Review Article

A Comprehensive Survey for Hadoop Distributed File System

ABSTRACT

In the last few days, data and the internet have become increasingly growing, occurring in big data. For these problems, there are many software frameworks used to increase the performance of the distributed system. This software is used for available ample data storage. One of the most beneficial software frameworks used to utilize data in distributed systems is Hadoop. This software creates machine clustering and formatting the work between them. Hadoop consists of two major components: Hadoop Distributed File System (HDFS) and Map Reduce (MR). By Hadoop, we can process, count, and distribute each word in a large file and know the number of affecting for each of them. The HDFS is designed to effectively store and transmit colossal data sets to high-bandwidth user applications. The differences between this and other file systems provided are relevant. HDFS is intended for low-cost hardware and is exceptionally tolerant to defects. Thousands of computers in a vast cluster both have directly associated storage functions and user programmers. The resource scales with demand while being cost-effective in all sizes by distributing storage and calculation through numerous servers. Depending on the above characteristics of the HDFS, many researchers worked in this field trying to enhance the performance and efficiency of the addressed file system to be one of the most active cloud systems. This paper offers an adequate study to review the essential investigations as a trend beneficial for researchers wishing to operate in such a system. The basic ideas and features of the investigated experiments were taken into account to have a robust comparison, which simplifies the selection for future researchers in this subject.
According to many authors, this paper will explain what Hadoop is and its architectures, how it works, and its performance analysis in a distributed systems. In addition, assessing each Writing and compare with each other.

Keywords: Hadoop, HDFS, Distributed File System.

1. INTRODUCTION

Data are now more redundant over the internet and dispersed systems. In general, on different servers, there are 4 petabytes of data. Technologies analyze this enormous data in a complicated manner. This data is saved in parallel processing in multiple distribution computers and access to the data. There is, therefore, increasing rivalry in similar processing systems for access to shared resource data. Many approaches can be employed to tackle these difficulties. The investigations tend to dismantle enormous quantities of data through a break-up and simultaneous resolution of the problem [1].

For example, a distributed cloud computing system provides an immense data processing mechanism. In addition, a combination of a distributed system and parallel processing may resolve some issues for clients remotely in a minimum of time. Shared memory systems and distributed memory systems are two more techniques in a similar hybrid processing system that solves complicated network challenges, such as data size restriction. A fundamental problem is the performance of the distributed system. For example, the performance of a three-stage 3TA Architecture system is better and more accurate than 2TA systems, which utilize Opnet as an evaluation and design tool. The system architecture has a significant effect on system performance. The performance of the distributed system is analyzed using several approaches. However, Hadoop is one of the most popular methods. It is a framework, and an open source software application that distributes processes that store and handle the large data application operated by the clustered system [1]. The research question addressed in this paper related to how to prepare a comprehensive study to address the Hadoop distributed file system. The trend of this paper is to help the new researchers in this field to have a broad scope of the Hadoop file system.

One of the latest jobs in software technology trends developed methods to store, manipulate and retrieve data from enormous data volumes. Hadoop [2] is a distributed filesystem and a multi-data processing and analysis platform based on the MapReduce methodology [3]. Ten years after it started as an open-source project, Hadoop became the most frequently used computer platform for distributed data storage and processing [4]. A key characteristic of Hadoop is the separation of data and processing over multiple (thousands) hosts and, in parallel with their data, the execution of application calculations [5]. A Hadoop cluster will improve computer power, storage capacity, and bandwidth by adding different commodity servers. Hadoop is an Apache project. The Apache open-source license allows all its modules to be distributed freely. Yahoo! established Hadoop's Centre and donated 80% of it (HDFS and MapReduce). HBase is currently a Microsoft division, developed at Powerset. Facebook has developed and generated Hive [6]. Pig [7], ZooKeeper [8], and Chukwa conceived and acquired Yahoo!. In cooperation with Cloudera, Avro was created at Yahoo!.

Hadoop was created, a practical open-source application, due to the necessity of MapReduce. Hadoop is currently utilized for backend data analysis by many business and academic users, developed in Java for cross-platform portability. The distributed Hadoop file system (HDFS) is a crucial component of Hadoop and is utilized for the storage of both input and output data for applications [9]. HDFS separates metadata and files of the device from metadata. HDFS maintains metadata on a dedicated server known as the Name Node same like other distributed file systems such as PVFS [10] and GFS [11]. Application data is saved on extra Data Nodes servers. Both servers are fully connected with each other and interact with each other using TCP protocols.

This paper consists of six sections. The rest are organized in the following style: Section two provides a detailed explanation of the background theory of the addressed subject. Section three represents reviewing the number of most previous works related to the Hadoop file system. A detailed discussion with a good comparison is browsed in section four. Necessary recommendations are given in section five. Finally, the conclusion of this comprehensive study is illustrated in section six.

2. BACKGROUND THEORY

2.1 Hadoop

A free, open-source Apache Foundation project, Hadoop, is a Java framework. It enables enormous volumes of data to be processed in a cluster of one or more hundred machines. This is the first technology that allows you to digitally store, manage and analyze an endless amount of data to allocate suitable work to the system concerned. TECHNO [12] includes Hadoop's two core services: Hadoop Distributed File System (HDFS) data storage and MapReduce technique, large-scale parallel data processing [13].

