A Comparative Analysis of Hadoop and Spark Frameworks using Word Count Algorithm

Yassine Benlachimi[1], Abdelaziz El Yazidi[2], Moulay Lahcen Hasnaoui[3]
ENSAM Moulay-Ismail University, LMMI Laboratory, Institute, Meknes, 50000, Morocco

***Abstract*—With the advent of the Big Data explosion due to the Information Technology (IT) revolution during the last few decades, the need for processing and analyzing the data at low cost in minimum time has become immensely challenging. The field of Big Data analytics is driven by the demand to process Machine Learning (ML) data, real-time streaming data, and graphics processing. The most efficient solutions to Big Data analysis in a distributed environment are Hadoop and Spark administered by Apache, both these solutions are open-source data management frameworks and they allow to distribute and compute the large datasets across multiple clusters of computing nodes. This paper provides a comprehensive comparison between Apache Hadoop & Apache Spark in terms of efficiency, scalability, security, cost-effectiveness, and other parameters. It describes primary components of Hadoop and Spark frameworks to compare their performance. The major conclusion is that Spark is better in terms of scalability and speed for real-time streaming applications; whereas, Hadoop is more viable for applications dealing with bigger datasets. This case study evaluates the performance of various components of Hadoop-such, MapReduce, and Hadoop Distributed File System (HDFS) by applying it to the well-known Word Count algorithm to ascertain its efficacy in terms of storage and computational time. Subsequently, it also provides an analysis of how Spark's in-line memory processing could reduce the computational time of the Word Count Algorithm.**

***Keywords*—Big data; Hadoop; spark; machine learning; Hadoop Distributed File System (HDFS)); MapReduce; word count**

I. INTRODUCTION

Due to the advancements in computational technology, hardware resources, and fast underlying networks, the world witnessing an explosion of Big Data generated by social media networks [1], Internet of Things (IoT)[2], streaming real-time applications [3], banking sector [4], industrial setups, and almost every notable R&D sector. According to [5] an estimate by a well-known online source, Social Media Today, 2.5 Exabyte (1018) data is generated per day, as of 2020. This data creation is expected to increase to 463 Exabytes per day by the end of 2025, according to Statista [6]. Consequently, it becomes extremely difficult to handle such enormous volumes of Big Data by using traditional methods and tools [7]. For example, the traditional database systems administering the legacy warehouses have become inefficient due to the utilization of conventional query tools. Venkatraman et al. [8] found multiple reasons for the failure of these tools. Firstly, the design of relational databases and data warehouses is not suitable to synthesize the new types of data with respect to volume, storage, veracity, and processing. Secondly, in traditional systems, the Structured Query Language (SQL) is utilized for communicating with databases. Thirdly, the maintenance of rational data-houses becomes very costly and unmanageable. Fourthly, traditional warehouses are based on organizing records in fields in a structured manner, while most of the Big Data on the Internet is unstructured by nature [9]. Therefore, traditional database management tools cannot efficiently be utilized in the case of Big Data, which is exponentially growing due to the surge in the number of Internet & social media users, and the development of new technologies like IoT, 5G networks, and Deep Learning (DL), etc. This entails an extremely competitive atmosphere among technology companies to provide accurate data in a minimum amount of time at a low cost. It is the only concept of Big Data that gives equal opportunity to everyone to extract the data and use the full value from in their particular organization or concerned field of interest. Chen et al. and Ward et al. [10, 11] more formally defined Big Data as "a set of several structured, unstructured data generated from different formats of various tasks with bulk volume that is uncontrollable by current traditional data-handling tools". Contrary to the conventional data handling tools, the Big Data analysts at Apache and the research community developed a very efficient framework—called Hadoop—that can process and manage huge volumes of data [12]. Primarily, the Hadoop (Highly Archived Object-Oriented Programming) framework spawns the input data to multiple distributed computing nodes and provides reliable and scalable computing results [12, 13]. Bangari et al. [14] uses simple programming models based on Java language for distributed processing of a large volume of datasets through the clusters of computers. The basic idea of Hadoop setup is to use a single server in order to handle a collection of slave workstation nodes in which each node contains its own local storage and computational resources. To process and store data, Hadoop utilizes the MapReduce algorithm, which divides any given task into smaller parts in order to distribute them across a set of cluster nodes [15, 16]. Sharmila et al. [17] showed another basic feature of Hadoop as is its file system known as the Hadoop Distributed File System (HDFS), which is an efficient storage system for cost-effective hardware. Although Apache Hadoop remained one of the most reliable frameworks to handle Big Data within a decade after its first release in 2006, its efficacy was reduced after the exponential growth of streaming real-time data, Machine/Deep Learning (M/DL) technologies, and the use of graphics in online games & other related applications [18]. Bell et al. [19] overcome these limitations by another open-source framework called Apache Spark that was developed, which incorporates diverse features such as better memory & storage management and more scalability.

The primary objective of the research were analyzing the processing times and file management systems of the heterogeneous environment to get better performance of Hadoop. This paper contributes by giving a comprehensive comparison between Apache Hadoop & Apache Spark in terms of their application, scalability, reliability, security, cost-effectiveness, and other features. Moreover, the discussion of primary components of these frameworks ascertains their performance. The study concludes that the Spark framework is more useful for streaming applications, and hence it is fast and scalable. On the other hand, Hadoop has better security features, and it can handle very large volumes of data. Furthermore in this paper, a case study discusses the performance of the Hadoop framework by implementing the famous WorldCount algorithm. The results show the effectiveness of Hadoop in terms of processing time and storage required to process large dataset files. In addition, an analysis was described by comparing these results with Spark's framework to determine how its features could reduce the processing time of the algorithm for the given files.

The remainder of this paper is organized into the following sections. In Section 2, the characteristics of Big Data are discussed in detail. Section 3 provides the related work. Section 4 discusses the major components of Hadoop. Section 5 describes the WordCount Algorithm. Section 6 details the Spark framework followed by Spark Components in Section 7 and its comparison with Hadoop in Section 8. Section 9 describes the experimental setup and results by implementing the WordCount application on the Hadoop cluster. Then Section 10 discusses how Spark implementation of the WordCount application could improve the execution times. Finally, Section 11 provides the conclusion and future work.

II. Characteristics of Big Data

The Data grows in three dimensions—also known as the 3Vs model—according to the Gartner research report; these three are Volume, Variety, and Velocity [10, 11]. It has been observed that many industries and organizations use the 3Vs model to analyze Big Data. However, it cannot be formally defined by merely 3Vs, and many other characteristics also exist to properly define Big Data [20]. Chen et al. [10] stated these characteristics are extended to 5Vs including the above-mentioned 3Vs. These are elaborated on the basis of line spacing, and typestyles. Examples of these type styles are given below and are depicted in Figure 1.

A. Veracity

It is one of the most important properties of Big Data tools and is defined as data accuracy and it is quality relative to its users. The veracity is ensured by providing accurate and clean data [10].

B. Volume

It may be defined as the bulk of data to be organized, stored, and processed. The data volume is exponentially growing, and it is expected to grow multiple times in the coming few decades. Chen *et al.* [10] Stated the current volume of Big Data generated per day is in the realms of Exabyte (1018).

