Big Data Testing Using Hadoop Platform

Tushar Kumar Sharma[1], Chirag Jindal[2], Akhil Saini[3], Satyam Gupta[4]

Computer Science and Engineering, Chandigarh University, Mohali India

---***---

Abstract — Big data analysis has emerged as a crucial technology in recent years due to the exponential growth of data generated from various sources. This data can come in structured, unstructured, or semi-structured formats and is generated from diverse channels such as social media platforms, smart city sensors, ecommerce websites, and numerous applications. These vast amounts of data encompass a wide range of formats, including text, images, audios, and videos. Hadoop provides a comprehensive ecosystem of tools and frameworks that enable efficient storage and processing of big data. One of the key components of Hadoop is the Hadoop Distributed File System (HDFS), which is designed to store and manage data across a cluster of commodity hardware. HDFS breaks down large files into smaller blocks and replicates them across multiple nodes, ensuring data reliability and availability. MapReduce allows for parallel computation of data across a cluster of machines, making it suitable for processing large-scale datasets. By breaking down complex tasks into smaller subtasks and distributing them across multiple nodes, MapReduce enables faster and more efficient data processing.As big data analysis continues to evolve, Hadoop has expanded its ecosystem with various technologies to enhance its capabilities. These include YARN (Yet Another Resource Negotiator), which serves as the cluster resource management framework, enabling efficient allocation of computing resources for different applications. Pig, an abstraction layer on top of Hadoop, provides a high-level language called Pig Latin for expressing data analysis tasks. MRjob is a Python framework that simplifies the development of MapReduce jobs. Zookeeper is a centralized service for maintaining configuration information, synchronization, and naming services. Hive offers a data warehouse infrastructure and a query language called HiveQL for querying and analyzing data stored in Hadoop. Apache Spark, a fast and general-purpose data processing engine, is integrated with Hadoop to provide faster in-memory computation capabilitiesIn this paper, our focus is on exploring big data analysis and demonstrating how Hadoop, along with its associated technologies, can be used for analyzing, storing, and processing large volumes of data. By leveraging the power of Hadoop's distributed architecture and the complementary tools within its ecosystem, organizations can effectively harness the potential of big data and derive valuable insights for various applications and industries.

Keywords— Big Data, Hadoop, HDFS (Hadoop Distributed File System), MapReduce, Software Testing, Yarn, Pig, MRjob, Zookeeper, Hive, Apache Spark, Hive, HBase

I. INTRODUCTION

Recently, information technology systems have been playing a major role in handling and giving insights to organization's business. This information can come from education, traffic, healthcare, or commerce sectors. This data can be structured or semi-structured and can be in the form of text, images, audio, video, and log files. The amount of data to be stored and processed is often in terabytes.[15] It becomes extremely hard for a single system to manage such big data.

3 V's, on the basis of which Big Data is defined includes– volume, velocity and variety.

<u>Volume</u> – The size of data from organizations in social media, healthcare, education, and business section comes is ever increasing. Big companies like Google and Facebook process information in petabytes. The IOT (sensors data) is also increasing day by day. So, it becomes difficult to manage such big data using traditions systems.

<u>Variety</u> – data in today's world is divided into structured (schema, columns), semi-structured (json, emails, xml) and unstructured (images, videos, audio) categories. This data can also be raw and requires heavy system work to convert it into useful information using traditional analytical systems.

<u>Velocity</u> – this concept defines the speed at which the data arrives from source destination and the speed with which it is processed. The size of the incoming data is huge andit needs to be processed at a similar speed.

Companies like Google, Facebook process petabytes of data on a daily basis.[23] There is certain software that come in use to operate and manage this big data. This paper focuses one such software called Hadoop and how it is used in big data testing. The paper first describes what is big data and then moves to technologies like Hadoop and its components –HDFS and MapReduce. There are several budding technologies which come into play like YARN, Pig, Spark, Zookeeper, Hive, Apache Spark and HBase. Yarn helps in enabling the processing and running of batch, stream, interactive, and graph data stored in HDFS.[27] Hive and Pig are two integral parts of the Hadoop ecosystem, but there are some key differences between them.