2.2 Fiber Optic Communication principles

Every Hadoop distribution has a Big Data system notion. Several studies to define and describe big data as a massive data volume have been undertaken. In addition, big-data features are speed, variety, and increasing data volume [14]. Big data are divided into three types: structured, unstructured, and half-structured data. Additional big data categories include pictures, video, audio, and natural language [15]. Highly organized structured data is, however unstructured data is not maintained systematically and clearly. For example, Wikipedia, Google, Facebook, and Amazon utilize unstructured data formats, whereas e-commerce businesses use structured data formats [12, 16]. NoSQL is a new database technology class designed to handle "Not Only SQL." Unstructured and semi-structured data eventually generates a variable number of data fields and diverse content, providing a challenge for the database model [17]. Current systems of NoSQL might be categorized into four large groups. Key/Value: This idea is like a distributed hash map. It maintains information as a key/value pair, where the value may be an integer or serialized object string [18].

Column-oriented: The data is kept in a row with columns. It simulates a relational database [19].

Document-oriented: The document-oriented model's ability to retrieve a hierarchically structured set of information using a single key distinguishes it [20].

Graph-oriented: the basis of graph theory is primarily based on the concept of nodes, connections, and attributes associated with the nodes [21].

2.3 HDFS

Hadoop distributed file systems need security solutions to safeguard their data while retaining high performance. To be secure, several researchers assume that HDFS is encrypted [13]. In this situation, big data is divided into 64MB or 128MB. Three duplicates are made of each block, and these copies are stored on three different computers [15]. When the system has a large amount of data and a simultaneous job, accessing the HDFS files may need multiple interacting Name Node and Data Nodes connections, which considerably lowers access speed [22].

Name Node maintains the hierarchical file tree structure in the filesystem. The files are stored as blocks on behalf of the customer by the Data Nodes [23]. Each block is reserved to the local node filesystem as a separate file. Data nodes do not need to be equal in their features since Data Nodes abstract the underlying filesystem details [24].

2.4 HDFS Architecture

The name node and data node are software components that may be executed on commodities machines. HDFS is created in Java language and can run the Nama node or Data node software on any Java-supporting device. The Java language is highly portable and allows for HDFS on multiple computers [25].

The Hadoop user can process the dataset on the local system on a single node by using local mode or stand-alone mode. The Hadoop Distributed File System (HDFS) and MapReduce are Hadoop's key components [26]. The HDFS is developed in Java and splits the dataset file into blocks according to data size. In processing the data set, the HDFS employs the Name and Data Node systems [27].

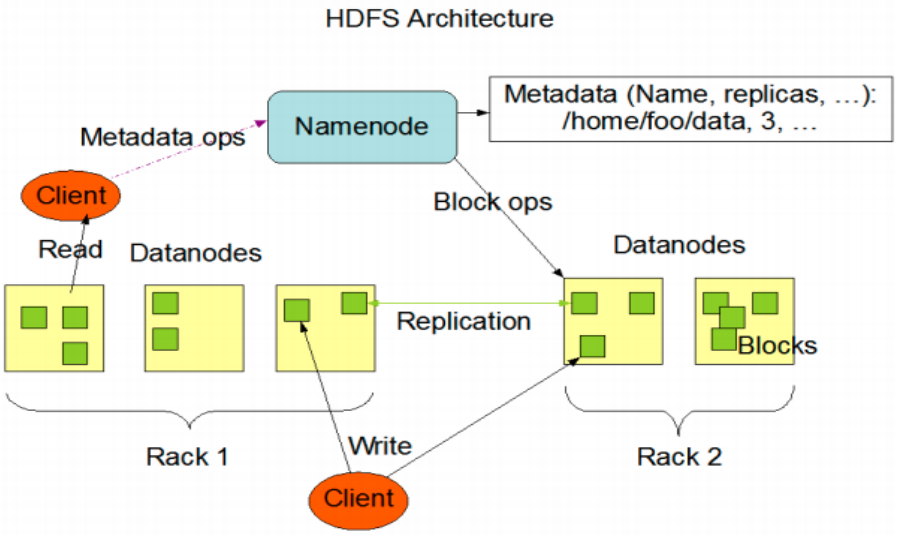

Fig. 1. HDFS Architecture [1]

2.4.1 Name Node:

The namespace for HDFS is a hierarchical file and directory. Files and folders on the Name Node are specified using inodes that hold attributes like rights, changes and access times, namespace, and disk space quotas [28].

The Name Node keeps the namespace tree and routes file blocks in Data Nodes 2 (the physical location of file data) [29]. When an HDFS customer wishes to read a file, they first contact the Name Node to identify the data blocks in the file and then receive the information on the block from the Data Node closest to the customer [1, 30]. The client asks the client to pick a group of three data nodes to host replicas of the block while typing data. The customer then transfers data in a pipeline format to the Data Nodes. Every cluster has its Name Node in the current arrangement [31].