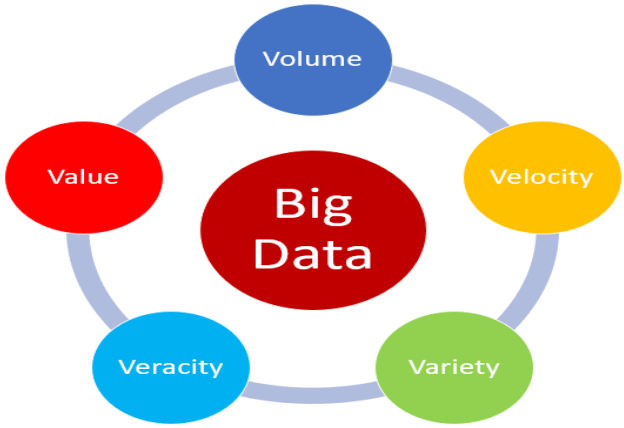

Fig. 1. The 5Vs of Big Data.

C. Variety

It refers to the various forms in which data is available to the users. Currently, the data can be structured, unstructured, or semi-structured[11], depending upon its organization. Moreover, it may be in the form of plain text, image, audio, video, etc., or any combination of these forms.

D. Value

This simply implies how important or critical the data is to the user. Data value describes its beneficence for a particular organization or individual. The ratio of the valuable data is inversely proportional to the total volume of data. Chen *et al.* and Ward *et al*. [10, 11] for instance, in an hour-long video, the ratio of valuable data can be of a few seconds.

E. Velocity

Velocity implies the rate at which the data is retrieved, and it assists to identify the difference between normal data and Big Data. The characteristic of velocity for the data warehouse is a very significant parameter in this competitive field. For instance, it is the velocity that plays a critical role in data retrieval or storage at a typical social media warehouse for its users to efficiently use it for socializing. The author in [10, 11, and 17] found that the users of Facebook, Twitter, Instagram, and other famous platforms expect to communicate with each other in real-time for their everyday experience.

Most of the data analysts and experts suggest utilizing a variety of open-source Big Data platforms in order to take benefit from Big Data analytics [10, 21]. Elgendy and Elragal [22] showed these platforms offer a mixture of hardware resources using state-of-the-art software tools for data storage, processing, analysis, and visualization. The opportunities generally vary based upon the phenomenon where data is being utilized, and its value depends upon the type of applications. For example, in the stock exchange system, where demands and consumption change at a fast rate, data has importance only for a limited period of time [20]. Furthermore, due to the enormity of data volumes and Big Data applications, it becomes extremely tough for managers to select a single or a small group of data platforms. Undoubtedly, the ever-growing competition and limited time make it inconvenient for them to work with a large number of Big Data platforms. However, they still require multiple platforms due to the demands of

repeated and multiple task solvers [22]. Nonetheless, it can be a very useful analysis to determine an optimum set of platforms for a given organization considering its R&D requirements and applications.

Another significant aspect of the Big Data paradigm is data security concerns created during the management, storage, and processing of data.

F. Management Issues

Data is normally retrieved or stored in structured, unstructured, or semi-structured modes in various organizations. It is difficult to manage such diversity in data in large volumes.

G. Storage Issues

The data sources are diverse, and it is normally retrieved from social media, streaming systems, mobile signal coordinates, sensor information, online recordings, and e-business exchange reports. Storing this data in various forms creates storage issues and requires standards to be properly implemented.

H. Processing Issues

Based on user requirements, the Big Data systems are expected to process data in volumes of Petabyte, Exabyte, or even in Zettabyte. This processing can be real-time or in batch mode. Therefore, Big Data systems must be able to cope with user requirements.

I. Security Issues

In government & private sectors, the data is normally vulnerable to malicious attacks and intrusions. Therefore, organizations are expected to carry effective intrusion detection systems and data integrity systems in order to ensure the safety of user data and avoid data exploitation.

All the above-mentioned issues and challenges are tackled by using efficient tools like Apache Hadoop & Spark. Currently, the most widely used framework is Hadoop, and it is particularly useful for processing large volumes of data by tech giants such as Twitter, LinkedIn, eBay, and Amazon [23]. A lot of research is performed to evaluate the performance of Hadoop, but there is a need to improve its functionality in terms of its time efficiency. [16,18] Identified that Hadoop is extremely powerful in the case of storage systems due to HDFS, but it struggles to compete with Spark in the processing part performed by its MapReduce algorithm. This work focused on the testing of MapReduce by recording the time elapsed by each processing step of the algorithm with various volumes of data downloaded in the form of data files from the Internet. The primary aim is to ascertain the performance of Hadoop compared to Spark to learn which application scenarios are suitable for a particular framework.

III. Related Work

Due to the enormous amount of data generated daily, the issues of its management, storage, and processing are not only significant for the academic community but also the industry. For instance, Zhao et al. [18] conducted a performance comparison between Hadoop and HAMR based on the running PageRank algorithm. HAMR is a new technology, which provides faster processing and memory utilization compared to Hadoop. The comparison parameters used in the research were memory usage, CPU consumption, and running time. Shah et al. [24] observed the performance of Hadoop in a heterogeneous environment with various types of hardware resources. They developed an algorithm called Saksham, which enables the rearrangement of the data blocks to optimize the performance of the Hadoop in homogeneous & heterogeneous environments. A region-based data placement policy is proposed by Muthukkaruppan et al. [25]. The main purpose of the proposed policy is to achieve high fault tolerance and data locality, which does not exist in the default policy. A specific region data block is placed in the contiguous data portion of nodes in the region-based policy. Qureshi et al. [13] described storage media-aware policy in order to improve the performance of Hadoop. This policy is known as the Robust Data Placement (RDP), which also handles the network traffic and unbalanced workload.

Meng et al. [26] proposed a strategy that places data blocks with disk utilization and network load while in default Hadoop, block placement is done by Round Robin that reduces the performance in a heterogeneous environment. This strategy improved the HDFS performance by enhancing the space storage utilization and throughput.

