International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST218475

Study on Big Data Frameworks

Adriano Fernandes*, Jonathan Barretto, Jonas Fernandes
Department of Computer Science, Don Bosco College of Engineering, Goa, India

Article Info
Volume 8, Issue 4
Page Number : 491-499

Publication Issue
July-August-2021

Article History
Accepted : 02 Aug 2021
Published : 08 Aug 2021

ABSTRACT

Big data analytics is becoming more and more popular every day as a tool for evaluating large volumes of data on demand. Apache Hadoop, Spark, Storm, and Flink are four of the most widely used big data processing frameworks. Although all four architectures support big data analysis, they vary in how they are used and the infrastructure that supports it. This paper defines a general collection of main performance metrics, which include Processing Time, CPU Use, Latency, Execution Time, Performance, Scalability, and Fault-tolerance, and contrasting the four big data architectures against these KPIs in a literature review. When compared to Apache Hadoop and Apache Storm frameworks for non-real-time results, Spark was found to be the winner over multiple KPIs, including processing time, CPU usage, Latency, Execution time, and Scalability. In terms of processing time, CPU consumption, latency, execution time, and performance, Flink surpassed Apache Spark and Apache Storm architectures.

Keywords - Latency, Scalability, Fault-Tolerance

I. INTRODUCTION

Roger Magoulas was the first to coin the word "big data" in 2005. He defines it as a large volume of data that conventional data processing methods cannot handle or process. This information comes from a variety of outlets, including social networking platforms, photographs, sensors, laptops, and search queries. There are some key characteristics that distinguish "large data" from other types of data, including its enormous scale and the fact that it is made up of diverse and distinct data sets. Furthermore, standard data processing methods cannot be used to process it.

Big data analysis techniques are now one of the most relevant developments as a result of massive data and the uncertainty that can occur as data is used for analyzing purposes. Rather than using conventional databases or programs, these tools allow users to organize and access all data. The aim of this literature review is to provide an analysis of four big data systems and compare them across a range of predefined main success metrics. The following sections make up this paper: the first part discusses the key features of big data, also known as the V's of big data and challenges in handling big data. Following that, several big data architectures are discussed, including Hadoop, Spark, Storm, and Flink.

II. CHARACTERISTICS OF BIG DATA

Big Data is composed of generic demands (volume, variety, and velocity), which are referred to as the 3Vs. [1]. The attributes of Big Data have recently progressed from the 3Vs to the 6Vs, with the addition of the features of value, veracity, and variability. Since joining the scheme, the last three are referred to as acquired Big Data specifications [1]. The 6Vs of Big Data are depicted in Figure 1.

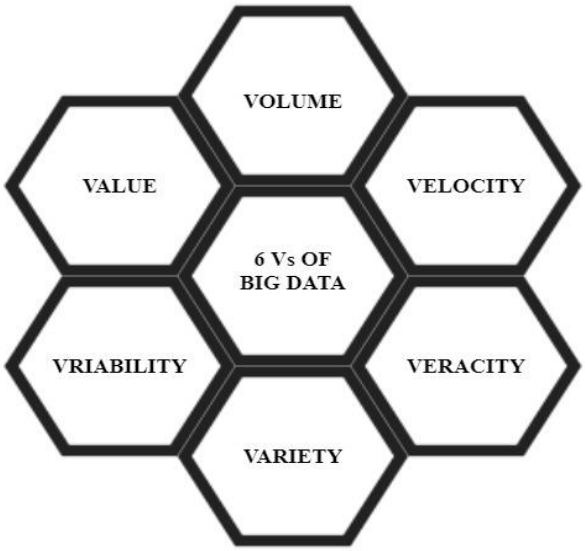

Figure. 1 6V's of big data

A. Volume

The amount or scale of the data is referred to as volume. Terabytes (TB), Petabytes (PB), Zettabytes (ZB), and Exabytes (EB) are the sizes of Big Data [1]. Organizations like Facebook, YouTube, Google, and NASA have vast volumes of data, posing new problems in terms of storing, retrieving, analyzing, and processing it. How we transfer and use data has evolved as a result of the use of Big Data rather than conventional storage [1].

