International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Propositional Aspects of Big Data Tools: A Comprehensive Guide to Apache Spark

Jyoti Chaudhary[1], Vaibhav Vyas[2]

Submitted: 27/11/2023 **Revised**: 30/12/2023 **Accepted**: 07/01/2024

Abstract: The industry market has been impacted by big data analysis. Large and diverse datasets are significantly impacted, revealing hidden patterns and other insights. Apache Spark is one of the most admired big data tools to process and execute massive amount. A consolidated large data analytics engine that offers independent data parallelism is Apache Spark. In this paper, an intensive examination has been conveyed on big data analytical technique. This examines a technical review on Apache Spark's in-memory computing capabilities, which make it noticeably faster than other equivalent frameworks for large data analytics. Moreover, Spark has outstanding batch processing and stream processing capability. Also, it talks about Apache Spark's multithreading and concurrency features. The central focus is the Apache Spark architecture, its evolution and ecosystem, application cases, Spark features, and need of Apache Spark for applications with a comparison with Apache Hadoop.

Keywords: *Apache Spark, Hadoop, Big data, Hive, Pig.*

1. Introduction

The word “Big Data” refers to a variety of unpredictable and enormous datasets in the modern world, produced from various sources and quickly changing cutting edge innovations. In general, Big data refers to the cluster of huge and sophisticated datasets that are hard to process through traditional database applications. While the criteria used to determine whether a particular dataset is considered as large dataset or not which is well defined and continues to evolve over time, the majority of researchers and professionals today allude to data indexes between 30 to 50 terabytes and various petabytes as gigantic amount of data.

Conventionally, big data is elucidated to 3Vs and 4Vs. whereas, 3Vs derive Volume, Velocity and Variety. Volume is the enormous quantity of data spawns every day while velocity is the growth rate and data gathering rate for analysis and the diverse data available is variety where data can be structured, unstructured or semi-structured. Further the 4th V refers to veracity that encompasses availability and accountability. The major intent of big data analysis is to process huge quantity data using profuse conventional and computational intelligent techniques [1].

Because of its enormous size, huge information makes the preparing and recovery complexities for the conventional Database the board frameworks and information handling applications. Therefore, the primary target of Big Data Analytics is to process the tremendous measure of information utilizing different conventional and smart computational techniques. It is to be noticed that all the enormous datasets accessible as Big Data isn't valuable for the examination and basic leadership.

Following figure shows the attributes of Big Data:

Fig.1 Characteristics of Big Data

[1]*Research Scholar, Department of Computer Science, Banasthali Vidyapith, Rajasthan*
Jyotichaudhary1410@gmail.com
[2]*Associate Professor, Department of Computer Science, Banasthali Vidyapith, Rajasthan*
vvaibhav@banasthali.in

The processing framework has been identified as crucially significant elements of big data systems. In order to ascertain what data is present, processing frameworks can employ non-volatile storage or ingest it into the system. Processing frameworks are classed according to the type and status of the data they are intended to work with. While several systems handle data in batches, others handle data as it enters the system in an uninterrupted stream. Certain systems can also manage data in both ways [1]. The three central categories of big data processing frameworks are batch-only framework, stream-only framework, and hybrid framework.

In a batch processing system, the entire data is gathered in one cluster and later saved and processed. Alternatively, real-time stream processing systems process the data immediately it arrives. Workloads in both the batch and stream modes can steer by hybrid processing systems. Despite the fact that we can utilize identical traits or APIs for both data equally, this falls out in a straightforward, more versatile data processing.

Apache Spark is an authoritative unified analytics engine for substantial distributed data processing and machine learning tasks. Programming languages like Python are now widely used to handle data science and engineering concerns. Big data workloads are strengthened by Apache Spark using methods like in-memory processing, stream processing, and batch processing. In Section III, these methods will be covered in more detail. In a short period of time, Apache Spark has been adopted by countless industries. Not only is it an Apache Software Foundation active project, but it is also a well-known open-source project. Big data is the process of collecting, analyzing, and storing massive volumes of data.