Similarly, Dai et al. [27] presented their proposed Replica Replacement Policy (RRP) developed in 2017 to improve the Hadoop performance by eliminating the utility of HDFS balancer; consequently, the replica is evenly distributed among the homogeneous and heterogeneous nodes. This policy generates better results of replica management as compared to the default replica management policy of the HDFS in Hadoop. Herodotou et al. [28] proposed a new tool to optimize the default parameters of Hadoop; for instance, the total number of map reduces, scheduling policy and the reuse of JVM to increase the performance. The tool is called Startfish, and its main purpose is to work with Hadoop phases such as placement, scheduling, and tuning of the assigned jobs to the computing nodes. Panatula et al. [29] worked on the performance of HadoopMapReduce Word Count Algorithm and presented it on the Twitter data. The experimental setup was based on a 4-node Hadoop system to analyze the performance of the algorithm. It was concluded that Hadoop can work efficiently with the setup of 3 or more nodes. Gohil et al. [15] processed a different set of applications of MapReduce including Word Count and Tera Sort etc. to evaluate Hadoop performance. The evaluation parameters were to set up a dedicated cluster and decrease the I/O latency of the network. Likewise, the in-house Hadoop cluster setup and Amazon EC2 instances are also used to evaluate the Hadoop performance. Khan et al. [30] have modeled the estimation of the provisioning of the resources and completion time of the jobs. Furthermore, the Hadoop and Spark-based distributed system performance has been evaluated by Taran et al. [31]. A performance evaluation framework is proposed for Hadoop by Lin et al. [16] on the basis of cluster computing nodes across a clustered network. A configuration strategy is proposed by Jain et al. [32], in which MapReduce parameters are configured with optimized tuning to improve the performance of Hadoop. To analyze the performance of Hadoop, several applications

were processed and tested repeatedly by Londhe et al. [33] using the Amazon platform for Hadoop. In [34], an analysis of the computational performance of the processors on a private network was presented in order to reduce the input/output latency of the network.

This work implemented a well-known WordCount Algorithm using Hadoop to evaluate the framework performance and provide a thorough analysis of the difference between Hadoop and Spark frameworks.

IV. HADOOP COMPONENTS

Hadoop framework allows the users to record & process Big Data in a distributed network, across several computing nodes using easy-to-use programming methods. It is an open-source framework designed by D. Cutting & M. Cafarella [14]. Most researchers and developers consider Hadoop the most efficient tool in the Big Data domain. It is sometimes misunderstood as merely a database, but it is a comprehensive ecosystem that allows distributing data for processing across thousands of servers and keeping the overall performance extremely optimized. As mentioned earlier, there are two basic components of the Hadoop system: HDFS and MapReduce algorithm [35].

The basic architecture of the Hadoop framework is depicted in Figure 2. It based on the Master-Slave system, in which the master node is called the name node, and the slave is the data node, which keeps and processes the actual data. For fault tolerance, the factor of replication is set at 3, where the MapReduce algorithm helps the replicated data to be processed in parallel mode [26]. In the following, the components of the Hadoop system describe in detail:

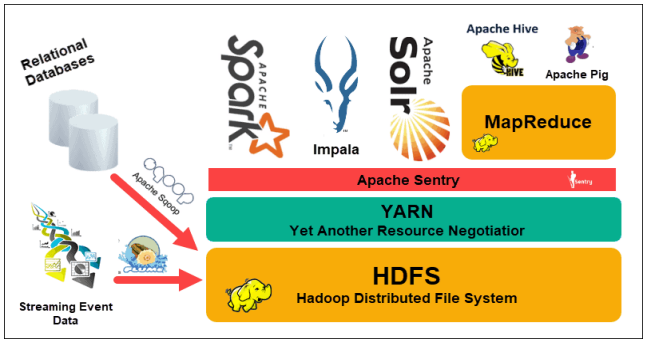

Fig. 2. Hadoop Architecture.

A. HDFS

A large amount of data in the form of sets is stored on HDFS which is a distributed file management system and works on the commodity hardware [38]. Thousands of nodes clustered in a distributed system can be supported using the HDFS in a reliable manner at a low cost. It can support large files containing volumes of data in terabytes. Furthermore, it provides portability for data across various platforms and nodes. However, the most important feature of HDFS is its ability to reduce traffic congestion across networks, because processing and data are moved closer to each other.

To perform low-latency computations, another significant open-source system has been developed on top of HDFS, which is called the HBase [36]. It is essentially a distributed non-relational database system using column-based value/key data presentation. [36, 37] found a developer can spawn data across distributed cluster nodes and update the data tables at very fast rates by using HBase. However, HDFS cannot be replaced and mixed with HBase due to its non-relational DBMS nature, although HBase can help to process the real-time data by using its in-memory processing engine [40]. The HDFS is used in many systems that consist of conventional file systems such as ext2 (Linux) or FAT (Windows). But HDFS is totally different from all these traditional file systems for the following main reasons:

HDFS is optimized to maximize data rates. The size of a data block is 64MB in HDFS as compared to 512 bytes to 4 KB in most of the traditional file systems, which significantly reduces the seek time. In addition, it is possible to further extend the size of a block to 128MB or 256MB. Moreover, [26] showed HDFS is a Write Once Read Many (WORM) file management system: any file can be written once but can be accessed several times.

Another prominent feature of HDFS is its fault tolerance where it furnishes a block-based replication framework with a configurable number of replications (by default, it is 3) [23, 26]. During the composing stage, every block compared to the record is imitated on isolated hubs in the bunch, which assists with ensuring its dependability and accessibility, when understanding the information. In the event that a block is inaccessible on one node, duplicates of that block are ensured to be accessible on a different node.

HDFS builds on the native OS file system to present a unified storage system built on a heterogeneous array of disks and file systems.

B. MapReduce

White [38] found MapReduce is the most significant component of the Hadoop framework. In the clusters of the Hadoop, the scalability is provided by MapReduce on thousands of computing servers. The term ‘MapReduce’ is constituted from two different words; Map and Reduce. Both these terms are attributed to performing different tasks in the Hadoop system. The job of mapping data is performed by the Map function. The map function converts the original data into the form of sets. The purpose of making sets is to make key pairs of all unique elements of data, which serves as the output of the map function. Subsequently, the output of the Map function becomes the input data of the Reduce function. The main job of the Reduce function is to reduce the number of key pairs of all unique elements in a key pair [15]. Figure 3 depicts the MapReduce process. The input data is split by the distributed file system and mapped to the Map node in key/value format. Then the Reduce node merges the key/value pairs to further reduce them and store them using the file system [38].

1) In the Hadoop system, the job of the Map function is performed firstly, followed by the Reduce function. Sometimes the map function job is enough to process the data efficiently. In Figure 3, the architecture and process of the MapReduce give a thorough overview.