HDFS holds all namespace in the RAM. Indole data and the list of blocks belonging to every file are the naming system's image metadata. A control point is a persistent picture archive saved on the default localhost filesystem [32]. The Name Node also keeps the record of changes to the picture, called the document, on the original file system of the localhost. Redundant checkpoint and diary copies can be generated to improve durability on different servers [33]. The Node name reads the namespace and replays the log to reset the namespace upon restarting. Block copies that vary over time and do not form part of the permanent checkpoint.

2.4.2 Data Nodes

In the native file system of localhost, three files support each Block Replication on the Data Node [34]. The first file contains the information, and the second file contains the block's metadata and gives the block data and the stamp creation [35]. The data file size equals the whole block length and does not need considerable space, as in traditional file systems, to round it to its nominal block size. Therefore, if a block is half-filled, the local drive requires just half the size of a complete block [36].

Each Data Node block reproduction is defined by two files in the default filesystem of the localhost. The Name Node does not contact Data Nodes directly [37].

It delivers commands in response to heartbeats to the data knots. The following guidelines include:

- Distribute blocks to other nodes.
- Get rid of local block replicas;
- Re-register the Node or shut it down;

- Submit a block report right away.

As these instructions are necessary to preserve the overall integrity of the device, it is crucial to maintain heartbeats even tiny clusters [38]. Despite interfering with other Name Node activities, the Name Node can handle miles of heartbeats every second [37].
Data Node without namespace ID was born in an era, and the cluster's namespace ID is permitted to enter and gather [36].

2.4.3 HDFS Client

HDFS organizes a group to read, edit, and delete files and transactions for creating and deleting folders, like most classic file systems [39]. The administrator uses namespace paths to connect to files and folders. The user application must not be aware that the metadata and persistence of the file system are hosted in various repositories or blocks that have several replicas [36].

2.4.4 Image and journal

The namespace image is the metadata system file defining how to organize data in folders and files. The image archive written to disk is a continuous checkpoint [37].

2.4.5 Checkpoint Node

The Node includes the current control point and diary for the regular construction of a new control point and a blank journal. Since the Checkpoint Node has the same RAM as the Name Node, it is generally run on a different host [40].

2.4.6 Backup Node

The Backup Node allows you to construct intermittent control points and maintains a current image in the memory of the file system's name that is always manfully aligned with the Name Node [37].

2.4.7 Upgrade, file system snapshot

Device updates design snapshots in HDFS to decrease the chance of data loss [41]. The snapshot feature allows admins to record frequently the current status of the file system, meaning that the upgrade may be rolled back and HDFS restored in the name and storage state of the file at the time of the shoot if a promotion leads to data failure or abuse [27].

HDFS was meant to store large files with data access streaming patterns. As said before [42]. In other words, as seen below, there are problems with minimum data:

- **High Name Node's memory consumption**

Node Name eats a lot of memory. Name Node stores metadata in the main memory. A total of around 250 bytes of main memory are used in a document's metadata. Metadata will absorb roughly 368 bytes of every block with the usual three copies [43].

- **Unacceptable storing time**

For example, 550,000 tiny files of between 1KB and 10KB in size are saved in HDFS in around 7.7 hours. On the other hand, it takes approximately 660 seconds to keep such data in a local file system, such as ext3 [44].

- **Name Node becomes the bottleneck**

Metadata maintenance in HdFS is a time-consuming operation, as it requires node coordination [31]. The HDFS client must first obtain the file's metadata from the Name Node to access a file. For tiny files, data transfer needs very little time, but disk search and metadata management constitute considerable overhead [32]. The HDFS client must call Name Node, often with a high number of tiny files, which can substantially influence the performance of Name Node [45].

For the management of a large number of tiny files, Hadoop Archive (HAR) gives Hadoop. The user groups and saves all the small files in a particular archive format (.her) [46]. The Hadoop archive command uses the HAR command to construct a task for MapReduce to compress a set of local files onto large files to enable the parallel (file extension free) and efficient retrieval of the original files [47].

2.5 MapReduce

MapReduce is a software framework that belongs to the Hadoop context. It can manage large quantities of data used in thousands of nodes in the terabyte range [48]. The MapReduce approach splits maps into maps and reduces the functionality [49]. Users supply a map function that processes a key/value pair to produce a collection of mid-key/value pairs and decreases a function that merges all the mid-term values with the same key [50]. MapReduce and HDFS are the fundamental components of Hadoop. In essence, HDFS employs a writing and reading process mechanism to disseminate data inside a local node (Single Node model) or via many nodes (Multiple Nodes Model) [14]. HDFS offers the MapReduce replication feature to enhance performance. MapReduce is a master node (Job Tracker) and multiple slave nodes (Task Trackers). A JobTracker is responsible for the task trackers group of slave nodes [15].

MapReduce operates on the acceptable items by first running the map function and reducing the unwanted things [51]. The implementation of MapReduce is based upon programs in many languages such as Java, C, or Python [50]. MapReduce is a Hadoop function that reorganizes a dataset's content [52]. A program code essentially comprises the Mapping funktion and subsequently the Reducer function for rearranging items in a dataset. The primary and value rules are applied on target objects to use Map and Reduce processes [48].