2. Literature Review

In [2], the authors propelled the Apache Spark project, which put forward a built-in analytic engine for a variety of distributed data processing. Spark enables simultaneous cluster programming. Even though it uses the same programming architecture as MapReduce, it extends its approach to include a simple data structure known as Resilient Distributed Datasets (RDDs). For extensive SQL, graph processing, stream processing and machine learning, Spark is the leading data processing technology. Therefore, the Spark model can efficiently support existing workloads and offer plenty of advantages to consumers.

The authors of [3] suggested strategies to deal with the considerable challenges encountered during large data processing. They exploit the Apache Storm framework and an illustration of Twitter data in their work. These difficulties were effectively met by Apache Storm, demonstrating its capacity to process real-time streams with extremely low latency.

[4] Describes how the PySpark on a solitary node was used to build a novel pipeline for functional magnetic resonance imaging (fMRI). PySpark is a data analysis and pipeline language that makes the Spark programming model accessible to python. In this pipeline template matching and the sum of squared differences (SSD) approach are used to extract the brain networks from the FMRI data. This pipeline is 4X faster than the python based one in terms of processing time. The concurrent in-memory data processing has been improved, the data has been transformed into resilient distributed datasets, and the results have been saved in other forms such data frames.

Gopalani et al. [5] compared Apache Hadoop's Map Reduce with Apache Spark framework primarily because both are used for big data processing. The Apache Spark framework, which is capable of in-memory processing, will bring about a significant change in the big data world, according to a study that also compares the two frameworks on a number of other factors and analyzes their performance using the KMeans method.

The authors of [6] placed a comparison between Apache Spark with Apache Flink. The research brings the facts for machine learning libraries in variegated frameworks for batch processing. Further the study embraces vector machines and linear regression which are the methods in machine learning. The study demonstrated by actual findings that spark works better than flink in terms of efficiency.

In [7] a smart grid is a fully automated system that integrates a large number of sensors into the existing electrical infrastructure to monitor and regulate it using contemporary information technology. These sensors provide vast amounts of data that meet the criteria for being referred to as big data. By instantly digesting and extrapolating new knowledge from this data, the Smart-grid may become smarter. The work has proffer Apache Spark as an amalgamate cluster computing platform which stores and processes data analytics on smart grid data for applications like real-time pricing and usual stipulate response.

Distributed solutions for data flow like Apache Hadoop MapReduce, Apache Spark and Apache flink were compared concerning the key of usability and ease of use [8]. Though MapReduce struggles with Scalability and built in redundancy, the later two concentrates on the requirement for effective data flow, data caching, and declarative data processing operators. The major goal is to highlight a course to pick a pertinent platform and to

improve the understanding for how long data processing systems runs.

3. Big Data Analysis Tools

The following provides a quick review of few selected big data analysis tools, an introduction to Apache Spark, and a comparison to its rivals to justify the use of Apache Spark.

Apache Hive

A data warehousing infrastructure called Hive is built on top of Hadoop. To organize, aggregate, and conduct data queries, it offers a language called Hive QL. Using a declarative programming model; Hive QL is comparable to SQL [7]. This follows procedural approach, which distinguishes it from Pig Latin. The conscious outcomes are described in simple query in HiveQL that is way similar to SQL. Alternatively, Pig Latin structures a query in sequence of assignment operations. Apache Hive allows the developers significantly SQL developers to design queries in HQL (Hive Query Language). Similarly, Hive can segment queries in HQL to make them allow interacting with several jobs running on MapReduce.