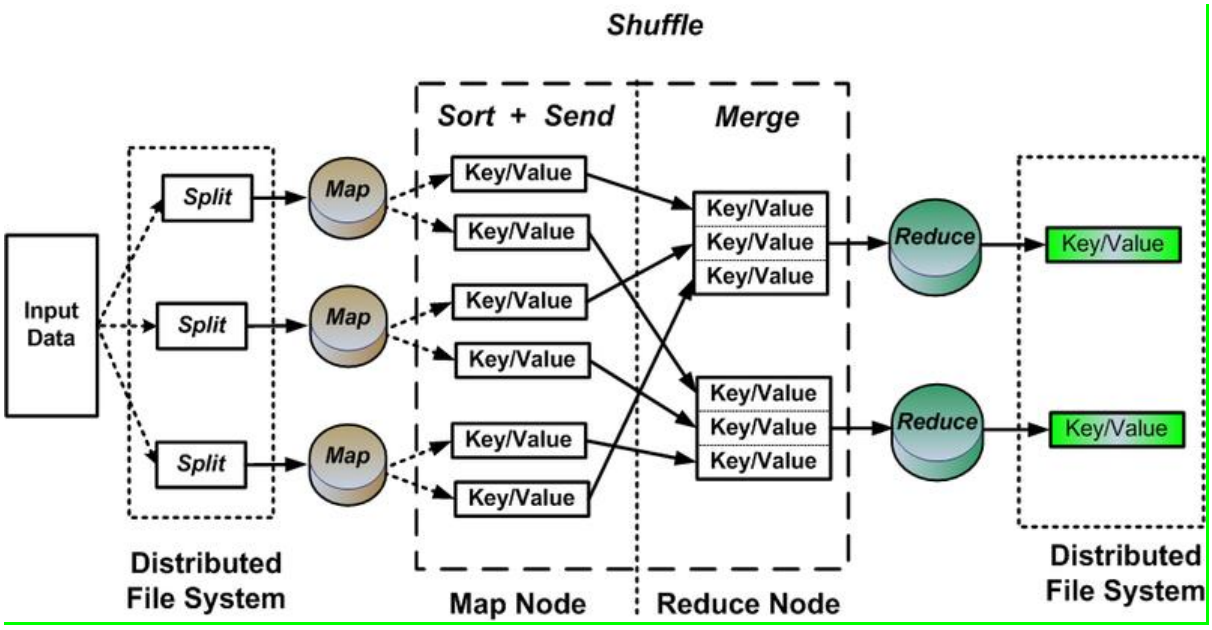

Fig. 3. MapReduce Process [39].

2) The tasks are distributed on multiple nodes in the MapReduce, which works as the computational model of Hadoop in order to process a large amount of data on clusters containing a large number of servers/nodes.

3) MapReduce works as the processing component of Hadoop because it processes the data concurrently on multiple nodes to reduce the overall computational cost and henceforth, increase the efficiency. When MapReduce gets data as input, it distributes it on the different nodes. The output of the mapper becomes intermediate data, which is in the split form. Subsequently, the shuffle process is used to exchange the data between various nodes. The data containing the same key is assigned to the same key of the reducer node and the output of the reducer is finally stored.

C. Hadoop WordCount

Lin, and Liu; Sharmila *et al.* [16, 17] stated that the Hadoop WordCount is a simple algorithm that is used to read the input text files, and count the number of unique words existing in a file.

Application	System Resource Utilization
WordSort	Sort Phase: IO-bound, Reduce Phase: Communication-bound.
Word Count	CPU-bound
TeraSort	Map Stage: CPU-Bound Reduce stage: IO-bound
NutchIndexing	IO-bound with high CPU utilizations in map stage. This workload is mainly used for web searching.
Kmeans	CPU-bound in the iteration phase, and IO-bound in the clustering phase. It is used for machine learning and data mining.

Fig. 4. Hadoop WordCount Resources Utilization [15].

In this algorithm, the input is a text file and different output files are generated in which information about words and their count is computed. On each level of this algorithm, different computing resources are utilized as depicted in Figure 5. In the Hadoop WordCount, the map and reduce functions perform in parallel. Figure 4 shows a comparison between some of the famous applications which are used to test the Hadoop system. It also presents the system resource utilization in the case of each application, and it can be noted that Word Count is a CPU-bound application.

V. The Word Count Algorithm

Figure 5 depicts WordCount Algorithm for counting the number of words occurrences in a file by using MapReduce programming.

```
1: Class Mapper <K1,V1,K2,V2 >
2: function map(K1,V1)
3: List words = V1.splitBy (token) ;
4: foreach K2 in words
5: write(K2,1) ;
6: end for
7: end function
8: end class
```

```
1: Class Reducer <K2,V2,K3,V3 >
2: function reduce(K2,Iterable<V2> itor)
3: sum = 0 ;
4: foreach count in itor
5: sum += count ;
6: end for
7: write(K3,sum) ;
8: end function
9: end class
```

Fig. 5. Algorithms of Word Count in Hadoop.

The principle is the same as above to count the integers: build pairs (key, list) where "key" is a word and "list" a list of 1 each designating an occurrence of a word.

The map step includes additional work which consists of breaking up the text file (or a simple character string) into words. A word is simply located between two patterns or the 'space' characters. The combined step is implicit before executing the reduce function toolbar. In Figure 5, the main job of the map function is to split the words of a line at a time through the tokens. A key value is assigned to each unique word and the output of this function is in the form of a key/value set, which contains the word and a key-value with the format: "<<word>, 1>. These key/value pairs are the inputs of the Reduce function. In the default order of the dictionary, the keys are sorted before the execution of the Reduce function. The reduce function job is to count the occurrence of each unique word. Finally, the Reduce function outputs the result on HDF. An example of the Map and Reduce function is shown in Figure 6. The original data is stored in a file containing different words, and after the WordCount algorithm is applied, the output is shown in the form of word counts.

In this example (depicted in Figure 6), the original data is shown in the left block. This data is retrieved from a file and mapped on the Map function. This file has different words such as Test, Reduce, Map, and HDFS. The map function splits these words using tokenization. In the map function, a key/value set is assigned to each unique word of the file which creates the key/value pairs and serves as the input of the Reduce function. The counting process is done by the Reduce function. The Reduce function count of each unique word is taken as the output, and it is stored on HDF to be later saved on the local storage.

Similar to the WordCount, there are many other applications of MapReduce, such as TeraSort and Sort. In the Hadoop MapReduce, TeraSort is used to sort the 1TB of data at an extremely fast rate. The HDFS file system makes it an ideal choice to fine-tune the configuration of a Hadoop cluster to quickly process these applications. In MapReduce, sort is an algorithm with an objective to process and analyze the data. According to [39], the improvement in MapReduce application is achieved at 59.15813, 64.23517, and 18.05475 for the TeraSort, Sort, and WordCount applications respectively, if the default settings of Hadoop at map level are considered.

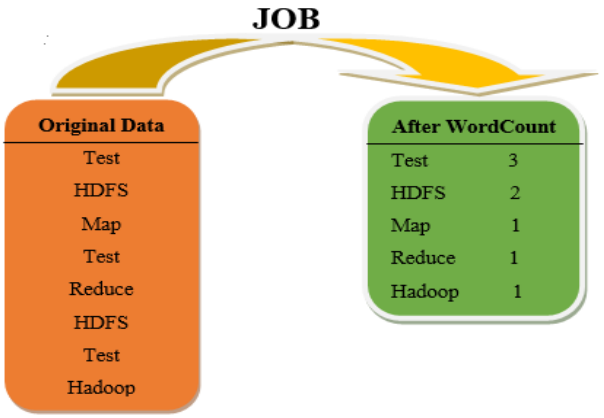

Fig. 6. Data Sample of WordCount.

VI. Spark based Big Data Analysis

Bell [19] establishes that the Apache Spark is a model-based open-source framework that is used to examine huge datasets in streaming applications. This framework was established in contrast with the Hadoop MapReduce model at UC Berkeley AMPLAB [19, 40] Architecture of Apache Spark is shown in Figure 7. Bell [19] showed that the main components of Apache Spark are the program of driver, initiators, cluster director, and the HDFS. The main program of Spark is its driver program. At the time of the startup of the Spark program, Spark Context is produced that plays a significant role in the whole implementation of the job [38]. The resources are managed in clusters when the Spark Context program is linked with the cluster manager. In order to store the app information and run the logic, the program is spawned through the cluster managers.