3. LITERATURE REVIEW

The distributed file system has always been necessary for continued progress and expansion since the 1990s. Chandrasekhar and others [24]. Proposed Extended Hadoop Distributed File System (EHDFS) to enable the combination of a vaster number of small files, increase access to small files efficiently and improve EHDFS metadata management for smaller files to improve memory usage HDFS resources.

The techniques of simulation and modeling relied on Mendoza and Lorene [53]. To examine the system's behavior inside a cluster of workstations, many simulations have been done. (HDFS) Model of colored Petri Networks (CPN) to study and evaluate the accessibility of workstations to a model by exploring different configurations and alternative approaches. The simulation findings show that acceptance or rejection of the Name Node pipeline is a key constraint.

The automatic benchmarking system was introduced by Kim et al. [54] (ABCM). This work developed the identification process for the set of settings parameters, reducing the benchmark runtime. Primarily TestDFSIO writes and reads the primary setup generated by ABCM to change the Benchmark Time. Optimal parameters have been determined, lowering by 32 percent the average execution time compared to the default set of Hadoop setup options. The four kinds of NoSQL databases are included by Erraissi and Belanger [12] as proposed. For significant data efforts, several companies use this software platform and its various components. Ethiopia and others [15] The complexity of the time of an algorithm indicate how long an algorithm takes to finish. O (long) time complexity is fair scheduling. Masmoudi and Almansouri [14]. Hadoop proposed on one node architecture allows the cost-effective analysis on a local workstation and automatic HDFS backup. The proposed encryption of Mahmoud et al. [54] HDFS files were encrypted with AES and OTP methods. Enhance the Encryption/Decryption file performance of this technique. Liao and al . [55] Presented a hierarchical approach to structural structure, which may be utilized to facilitate

the processing of HDFS data and B-tree and R-tree variants. The distributed caching system built on top of the HDFS dubbed HDCache was described by Zhang et al. [22]. Use standard memory to compensate for the shortcomings in performance (HDFS). Vijayakumari and others[56]. Apache-owned components of the Hadoop Project include Hadoop Distributed File System and MapReduce. Comparisons are made between these two file systems with specific parameters (Security, File serving ... etc.).
Wang et al. [57] proposed Zput's Remote Block Placement support and design. Zput's significant benefit over HDFS-put is an improved upload efficiency while avoiding adverse effects. They provided Hua et al. [58] with rigorous interaction tasks. The HDFS modifications are: (2) the caching on each rack to increase I/O functionality in accessing interaction-intensive files; (3) the use of PSO-based methods to establish a near-optimum storage allocation plan for incoming documents. Shahabinejad et al., respectively [59]. They have suggested a locally repairable binary code (BLRC) since it does not include finite set multiplication, encoding, decoding, and repair, which saves considerable time. Krishna and others [60]. They realized that HDFS works well for files more extensive than the default blocks and poorly for files smaller than the standard blocks. Clubric et al. [61] The security of essential data, not accomplished by Kerberos, at an HDFS storage level. Day and al.[62] Suggest a novel replica placement method for HDFS that addresses load balancing through the consistent distribution of replicas onto cluster nodes and hence eliminates the need to provide any load balancing utility. Qu et al. Qu et al. [63] The DRS approach based on an improved Markov model of the chain is proposed. The Markov model distinguishes between various data kinds and changes the copies dynamically, which will then be spread equitably throughout the rack, depending on the connection between the files. Zebedie and al. [1] In distributed systems, Hadoop's performance is higher than other technologies used for the same purpose. Several important companies, like Facebook, have implemented Hadoop.

4. DISCUSSION AND COMPARISON

This article must emphasize that every author has a distinct perspective but identical aims while analyzing HDFS performance. Hadoop's adaptation to distributor systems aims to speed up massive data storage, processing, analysis, and management. The Hadoop in an original manner, including Twenty researchers covered in the literature review (architecture and operation). An overview of the comparison of twenty prior publications is provided in Table 1. They were comparing the performance of Hadoop in the distributed system area with previously dependent approaches. The comparison focuses on the dependent instruments, the aims attained, and the actual results for each study. Each writer describes Hadoop's architecture and operation. Comparing Hadoop's performance with other previous techniques in the distributed system

Table 1. Comparisons of related works

Ref. no.	Tools	Achieved Objectives	Significant Results
Mendoza and Quesada **[53]**	CPN the Colored Petri Nets combined with the CPN ML programming language	The feasibility of exploiting the idle computational storage in a large Cluster of Workstations (COW).	To achieve a reliable service for Writing and reading files despite the random failures due to the turning on and off of the computers in a COW with hundreds of machines.
Aswan et al. **[64]**	small single rack implementation and multi-rack implementation of the HDFS	the high overview of Hadoop Distributed File System architecture and different server roles	MapReduce is used for implementation, and HDFS is in charge of storing massive datasets.
Chandra sekhar et	Extended Hadoop Distributed File System (EHDFS).	To minimize the file count, a collection of associated files discovered by the client is	EHDFS can minimize metadata while increasing the efficiency of storing and accessing a large number of tiny files.