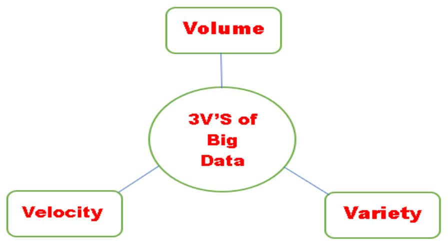


Fig 1: 3V's of Big Data

Spark is an free to use framework which is dedicated for providingto-way communicative questioning, machine learning, and live workloads. Apache Zookeeper is a service that provides operational support to Hadoop clusters, offering a distributed configuration service, a synchronization tool, and a naming registry.[7] Finally, Apache HBase is a free-to-use, NoSQL, distributed, warehouse for big data that provides consistent live access to zetabytes of data. It is effective for managing large sparse datasets. [5] The paper will focus on practical usage of these technologies and compare them performance wise.

However, with growing number of business storage this technology has also become quite vulnerable to attacks. The paper will focus on describing challenges, security measures and vulnerability of this architectureand how to tackle them.

II. CHALLENGES ASSOCIATED WITH BIG DATA

With the humongous growth in the big data technology, there is massive rush of unstructured data in various forms and size. It becomes quite difficult to handle such data. Hence, many of the existing problems related to big data such as analysis methods and applications, storage models, privacy and security are discussed below.

A. Analysis methods and Applications

It is important to analyze big data to gain further meaningful and useful insights. To handle a complex, inconsistent and unstructured data, an accurate method is required. To work with such an inconsistent and complicated data, poses another problem for the data scientists. There are methods such as inquisitive, predictive, prescriptive, and pre-emptive analytics but the skills, tools and knowledge required to work on them are limited.[21]

some of the challenges in dealing with big data are live or on-the-spot data transfer, situation discovery, research, insight gaining and responses. Their application require a more than sufficient knowledge to apply an efficient approach. Conventional applications are limited by processing limitations, high computational processing, and memory constraints.

B. Storage models

Millions of new IOT devices are being linked to the internet every day. Social media apps like Instagram, Facebook are an explosive contributor to the big data. Following this burst in data, many companies switch from traditional data storage systems towards cloud storage.[9] It is still a challenge to upload to cloud storage in real time because the data is huge. During the real time uploading of IOT devices associated with floods and debris flow management to the cloud, the sensors may be sending big chunks of data which can cause timely processing and processing computationally challenging.

C. Privacy and data security

Despite the rise in popularity of big data analytics, the problem of how to deal with a lot of data while maintaining privacy is still unresolved. Data are decentralized and originate from a variety of sources, including sensors, mobile devices, and Internet of Things (IoT) devices, which explains why. A privacy and security issue has also arisen from the analysis of heterogeneous data sources as a result of contact with other external systems.[9] It is also essential to ensure that the source is not subject to any attacks. Illegal transfer of backup data, for instance, becomes a major worry in big data research for the healthcare industry. To stop a new breach of patient security and privacy, this situation must be fully rectified. This event forces big data to reconsider analytics and developer privacy.

III. RELATED WORK

A. Hadoop and budding tools

Hadoop, developed by the Apache Software Foundation, is a distributed computing framework that enables unrestricted access to large-scale data processing. It is widely used by global corporations such as eBay, Facebook, and Yahoo. Hadoop comprises several components, including HDFS for user interface and storage, MapReduce for data processing, and YARN for system management. Initially created in 2005 to support Yahoo's Nutch search engine project, Hadoop was published in 2011 and can scale up to 4000 nodes per cluster, enabling fast processing of vast amounts of data.[13] To enhance security and administration, vendor-specific Hadoop distributions have been developed. The key components of Hadoop include the Name Node, Data Node, Job Tracker, and Task Tracker. However, Hadoop's lack of internal cluster security makes it susceptible to threats.