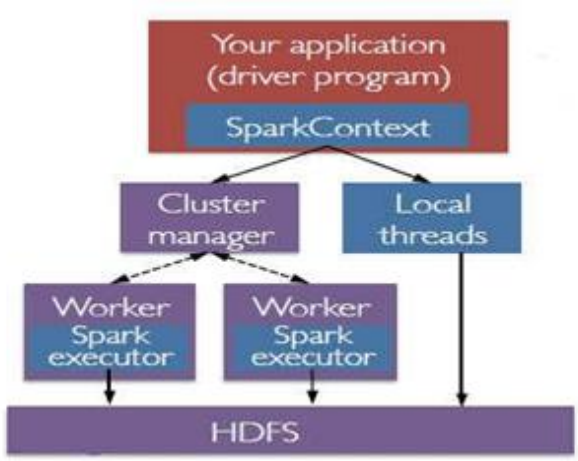

Fig. 7. Architecture of Spark.

Bell [19] said that Spark can be operated using multiple programming languages such as Scala, Java, Python, and R. Each language is provided with multiple libraries & APIs to support Big Data analytics. However, the most popular language in the Spark community is Scala, as Spark's core is implemented in Scala. Moreover, scalability is always a great concern in the case of Big Data analytics, which is handled efficiently in the Scala language. It is more compact compared to Java; a program written in Java is much larger in size compared to its equivalent of Scala. For comparison, it can be noted that one line of code in Scala is equal to 20-25 lines of code in Java. In order to enhance efficiency and reliability, numerous programs are being transformed from Java to Scala [33].

Bell [19] stated that the basic framework of Spark consists of two major technologies namely, Resilient Distributed Dataset (RDD) & Directed Acyclic Graph (DAG). The description of these two technologies as follows:

A. Resilient Distributed Datasets (RDD)

The Resilient Distributed Datasets (RDDs) are used to gather elements that are fault-tolerant and continue in parallel fashion as a primary concept of Spark [41]. Once the RDDs are created, they cannot be changed. It is impossible to change

them even with their ability to transform and perform actions. These datasets help to reorganize the computations and data processing enhancement [41]. Ganesh *et al.* [29] found a typical RDD provides information to regenerate on multiple nodes, and that is the primary reason for their fault tolerance. By transforming the current lists of data or changing the files in HDFS, the RDD is created. In case of misinformation of specific compartments of RDD, the default value is used by a spark programmer. A typical RDD executes two types of methods given as follows:

B. Transformation

A new RDD instead of a single value run at the time of performing changes on RDD. The analyses of computation are not sudden to transform, because the computation speed is very slow. When a program runs on this, then they are implemented. The following functions of transformation are performed: Mapping, Filtering, Reduce ByKey, FlatMap, and Group ByKey.

1) Action: A single value is examined and run when action methods are used on RDDs. At the time of an action method, the computation of data processing is performed and sent to a resultant value. In this respect, few action techniques are primary, take, decrease, gather, count, for each, and Count ByKey.

C. Directed Acyclic Graph (DAG)

Directed Acyclic Graph (DAG) engine is an updated Spark method to help cyclic data flow [40,41]. DAG consists of task phases that should be performed on the cluster of Spark nodes. Spark produces several DAGs of the input data that consist of an arbitrary number of steps, while DAG is further processed by the MapReduce which consists of two steps of Map and Reduce, as described in Section IV. Gohil *et al.* [15] showed that after the completion of a single step, this process allows for the processing of a simple task as compared to a complex task to be processed in multiple steps in a single run.

VII. Components of Apache Spark

Spark's underlying architecture is just like Hadoop, but it uses the in-memory system to process the streaming and graphics data. Due to the provision of in-memory processing, it can handle complex analyses on large volumes of data. There are multiple subsystems for memory handling, job assignment, fault tolerance, and storage, etc. Moreover, Spark can access information stored on different platforms such as Hadoop HDFS, Mesos, Mongo DB, Cassandra, H-Base, Amazon S3, and the data sources from standard cloud interfaces [34]. Figure 8 depicts major components of the Spark system. These components include Spark SQL, Spark Streaming, MLib, and GraphX [16]. The brief discussion these components are as follows:

A. Directed Acyclic Graph (DAG)

The primary component of this framework is Spark core. It handles all the basic functions related to I/O, dispatching, and arrangement of the distributed tasks. All the other components interact with the core to invoke their basic functionalities.

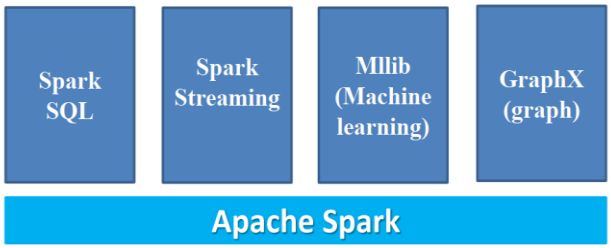

Fig. 8. Components of Spark.

B. Spark SQL

Spark SQL combines the traditional relational tables with RDDs for users to use SQL commands to communicate with large datasets for complex analytics [42]. It permits the use of old BI and visualization tools to allow performing SQL queries on Spark. An advanced RDD data concept method is introduced to provide support for structured and semi-structured data.

C. Spark Streaming

Kroß and Krcmar [43] stated that Spark Streaming is used for the processing of real-time data. To support the processing of real-time data of RDD, it uses an API called DStream. This component has the ability to seamlessly parallelize the data for streaming applications.

D. Spark GraphX

GraphX is a library that introduces 'the Resilient Distributed Property Graph' introduced by GraphX which has the ability to work with graphics and related computations [44]. Aggregate messages, sub-graphs and join vertices, and an optimized variant of Pregel API are various operators of GraphX. It also provides various graph-related algorithms and builders to simplify the graph analytics tasks.

E. MLlib

Meng *et al.* [45] found MLlib is an extensive framework to invoke Machine Learning (ML) options in Spark. It enables several basic ML algorithms like clustering, classification, and regression, etc. In Big Data, mining requires automation-based systems to dig out certain patterns of data, and MLlib provides a platform for data mining.

MLlib is a relatively new technology in Spark as compared to Mahout [46], which has been utilized as a set of Java APIs for machine learning. Mahout provides several optimized ML algorithms, thereby relieving its users to worry about the development of basic algorithms. Instead, they can work on their analytical problem on their datasets. The basic advantage of MLlib over Mahout is that it offers regression models [45,46].