al. **[24]**		merged into a huge file.	
Kim et al. **[54]**	small single rack implementation and multi-rack implementation of the HDFS	the high overview of HDFS architecture and different server roles	MapReduce is used for implementation, although suitable for dataset management and storage, HDFS assumes the job of storing massive datasets.
Manias and Schroeder [4]	(HDFS)'s code evolution. based on the reports and patch files (patches)	classify the root causes of issues at a finer granularity than prior work	having an ever-increasing pace through time. Furthermore, the total breadth and complexity of reports and patch files stay relatively consistent through the lifespan of HDFS.
Erase and Selangor **[12]**	Model-Driven Engineering (MADE)	the storage layer is very useful and is essential	Model-Driven Engineering (MDE) is used to offer universal Metamodeling for the storage layer of a Big Data system.
Hussain et al. **[15]**	introduce a job scheduling algorithm	Scheduling is more complicated since it is Required in significant data time optimization.	The approach has lowered the number of iterations while increasing time efficiency
Almansouri and Masmoudi **[14]**	Illustrated the main steps to setup Hadoop and MapReduce	Hadoop for extensive data analysis has provided critical information that may be used for analysis.	Hadoop employs MapReduce, in which the dataset is processed in the Mapping phase before being reduced in the Reducing phase.
Mahmoud et al. **[13]**	encryption /Decryption file by using AES and OTP algorithms	Because of enormous data, Cloud computing provides users with on-demand, reliable, flexible, and low-cost services	As the size of the encrypted file expanded by 20% from the initial file size, This ratio improved.
Liao et al. **[55]**	built-in block-based hierarchical index structures, like R-tree	to arrange data sets in one, two or more dimensions	increase query performance for commonly used query types (e.g., point Query, range query) on (HDFS).
Zhang et al. **[22]**	HDFS-based Distributed Cache System (HDCache).	Files loaded from HDFS are cached in shared memory and may be accessed immediately By a client library.	cache system can store files with a wide range in their sizes
Vijayakumari et al. **[56]**	Cloud computing, Google File System (GFS), and (HDFS)	Hadoop MapReduce is based on the Google MapReduce concept.	GFS is built to function in data centers to provide exceptionally high data throughputs, minimal latency, and the ability to withstand individual Server outages.
Wang et al. **[57]**	Sport.	which can substantially speed up uploading by employing a metadata mapping strategy	the remote block placement can boost the course of block
Hua et al. **[58]**	interaction-intensive files	The paper addresses the throughput degradation problem while reading interaction-intensive files and proposes an improved HDFS design,	HDFS throughput for interaction-intensive files rise by 300 %, with just a the little performance hit for big data set workloads.
Shahabinejad et al. **[59]**	locally repairable codes (LRCs), binary locally repairable codes (BLRC)	(LRCs) computational complexity reduction can be attractive. With regard to the immense size of modern energy-hungry HDFS	legend has lower complexity than most recent non-binary LRC desirable requirements in HDFS, such as storage overhead and reliability.
Krishna et al. **[60]**	Apache Hadoop project	The computation in HDFS is They were done at the nodes where the necessary data is stored.	For files more extensive than the default block size, HDFS operates admirably.
Shetty and	data security is to encrypt the data that	Data security will be a crucial consideration when storing	Hadoop security features include Kerberos

Manjaiah , **[61]**	is stored in Hadoop,	sensitive data on Hadoop.	And Transparent Data Encryption (TDE) for Hadoop and security in the Hadoop Distributed File System.
Dai et al. **[62]**	placement of data replicas	Another placement policy that approaches the sender of information from an entirely different angle	HDFS replica placement policy, capable of generating replica distributions that fulfill all HDFS replica placement standards
Qu et al. **[63]**	DRS, a dynamic replica strategy based on improved Markov model	DRS may dynamically increase or decrease the number of replicas when the Data becomes hot or cold.	DRS is effective, and it outperforms HDFS's static replica Method.
Zeebare e et al. **[1]**	Hadoop Distributed File System (HDFS) and Map Reduce (MR).	HDFS, analyze, process and manage extensive data and very easy and fast to access data on different servers in the clustered system	HDFS is used to process and compute The number of words in an extensive database. Hadoop out-performs alternative software used for The same goal.

5. RECOMMENDATIONS

Depending on the addressed trends toward producing efficient Hadoop file system by many previous works, it is recommended to take care of the following concepts when treating with this subject: the relations between Hadoop file system in one side with the overall cloud systems for excellent efficiency. Considering the effects of Fog computations. Going towards the dew computations that will provide broad scope for combining Hadoop techniques with the dew strength.