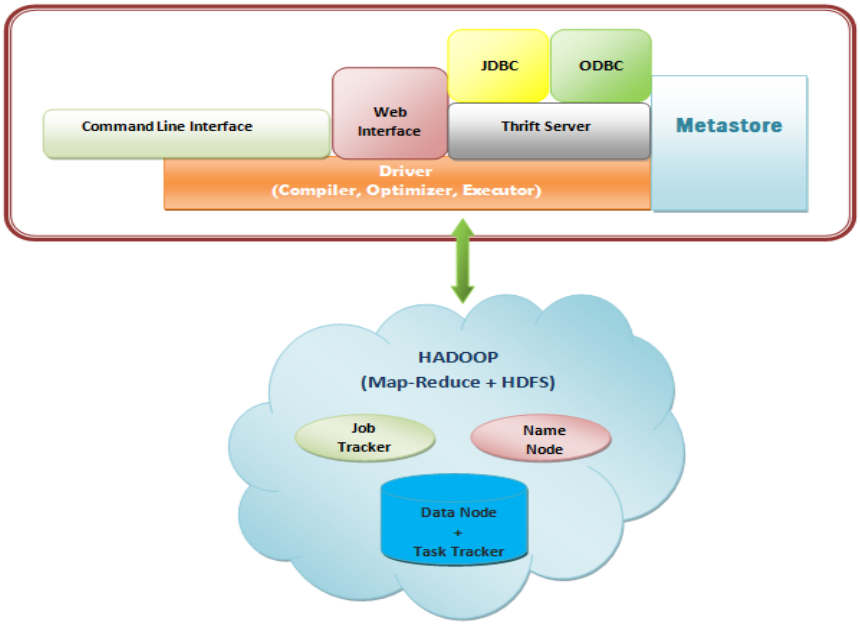

Fig.2 Hive Architecture

The internal working of Hive [18] can be summarized with few keywords that are

1. UI: as the name UI stands for user interface which allows users to submit the query for further processing.
2. Driver: this fundamental receives the query from the user and fetch the API modeled on JDBC/ODBC interfaces to execute the query.
3. Compiler: complier is used to parse the query by semantic analysis of each module of query and ultimately finds the execution plan through the parse table.
4. Meta store: this component works on storing the information about all the parse tables along with the column information. And the information of serializer and de-serailizer with requires to perform all read and write operations.

Execution engine: it is the last phase for any query where it is executed according to the complier plan. This manages the dependency of one operation on another and same for every stage included in the execution.

Apache Pig

Pig is a tool or, more precisely, a platform for scrutinize bulk size of large data. Pig program's ability to handle large data sets is made possible by the substantial parallelization of tasks [19]. Although Pig and Hive are intended to carry out comparable jobs. Pig founds to be reliably suitable for the data devising stage of data processing, while Hive is additionally suitable for the data warehousing and display scenario.

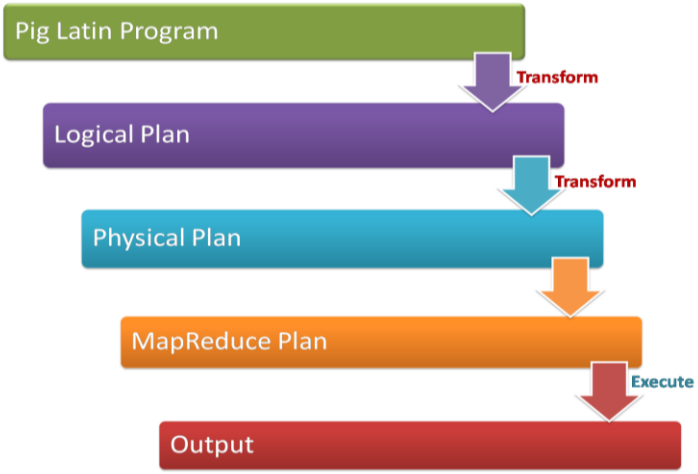

Fig.3 Pig Latin Workflow

The plan is for data to be cleaned up using the tools given by Pig before being saved as it is gradually gathered. After then, Hive avails of executing ad-hoc searches and probe the data. The data warehouse render inoperative throughout the task of gradual and progressive construction, whereas, Pig carries out both data preparation and querying effectively. Testing has to be done to see whether utilizing Pig and Hive together is practical [20].