B. Variety

The various types of data produced are referred to as variety. Different measurements may be used to quantify variety, such as structure, which allows one to distinguish between structured, semi-structured, and unstructured data, or processing volume, as in batch versus stream processing.

C. Variability

Variability applies to data that isn't consistent, can't be quickly processed, and is difficult to handle. Researchers face a big challenge in explaining variable data [1].

D. Velocity

Velocity indicates the rate at which Big Content is generated, transferred, stored, and examined. Because of the high prices, Velocity poses new testing problems for data scientists . The data is left behind when the user has to retrieve or manipulate the data and the operation is too slow [1].

E. Veracity

The consistency of data being analyzed is referred to as veracity. The veracity of the data source is therefore determined by the precision of the data [1].

F. Value

The intent or business result that the data carries in to aid the decision-making process is referred to as value [1].

III. CHALLENGES IN HANDLING BIG DATA

A. Storage

While today's hard drives have capacities in the terabytes, the quantity of data created over the internet on a daily basis is on the scale of exabytes. Though the data created in education is not as vast as all of the data generated on the internet, it is sufficient and will continue to grow in the future. Traditional RDBMS technologies will be unable of storing or processing such large amounts of data. Databases that don't utilize standard SQL-based queries are used to solve this problem. Data is compressed at rest and in memory using compression technologies [5].

G. Analysis

Because the data created by various types of online learning sites differs in structure and amount, analyzing the data might take a long time and require a lot of resources. Scaled out architectures are used to process data in a distributed manner to solve this. Data is broken down into smaller chunks and processed by a large number of computers spread throughout the network, then aggregated [5].

H. Reporting

Statistical data is displayed in the form of numbers in traditional reports. Traditional reports become difficult to comprehend by humans when vast amounts of data are involved. In certain instances, the reports must be presented in a way that allows them to be easily identified by simply glancing at them [5].

IV. BIG DATA FRAMEWORKS

A. Hadoop

Apache Hadoop is a well-known Big Data platform with a sizeable community of users. It was created to prevent the low performance and complexity that come with utilizing existing technologies to handle and analyze large amounts of data. Because of its parallel clusters and distributed file system, Hadoop has the ability to process massive data volumes quickly. Hadoop, does not copy the whole remote data into memory to do computations. it runs operations on stored data. As a result, Hadoop relieves the network and servers of a significant amount of communication burden. Another feature of Hadoop is its ability to run applications in a distributed environment while providing fault tolerance. To ensure this, it replicates data on servers to prevent data loss[4].

Hadoop architecture consists of 3 main layers namely HDFS, YARN and MAP REDUCE shown in fig. 2 [1].

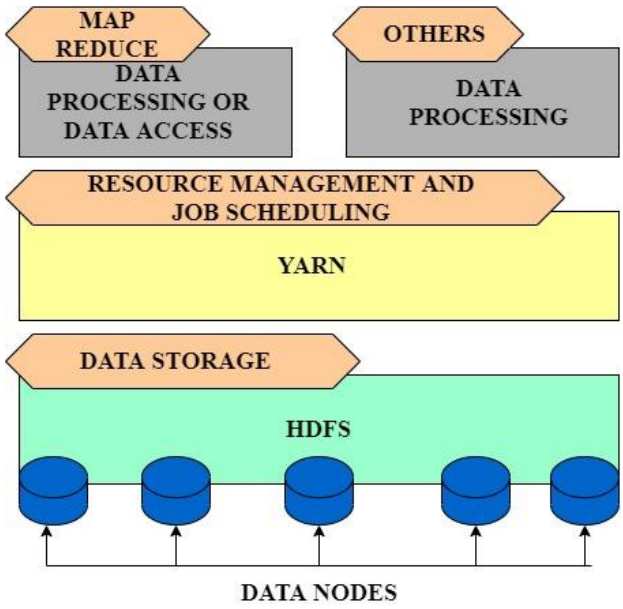

Figure. 2 Hadoop architecture

HDFS:

Hadoop applications use the Hadoop Distributed File Solution (HDFS) as their primary data storage system. HDFS implements a distributed file system that enables high-performance access to data across highly scalable Hadoop clusters using a Name Node and Data Node architecture.