6. CONCLUSION

From the previous works addressed in this paper, and depending on the summarized comparison in section 4, it can be concluded that the earlier researchers focused on: the unstructured data sets are processed quickly using the programming paradigm and HDFS of Hadoop MapReduce. Also, Hadoop enables you to work with the MapReduce framework while masking the complexity in a public or private cloud to install, configure and operate computer modules. The users can build a cluster of commodity servers using Hadoop. Some researchers have developed MapReduce as a stand-alone, service-like platform, which can be adapted to fit the demands of cloud providers. It also enables consumers to gather and analyze data. Adding to that, the HDFS is a fast-changing technology for storing and managing extensive data. Finally, HDFS is master/slave design-based and more potent for read-intensive business intelligence systems databases.

Adding to the above conclusion, it can be shown clearly that previous researchers focussed on achieving the following outcomes and objectives: The feasibility of exploiting the idle computational storage in a large Cluster of Workstations. They are classifying the root causes of issues at a finer granularity than prior work. Scheduling is more complicated since it is required in significant data time optimization. Because of enormous data, Cloud computing provides users with on-demand, reliable, flexible, and low-cost services. Data security will be a crucial consideration when storing sensitive data on Hadoop.

COMPETING INTERESTS DISCLAIMER:

Authors have declared that no competing interests exist. The products used for this research are commonly and predominantly used in our research area and country. There is absolutely no conflict of interest between the authors and producers of the products because we do not intend to use these products as an avenue for litigation but for the advancement of knowledge. Also, the research was not funded by the producing company rather it was funded by personal efforts of the authors.

REFERENCES

[1] S. R. Zeebaree, H. M. Shukur, L. M. Haji, R. R. Zebari, K. Jacksi, and S. M. Abas, "Characteristics and analysis of hadoop distributed systems," *Technology Reports of Kansai University,* vol. 62, pp. 1555-1564, 2020.

[2] S. Wadkar, M. Siddalingaiah, and J. Venner, *Pro Apache Hadoop*: Springer, 2014.

[3] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM,* vol. 51, pp. 107-113, 2008.

[4] S. Maneas and B. Schroeder, "The evolution of the hadoop distributed file system," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2018, pp. 67-74.

[5] H. I. Dino, S. Zeebaree, O. M. Ahmad, H. M. Shukur, R. R. Zebari, and L. M. Haji, "Impact of load sharing on performance of distributed systems computations," *International Journal of Multidisciplinary Research and Publications (IJMRAP),* vol. 3, pp. 30-37, 2020.

[6] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony*, et al.*, "Hive: a warehousing solution over a map-reduce framework," *Proceedings of the VLDB Endowment,* vol. 2, pp. 1626-1629, 2009.

[7] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston*, et al.*, "Building a high-level dataflow system on top of Map-Reduce: the Pig experience," *Proceedings of the VLDB Endowment,* vol. 2, pp. 1414-1425, 2009.

[8] F. P. Junqueira and B. C. Reed, "The life and times of a zookeeper," in *Proceedings of the 28th ACM symposium on Principles of distributed computing*, 2009, pp. 4-4.

[9] J. Shafer, S. Rixner, and A. L. Cox, "The hadoop distributed filesystem: Balancing portability and performance," in *2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*, 2010, pp. 122-133.

[10] W. Tantisiriroj, S. Patil, and G. Gibson, "Data-intensive file systems for internet services: A rose by any other name," Technical Report CMUPDL-08-114, Parallel Data Laboratory, Carnegie Mellon …2008.

[11] K. McKusick and S. Quinlan, "GFS: evolution on fast-forward," *Communications of the ACM,* vol. 53, pp. 42-49, 2010.

[12] A. Erraissi and A. Belangour, "Capturing hadoop storage big data layer meta-concepts," in *International Conference on Advanced Intelligent Systems for Sustainable Development*, 2018, pp. 413-421.

[13] H. Mahmoud, A. Hegazy, and M. H. Khafagy, "An approach for big data security based on Hadoop distributed file system," in *2018 International Conference on Innovative Trends in Computer Engineering (ITCE)*, 2018, pp. 109-114.

[14] H. T. Almansouri and Y. Masmoudi, "Hadoop Distributed File System for Big data analysis," in *2019 4th World Conference on Complex Systems (WCCS)*, 2019, pp. 1-5.

[15] R. Hussain, M. Rahman, K. I. Masud, S. M. Roky, M. N. Akhtar, and T. A. Tarin, "A Novel Approach of Fair Scheduling to Enhance Performance of

