms but to the intelligent cloud environments these frameworks are operating in.

Architecture:

All frameworks that were discussed have distinct architectures,with their features. For instance, a driver application, cluster manager, worker nodesand RDDs serve as the main data abstraction in Apache Spark's master-worker architecture [18]. It supports in-memory processing Spark's design enables it to run interactive and iterative workloads effectively. On the other hand, the master-worker design of Apache Flink additionally includes a Job Manager, Resource Manager, and Task Managers [18]. The architecture of Flink includes cutting-edge features like event time processing and support for stateful computations, and it is intended for both batch and stream processing [10]. Flink is suited for real-time streaming applications because it places an emphasis on low-latency processing and effective memory management.The master-worker design of Hadoop MapReduce, on the other hand, uses a Job Tracker and Task Trackers together with a master worker [19]. It is driven by disk-based processing and focuses on massive batch processing. MapReduce uses the map and reduce functions to handle data that is stored in the Hadoop Distributed File System (HDFS) [19]. Overall, the design of Spark's quick in-memory processing, that of Flink emphasizes low-latency stream processing, and that ofMapReduce emphasizes batch processing with disk-based operations.

II. CONCLUSION

This study offers a thorough analysis of the performance, scalability, fault-tolerance, and comparison of Apache Spark, Apache Flink, and Hadoop MapReduce architectures. By synthesizing existing literature and conducting comprehensive comparative analyses, the research offers a unique perspective on the strengths and limitations of each framework, thereby contributing to the advancement of knowledge in the field of distributed data processing. The outcomes show that switching from MapReduce technology to Apache Spark or Apache Flink can result in significant performance gains. When evaluating the migration, it is crucial to consider the work needed to adapt MapReduce workloads to the new APIs. Due to its maturity, size of the Apache project, market share, and community, Spark currently stands out as the best framework overall. In comparison to Flink, Spark provides a greater set of operations and a wider variety of tools. Nevertheless, Flink has offered novel concepts that have influenced Spark's advancement. Garbage collection cost is reduced by Flink's use of transparent persistent memory management and customized object serialization. Furthermore, explicit iterators in Flink have demonstrated substantial advantages for iterative algorithms, leading to noticeably quicker execution times as compared to MapReduce and Spark. The selection of a framework is based onparticular use cases and specifications for performance, fault tolerance, and real-time processing. Overall, this analysis clarifies the relative merits and contributions of Spark, Flink, and MapReduce, emphasizing Spark as the present best option while recognizing Flink's creative concepts and future development possibilities. Scientific contribution of the study lies in its evaluation of these frameworks, shedding light on their relative merits and providing recommendations for businesses navigating the complex landscape of data processing technologies.

REFERENCES

[1] Abeni L., Andreoli R., Gustafsson H., Mini R., Cucinotta T. Fault Tolerance inReal-Time Cloud Computing // Proc. of the 2023 I E EE 26th International Symposium on Real-Time Distributed Computing (ISORC). – 2023. – P. 170-175;

[2] Azhir E., Hosseinzadeh M., Khan F., Mosavi A. Performance Evaluation of Query Plan Recommendation with Apache Hadoop and Apache Spark // Mathematics.– 2022. – Vol. 10, No. 19. – P. 3517;

[3] Babu P. R., Jimalo K. M., Gadiparthi M., Kumar K. R. N. K. DistributedConsensus and Fault Tolerance Mechanisms Using Distributed Machine Learning //Proc. of the 2023 International Conference on Disruptive Technologies (ICDT). –2023. – P. 11 9-123;

[4] Carbone P., Katsifodimos A., Kth K., Sweden S., Ewen S., Markl V., Haridi S.,Tzoumas K. Apache Flink: Stream and batch processing in a single engine // I E E E Data Engineering Bulletin. – 2015. – Vol. 38;

[5] Ghazi M. R., Gangodkar D. Hadoop, MapReduce and HDFS: A developer'sperspective // Elsevier B.V. – 2015. – Vol. 48. – P. 45–50;

[6] George M. M., Rasmi P. S. Performance Comparison of Apache Hadoop andApache Spark for COVID-19 data sets // Proc. of the 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT). – 2022;Institute of Electrical, Electronics, and Engineers et al. Symposium on Colossals

[7] Data Analysis and Networking (CDAN). – 2016

[8] Jaggi H. S., Kadam S. S. Integration of Spark framework in supply chain management // Procedia Computer Science. – 2016. – Vol. 79. – P. 1013–1020;

[9] Joshi B. Y., Shankar P., Sawai D. Performance Tuning Of Apache SparkFramework In Big DataProcessing with Respect To Block Size And Replication Factor

[10] Katsifodimos A., Schelter S. Apache Flink: Stream analytics at scale // Proc. of the 2016 I E EE International Conference on Cloud Engineering Workshop (IC2EW).

[11] Merla P., Liang Y. Data analysis using Hadoop MapReduce environment// Proc.of the 2017 I E EE

International Conference on Big Data (Big Data). – 2017. – P.4783–4785;

[12] Natesan P., Sathishkumar V. E., Mathivanan S. K., Venkatasen M., Jayagopal P., Allayear S. M. A Distributed Framework for Predictive Analytics Using Big Data and MapReduce Parallel Programming // Mathematical Problems in Engineering. –2023. – P. 1–10;

[13] Salloum S., Dautov R., Chen X., Peng P. X., Huang J. Z. Big data analytics on Apache Spark // International Journal of Data Science and Analytics. – 2016. – Vol. 1,No. 3–4. – P. 145–164;

[14] Shaikh E., Mohiuddin I., Alufaisan Y., Nahvi I. Apache Spark: A big data processing engine // Proc. of the 2019 2nd I E EE Middle East and North Africa Communications Conference (MENACOMM). – 2019. – P. 1–6;

[15] Tran Q., Nguyen B., Nguyen L., Nguyen O. Big Data Processing With Apache Spark // TraVinh University Journal Of Science. – 2023;

[16] Ullah F., Dhingra S., Xia X., Babar M. A. Evaluation of distributed data processing frameworks in hybrid clouds // Journal of Network and Computer Applications. – 2024. – Vol. 224. – P. 103837;

[17] Apache Software Foundation. Apache Flink Documentation. – 2024. –Available at:https://nightlies.apache.org/flink/flink-docs-stable/;– 2016. – P. 193–193;Apache Software Foundation. Apache Spark Documentation. – 2023. Available at: https://spark.apache.org;

[18] Aziz K., Zaidouni D., Bellafkih M. Big Data Optimisation Among R D D s Persistence in Apache Spark// Communications in Computer and Information Science. 2018. – P. 29–40