WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A Comparative Study on Hadoop MapReduce and Apache Spark Framework for Big Data Analytics

Dr. Vandana Vijay [a], Dr. Vaibhav Sharma [b], Dr. Vandana Srivastava [c], Dr. Vipin Kumar Jain [d]

[*]*Assistant Professor, S. S. Jain Subodh P G College, Jaipur, India*

[a]*vandanavijay161978@gmail.com*, [b]tamvaibhav238@gmail.com, [c]svandana94@gmail.com, [d]*prateeshvip79@gmail.com*

DOI: https://doi.org/10.55248/gengpi.5.0224.0601

ABSTRACT

In today internet world, due to the current advent of new technologies, mobile devices, and communication media like social networking sites, the amount of data generated every year is growing at a very high rate. The growth of this generated data is beyond our imagination. It is impossible to store these huge data sets in RDMBSs like MySQL, as there are no specific formats of the data and that can be in either text or image formats. It requires the need of technologies which can easily manage and process huge volumes of structured and unstructured data in real-time and can protect data privacy and security. Big data technologies like MapReduce, Apache Flume, and Apache Spark can capture, store, and analyze this huge amount of data in very efficient and less costly manner. Spark and MapReduce programming frameworks provide an effective open-source solution for managing and analyzing the Big Data. MapReduce is a high-performance distributed Big Data programming framework. It processes the data in batch processing environment. On the other hand, Apache Spark is a scalable distributed in-memory data processing engine. It processes the data in both batch and real time environment. It uses Resilient Distributed Datasets (RDD) and Directed Acyclic Graph (DAG) for data processing. In this paper, a review on Hadoop MapReduce and Apache Spark have been made by comparing them on various parameters like performance, streaming, fault tolerance, storage, language support, and reliability.

Keywords: Big Data Analytics, Hadoop MapReduce, Spark Framework.

1. Introduction to Big Data and Hadoop

Big data is a wide term that covers the nontraditional strategies and technologies used to process, organize, and gather insights from large datasets. It is not shocking or new that to work with the data that is not in the capability of a single computer system was a tedious task. With the introduction of big data with Hadoop, gave a lot of ease and flexibility to store this 'big data'. The data which follows following criteria is considered as big data, The most important feature of Hadoop which makes it different from spark is Hadoop works on batch processing and spark works on stream processing. Big data' could be found in three forms: Structured, Un-structured, Semi-structured [7]. The Apache Hadoop software library is a framework that allows for the distributed processing of big data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly available service on top of a cluster of computers, each of which may be prone to failures.

The organization of paper is as follows: Section 1 describes the concepts of Big Data and its technologies. Section 2 discuss the Hadoop MapReduce programming framework along with its two important components. Section 3 explain the Apache Spark framework working model. Section 4 provides a comparative analysis of Hadoop MapReduce and Apache Spark on basis of different parameters. Finally, the paper is concluded in Section 5.

2. Hadoop MapReduce

Hadoop is an Apache open-source framework written in Java. It allows distributed computing environment for processing of large amount of data across cluster of computers in parallel manner [6]. It has two important components: (1) Hadoop Distributed File System (HDFS) and (2) MapReduce framework. HDFS component of Hadoop supports the storage of the big data within distributed environment while the MapReduce component provides the processing of the information or data in an efficient manner. Hadoop distributed file system cluster is composed of a single NameNode and multiple DataNodes. NameNode is a master server, which manages system metadata. It also maintains file system namespace and maps data from database to DataNode. MapReduce is a programming framework which allows performing parallel and distributed processing on large data sets in a distributed environment using many nodes [5]. It consists of two important parts: A Job Tracker (Master Node) and multiple Task Trackers (Slave Nodes). It is shown in figure 1. Apart from these components, Hadoop has grown into a complex ecosystem, including a range of software projects and some commercials tools.