Apache Hadoop

A well-known framework for batch processing is Hadoop. The Hadoop distributed file system and MapReduce are the primary components of Hadoop. This framework is developed by apache Hadoop based on MapReduce. Nodes are the individual computers that makeup the cluster. Performance and node count are directly inversely correlated; the more nodes, the higher performance. Hadoop operates on the distributed and parallel computing model, processing data concurrently across a number of devices.

The Map phase and Reduce phase are the two steps that comprise Mapreduce. The input data is processed using a map task and is stored in HDFS as files. The input records are transformed into intermediate records using the map. A single call to the setup method precedes the map method, which is then followed by a single call to the cleanup function for each key/value combination.

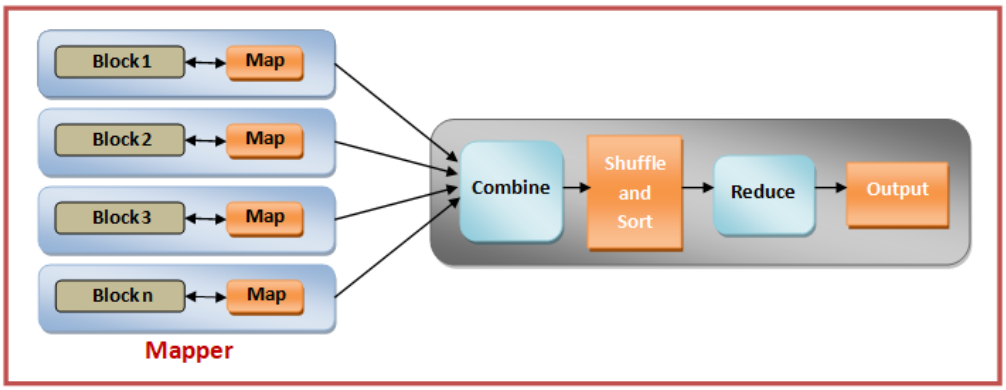

Fig.4 How Map-Reduce Works

MapReduce by default accepts the text input format, which accepts key as byte offset and value as text. In the word count task, the mapper and reducer can communicate with the rest of Hadoop system since the key will be long variable, the value will be the text, and context. And processing of the data occurs at reduce stage. It creates the output, which will be stored in the HDFS, using the intermediate key value pair from the mapper. Basically, this summarizes the data [21].

Evolution of Apache Spark

The Hadoop computational computing software process has been expedited by the Apache software foundation [9]. Though spark has its own cluster therefore, it is not reliant on Hadoop and hence not amend on Hadoop, it is solely a method to implement spark. Hadoop is used by Spark for processing and storage, respectively. Having the capability of its own cluster computation, Hadoop is just used by it for storage. The core component of Spark is its

in-memory cluster computing, which accelerated application processing. It is purposive to handle a variety of workloads, like batch applications, iterative queries and streaming. Along with handling each workload, it relieves the management strain of keeping distinct tools up to date.

At UC Berkeley's AMPLab, Matei Zaharia created Spark, one of Hadoop's side projects in 2009. It was made available as open source under a BSD license in 2010. After being given to the Apache Software Foundation in 2013, Apache Spark has become a top-level Apache project as of February 2014 [10].

Apache Spark, a potent framework that expedites distributed computing on massive data, has now integrated into the Hadoop environment [11]. It achieves this by utilizing in-memory elementary, which enables it to run applications 100 times quicker than Hadoop. This technology is especially proficiently suitable for online and iterative processing, as it permits client programs to load information into memory and execute repeated queries. Spark is also adaptable in that it enables the implementation of a number of distributed programming models, including Pregel and MapReduce [12].

As Spark employs "in-memory computing" as averse to MapReduce's more traditional read from and write to the disc method, it can be expeditious than MapReduce for identical batch processing operation. Spark can work unrestrained or in tandem with Hadoop to re-implement the MapReduce engine [13].