Hadoop Distributed File System," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1-6.
[16] F. Q. Kareem, S. R. Zeebaree, H. I. Dino, M. A. Sadeeq, Z. N. Rashid, D. A. Hasan*, et al.*, "A survey of optical fiber communications: challenges and processing time influences," *Asian Journal of Research in Computer Science,* pp. 48-58, 2021.
[17] Z. Ageed, M. R. Mahmood, M. Sadeeq, M. B. Abdulrazzaq, and H. Dino, "Cloud computing resources impacts on heavy-load parallel processing approaches," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 22, pp. 30-41, 2020.
[18] M. Seeger and S. Ultra-Large-Sites, "Key-Value stores: a practical overview," *Computer Science and Media, Stuttgart,* 2009.
[19] I. Robinson, J. Webber, and E. Eifrem, *Graph databases: new opportunities for connected data*: " O'Reilly Media, Inc.", 2015.
[20] D. Abadi, P. Boncz, S. H. Amiato, S. Idreos, and S. Madden, *The design and implementation of modern column-oriented database systems*: Now Hanover, Mass., 2013.
[21] A. Issa and F. Schiltz, "Document oriented databases," ed: Universite Libre de Bruxelles Retrieved from http://cs. ulb. ac. be/public …, 2015.
[22] J. Zhang, G. Wu, X. Hu, and X. Wu, "A distributed cache for hadoop distributed file system in real-time cloud services," in *2012 ACM/IEEE 13th International Conference on Grid Computing*, 2012, pp. 12-21.
[23] Z. S. Ageed, S. R. Zeebaree, M. M. Sadeeq, S. F. Kak, H. S. Yahia, M. R. Mahmood*, et al.*, "Comprehensive survey of big data mining approaches in cloud systems," *Qubahan Academic Journal,* vol. 1, pp. 29-38, 2021.
[24] S. Chandrasekar, R. Dakshinamurthy, P. Seshakumar, B. Prabavathy, and C. Babu, "A novel indexing scheme for efficient handling of small files in hadoop distributed file system," in *2013 International Conference on Computer Communication and Informatics*, 2013, pp. 1-8.
[25] H. R. Abdulqadir, S. R. Zeebaree, H. M. Shukur, M. M. Sadeeq, B. W. Salim, A. A. Salih*, et al.*, "A study of moving from cloud computing to fog computing," *Qubahan Academic Journal,* vol. 1, pp. 60-70, 2021.
[26] M. Afzali, N. Singh, and S. Kumar, "Hadoop-MapReduce: A platform for mining large datasets," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 1856-1860.
[27] U. R. Pol, "Big data analysis using Hadoop MapReduce," *Am. J. Eng. Res. AJER,* vol. 5, pp. 146-151, 2016.
[28] I. M. Ibrahim, "Task scheduling algorithms in cloud computing: A review," *Turkish Journal of Computer and Mathematics Education (TURCOMAT),* vol. 12, pp. 1041-1053, 2021.
[29] F. E. F. Samann, S. R. Zeebaree, and S. Askar, "IoT provisioning QoS based on cloud and fog computing," *Journal of Applied Science and Technology Trends,* vol. 2, pp. 29-40, 2021.
[30] Z. S. Ageed, R. K. Ibrahim, and M. Sadeeq, "Unified ontology implementation of cloud computing for distributed systems," *Current Journal of Applied Science and Technology,* pp. 82-97, 2020.

[31] H. Shukur, S. Zeebaree, R. Zebari, O. Ahmed, L. Haji, and D. Abdulqader, "Cache coherence protocols in distributed systems," *Journal of Applied Science and Technology Trends,* vol. 1, pp. 92-97, 2020.

[32] L. M. Haji, S. Zeebaree, O. M. Ahmed, A. B. Sallow, K. Jacksi, and R. R. Zeabri, "Dynamic resource allocation for distributed systems and cloud computing," *TEST Engineering & Management,* vol. 83, pp. 22417-22426, 2020.

[33] M. M. Sadeeq, N. M. Abdulkareem, S. R. Zeebaree, D. M. Ahmed, A. S. Sami, and R. R. Zebari, "IoT and Cloud computing issues, challenges and opportunities: A review," *Qubahan Academic Journal,* vol. 1, pp. 1-7, 2021.

[34] H. Shukur, S. R. Zeebaree, A. J. Ahmed, R. R. Zebari, O. Ahmed, B. S. A. Tahir*, et al.*, "A state of art survey for concurrent computation and clustering of parallel computing for distributed systems," *Journal of Applied Science and Technology Trends,* vol. 1, pp. 148-154, 2020.

[35] O. Alzakholi, H. Shukur, R. Zebari, S. Abas, and M. Sadeeq, "Comparison among cloud technologies and cloud performance," *Journal of Applied Science and Technology Trends,* vol. 1, pp. 40-47, 2020.

[36] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, 2010, pp. 1-10.

[37] M. Maurya and S. Mahajan, "Performance analysis of MapReduce programs on Hadoop cluster," in *2012 World Congress on Information and Communication Technologies*, 2012, pp. 505-510.

[38] H. S. Yahia, S. R. Zeebaree, M. A. Sadeeq, N. O. Salim, S. F. Kak, A.-Z. Adel*, et al.*, "Comprehensive survey for cloud computing based nature-inspired algorithms optimization scheduling," *Asian Journal of Research in Computer Science,* pp. 1-16, 2021.

[39] S. R. Zeebaree, "Remote controlling distributed parallel computing system over the cloud (RCDPCSC)," in *2020 3rd International Conference on Engineering Technology and its Applications (IICETA)*, 2020, pp. 258-258.

[40] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden*, et al.*, "A comparison of approaches to large-scale data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 2009, pp. 165-178.

[41] S. H. Haji, S. R. Zeebaree, R. H. Saeed, S. Y. Ameen, H. M. Shukur, N. Omar*, et al.*, "Comparison of software defined networking with traditional networking," *Asian Journal of Research in Computer Science,* pp. 1-18, 2021.