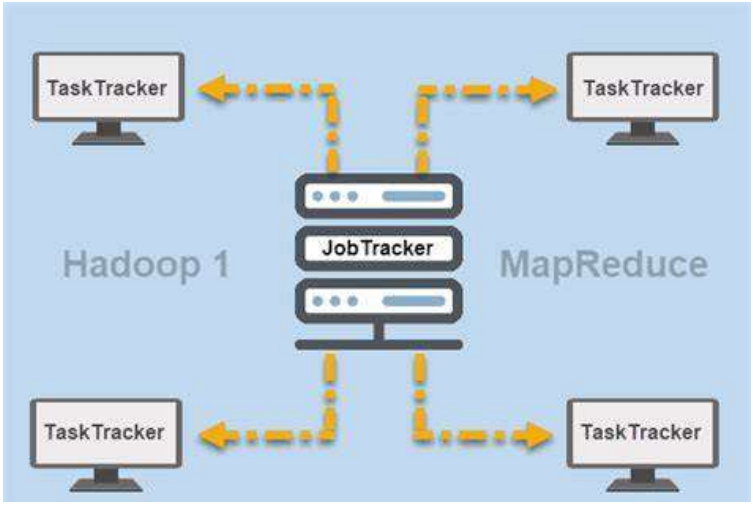

Fig 1. **Hadoop MapReduce**

2. Apache Spark

Apache Spark is a data processing framework that can rapidly operate processing duties on very massive information sets and can additionally distribute information processing duties throughout a couple of computers, either on its very own or in tandem with different allotted computing tools. It is a framework for performing general data analytics on distributed computing cluster like Hadoop. It provides in memory computations for increase speed and data process over MapReduce. It runs on top of existing Hadoop cluster and access Hadoop data store (HDFS), can also process structured data in Hive and Streaming data from HDFS, Flume, Kafka, Twitter [1]. The spark context is the driver program that is responsible for the creation of RDD (resilient Distributed Datasets).The RDD represents a collection of items distributed across the cluster that can be manipulated in parallel. The datasets get converted into blocks of data but into same RDD. The blocks are called atoms of the dataset. Spark context assigns an executor to each worker node for instance py4J whose responsibility is to transform by default spark context session into java-spark session for further processing of data. The transformation used most commonly are filter (), map(),join(), flatmap(). The link between spark context and worker node is cluster manager. This manager works in three modes- standalone YARN, MESOS. It is shown in figure 2.

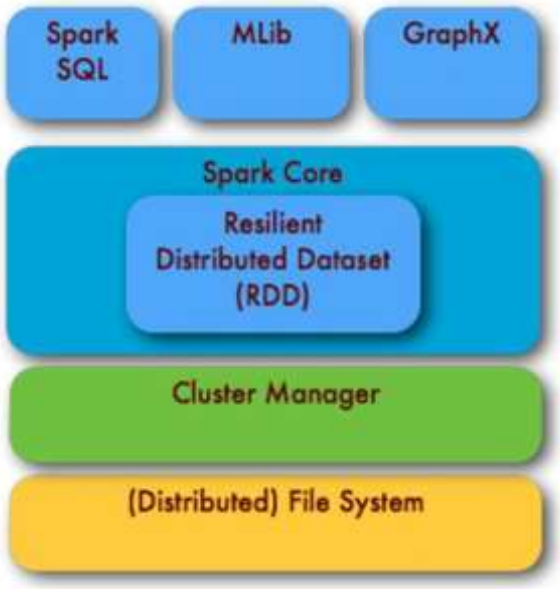

Fig 2. **Apache Spark**

3. Difference between Hadoop MapReduce and Apache Spark

Table 1. provides a comparative analysis of Hadoop MapReduce and Apache Spark on basis of different parameters namely speed, Easy to Manage, Real-time analysis, latency, Interactive mode, Streaming, Ease of use, Recovery, Scheduler, Fault tolerance, Security, etc.

Table 1: Difference between Hadoop MapReduce and Apache Spark

S.No.	Parameter	Apache Spark	MapReduce
1	**Speed**	Apache Spark runs applications up to 100x faster in memory and 10x faster on disk than Hadoop. Because of reducing the number of the reading/write cycle to disk and storing intermediate data in-memory Spark makes it possible.	MapReduce reads and writes from disk, as a result, it slows down the processing speed.
2	**Difficulty**	Spark is easy to program as it has tons of high-level operators with RDD – Resilient Distributed Dataset.	In MapReduce, developers need to hand code each operation which makes it very difficult to work.
3	**Easy to Manage**	Spark can perform batch, interactive and Machine Learning and Streaming all in the same cluster. As a result, makes it a complete Data analytics engine. Thus, no need to manage different component for each need.	As MapReduce only provides the batch engine. Therefore, dependent on other engines.
4	**Real-time analysis**	It can process real-time data i.e., data coming from the real-time event streams at the rate of millions of events per second.	MapReduce fails when it comes to real-time data processing as it was designed to perform batch processing on voluminous amounts of data.
5	**latency**	Spark provides low-latency computing.	MapReduce is a high latency computing framework.
6	**Interactive mode**	Spark can process data interactively	MapReduce doesn't have an interactive mode.
7	**Streaming**	Spark can process real-time data through Spark Streaming.	MapReduce, can only process data in batch mode.
8	**Ease of use**	Spark is easier to use. Since its abstraction (RDD) enables a user to process data using high-level operators.	MapReduce is complex. Therefore, require low-level APIs to process the data, which requires lots of hand coding.
9	**Recovery**	RDDs allows recovery of partitions on failed nodes by re-computation of the DAG while also supporting a more similar recovery style to Hadoop by way of checkpointing, to reduce the dependencies of an RDDs.	MapReduce is naturally resilient to system faults or failures. So, it is a highly fault-tolerant system.
10	**Scheduler**	Due to in-memory computation spark acts its own flow scheduler.	MapReduce needs an external job scheduler for example, Oozie to schedule complex flows.
11	**Fault tolerance**	Spark is fault-tolerant. Therefore, there is no need to restart the application from scratch in case of any failure.	Like Apache Spark, MapReduce is also fault-tolerant, so there is no need to restart the application from scratch in case of any failure.
12	**Cost**	As spark requires a lot of RAM to run in-memory. Thus, increases the cluster, and its cost.	MapReduce is a cheaper option available while comparing it in terms of cost.