VIII. Hadoop vs Spark

In the Big Data domain, the need for efficient, accurate, and reliable analytics along with excellent data visualization is increasing with every passing day [10, 11, 17]. Geczy [47] Identified that Tech giants like Yahoo, Amazon, Uber, eBay, Facebook, and Twitter, etc. heavily rely on these analytics for their smooth operations. These companies need optimized costs, real-time analytics, and fault tolerance with their data

processing. For more than a decade, Hadoop served as a prominent platform in order to provide these options to tech companies, government organizations like NASA & CERN, but the technological needs are becoming more and more advanced. Nowadays, Big Data analytics also require Machine Learning (ML) capabilities, streaming systems, and graphics processing [43–45]. Hadoop is an excellent platform for batch processing and large volumes of data, but it struggles with streaming applications and Machine Learning capabilities, etc. [10, 45]. Ward, and Barker [11] showed to overcome these limitations in Hadoop, Spark was introduced as another open-source project. It has built-in libraries for streaming, graphics, and ML capabilities, and it is designed to work on multiple programming languages.

Currently, both Hadoop and Spark are being used globally for different reasons, and they are the most prominent platforms for Big Data analytics and processing. Therefore, it is very hard to figure out which of these two is better. In [35], a comparison in terms of optimization is presented, and it is concluded that Spark performs much better in terms of accessing the storage systems and utilization of memory bandwidth. Table 1 present a brief comparison of Hadoop with Spark in terms of various features. The selection of these features from state-of-the-art research [2].

As shown in Table 1, the major difference between Hadoop and spark is the underlying methodology used to process the data [48]. Spark has the capability to perform Batch & Stream processing, whereas Hadoop can only operate with Batch processing. This particular capability is invoked in Spark by allowing in-memory processing, instead of disk-based computations in Hadoop. Hence, in the case of Hadoop, read and write operations are performed by seeking data available on a disk. This provides a fundamental advantage to Spark over Hadoop, as this difference in the technique of processing greatly affects the processing speeds. As all the processing in the case of Spark is performed in the memory, it takes lesser time to process data. According to [35], Spark is 100 times faster as compared to Hadoop.

Hadoop is most suitable in applications where the results are not immediately required in a short time. Contrarily, Spark is useful in applications when the results are needed in real-time. For instance, Spark is a good platform for stock price evaluation and enterprises [35].

Another major difference between the two frameworks is the amount of data to be processed. Htay and Phyu [39] indicated that Hadoop can process a massive amount of data as compared to Spark. Initially, Hadoop was also utilized for data archiving because Hadoop has the ability to store the data in large volumes due to HDFS. Using Hadoop, a user can store the data not only for a few years but for decades in its original and archiving form [1]. Chen and Zhang [7] stated that spark is simple to use as compared to Hadoop due to its easy-to-use APIs and many friendly features. If cost is taken in consider, Hadoop is less costly as compared to Spark. Similarly, Hadoop has better security features compared to Spark. Hadoop does not support data caching whereas Spark supports caching of data memory. Hadoop has a higher latency of computing as it has less data interactivity, as opposed to Spark [48].

As presented in Table 1, both Hadoop and Spark support auto-scaling, which means that if the data requires more nodes, then the frameworks give provision to add more nodes for distributing the workload automatically. Furthermore, as discussed above, Spark is flexible to support multiple programming languages in contrast to Hadoop only allowing Java programming.

Based on this comparison, it is very difficult to determine which framework is better. But generally, it can ascertain that both are useful for their own set of applications. For instance, Spark is better in real-time applications, where streaming data is required to be processed, whereas Hadoop is useful for processing large volumes of data when it is not strictly time-bound [1]. Furthermore, Spark provides excellent built-in APIs and libraries to optimize most of the user tasks, which can help to save a lot of programming effort. It has higher interactivity of data. Finally, it offers options for batch processing and streaming, as well as ML & graphic processing. Hence, Spark is obviously more advanced.

TABLE I. COMPARISON BETWEEN SPARK AND HADOOP

Features	**Hadoop**	**Spark**
Processing Mode	Batch	Batch and Stream
Scalability	Horizontal	Horizontal
Message Delivery Guarantees	Exactly once	Exactly once
Computation Mode	Disk-based	In memory
Auto-scaling	Yes	Yes
Iterative Computation	Yes	Yes
Speed	Slow	Fast
Amount of Data for Processing	More	Less
Security	More Secure	Less Secure
Cost	High	Low
Performance	Fair	Good
Language	Scala	Java
Data Caching	Support	No Support

IX. EXPERIMENTS AND RESULTS

To evaluate the performance of the MapReduce algorithm, the study used the Hadoop framework by implementing the WordCount algorithm in Java and performed the corresponding data analysis. The dataset consists of four different text files named chas.txt, ac.txt, xad.txt, and xaa.txt; all downloaded from the Internet. The corresponding file sizes are 202MB, 137MB, 69MB, and 34MB respectively. This test dataset contains random text comprising diverse words. Table 1 presents the study dataset.

To execute the data, a number of computing machines (or nodes) clustered across a network work together in MapReduce fashion. This number can be as large as in millions if the volumes of data are extremely large. The Map and Reduce functions may work on the same machine or on separate machines. The management of these machines is performed by master node(s).

A developed of a cluster of three machines to crunch the data given in Table 2 that used in the experiment. The hardware resources of the setup are presented in Table 3. The master node had four cores with main memory of 2GB, and it handled all the control processes. The other two nodes were single-core machines with 1GB of memory each.

The cluster executed the WordCount application, and it recorded the total time elapsed by the MapReduce function with respect to the size of each file. The algorithm first loads the text data into the HDFS data storage. When this data is available on the cluster, it can be used for analysis. Subsequently, the algorithm reads the text files and counts all the words in the given text files using MapReduce functions. The results are presented in Table 4. The investigation evaluated the time of Map and Reduce functions separately in the case of each file. The file sizes of the 202MB, 137MB, 69MB, and 34MB take 954645ms, 153826ms, 86224ms, and 57508ms to map all the words in each file by the Map function, respectively. On the other hand, the Reduce function takes 57964ms, 51089ms, 5095ms, and 6472ms, respectively.

TABLE II. TEST DATASET FOR WORD COUNT ALGORITHM

File name	File size (in MB)
chast.txt	202
ac.txt	137
xad.txt	69
xaa.txt	34

TABLE III. HARDWARE RESOURCES FOR TEST CLUSTER

Machine	Cores	Memory	Network
Master	4	2GB	192.168.1.4
Slave1	2	1GB	192.168.1.6
Slave2	2	1GB	192.168.1.7

TABLE IV. HADOOP-MAPREDUCE (WORDCOUNT) IMPACT ON FILES SIZE

Data File (in MB)	Time elapsed by Map function (ms)	Time elapsed by Reduce function (ms)
202 (chast)	954645	57964
137 (xac)	153826	5108
69 (xad)	86224	5095
34 (xaa)	57508	6472

Figure 9 shows the graphical representation of the MapReduce function time on the WordCount algorithm. The slope of the graph shows that the map function takes more time compared to the reduced function for the WordCount algorithm. It can be noted that the time elapsed by the Map and Reduce functions are directly proportional to the size of the file. As the size of the file increases, the time elapse of the Map and Reduce functions is also increased. The slope of the reducer function goes upward slightly as the size of the file increases, but the time slope of the mapper function goes upward at 75 degrees as the file size increases from 137MB to 202MB. The difference between these two files is not too much. The performance of the mapper function is not very prominent in case of an increase in the file size. This is because Hadoop is normally optimized for the data file sizes in GBs and TBs. Another reason for the timing results is the size of the cluster that have used. Hadoop is normally utilized for clusters with thousands of nodes, crunching data volumes in the range of gigabytes & terabytes, etc.