[42] H. Shukur, S. Zeebaree, R. Zebari, D. Zeebaree, O. Ahmed, and A. Salih, "Cloud computing virtualization of resources allocation for distributed systems," *Journal of Applied Science and Technology Trends,* vol. 1, pp. 98-105, 2020.

[43] K. Shvachko, "Name-node memory size estimates and optimization proposal," *Apache Hadoop Common Issues, HADOOP-1687,* 2007.

[44] X. Liu, J. Han, Y. Zhong, C. Han, and X. He, "Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS," in *2009 IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1-8.

[45] C. Vorapongkitipun and N. Nupairoj, "Improving performance of small-file accessing in Hadoop," in *2014 11th international joint conference on computer science and software engineering (JCSSE)*, 2014, pp. 200-205.
[46] Y. S. Jghef and S. Zeebaree, "State of art survey for significant relations between cloud computing and distributed computing," *International Journal of Science and Business,* vol. 4, pp. 53-61, 2020.
[47] M. A. Ahad and R. Biswas, "Handling small size files in hadoop: Challenges, opportunities, and review," *Soft computing in data analytics,* pp. 653-663, 2019.
[48] M. R. Ghazi and D. Gangodkar, "Hadoop, MapReduce and HDFS: a developers perspective," *Procedia Computer Science,* vol. 48, pp. 45-50, 2015.
[49] L. M. Haji, O. M. Ahmad, S. Zeebaree, H. I. Dino, R. R. Zebari, and H. M. Shukur, "Impact of cloud computing and internet of things on the future internet," *Technology Reports of Kansai University,* vol. 62, pp. 2179-2190, 2020.
[50] H. J. Watson, "Tutorial: Big data analytics: Concepts, technologies, and applications," *Communications of the Association for Information Systems,* vol. 34, p. 65, 2014.
[51] B. R. Ibrahim, S. R. Zeebaree, and B. K. Hussan, "Performance Measurement for Distributed Systems using 2TA and 3TA based on OPNET Principles," *Science Journal of University of Zakho,* vol. 7, pp. 65-69, 2019.
[52] Z. N. Rashid, S. R. Zebari, K. H. Sharif, and K. Jacksi, "Distributed cloud computing and distributed parallel computing: A review," in *2018 International Conference on Advanced Science and Engineering (ICOASE)*, 2018, pp. 167-172.
[53] L. Aguilera-Mendoza and M. T. Llorente-Quesada, "Modeling and simulation of Hadoop Distributed File System in a cluster of workstations," in *International Conference on Model and Data Engineering*, 2013, pp. 1-12.
[54] J. Kim, T. A. Kumar, K. George, and N. Park, "Performance evaluation and tuning for MapReduce computing in Hadoop distributed file system," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, 2015, pp. 62-68.
[55] H. Liao, J. Han, and J. Fang, "Multi-dimensional index on hadoop distributed file system," in *2010 IEEE Fifth International Conference on Networking, Architecture, and Storage*, 2010, pp. 240-249.
[56] R. Vijayakumari, R. Kirankumar, and K. G. Rao, "Comparative analysis of google file system and hadoop distributed file system," *International Journal of Advanced Trends in Computer Science and Engineering,* vol. 3, pp. 553-558, 2014.
[57] Y. Wang, W. Wang, C. Ma, and D. Meng, "Zput: A speedy data uploading approach for the hadoop distributed file system," in *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, 2013, pp. 1-5.
[58] X. Hua, H. Wu, Z. Li, and S. Ren, "Enhancing throughput of the Hadoop Distributed File System for interaction-intensive tasks," *Journal of Parallel and Distributed Computing,* vol. 74, pp. 2770-2779, 2014.

[59] M. Shahabinejad, M. Khabbazian, and M. Ardakani, "An efficient binary locally repairable code for hadoop distributed file system," *IEEE communications letters,* vol. 18, pp. 1287-1290, 2014.

[60] T. L. S. R. Krishna, T. Ragunathan, and S. K. Battula, "Performance evaluation of read and write operations in hadoop distributed file system," in *2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming*, 2014, pp. 110-113.

[61] M. M. Shetty and D. Manjaiah, "Data security in Hadoop distributed file system," in *2016 International Conference on Emerging Technological Trends (ICETT)*, 2016, pp. 1-5.

[62] W. Dai, I. Ibrahim, and M. Bassiouni, "A new replica placement policy for hadoop distributed file system," in *2016 IEEE 2nd international conference on big data security on cloud (bigdatasecurity), IEEE international conference on high performance and smart computing (HPSC), and IEEE international conference on intelligent data and security (IDS)*, 2016, pp. 262-267.

[63] K. Qu, L. Meng, and Y. Yang, "A dynamic replica strategy based on Markov model for hadoop distributed file system (HDFS)," in *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)*, 2016, pp. 337-342.

[64] V. Sajwan, V. Yadav, and M. Haider, "The Hadoop Distributed File System: Architecture and Internals," *International Journal of Combined Research & Development (IJCRD),* vol. 4, pp. 541-544, 2015.