Ecosystem of Apache Spark:

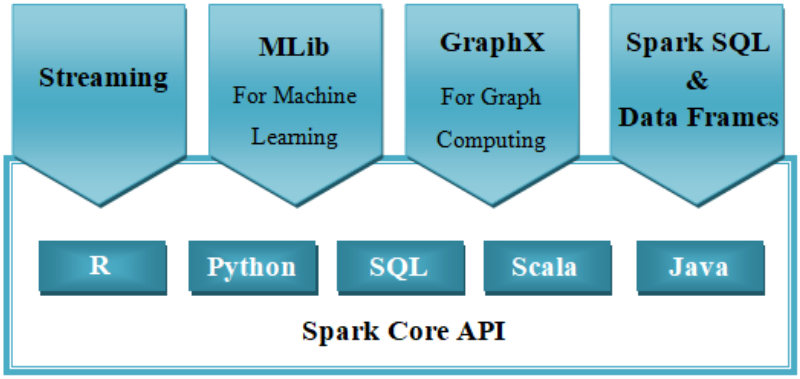

Fig.5 Apache Spark Ecosystem

Now, to know Apache spark in a better way, let us consider its components that take part in the Apache Spark model. Apache Spark consists of four major components which are Streaming, MLib, Spark SQL, and GraphX. Spark Streaming is beneficial to the API for center Spark which avails the process of continuous streaming data. The design of Spark streaming depends on the series of RDD to process the real-time data. This is helpful in fault tolerance and increases the throughput for stream processing of any live real data [14][13]. For example, to implement the streaming over Spark, the data is split into small batches which will be used to generate new results by combining the current state with the state that is already stored in RDD. Spark streaming is beneficial to the Spark API, which is used to process continuously streaming data and a Spark component. Concerning that working with the Spark SQL package's sorted-out data is the real goal. The Apache Hive or SQL-based HQL (Hive Query Language) is a variant of Spark SQL that makes it possible to address data [14]. Spark SQL connects the Hive table, Parquet, and JSON by highlighting the significant differences between the various data sources.

In addition to providing Spark with a standard SQL interface, Spark SQL also enables planners to support various SQL requests using programmed data controls that is re-enforced by RDDs in Scala, Python, and Java. Since everything falls in a solitary application, it amalgamates SQL with in-depth analysis. This aspect of firmly consolidating the environment with the affluent and drive handling condition raised by Spark makes it superior to additional open-source information stockroom devices now in use [15]. According to benchmarks, MLib been tested against Alternating Least Squares (ALS) implementations by the MLlib developers. Spark MLlib is nine times faster than Apache Mahout's Hadoop disk-based variant (before Mahout procures a Spark interface) [16]. Machine Learning Library points for MLlib. Apache Spark MILib is used for performing machine learning. Python and R, are two more languages that may be used to implement machine learning, both of which offer improved visibility and graphical representation [13]. The Apache Spark includes several APIs, one of which is GraphX, which is used for graphs and parallel graphs for computing. This is the reason for extending the Spark RDD feature of graph. Graphs have the ability to have many edges running concurrently, each edge and vertex having a user-defined characteristic, and the parallel edges having numerous relationships between the same vertices.

This property is also referred to as a multi-graph. The Spark component GraphX greatly expands the Spark RDD abstraction by attaching the directed multigraph attributes to each and every vertex and edge utilizing the Resilient Distribution property of Graph. With the purpose of streamlining graph analytics activities, it incorporates an emergent collection of builders and graph algorithms. It also optimizes a number of Pregel APIs and displays a range of operators such as map-reduce triplets, sub-graphs, and join vertices [14].

Architecture of Apache Spark:

Fig. 3 illustrates the architecture of Apache Spark, which include a master node and a driver program that is conscientious for calling an application's primary program. If an interactive shell is used, the driver program is moreover user written code. This driver program creates the spark context. A doorway to all of Spark's functionality is provided by the spark context. It works together with the cluster manager, who is accountable of managing a number of tasks [17]. The driver software collaborates with the spark context to run the job within the cluster.

