Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Big Data Analytics Using Hadoop and Spark: Applications, Challenges, and Future Direction

Surya Veera Brahmaji Rao Sunnam
Lead Data Engineer, Bank of America, USA
Corresponding Author: Surya Veera Brahmaji Rao Sunnam, **E-mail**: reachsuryasunnam@gmail.com

| ABSTRACT
Big data describes large datasets that are difficult to manage owing to their diversity, size, and complexity in terms of storage, analysis, and visualisation for future operations. Artificial intelligence (AI) based big data analytics have revolutionised the processing of massive data sets by leveraging distributed computing platforms like Apache Spark and Apache Hadoop. Both HDFS and MapReduce leverage the 5Vs of Hadoop—Volume, Velocity, Variety, and Veracity—to store files and perform batch analysis on massive, heterogeneous datasets. Spark expands upon these capabilities with libraries for graph analytics, streaming, machine learning, in-memory computing, and directed acyclic graphs (DAGs). This paper gives a general outline of the Hadoop and Spark ecosystem, as well as their architecture and how they are used in distributed machine learning, real-time analytics, NLP, and anomaly detection. It also addresses such critical issues as scalability limitations, data protection, resource administration, and complexity of infrastructures. Lastly, the future directions are discussed with a focus on cloud-native designs, edge computing, and hardware acceleration as well as intelligent resource optimization to improve the performance, efficiency, and flexibility of the next generation Big Data systems.

| KEYWORDS

AI–Big Data Analytics, Apache Hadoop, Apache Spark, Streaming Analytics, Data Mining, MapReduce

| ARTICLE INFORMATION

ACCEPTED: 02 January 2021 **PUBLISHED:** 28 February 2021 **DOI:** 10.32996/jcsts.2021.3.1.7

I. Introduction

The blistering development of the Internet, cloud computing, mobile technologies and interconnected devices have led to unprecedented growth of digital data. Big Data: Traditionally, data have been managed through conventional systems that were unable to condense the sheer size and complexity of such datasets [1]. Traditional methods of data storage and processing have proven ineffective and counterproductive when faced with Big Data's unprecedented pace, diversity, and volume. With data ever increasing exponentially in most industries including healthcare, finance, manufacturing, and smart infrastructure among others, the use of sophisticated analytics structures has become a necessity to draw meaningful insights.

Big Data analytics involves analysing large, complex datasets using computational analytical models and scalable systems. Technologies like Apache Spark and Apache Hadoop, which are distributed computing platforms, have become crucial in this category [2]. Hadoop is an excellent option for consistently processing large batches because of its HDFS and parallel processing capabilities, which are made possible by its MapReduce programming methodology [3], [4]. Spark further enhances these functions by including in-memory computation and Resilient Distributed Datasets (RDDs) which are much more efficient in regard to iterative and real-time analytics.

Data mining from massive datasets is also made easier with the integration of AI and Big Data analytics technologies [5]. Automated decision-making, anomaly detection, trend forecasting, and latent trend discovery are all goals of AI-powered analytics, which employ ML, DL, and natural language processing techniques [6]. Analysis performed on scalable platforms is more

accurate, faster, and more scalable when distributed systems like Hadoop and Spark are used for training and deploying AI models.

A. Structure of the paper

The following describes the content of the paper: In Section II, cover the Big Data aspects, Hadoop architecture, HDFS, MapReduce, ecosystem components, and noteworthy AI-Big Data Analytics applications that utilize Hadoop. Section III is a discussion of AI–Big Data Analytics with Spark, which involves Spark architecture, ecosystem modules, MapReduce to in-memory computing transition, and the use of emerging technologies. The section IV presents challenges and future perspectives of Hadoop- and Spark-based Big Data systems based on scalability, security, resource management, and complexity of infrastructure. Section V gives a summary of literature review and Section VI brings the study to the end by giving the important findings and future research directions.

II. AI-Big Data Analytics Using Hadoop

Apache Hadoop AI-Big Data Analytics use a combination of HDFS, a distributed storage system, and MapReduce, a parallel processing system, to store and analyse massive, diverse datasets that meet the 5Vs: volume, velocity, variety, veracity, and value. The fault-tolerant and scalable architecture of Hadoop is capable of performing efficient data mining, large-scale machine learning, and enterprise-scale analytics across such domains as healthcare, finance, education and cybersecurity.

A. Characteristics of Big Data

The following features, illustrated in Fig. 1, characterize big data:

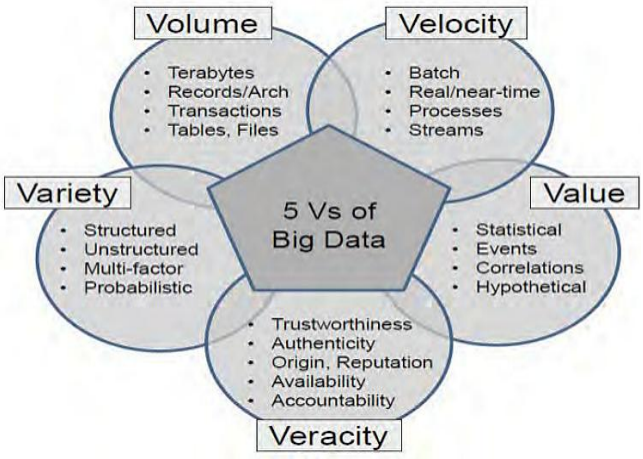

Fig. 1. 5Vs of Big Data.