Hadoop is an open source distributed processing platform for large data applications that controls data processing and storage. HDFS is a vital part of the numerous Hadoop ecosystem technologies. It provides a dependable method for managing huge data pools and supporting related big data analytics applications.

Map Reduce:

The Apache Hadoop ecosystem includes MapReduce. In the Hadoop ecosystem, the MapReduce component improves the processing of large amounts of data by employing distributed and parallel algorithms. This programming approach is used to evaluate massive amounts of data gathered from internet users in social platforms and e-commerce.

MapReduce has two basic tasks: map and reduce. We complete the first job before moving on to the second. We divided the incoming dataset into pieces in the

map task. These chunks are processed in parallel by the Map job. We utilize outputs as inputs for the reduce jobs in the map. Reducers break down the intermediate data from the maps into smaller tuples, which decreases the number of jobs and leads to the framework's ultimate output.

The MapReduce framework improves job scheduling and management. The framework re-executes the unsuccessful tasks. This framework is simple to use, even for programmers who are unfamiliar with distributed processing[1].

Yarn:

Hadoop's cluster resource management mechanism is Apache YARN (Yet Another Resource Negotiator). YARN was added to Hadoop to enhance the MapReduce implementation, but it's flexible enough to accommodate other distributed computing paradigms as well.

YARN provides APIs for obtaining and dealing with cluster resources, although user code seldom uses these APIs directly. Users instead utilize higher-level APIs offered by distributed computing frameworks, which are based on YARN and mask resource management specifics from the user.

B. Spark

Spark , a MapReduce-based project that was first created at the University of California, Berkeley and is now an Apache top-level project, is based on MapReduce but tackles a lot of the shortcomings. like Hadoop, it allows for iterative computing. It uses in-memory computing to enhance performance and resource use. Both science and industry have adopted Spark's processing methodology. Resilient Distributed Datasets (RDD), which store data in memory and enable fault tolerance without replication , are the major abstractions utilized in this research. Read-only distributed shared memory is how RDDs are defined. This system, shown in Figure 3 [2], simplifies the learning process by storing intermediate results in memory, reducing the number of read and write operations required substantially. Spark's speed was shown when it won the Daytona GraySort Benchmark Contest in October 2014 . Hadoop/MapReduce previously held the record for sorting 102.5 TB on 2100 nodes in 72 minutes. Spark sorted 100 TB on 206 nodes in 23 minutes, three times quicker with a tenth of the servers.

Furthermore, it has been highlighted that Spark is easier to program, with part of the rationale being that it can be programmed in Java, R, Python, or Scala. Spark includes the MLlib and GraphX libraries for machine learning tasks, as well as a number of Spark implementations in the current version of the Mahout library[2].