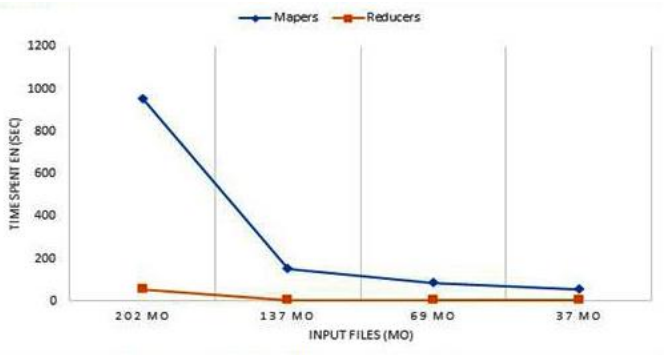

Fig. 9. WordCount Algorithm Time Impact.

X. DISCUSSION

The study implemented the WordCount application using Hadoop and executed it in a cluster, as discussed in Section IX. If all the parameters and hardware resources are kept the same, and the WordCount algorithm is implemented on the Spark framework, the study note that the overall processing time is reduced by a factor of 10. Similarly, the in-line memory processing option in Spark makes it 100x faster for memory operations. Another advantage of Spark is the reduced number of Lines of Codes (LoCs) of Scala. The Hadoop implementation took 60 LoCs in Java to write the MapReduce function, but the same code took only 5 LoCs in Scala in the case of Spark. Therefore, the only complication in the case of Spark is learning a Scala, but this learning can reduce a lot of effort.

XI. CONCLUSION AND FUTURE WORK

This article described two frameworks, Hadoop and Spark, which allow the processing of Big Data on clusters of computing machines. The study first discussed the main characteristics of Big Data followed by the primary components of Hadoop and Spark. Then it discussed the underlying storage architecture called HDFS in both frameworks. Afterwards it detailed the MapReduce algorithm used as a core program in Hadoop operations. Subsequently, it differentiated both the frameworks on the basis of various features like cost, scalability, programming languages, and processing modes, etc. The enquiry concluded that Spark is better for streaming applications, while Hadoop is better in the case of processing large datasets in batch processing mode. One of the main objectives of this work is to evaluate the performance of the WordCount application in terms of processing time for both frameworks. Finally, the study provided a discussion on the limitations of Hadoop as compared to Spark while processing the WordCount application. The investigation concludes that the performance of Hadoop can be measured on the basis of different aspects, such as tuning of the MapReduce parameters and the total

number of nodes, etc. In the future, it intend to extend this work by taking big datasets on a larger cluster and report the memory speed and execution times for more complex applications. The research will compare the performance of Hadoop and Spark for a given set of application parameters, and propose an optimization function to choose a particular framework for a given application. It will evaluate the performance of these applications in terms of data growth, the number of iterations, and data processing in real-time. Unfortunately, the current version of Spark is unable to handle a large number of workloads using SQL-like queries, as required by tech giants like Google, Facebook, and Yahoo, etc. Consequently, some modifications are suggested in newer Spark versions to allow multithreading options to spawn application tasks more efficiently. The research intends to further explore Spark to find out optimized options for application submission using large datasets.

ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”.Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

[1] Z. Tufekci, “Big Questions for social media big data: Representativeness, validity and other methodological pitfalls,” Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014, pp. 505–514, 2014.

[2] R. Sakthivel, V. Parthipan, and D. Dhanasekaran, “Big data analytics on smart and connected communities using Internet of Things,” International Journal of Pharmacy and Technology, vol. 8, no. 4, pp. 19590–19601, 2016.

[3] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, “Deep learning for IoT big data and streaming analytics: A survey,” ArXiv, vol. 20, no. 4, pp. 2923–2960, 2017.

[4] U. Srivastava, and S. Gopalkrishnan, “Impact of big data analytics on banking sector: Learning for Indian Banks,” Procedia Computer Science, vol. 50, pp. 643–652, 2015, doi:10.1016/j.procs.2015.04.098.

[5] https://www.socialmediatoday.com/news/10-social-media-statistics-you-need-to-know-in-2019-infographic/559181/.

[6] https://www.statista.com/markets/.

[7] C.L. Philip Chen, and C.Y. Zhang, “Data-intensive applications, challenges, techniques and technologies: A survey on Big Data,” Information Sciences, vol. 275, pp. 314–347, 2014, doi:10.1016/j.ins.2014.01.015.

[8] S. Venkatraman, K.F.S. Kaspi, and R. Venkatraman, “SQL Versus NoSQL Movement with Big Data Analytics,” International Journal of Information Technology and Computer Science, vol. 8, no. 12, pp. 59–66, 2016, doi:10.5815/ijitcs.2016.12.07.

[9] T.K. Das, and P. Mohan Kumar, “Big data analytics: A framework for unstructured data analysis,” International Journal of Engineering and Technology, vol. 5, no. 1, pp. 153–156, 2013.

[10] M. Chen, S. Mao, and Y. Liu, “Big data: A survey,” Mobile Networks and Applications, vol. 19, no. 2, pp. 171–209, 2014.

[11] J.S. Ward, and A. Barker, “Undefined By Data: A Survey of Big Data Definitions,” ArXiv Preprint ArXiv:1309.5821, 2013.

[12] D. Zhao, “Performance comparison between Hadoop and HAMR under laboratory environment,” Procedia Computer Science, vol. 111, pp. 223–229, 2017, doi:10.1016/j.procs.2017.06.057.

[13] N.M.F. Qureshi, and D.R. Shin, “RDP: A storage-tier-aware robust data placement strategy for hadoop in a cloud-based heterogeneous environment,” KSII Transactions on Internet and Information Systems, vol. 10, no. 9, pp. 4063–4086, 2016, doi:10.3837/tiis.2016.09.003.

[14] K. Bangari, S. Meduri, and C.C.Y. Rao, “Implementation of Word Count-Hadoop Framework with Map Reduce Algorithm.” International Journal of Computer Trends and Technology (IJCTT), vol. 49, no. 3, pp. 179-182, 2017.

[15] P. Gohil, D. Garg, and B. Panchal, “A performance analysis of MapReduce applications on big data in cloud based Hadoop,” in 2014 International Conference on Information Communication and Embedded Systems, ICICES 2014, IEEE: pp. 1–6, 2015, doi:10.1109/ICICES.2014.7033791.

[16] W. Lin, and J. Liu, “Performance analysis of MapReduce program in heterogeneous cloud computing,” Journal of Networks, vol. 8, no. 8, pp. 1734–1741, 2013, doi:10.4304/jnw.8.8.1734-1741.

[17] K. Sharmila, S. Kamalakkannan, R. Devi, and C. Shanthi, “Big data analysis using apache hadoop and spark,” in International Journal of Recent Technology and Engineering, IEEE: pp. 167–170, 2019, doi:10.35940/ijrte.A2128.078219.