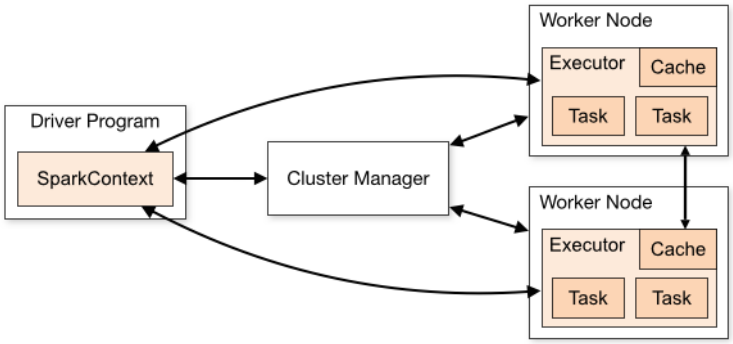

Fig.3 Apache Spark Architecture

The cluster manager first handles the task of assigning resources. Dividing the jobs into several tasks is then carried and forward to the slave or worker nodes. As soon as an RDD is formed in the Spark context, it may be distributed to the various slave nodes and cached there as well. The slave nodes participate in carrying out the duties that the cluster management gave them. These tasks are then reinstating to the spark context. The tasks are accomplished by the executor. The executors have the equal duration as Spark.

To ameliorate the system performance, expanding the number of worker nodes must be inflated to further separate the jobs into more rational chunks [17].

Why Apache Spark?

Spark utilizes the conviction of the RDD concept, which enables to persist the data as needed and store it in memory. As a result, batch processing task performance can be significantly increased (ten to one hundred times greater than that of traditional Map Reduce).

Furthermore, spark gives us the ability to cache data in memory, which is advantageous for iterative algorithms like those employed in machine learning. Conventional MapReduce and DAG engines don't work well for such utilization since they are built on acyclic data flow. An alternative is to start an application as a sequence of independent tasks, data reading will be taken from steady storage (like a distributed file system) and writes final transaction back to this steady storage. They acquire large expenses each time they load the data and write it back to persistent storage.

Spark enables stream processing with enormous input data and on-the-fly handling of just a small portion of the data. Online machine leaning can utilize this and is ideal for instance requiring real-time analysis, which is actually a nearly universal requirement in the sector.

In particular, multi-pass applications that demand low-latency data sharing across numerous concurrent operations are inefficient for MapReduce. These applications, which are extremely widespread in analytics, include: Iterative algorithms, which include a variety of machine learning algorithms and graph algorithms like PageRank.

Interactive data mining, which allows users to repeatedly query data that, has been loaded into RAM across a cluster. And, streaming program that continuously save aggregate state [5].

Comparison between Apache Spark and Apache Hadoop

Comparative Parameters	**Hadoop**	**Spark**
Category	Hadoop is the essential information preparing motor.	Spark is the information examination motor.
Usage	Hadoop chips away at Batch preparing with a tremendous volume of information.	Flash chips away at continuous information from ongoing occasions like Twitter, Facebook.
Latency	Processing inertness of Hadoop is extremely high.	On account of Spark, figuring idleness of Spark is low.
Data	Hadoop process the information as cluster mode.	Flash can process intelligently.
Ease of Use	Hadoop is likewise MapReduce model and that is perplexing, it is have to deal with low level APIs.	Sparkle is exceptionally simpler to utilize, reflection empowers a client to process information utilizing significant level administrators.
Scheduler	In Hadoop outside occupation scheduler is required.	In memory calculation of Spark, no outer scheduler required.
Security	Hadoop is profoundly secure.	Flash is less secure as contrast with Hadoop.
Cost	Hadoop is less exorbitant since MapReduce model give a less expensive technique.	Sparkle is costlier that Hadoop since it has an in-memory arrangement.