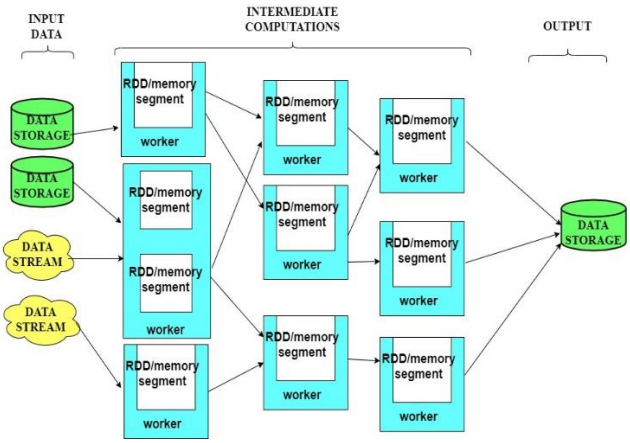

Figure. 3 processing model of spark

C. Storm

Storm is a real-time data processor that was designed to solve the shortcomings of existing processors in gathering and analyzing social media streams. Storm development began at BackType, a social media analytics firm, and was continued by Twitter following a 2011 acquisition. In September 2014, the project was open sourced and became an Apache top-level project . Real-time processing is becoming increasingly important in the machine learning field , and as a result, Storm is seeing greater use in both commercial and research contexts. Spouts and bolts make up the Storm's architecture. Topologies are networks of spouts and bolts that are represented as directed graphs. Figure 4 [2] shows an example of this. The project is largely written in Clojure,

however all APIs were initially written in Java to encourage wider adoption. It now incorporates Thrift , a cross-language development framework that enables topologies to be created and submitted in any programming language . Storm employs real-time streaming but also provides micro-batch via its Trident API [2].

Storm was designed to be a stand-alone system independent of Hadoop, however since Hadoop's migration to YARN, effort has been done to connect the two projects[2].

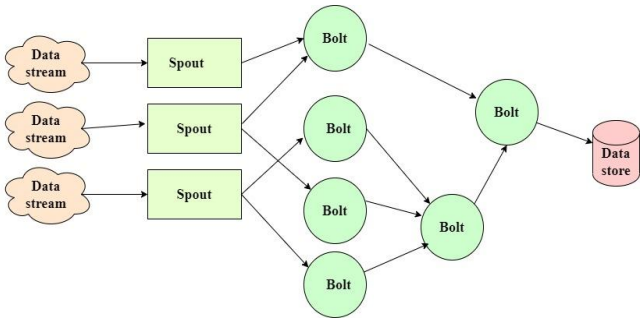

Figure 4. processing model of storm

D. Flink

Flink is a free and open source framework for batch and real-time data processing. It has numerous advantages, including fault tolerance and large-scale computing. Flink uses a programming approach similar to MapReduce. Flink, unlike MapReduce, provides extra high-level operations including join, filter, and aggregate. On a more abstract level, it provides many APIs that allow users to begin distributed computation in a transparent and simple manner. Flink ML is a machine learning package that includes a variety of learning algorithms for building scalable and quick Big Data applications [6]. Flink's architecture is depicted in Figure 5 [3]. The base layer's storage layer will read and write data from various sources like HDFS, local files, and so on. The cluster manager is located in the deployment and resource management layer, and it is responsible for handling the tasks of planning, tracking, and managing resources. This layer also includes the program's execution environment, which is made up of clusters or cloud environments. It has a distributed stream data flow engine with a kernel layer for real-time processing. Interface layers for batch and streaming operations are included in the framework software. In the top layer, which is a library, the system is developed in Java or Scala programming language. It is then submitted to the compiler for conversion using the Flink optimizer to improve its performance [1].