B. Hadoop Distributed File System (HDFS)

HDFS is designed to efficiently handle large-scale data distributed across clusters of commodity computers. This data can originate from various sources such as IoT sensors, social media platforms, and e-commerce servers. HDFS breaks down big data into smaller blocks, typically weighing 128MB, and distributes multiple copies of each block across the cluster for fault tolerance. If a machine fails, the data can be retrieved from other machines.

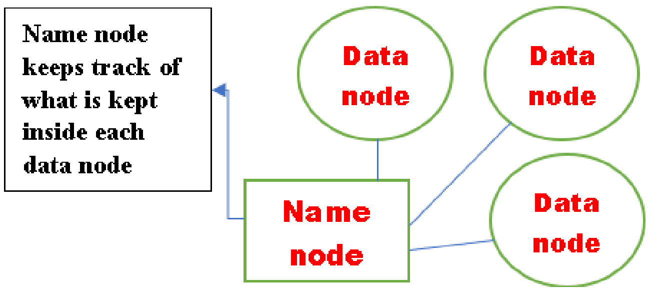


Fig 2: HFDS Nodes

The architecture of HDFS consists of three main components: the Name Node, Client Node, and Data Nodes. The Name Node keeps track of the files present on the Data Nodes and their locations within the cluster. However, the Name Node itself does not store data. The Client Node is responsible for reading and writing files in HDFS.[34] When reading a file, the Client Node requests the file's location from the Name Node, which provides the information needed to access the file from the Data Nodes. When writing a file, the Client Node informs the Name Node to create a new file, and the data is then sent to the Data Nodes, which communicate with each other to create backups of the file. Various methods can be used to interact with HDFS, including command-line interface, user interface (such as Ambari), Java interface, and HTTP/HDFS proxies. The command-line interface offers several commands for working with HDFS.

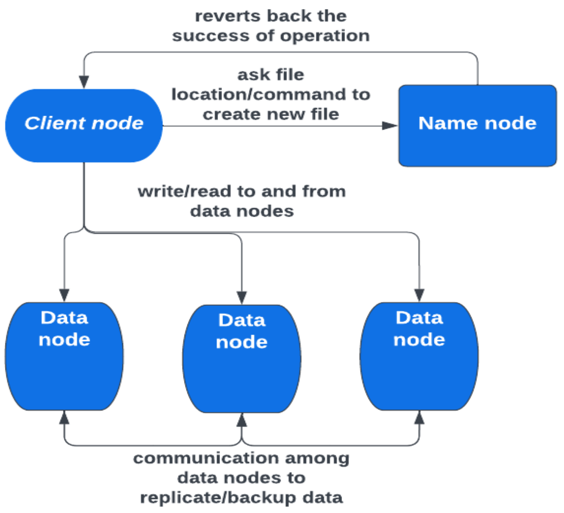

reverts back the success of operation
ask file location/command to create new file
Client node
Name node
write/read to and from data nodes
Data node
Data node
Data node
communication among data nodes to replicate/backup data

Fig 3: Functions of different nodes in HDFS

C. MapReduce

MapReduce is a parallel programming paradigm and processing approach for building distributed applications that efficiently handle large volumes of data on clusters of shared hardware.[35] It was originally developed at Google and is powered by Hadoop, an open-source platform developed by Apache, primarily based on Java. The MapReduce algorithm consists of two key elements: Map and Reduce. Using tuples to represent each element, the Map operation transforms one collection of data into another, while the Reduce operation combines the output of the Map into a smaller set of tuples. The MapReduce paradigm allows easy scaling of data processing across multiple CPU nodes. Mappers and reducers are the basic components used in the MapReduce model for data processing. A MapReduce application can be easily scaled by updating the configuration, allowing it to run on hundreds or even thousands of servers in a cluster.