The above table explains the various features of spark and Hadoop [21]. Therefore, quantification of qualitative outcomes is accomplished using likert scale. This will easy the evaluation of performances among Apache Hadoop and Apache Spark considering various parameters and identify the best suitable big data technology to meet an individual's requirements. Since various researchers have contrasted these big data tools by measuring different factors, but the objective of this study is to identify best performing big data tool for massive amount of data in terms of usage, latency, data, usability, and processing time. This study has gone through various research papers like [5][8][19][21][23] to find the mentioned values.

Table 1. Feature comparison of two popular big data Platforms

Tools/Features	Usage	Latency	Data	Ease of Use	Processing time
Apache Hadoop	Variable	High	High	Moderate	Moderate
Apache Spark	High	Moderate	High	High	Very low

Table 2. Quantification of the qualitative standards

Criterion values with quantifiable range	Criteria covered	Explanation
High-5, Variable- 4, Moderate-3, Low-2, Very low-1	usage, latency, data, ease of use, and processing time	From Table-3, move from the greatest to the lowest degree of assistance with 5 grades 5-1.

Table 3. Summary overview based on features

Features / **Tools**	Usage	Latency	Data	Ease of Use	Processing time
Apache Hadoop	4	5	4	3	3
Apache Spark	5	3	5	4	1

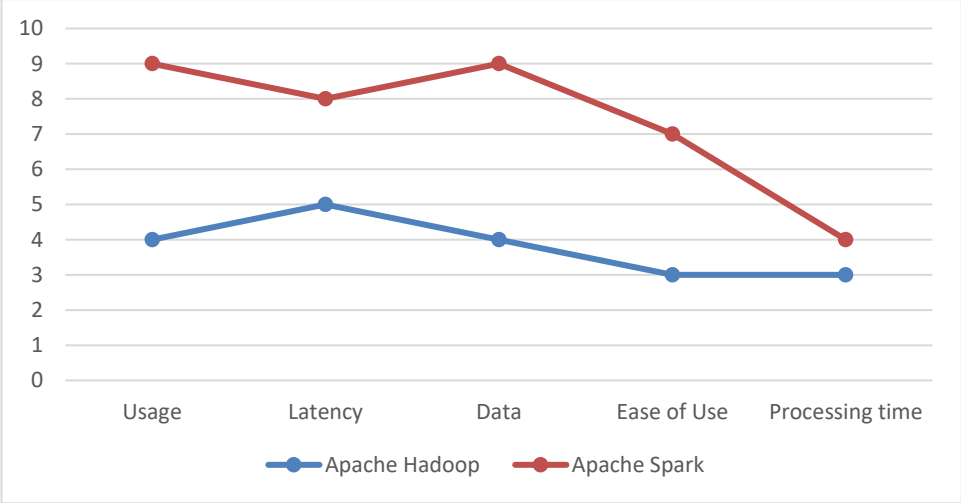

Fig. 2**.** Graphical analysis of features in Big data tools

4. Conclusion

For the future generation of computer applications, scalable data processing will be crucial, but it often entails a challenging workflow of operations using several computing platforms. With the introduction of a unified programming paradigm and engine for large data applications, the spark project made this process easier. Our research demonstrates that a model like this may effectively handle the demands of today while also providing users significant advantages. We believe that Apache Spark emphasizes the significance of compos-ability in large data programming libraries and stimulates the creation of more readily interoperable libraries. The qualitative approach has explained the suitability of Apache Spark for massive data and can be used with Databases like NoSQL as well to outperform the operations on large scale as these databases have efficient performance [22].

References:

[1] Acharjya, D. P., & Ahmed, K. (2016). A survey on big data analytics: challenges, open research issues and tools. International Journal of Advanced Computer Science and Applications, 7(2), 511-518.

[2] Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Stoica, I. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM*, *59*(11), 56-65.

[3] Iqbal, M. H., & Soomro, T. R. (2015). Big data analysis: Apache storm perspective. *International journal of computer trends and technology*, *19*(1), 9-14.