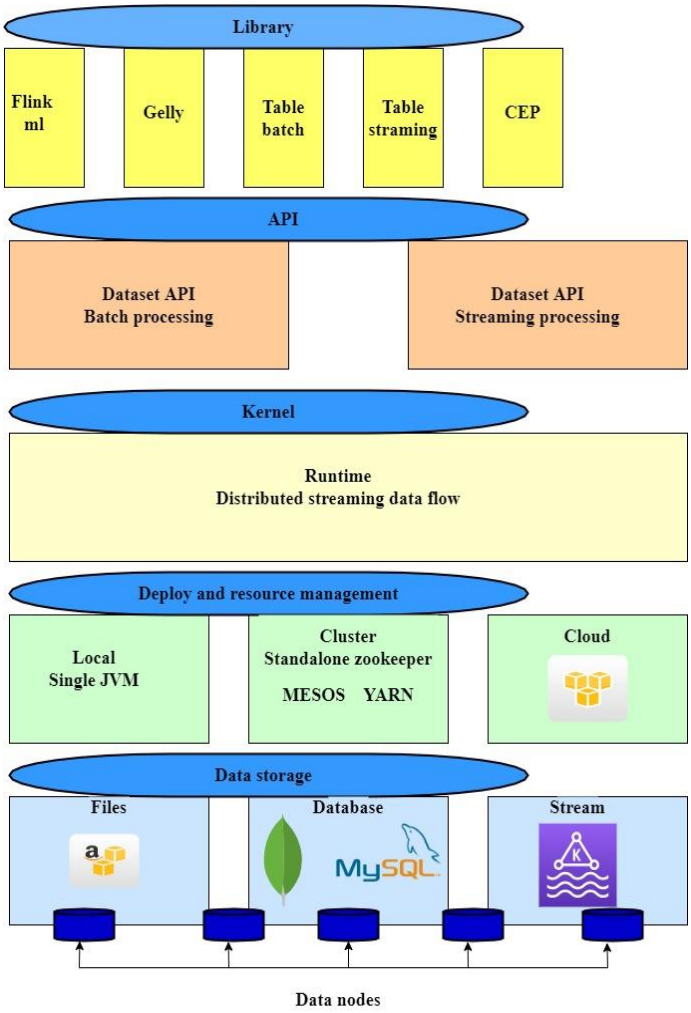

Figure. 5. Flink architecture

V. ANALYSIS OF BIG DATA FRAMEWORKS

A. Scalability

The capacity of a system to react to increasing loads is known as scalability. Scale up (vertically) and scale out are the two styles (horizontally). Scale up refers to upgrading the hardware setup, while scale out refers to adding additional hardware. Our study's four systems are all horizontally scalable. It implies that

we can add as many nodes to the cluster as required [1].

B. Data delivery guarantee

In the case of a malfunction, message transmission assurances are used. It can be divided into two forms based on the four frameworks listed above: exactly once delivery and at-least-once delivery. The term "precisely once delivery" suggests that the information will not be duplicated or misplaced, and will only be sent to the intended recipient once. At-least-once delivery, on the other hand, means that the message is delivered several times and at least one of them is successful. Furthermore, the message can be duplicated without losing its integrity[1].

C. Computation mode

In-memory computing or the more "traditional" mode, where computation results are written back to the disc, are two options for computation mode. In-memory computing is quicker, but it has the drawback of causing the contents to be lost if the computer is switched off [1].

D. Auto scaling

Auto-scaling is the process of automatically scaling cloud services up or down depending on the situation [1].

E. Iterative computation

In the absence of a real solution or where the expense of a real solution is prohibitively high, iterative computation refers to the use of an iterative approach to estimate an estimated solution [1].

TABLE 1. BIG DATA FRAMEWORK FEATURES OVERVIEW

Features	**Hadoop**	**Spark**	**Storm**	**Flink**
Processing mode	Batch	Batch and stream	Stream	Batch and stream
Scalability	Horizontal	Horizontal	Horizontal	Horizontal
Data delivery guarantee	precisely once	Precisely once	At least once	Precisely once
Computation mode	Drive based	In memory	In memory	In memory
Auto scaling	Yes	Yes	No	No
Supported programing languages	Java	Java, Scala, python	Java	Java

VI. LITERATURE REVIEW OF BIG DATA FRAMEWORKS

A. Spark vs Storm

Spark and Storm both have fault tolerance and scalability, but they use different processing models. Spark processes events in micro-batches, whereas Storm processes them one by one. Because of this disparity in process management, Spark has a latency of seconds whereas Storm has a latency of milliseconds.

The Spark method allows you to build streaming tasks in the same manner that batch jobs are written, allowing you to reuse the majority of the code and business logic. Storm focuses on stream processing. This framework employs a fault-tolerant method to finish calculations or pipeline multiple computations on an event as it enters the system.

B. Hadoop vs Spark vs Flink

1) Data Processing:

Apache Hadoop is a batch processing framework. It takes a big data collection as input, processes it all at once, and outputs the result. Batch processing is highly efficient when dealing with large amounts of data. Because of the quantity of the data and the

computing capability of the system, an output is delayed. Apache Spark is a Hadoop Ecosystem component. It's mostly a batch processing system, although it can also handle stream processing. Apache Flink is a single runtime that can handle both streaming and batch processing.