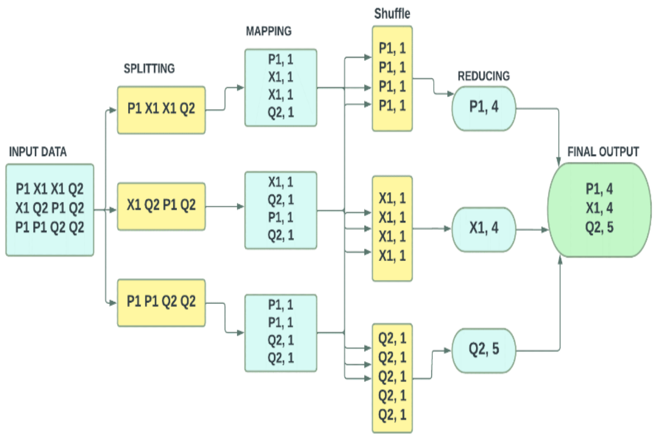


Fig 4: MapReduce flow chart

The MapReduce program's execution consists of two stages: the Map stage and the Reduce stage.[28] The Map stage processes the input data, typically stored in Hadoop Distributed File System (HDFS), and passes it character by character to the mapper function, producing intermediate data. The Reduce stage combines the intermediate outputs and produces a final output, which is then stored back in HDFS. MapReduce follows a master-slave paradigm, where the master (JobTracker) divides the problem into jobs and sends them to various Data Nodes (TaskTracker) for parallel processing. The map jobs often run on the same cluster nodes as the processed data, and if a node is overloaded, another node close to the data is selected. The intermediate results of the Map and Shuffle phases are combined to reduce the amount of data transferred between map tasks and reduce tasks. While the input data and the final output are stored in HDFS, the intermediate results are stored in the local file systems of the Data Nodes.

The JobTracker is responsible for monitoring the Data Nodes, managing the TaskTrackers, and ensuring the completion of jobs. If a task fails or a Data Node becomes unresponsive, the JobTracker restarts the job on another server, preferably one with a copy of the data. Additionally, if a task is running slowly, the JobTracker can initiate speculative execution by starting the same task on a different server to complete the job within the allocated time.

D. MrJob

MrJob is a Python framework that simplifies the development and execution of MapReduce tasks on Hadoop servers. It provides a high-level API for creating MapReduce jobs and abstracts the complexities of Hadoop. Some key features and benefits of MrJob include:

i. Compatibility with various Hadoop versions, including Amazon Elastic MapReduce (EMR), local computers, and Docker containers.
ii. High-level API that eliminates the need for dealing with low-level Hadoop details.
iii. Built-in test framework for testing MapReduce tasks before deploying them to a Hadoop cluster.

To create and run a job using MrJob, one can use commands such as creating a new job, running a job locally or on a Hadoop cluster, and testing a job locally.[19]

Creating a new job: python mrjob create-job my_job

Running a job locally: python my_job.py -r localinput_file

Running a job on a Hadoop cluster: python my_job.py-r hadoop input_file

Running a job on Amazon Elastic MapReduce: python my_job.py -r emr s3://input_bucket/input_file

Testing a job locally: python my_job_test.py

E. YARN

YARN, which stands for Yet Another Resource Negotiator, is a cluster management technology used in Hadoop for resource allocation and application scheduling. It offers the following characteristics and advantages:

i. Resource management: YARN enables efficient sharing of cluster resources among multiple applications.
ii. Centralized scheduler: YARN includes a centralized scheduler that controls resource allocation and schedules applications on the cluster.
iii. Fault tolerance: YARN incorporates fault tolerance mechanisms to ensure uninterrupted application execution even in the presence of node failures.
iv. Scalability: YARN is designed to handle large clusters with thousands of nodes and millions of tasks.

To interact with YARN, various commands can be used, such as starting the ResourceManager, starting the NodeManager, submitting MapReduce or Spark jobs, and monitoring application progress.

petabytes of data in awidely distributed way.