13	**Security**	Spark is little less secure in comparison to MapReduce because it supports the only authentication through shared secret password authentication.	Hadoop MapReduce is more secure because of Kerberos, and it also supports Access Control Lists (ACLs) which are a traditional file permission model.
14	**Language Developed**	Spark is developed in Scala	Hadoop MapReduce is developed in Java.
15	**Category**	It is data analytics engine.	It is basic data processing engine.
16	**Line of code**	Apache Spark is developed in merely 20000 lines of codes.	Hadoop 2.0 has 1,20,000 lines of codes
17	**OS support**	Spark supports cross-platform	Hadoop MapReduce also supports cross-platform.
18	**Programming Language support**	Scala, Java, Python, R, SQL	Primarily Java, other languages like C, C++, Ruby, Groovy, Perl, Python are also supported using Hadoop streaming.
19	**SQL support**	It enables the user to run SQL queries using Spark SQL.	It enables users to run SQL queries using Apache Hive.
20	**Scalability**	Spark is highly scalable.	MapReduce is also highly scalable.
21	**Machine Learning**	Spark has its own set of machine learning ie MLlib.	Hadoop requires machine learning tool for example Apache Mahout.
22	**Caching**	Spark can cache data in memory for further iterations. As a result, it enhances system performance.	MapReduce cannot cache the data in memory for future requirements. So, the processing speed is not that high as that of Spark.
23	**Hardware Requirement**	Spark needs mid to high-level hardware.	MapReduce runs very well on commodity hardware.

4. Conclusion

We have entered in an era of billion or trillions of data that is also called Big Data. The paper describes the concept of Big Data and the technology associated to deal with big data (Spark, Hadoop, HDFS, MapReduce). The differences between Apache Spark vs Hadoop MapReduce shows that Apache Spark is much-advance cluster computing engine than MapReduce. Moreover, Spark can handle any type of requirements (batch, interactive, iterative, streaming, graph) while MapReduce limits to Batch processing. Also, Apache Spark is growing very quickly and replacing MapReduce. The study confirms that Apache Spark outperforms MapReduce by a dramatic margin, and as the data grows Spark becomes more reliable and fault tolerant. Spark performs far better than MapReduce. It demonstrates that Spark will become a possible replacement of MapReduce soon.

References:

1. *Vijay, V. (2019). Survey on Role of Hadoop in Cloud Computing Environment. Journal of Emerging Technologies and Innovative Research ISSN, 2349, 5162.*
2. *Srivastava, V., Vijay, V., & Sharma, V. (2023, September). A PROPORTIONAL STUDY OF CLOUD, FOG AND MIST TECHNOLOGY FOR IOT BASED APPLICATIONS. International Research Journal of Modernization in Engineering Technology and Science, Volume:05, Issue:09, Impact Factor- 7.868, e-ISSN: 2582-5208.*
3. *Singh, S., & Pareek, A. (2022). Multilingual Sentiment Analysis of Tweet using a new Model SATV. NeuroQuantology, 20(6), 7478.*
4. *Vijay, V., & Nanda, R. (2021). Query caching technique over cloud-based MapReduce system: A survey. In Rising Threats in Expert Applications and Solutions: Proceedings of FICR-TEAS 2020 (pp. 19-25). Springer Singapore.*
5. *Vijay, V., & Nanda, R. (2022, July). A Priori Study on Factors Affecting MapReduce Performance in Cloud-Based Environment. In Proceedings of Seventh International Congress on Information and Communication Technology: ICICT 2022, London, Volume 3 (pp. 509-515). Singapore: Springer Nature Singapore.*

6. *Verma, A., Mansuri, A. H., & Jain, N. (2016, March). Big data management processing with Hadoop MapReduce and spark technology: A comparison. In 2016 symposium on colossal data analysis and networking (CDAN) (pp. 1-4). IEEE.*

7. *Vijay, V., and Nanda, R. Study on Hadoop-MapŘeduce in Context of Cloud environment. Journal of Science & Technology, (2020), ISSN 2319-2607, Vol.9 (1), pp 24-28.*

8. *Singh, S., Pareek, A. (2022). A New Model SATV for Sentiment Analysis of Hinglish Sentences. In: Rathore, V.S., Sharma, S.C., Tavares, J.M.R., Moreira, C., Surendiran, B. (eds) Rising Threats in Expert Applications and Solutions. Lecture Notes in Networks and Systems, vol 434. Springer, Singapore. https://doi.org/10.1007/978-981-19-1122-4_3.*

9. *Sharma, V., V. Nigam, and S. Vaibhav. "Strategic analysis on big data in Indian technological scenario." International Journal of Research in Computer Application & Management", ISSN 2231– 1009/14-17, Volume No. 8, Issue No 10 (2018).*

10. *Jain, V. K., & Vijay, R. (2013). Indian currency denomination identification using image processing technique. International Journal of Computer Science and Information Technologies, 4(1), 126-128*