[18] V. Kalavri, and V. Vlassov, “MapReduce: Limitations, optimizations and open issues,” in Proceedings - 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013, IEEE: pp. 1031–1038, 2013, doi:10.1109/TrustCom.2013.126.

[19] J. Bell, “Apache Spark,” Machine Learning, vol. 17, pp. 275–314, 2015, doi:10.1002/9781119183464.ch11.

[20] B. Data, “Big data characteristics and sources,” The Macrotheme Review, vol. 3, no. 6, pp. 8–10, 2017.

[21] Begum, F. Fatima, and R. Haneef, “Big Data and Advanced Analytics,” in Advances in Intelligent Systems and Computing, IEEE: pp. 594–601, 2019, doi:10.1007/978-3-030-11890-7_57.

[22] N. Elgendy, and A. Elragal, “Big data analytics: A literature review paper,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer: pp. 214–227, 2014, doi:10.1007/978-3-319-08976-8_16.

[23] J.R. Saura, B.R. Herraez, and A. Reyes-Menendez, “Comparing a traditional approach for financial brand communication analysis with a big data analytics technique,” IEEE Access, vol. 7, pp. 37100–37108, 2019, doi:10.1109/ACCESS.2019.2905301.

[24] Shah, and M. Padole, “Saksham: Resource Aware Block Rearrangement Algorithm for Load Balancing in Hadoop,” Procedia Computer Science, vol. 167, pp. 47–56, 2020, doi:10.1016/j.procs.2020.03.181.

[25] K. Muthukkaruppan, K. Ranganathan, and L. Tang,. U.S. Patent Application No. 14/996,627, 2016.

[26] L. Meng, W. Zhao, H. Zhao, and Y. Ding, “A network load sensitive block placement strategy of HDFS,” KSII Transactions on Internet and Information Systems, vol. 9, no. 9, pp. 3539–3558, 2015, doi:10.3837/tiis.2015.09.014.

[27] W. Dai, I. Ibrahim, and M. Bassiouni, “An Improved Replica Placement Policy for Hadoop Distributed File System Running on Cloud Platforms,” in Proceedings - 4th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2017 and 3rd IEEE International Conference of Scalable and Smart Cloud, SSC 2017, IEEE: pp. 270–275, 2017, doi:10.1109/CSCloud.2017.65.

[28] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F.B. Cetin, and S. Babu, “Starfish: A Self-tuning System for Big Data Analytics.,” in Cidr, pp. 261–272, 2011.

[29] P. Ganesh, K. Sailaja Kumar, D. Evangelin Geetha, and T. V. Suresh Kumar, “Performance evaluation of cloud service with hadoop for twitter data,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 13, no. 1, pp. 392–404, 2019, doi:10.11591/ijeecs.v13.i1.pp392-404.M. Khan, Y. Jin, M. Li, Y. Xiang, and C. Jiang, “Hadoop Performance Modeling for Job Estimation and Resource Provisioning,” IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 2, pp. 441–454, 2016, doi:10.1109/TPDS.2015.2405552.

[30] V. Taran, O. Alienin, S. Stirenko, Y. Gordienko, and A. Rojbi, “Performance evaluation of distributed computing environments with hadoop and spark frameworks,” in arXiv, IEEE: pp. 80–83, 2017.

[31] Jain, and M. Choudhary, “Analyzing & optimizing hadoop performance,” in Proceedings of the 2017 International Conference On

Big Data Analytics and Computational Intelligence, ICBDACI 2017, IEEE: pp. 116–121, 2017, doi:10.1109/ICBDACI.2017.8070820.

[32] S. Londhe, and S. Mahajan, "Effective and Efficient Way of Reduce Dependency on Dataset With the Help of Mapreduce on Big Data," International Journal of Students' Research in Technology & Management, vol. 3, no. 6, pp. 401, 2015, doi:10.18510/ijsrtm.2015.364.

[33] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[34] M. Santos, K. Santos, E. Alves, and A. Dantas, "CPU Bound Analysis of Wordcount Application in Hadoop Yarn Virtualized Nodes Using the Xen Platform," in 2018 Symposium on High Performance Computing Systems (WSCAD), IEEE: pp. 274–274, 2019, doi:10.1109/wscad.2018.00056.

[35] P.J. Morris, "The dawn of big data.," in North Carolina medical journal, IEEE: pp. 177, 2014, doi:10.18043/ncm.75.3.177.S. Nishimura, S. Das, D. Agrawal, and A. El Abbadi, "MD-HBase: A scalable multi-dimensional data infrastructure for location aware services," in Proceedings - IEEE International Conference on Mobile Data Management, IEEE: pp. 7–16, 2011, doi:10.1109/MDM.2011.41.

[36] T. White, "Hadoop: The definitive guide 4th Edition," Online, vol. 54, pp. 258, 2012, doi:citeulike-article-id:4882841.

[37] T.T. Htay, and S. Phyu, "Improving the performance of Hadoop MapReduce Applications via Optimization of concurrent containers per Node," in 2020 IEEE Conference on Computer Applications, ICCA 2020, IEEE: pp. 1–5, 2020, doi:10.1109/ICCA49400.2020.9022836.

[38] R.K. Chawda, and G. Thakur, "Big data and advanced analytics tools," in 2016 symposium on colossal data analysis and networking (CDAN), IEEE: pp. 1–8, 2016.

[39] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M.J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in Proceedings of NSDI 2012: 9th USENIX Symposium on Networked Systems Design and Implementation, pp. 15–28, 2012.

[40] M. Armbrust, R.S. Xin, C. Lian, Y. Huai, D. Liu, J.K. Bradley, X. Meng, T. Kaftan, M.J. Frankliny, A. Ghodsi, and M. Zaharia, "Spark SQL: Relational data processing in spark," in Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 1383–1394, 2015, doi:10.1145/2723372.2742797.

[41] J. Kroß, and H. Krcmar, "Modeling and Simulating Apache Spark Streaming Applications," Softwaretechnik-Trends, vol. 36, no. 4, pp. 1–3, 2016.

[42] J.E. Gonzalez, R.S. Xin, A. Dave, D. Crankshaw, M.J. Franklin, and I. Stoica, "GraphX: Graph processing in a distributed dataflow framework," in Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2014, pp. 599–613, 2014.

[43] Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D.B. Tsai, M. Amde, and S. Owen, "Mllib: Machine learning in apache spark," The Journal of Machine Learning Research, vol. 17, no. 1, pp. 1235–1241, 2016.

[44] D. Lyubimov, and A. Palumbo, Apache Mahout: Beyond MapReduce, CreateSpace Independent Publishing Platform, 2016.

[45] H. Schildt, "Big data and organizational design–the brave new world of algorithmic management and computer augmented transparency," Innovation: Management, Policy and Practice, vol. 19, no. 1, pp. 23–30, 2017, doi:10.1080/14479338.2016.1252043.

[46] P. Geczy, "Big data characteristics," The Macrotheme Review, vol. 3, no. 6, pp. 94–104, 2014.