Start ResourceManager: yarn resourcemanager

Start NodeManager: yarn nodemanager

Submit MapReduce job: yarn jar mapreduce.jar MyJob input output

Submit Spark job: spark-submit --master yarn -- deploy-mode cluster myapp.py input output

Monitor application progress: yarn application - status <application_id>

F. Pig

Pig is a framework for analyzing large datasets using Pig Latin, a high-level scripting language. Pig Latin provides a simplified syntax for expressing data analysis tasks, allowing users to focus on the analysis logic rather than low-level implementation details.[9] Pig handles data storage and processing complexities, making data handling easier. It offers a scripting interface for conducting data analysis jobs and is designed to operate on large clusters and handle petabytes of data in a distributed manner. Some common Pig commands include loading data from a file, filtering data, grouping and aggregating data, and writing data to a file.

Start Pig shell: pig

Load from a file: mydata = LOAD 'mydata.txt' USING PigStorage(',') AS (col1:int, col2:chararray,col3:double);

Filter data: mydata_filtered = FILTER mydata BY col1 > 100;

Group & aggregate data: mydata_grouped = GROUP mydata BY col2; mydata_aggregated = FOREACH mydata_grouped GENERATE group,AVG(mydata.col3);

Write to a file: STORE mydata_aggregated INTO 'output' USING PigStorage();

G. Hive

Hive is a data storage utility that provides a SQL-like interface for querying and analyzing large Hadoop datasets. Hive translates SQL queries into MapReduce tasks, enabling developers and analysts to interact with Hadoop using familiar SQL syntax. Key characteristics and benefits of Hive include:

i. SQL-like interface: Hive offers a user-friendly SQL-like interface that simplifies the creation and execution of queries on large databases.
ii. Scalability: Hive is designed to handle big clusters and process petabytes of data in a distributed and parallel manner.
iii. Data warehousing features: Hive supports features like table creation and partitioning for effective data warehousing.

To use Hive, one can start the Hive shell, create tables, load data into tables, and run queries using SQL syntax.

Starting the Hive shell: hive

Creating a table: CREATE TABLE mytable (col1 INT, col2 STRING, col3 DOUBLE);

Loading data into a table: LOAD DATA LOCAL INPATH 'data.txt' INTO TABLE mytable;

Running a query: SELECT col1, AVG(col3) FROM mytable GROUP BY col1;

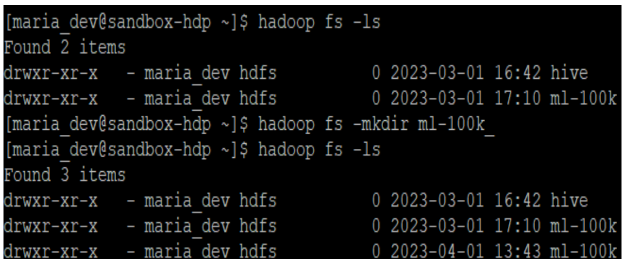
Overall, these tools and technologies, such as Hadoop, HDFS, MapReduce, MrJob, YARN, Pig, and Hive, provide powerful capabilities for distributed data processing and analysis, making it easier to handle large datasets.

IV. IMPLEMENTATION OF BIG DATA PROCESSING

This paper runs a MapReduce job using sandbox environment to better understand the usage of Hadoop HDFS and MapReduce commands. A sample movie dataset is used to apply various functions and commands on. With the help of Ambari (Hadoop user interface) bash command line we type the following HDFS commands:

i. Hadoop fs -ls (lists files in the cluster)