2) Streaming engine:

Map-reduce is a batch-oriented processing technology used by Hadoop. It accepts a huge data collection as input, processes it, and outputs the result all at once. Apache Spark Streaming is a micro-batch processing system for data streams. Each batch is made up of events that occurred within the batch period. However, for use cases where we need to analyze massive amounts of live data and give results in real time, this is insufficient. Flink is a streaming engine that supports streaming, SQL, micro-batch, and batch workloads . A batch is a collection of streaming data that has been limited in size.

3) Computation Model:

Hadoop's MapReduce uses a batch-oriented approach. Batch is a method of processing data while it is still in motion. It takes a vast quantity of data, processes it, and then outputs the results. Spark supports Micro-batching . Micro-batches are a type of computer model that essentially collects and then processes data. Flink uses an operator-based streaming paradigm with a continuous flow. A continuous flow operator processes data as soon as it enters, without having to wait for it to be collected or processed.

4) Performance:

Hadoop Only supports batch processing . Because it does not handle streaming data, it performs slower than Spark, and Flink. Apache Spark is regarded as the most developed community. However, because it employs micro-batch processing, its stream processing is less efficient than Apache Flink. When compared to other data processing systems, Apache Flink performs quite well. When comparing Hadoop versus Spark versus Flink, Apache Flink employs native closed loop iteration operators, which makes machine learning and graph processing quicker.

5) Fault tolerance:

Hadoop's MapReduce is a fault-tolerant programming model. In the event of a Hadoop failure, there is no need to restart the programme from the beginning. Apache Spark recovers lost work and provides exactly-once semantics out of the box with no additional code or setup. Apache Flink's failure tolerance technique is based on Chandy-Lamport distributed snapshots. Because the method is lightweight, it can sustain high throughput rates while also providing excellent consistency guarantees.

6) Iterative Processing:

In Hadoop Iterative processing is not supported. Spark iterates data in bunches. Each iteration must be scheduled and executed independently in Spark. Flink uses a streaming design to iterate data. Flink may be told to process only the sections of the data that have changed, resulting in a substantial increase in work performance.

7) Latency:

Hadoop's MapReduce architecture is slower than others since it is designed to accommodate a wide range of data formats, structures, and volumes. As a result, Hadoop has a longer latency than Spark and Flink. Spark is also a batch processing system, but it is quicker than Hadoop's MapReduce since it caches much of the input data in memory through RDD and maintains intermediate data in memory, finally writing the data to disc upon completion or whenever needed. Apache Flink's data streaming runtime delivers low latency and high throughput with minimal setup work.

8) Processing Speed:

MapReduce is slower than Spark and Flink in Hadoop. The slowness is due to the nature of MapReduce-based processing, which creates a lot of intermediate data and a lot of data transferred between nodes, resulting in a lot of disc IO delay. In addition, it must store a large amount of data on disc for synchronization between stages in order to allow

Job recovery from errors. In addition, MapReduce does not support caching all subsets of data in memory. Apache Spark is quicker than MapReduce because it caches much of the input data in memory using RDDs and maintains intermediate data in memory before writing it to disc when it's done or when it's needed. Spark is 100 times quicker than MapReduce, demonstrating the superiority of Spark over Hadoop MapReduce. Because of its streaming nature, Flink analyses data quicker than Spark. Flink improves the job's performance by telling it to only process the data that has changed.

9) Easy to use:

Hadoop's MapReduce developers must manually code each action, making it extremely difficult to work with. In Spark It's simple to programme thanks to its large number of high-level operators. Flink has high-level operators as well.

10) Cost:

In Hadoop since MapReduce does not seek to keep everything in memory, it may generally run on less costly hardware than other competitors. Since Spark requires a lot of RAM to function in-memory, increasing the amount of RAM in the cluster gradually raises the cost. Because Apache Flink takes a lot of RAM to function in-memory, its price progressively rises.