[4] S. Sarraf and M. Ostadhashem, "Big data application in functional magnetic resonance imaging using apache spark," in 2016 Future Technologies Conference (FTC), Dec 2016, pp. 281–284.

[5] Gopalani, S., & Arora, R. (2015). Comparing apache spark and map reduce with performance analysis using k-means. *International journal of computer applications*, *113*(1).

[6] García-Gil, D., Ramírez-Gallego, S., García, S., & Herrera, F. (2017). A comparison on scalability for batch big data processing on Apache Spark and Apache Flink. *Big Data Analytics*, *2*(1), 1-11.

[7] Shyam, R., HB, B. G., Kumar, S., Poornachandran, P., & Soman, K. P. (2015). Apache spark a big data analytics platform for smart grid. *Procedia Technology*, *21*, 171-178.

[8] Akil, B., Zhou, Y., &Röhm, U. (2017, December). On the usability of Hadoop MapReduce, Apache Spark & Apache flink for data science. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 303-310). IEEE.

[9] http://spark.apache.org/

[10] Jonnalagadda, V. S., Srikanth, P., Thumati, K., Nallamala, S. H., &Dist, K. (2016). A review study of apache spark in big data processing. *International Journal of Computer Science Trends and Technology (IJCST)*, *4*(3), 93-98.

[11] Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). *Learning spark: lightning-fast big data analysis*. " O'Reilly Media, Inc."

[12] Ramírez-Gallego, S., Mouriño-Talín, H., Martinez-Rego, D., Bolón-Canedo, V., Benítez, J. M., Alonso-Betanzos, A., & Herrera, F. (2017). An information theory-based feature selection framework for big data under apache spark. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*(9), 1441-1453.

[13] Shaikh, E., Mohiuddin, I., Alufaisan, Y., &Nahvi, I. (2019, November). Apache spark: A big data processing engine. In *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)* (pp. 1-6). IEEE.

[14] Ahmed, D. N., Aftab, A., &Nezami, M. M. (2020). A technological survey on apache spark and hadoop technologies. IJSTR, 9(01), 3100-3109.

[15] Han, Z., & Zhang, Y. (2015, December). Spark: A big data processing platform based on memory computing. In *2015 Seventh International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)* (pp. 172-176). IEEE.

[16] Jonnalagadda, V. S., Srikanth, P., Thumati, K., Nallamala, S. H., &Dist, K. (2016). A review study of apache spark in big data processing. *International Journal of Computer Science Trends and Technology (IJCST)*, *4*(3), 93-98.

[17] Anuraag Garg, "Apache spark architecture," Website, 2023. [Online]. Available: https://intellipaat.com/blog/tutorial/spark-tutorial/spark-architecture/.

[18] Puspalatha, N., & Sudheer, P. (2015). Data processing in big data by using Hive interface. *International Journal of advance research in computer science and management studies*, *3*(4).

[19] Hussain, T., Sanga, A., & Mongia, S. (2019, October). Big data hadoop tools and technologies: A review. In *Proceedings of International Conference on Advancements in Computing & Management (ICACM)*.

[20] Shoro, A. G., & Soomro, T. R. (2015). Big data analysis: Apache spark perspective. *Global Journal of Computer Science and Technology*, *15*(C1), 7-14.

[21] Singh, A., Khamparia, A., & Luhach, A. K. (2019, June). Performance comparison of apachehadoop and apache spark. In *Proceedings of the Third International Conference on Advanced Informatics for Computing Research* (pp. 1-5).

[22] Chaudhary, J., Vyas, V., & Jha, C. K. (2022). Qualitative Analysis of SQL and NoSQL Database with an Emphasis on Performance. In *IOT with Smart Systems: Proceedings of ICTIS 2022, Volume 2* (pp. 155-165). Singapore: Springer Nature Singapore.

[23] KE, K., Balaji, A., & Sajith, A. (2018). Performance comparison of apache spark and Hadoop based large scale content-based recommender system. In *Intelligent Systems Technologies and Applications* (pp. 66-73). Springer International Publishing.