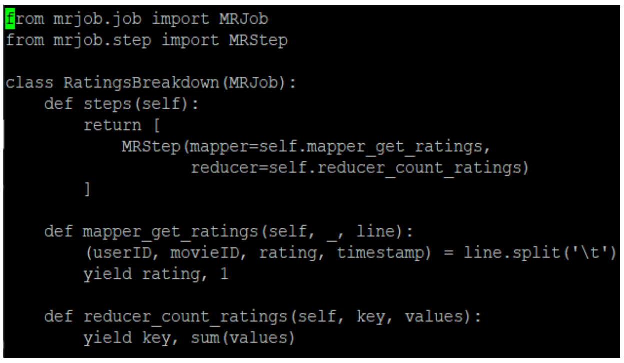
ii. Hadoop fs -mkdir ml-100k(makes a new folder named ml-100k)



```
[maria_dev@sandbox-hdp ~]$ hadoop fs -ls
Found 2 items
drwxr-xr-x   - maria_dev hdfs          0 2023-03-01 16:42 hive
drwxr-xr-x   - maria_dev hdfs          0 2023-03-01 17:10 ml-100k
[maria_dev@sandbox-hdp ~]$ hadoop fs -mkdir ml-100k_
[maria_dev@sandbox-hdp ~]$ hadoop fs -ls
Found 3 items
drwxr-xr-x   - maria_dev hdfs          0 2023-03-01 16:42 hive
drwxr-xr-x   - maria_dev hdfs          0 2023-03-01 17:10 ml-100k
drwxr-xr-x   - maria_dev hdfs          0 2023-04-01 13:43 ml-100k
```

Fig 5: MrJob commands in Ambari Bash Shell

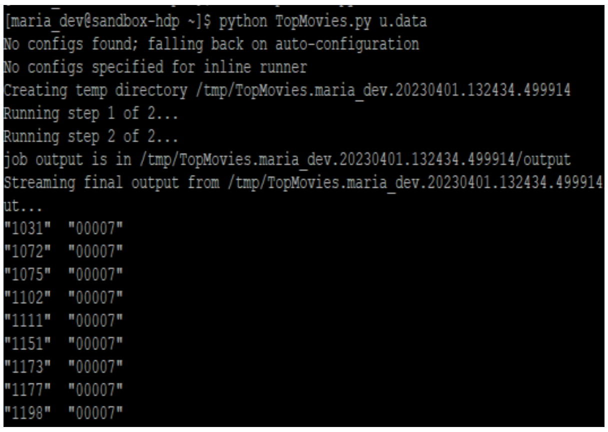
With the help of nano text editor we input the functions for sorting and adding the movie ratings according to the user id. This task is completed with the help of a python library called MRjob. Below are the functions of MRjob:



```
from mrjob.job import MRJob
from mrjob.step import MRStep

class RatingsBreakdown(MRJob):
    def steps(self):
        return [
            MRStep(mapper=self.mapper_get_ratings,
                   reducer=self.reducer_count_ratings)
        ]

    def mapper_get_ratings(self, _, line):
        (userID, movieID, rating, timestamp) = line.split('\t')
        yield rating, 1

    def reducer_count_ratings(self, key, values):
        yield key, sum(values)
```

Fig 6: MrJob functions in nano-text editor

Once the functions are defined in the editor we make sure to save the python file.[12] Then we run the MRjob python file on the movie dataset only to view the required results:



```
[maria_dev@sandbox-hdp ~]$ python TopMovies.py u.data
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/TopMovies.maria_dev.20230401.132434.499914
Running step 1 of 2...
Running step 2 of 2...
job output is in /tmp/TopMovies.maria_dev.20230401.132434.499914/output
Streaming final output from /tmp/TopMovies.maria_dev.20230401.132434.499914
ut...
"1031"  "00007"
"1072"  "00007"
"1075"  "00007"
"1102"  "00007"
"1111"  "00007"
"1151"  "00007"
"1173"  "00007"
"1177"  "00007"
"1198"  "00007"
```

Fig 7: Required Result

After running the Python commands we get the MapReduced result from the dataset which contains the sorted (ascending order) ratings of the movies (right) according to the user ids(left).