11) Scalability:

In Hadoop MapReduce offers tremendous scalability potential and has been deployed on tens of thousands of nodes in production. Spark is very scalable; we can keep adding nodes to the cluster indefinitely. Apache Flink is likewise very scalable; we can keep adding nodes to the cluster indefinitely.

12) Real time analysis:

In Hadoop, MapReduce fails when it comes to real-time data processing since it was built to process large volumes of data in batches. Spark is capable of processing real-time data, such as data from real-time event streams, at a pace of millions of events per second. Flink is mostly used for real-time data analysis, although it can also handle bulk data processing .

13) Caching:

In Hadoop MapReduce is unable to cache data in memory for future needs. Spark has the ability to store data in memory for future iterations, which improves performance. To improve performance, Flink can store data in memory for future iterations .

14) Duplication elimination:

Hadoop does not support duplicate removal. Spark processes each record just once, preventing duplication. Apache Flink processes each record precisely once, preventing duplication. Streaming programmes have the ability to keep a custom state throughout computation. In the event of a failure, Flink's checkpointing system ensures that the state is only updated once.

TABLE 2. Comparing The Four Big Data Frameworks

	Hadoop	**Spark**	**Flink**	**storm**
Cost	Cost efficient	Expensive	Expensive	Not compared
Performance	Less efficient	Efficient	Most efficient	Not compared
Processing time	Slower	Faster than Hadoop	Faster than spark	Not compared
CPU consumption	CPU consumption is high	CPU consumption is high	CPU consumption is low	Not compared

Iterative processing	Does not support	Supports	Supports	Not compared
latency	Latency is high	Latency is large	Latency is low	Latency is low
Execution time	Execution time is high	Execution time is low	Execution time is low	Not compared
Duplicate elimination	Does not support	supports	supports	Not compared
Supports real time analysis	No	Yes	Yes	Not compared

VII. CONCLUSION

In this article, we examined and compared the efficiency of four frameworks: Hadoop, Flink, Spark, and Storm, using various primary performance indicators. The findings of this analysis indicate that Flink outperformed the other structures indicating that it is the strongest. Finally, they can achieve high computational efficiency (HPC). In the future, by taking these metrics into account while evaluating the efficiency of the four structures, each system will be able to improve on any metric that has a low impact on obtaining HPC. As such, we aspire to seeing enhancements in some of these frameworks.

VIII. REFERENCES

[1]. Safaa Alkatheri,Samah Abbas,Muazzam Siddiqui "A comparative study of big data frameworks" International Journal of Computer Science and Information Security, Vol 17,No.1,pp. 66-71 2019.

[2]. Sara Landset, Taghi M. Khoshgoftaar, Aaron N. Richter* and Tawfiq Hasanin "A survey of open source tools for machine learning with big data in the Hadoop ecosystem" pp. 13-16 ,2015

[3]. P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," Bull. IEEE Comput. Soc. Tech. Comm. Data Eng., vol. 36, no. 4, pp. 30, 2015.

[4]. Ahmed Oussous , Fatima-Zahra Benjelloun , Ayoub Ait Lahcen , Samir Belfkih , "Big Data technologies: A survey" , pp.437-438,2018.

[5]. Katrina Sin , Loganathan Muthu "Application of big data in education data mining and learning analytics – a literature review" Vol. 5, No. 4, pp. 1035-1036 , 2015.

[6]. Wissem Inoubli,Haithem Mezni,Sabeur Aridhi,Alexander Jung "Big Data Frameworks:A Comparitive Study" ,Future Generation Computer Systems, pp. 6-7, 2018.

Cite this article as :

Adriano Fernandes, Jonathan Barretto, Jonas Fernandes "Study on Big Data Frameworks", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 8 Issue 4, pp. 491-499, July-August 2021. Available at doi : https://doi.org/10.32628/IJSRST218475
Journal URL : https://ijsrst.com/IJSRST218475