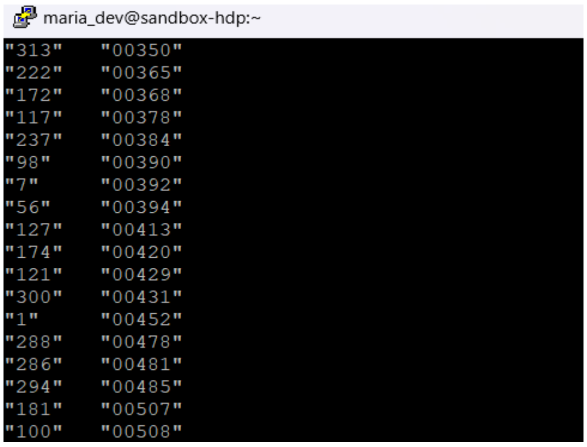

```
maria_dev@sandbox-hdp:~
"313"   "00350"
"222"   "00365"
"172"   "00368"
"117"   "00378"
"237"   "00384"
"98"    "00390"
"7"     "00392"
"56"    "00394"
"127"   "00413"
"174"   "00420"
"121"   "00429"
"300"   "00431"
"1"     "00452"
"288"   "00478"
"286"   "00481"
"294"   "00485"
"181"   "00507"
"100"   "00508"
```

Fig 8: Required Result

V. CONLUSION AND FUTURE WORK

Big Data is a type of data whose size, variety, and complexity necessitate the development of new design, methods, algorithms, and analytics in order to handle it and derive value and secret knowledge from it. Data is now produced from a variety of sources and can enter the system at varying speeds.[4] Today, processing huge amounts of data is a major problem. We covered the Hadoop tool for Big Data in depth in this article. Hadoop is the foundational framework for structuring Big Data and solving the issue of making it usable for analytics. We also talked about some Hadoop components that help with the handling of big data sets in dispersed computing settings. In the future, we can apply some clustering methods and test their performance.

Hadoop offers several advantages for big data testing, including scalability, failure tolerance, and support for different data types. However, due to the platform's complexity and the diversity of tools accessible, testing big data on Hadoop can be difficult. To guarantee the accuracy and dependability of big data processed with Hadoop, it is critical to have a comprehensive grasp of its architecture, testing methods, and tools.

As the demand for big data analytics grows rapidly across various sectors, the potential scope of big data testing using Hadoop is important. As more businesses implement big data technologies, the need for efficient testing strategies and tools grows.[17] Because of its scalability, fault tolerance, and support for different data types, Hadoop has become a common tool for big data handling and analysis. Hadoop is anticipated to provide more sophisticated testing methods and instruments to guarantee the accuracy and dependability of big data as it evolves.

REFERENCES

[1] Bhosale, H. S., Gadekar, P. D. (2014). "A Review Paper on Big Data and Hadoop." International Journal of Scientific and Research Publications, 4(10).

[2] Smitha, T., Kumar, V. S. (2013). "Application of Big Data in Data Mining." International Journal of Emerging Technology and Advanced Engineering, 3(7).

[3] IBM Big Data Analytics HUB. (n.d.). "Four V's of Big Data." Retrieved from www.ibmbigdatahub.com/infographic/four-vs-big-data.

[4] Mridul, M., Khajuria, A., Dutta, S., Kumar, N. (2014). "Analysis of Big Data using Apache Hadoop and MapReduce." International Journal of Advance Research in Computer Science and Software Engineering, 4(5).

[5] Apache Hadoop Project. (n.d.). "Apache Hadoop." Retrieved from http://hadoop.apache.org/.

[6] Smitha, T., Sundaram, V. (2012). "Classification Rules by Decision Tree for Disease Prediction." International Journal for Computer Applications, 43(8).

[7] Baker, N. (2017). "3 Universities That Are Using Big Data - QS." Retrieved from https://www.qs.com/3-universities-thatare-using-big-data/?fbclid=IwAR2eH_8StopgXHIlnGn5IQvnwIKmDQVM_xTWZc5oq6FDp4_vwNkUzG4xZOQ.

[8] Information Technology Research and Development Centre, University of Kufa. (2018). "E-learning server statistics." Retrieved from http://elearning.uokufa.edu.iq/?page_id=100.

[9] Parsola, J., et al. (2018). "Post Event Investigation of Multi-stream Video Data Utilizing Hadoop Cluster." International Journal of Electrical and Computer Engineering, 8, 5089.

[10] Apache Hadoop. (2018). "Hadoop – Apache Hadoop 2.9.2." Retrieved from https://hadoop.apache.org/docs/r2.9.2/.

[11] Tom, W. (2015). "Hadoop: The Definitive Guide." Fourth Edition The Definitive Guide Storage and Analysis at Internet Scale.

[12] Insights SAS. (n.d.). "What is Hadoop?" Retrieved from https://www.sas.com/en_us/insights/big-data/hadoop.html.

[13] Yang, H. C., Dasdan, A., Hsiao, R. L., Parker, D. S. (2007). "Map–reduce–merge: Simplified relational data processing on large clusters." Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data.

[14] Hunt, P., Konar, M., Junqueira, F. P., Reed, B. (2010). "ZooKeeper: Wait-free coordination for internet-scale systems." Proceedings of the USENIX Annual Technical Conference.

[15] Junqueira, F., Reed, B. (2013). "ZooKeeper: Distributed Process Coordination." O'Reilly Media, Inc.

[16] Ranjan, R. (2014). "Streaming big data processing in datacenter clouds." IEEE Cloud Computing, 1(1), 78-83.

[17] Bahga, A., Madisetti, V. (2014). "Internet of Things: A Hands-On Approach." Vpt.

[18] Meng, X., et al. (2016). "MLlib: Machine learning in Apache Spark." Journal of Machine Learning Research, 17(1), 1235-1241.

[19] Gonzalez, J. E., et al. (2014). "Graphx: Graph processing in a distributed dataflow framework." Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation.

[20] Kulkarni, S., et al. (2015). "Twitter Heron: Stream processing at scale." Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data.

[21] Aji, A., et al. (2013). "Hadoop GIS: A high-performance spatial data warehousing system over MapReduce." Proceedings of the VLDB Endowment, 6(11), 1009-1020.

[22] Apache Hadoop. (2016). Retrieved from http://hadoop.apache.org/.

[23] Apache Hive. (2016). Retrieved from https://hive.apache.org/.

[24] Apache Hadoop HDFS. (2016). Retrieved from http://hadoop.apache.org/hdfs.

[25] Dean, J., Ghemawat, S. (2008). "MapReduce: Simplified data processing on large clusters." Communications of the ACM, 51(1), 107-113.

[26] White, T. (2012). "Hadoop: The Definitive Guide." O'Reilly Media, Inc.

[27] Zaharia, M., et al. (2010). "Spark: Cluster computing with working sets." HotCloud, 10(10-10), 95.

[28] Akbarinia, R., Pournaras, E., Aberer, K. (2013). "Dynamic load balancing in distributed stream processing systems." IEEE Transactions on Parallel and Distributed Systems, 24(7), 1362-1371.

[29] Zaharia, M., et al. (2016). "Apache Spark: A unified engine for big data processing." Communications of the ACM, 59(11), 56-65.

[30] Kshemkalyani, A. D., Singhal, M. (2010). "Distributed Computing: Principles, Algorithms, and Systems." Cambridge University Press.

[31] Abadi, D. J., Chu, A., Eksombatchai, P. (2013). "The power of comparative reasoning." Communications of the ACM, 56(3), 70-77.

[32] Chen, Q., et al. (2014). "A survey of big data storage and computational frameworks." Journal of Computer Science and Technology, 29(2), 165-182.

[33] Qiu, M., et al. (2014). "Performance modeling and analysis of big data processing in cloud systems." IEEE Transactions on Parallel and Distributed Systems, 25(9), 2193-2203.

[34] Bhatia, R., Kumar, S., Goyal, P. (2013). "Hadoop: A framework for big data analytics." International Journal of Emerging Technology and Advanced Engineering, 3(3), 238-241.

[35] Vavilapalli, V. K., et al. (2013). "Apache Hadoop YARN: Yet Another Resource Negotiator." Proceedings of the 4th Annual Symposium on Cloud Computing, 5(7), 1-16.