1) Volume

The biggest danger that this book poses to conventional IT infrastructures is the knowledge it contains. This is the image that most people see when they hear the term "Big Data." [7]. Even though they don't have the resources to process it, many businesses already have massive volumes of log data stored away. Big data analytics is appealing because of the advantages it offers in processing massive amounts of data.

2) Velocity

Data creation, processing, storage, and analysis speeds are all being enhanced by relational databases, which is why the term "velocity" is used to describe this trend. Both the rate of data generation and its rate of dissemination are measured in terms of velocity. Concerning the rapid spread of messages via social media. Wal-Mart's data warehouse held 1 million gigabytes of information in 1999. Its data access in 2012 exceeded 2.5 petabytes, or 2,500,000 gigabytes.

3) Variety

The diversification of Big Data is the next characteristic to consider. Problems can arise when integrating large datasets with relational databases, and structured data isn't always considered big data. Understanding that Big Data is a subset of a larger category is thus essential for data analysts. Big Data storage and analysis become much more complicated when dealing with different types of data, both organized and unstructured. Unstructured data makes up 90% of all data.

4) Veracity

Dirty data is inevitable when dealing with high-velocity, high-variety, high-volume data. No data set is ever going to be completely accurate. The provided data has a very wide range of quality. The study's accuracy depends on the reliability of the data used.

5) Value

A huge data's value is the most crucial factor. The prospective worth of Big Data, however, is enormous. Having access to enormous data is great and all, but it worthless if can't do anything with it. Big data storage IT infrastructure solutions are expensive to deploy, and companies wants a return on their investment.

B. Hadoop Architecture

Hadoop, a distributed object-oriented programming framework, was developed in 2005 by Goug Cutting and Mike Cafarella. A distributed search engine project served as its inspiration. With this open-source Java platform, large data sets can be stored, retrieved, and used in a distributed fashion that is characterised by fault tolerance and high availability at a low cost. Data from sensors, emails, chats, videos, audio, folders, files, programs, structured searches, unstructured data, and any other source that doesn't fit neatly into any one category are all part of the massive amounts of data that Hadoop manages [8]. Hadoop clusters eliminate the need to collect resources from several systems by storing them all in one place without schema representation. Hadoop is comprised of the following several components: Lucene, Avro, Chukwa, Flume, Sqoop, Oozie, Pig, and HBase. Work scheduling, location awareness, documentation, and source code are all part of the Hadoop package.

A Hadoop cluster is made up of multiple Slave nodes and one Master node. Slave nodes serve as both task trackers and data nodes, storing compute-only and data-only worker nodes, respectively, while master nodes contain data, names, job trackers, and task trackers. Schedules for jobs are handled via the Job Tracker. Hadoop consists of two primary parts. Map Reduce and HDFS are two of them. HDFS handles data storage in a clustered environment, whereas Map Reduce conducts data processing. See the Hadoop infrastructure in action in Fig. 2.

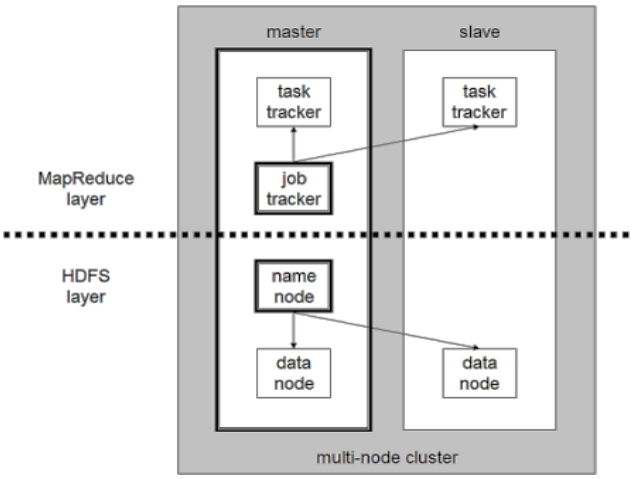

Fig. 2. Architecture of Hadoop

C. Hadoop Distributed File System (HDFS)

One part of Hadoop that can withstand storage failures is the HDFS. The HDFS can withstand the breakdown of the majority of storage infrastructure, grow incrementally, and store massive amounts of data without data loss. Hadoop facilitates the coordination of several computer operations by grouping them into clusters. Using low-cost computers, clusters can be constructed [9]. Hadoop can keep the cluster running even if one server fails by distributing the workload across the remaining servers. This prevents data loss and uninterrupted work. For cluster storage management, HDFS uses a technique called "blocking," which involves dividing incoming files into smaller chunks and then storing each block redundantly among all pool observers. Figure 3 shows that HDFS usually keeps three full copies of each file by copying each part to three different computers:

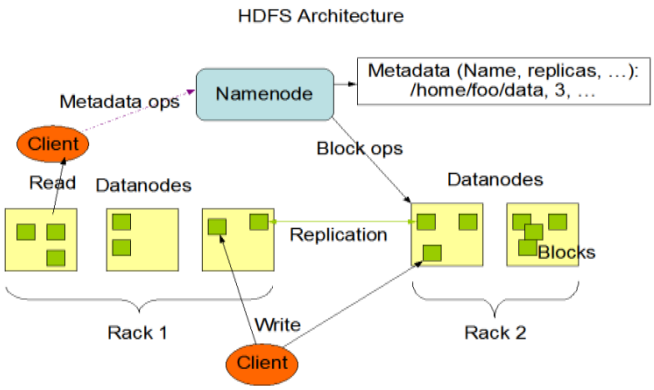

Fig. 3. HDFS Architecture

D. Big Data Analytics Hadoop MapReduce

Big Data processing is made easier with Adoop, an open-source program. Big Data analysis is widely used by organizations and researchers. Hadoop relies on Google's File System and MapReduce for its foundation. Hadoop uses a distributed computing environment to manage massive datasets. Elastic MapReduce, HDFS, Apache Hive, Hadoop Kernel, Base, and Zookeeper are all components of the Apache Hadoop ecosystem.

Hadoop is made up of two primary components:

- **Storage:** HDFS is a decentralized system that may be used on inexpensive hardware and offers fault tolerance [10]. The high speed access that HDFS provides to application data makes it a good fit for applications with large data collections [11]. Thousands of servers can store data in HDFS. The HDFS architecture is master/slave. Data stored in HDFS is organized into blocks of specified size whenever new files are added. The default value for the customizable block size is 64 MB.
- **MapReduce Processing:** This programming style was developed by Google in 2004 with the goal of simplifying the development of error-tolerant systems that handle massive amounts of data continuously across several devices. This partitions the issue and data sets and runs them in parallel, handling massive data sets. The following are two of MapReduce's functions:
  - Commonly used for data filtering, transformation, and parsing, the Map function is called first. The result of Map is subsequently used as the input to Reduce.
  - Reduce-Utilising the Reduce function to condense information retrieved via the Map function is common practice, while it is not necessary.

E. Big Data Components on Hadoop Framework

Apache Hadoop is an open-source, free, and platform for managing distributed computing over computer clusters with massive datasets using simple programming concepts. The Apache Hadoop software framework enables the distributed processing of large datasets across multiple machines in a network. Data processing and storage can be easily scaled up or down from a single server to thousands of computers using Hadoop. An advantage of Hadoop is its scalability, which allows it to work with anything from a single server all the way up to thousands of devices all using local storage and compute. The fact that Hadoop can operate on reasonably priced hardware is just one of its several advantages (6.8.9).

F. Hadoop includes these modules.

1) Hadoop Distributed File System (HDFS)

A distributed file system enables application data access with high throughput. Because of its fault tolerance and distributed nature, this file system is reliable enough to hold enormous amounts of data. The name node is another name for the master node; it is responsible for overseeing the cluster's metadata [12]. In a Master/Slave configuration, one device (the master) controls another device (the slave) or devices (the servants). Slave nodes that store data are called data nodes. The programming language Java underpins this file system.

2) Hadoop YARN / Map Reduce

A system for managing cluster resources and scheduling jobs. A programming style that manages cluster resources and processes massive datasets concurrently on a cluster; it is based on the Hadoop concept.

3) HBase

This distributed database is both scalable and capable of storing big tables of structured data. The ability to perform updates, inserts, removals, etc., is a transactional capability it offers. HBase is a NoSQL database used in conjunction with HDFS. HBASE provides quick access, fault tolerance, and columnar storage, enabling real-time reading and writing of massive datasets.

4) Pig

Apache PIG is an environment for high-level data flows and parallel processing. It makes it easy to implement advanced MapReduce functions like summarizing/aggregating, merging, sorting, and more. Parallel processing, one of PIG's key advantages, allows it to manage extremely big datasets.

5) Hive

Data warehouses enable data aggregation and ad hoc querying. Hive is a data warehouse software package that can use to organise, access, summarise, and analyse a lot of data. With HiveQL, a SQL-like language, and can query petabytes of data in Hive. With its complete support for map/reduce, it is ideal for HDFS data analysis. The benefits of Hive include its resemblance to the classic SQL language, its speed over large datasets, its scalability, its extensibility, and the wide range of reports it gives.

6) Sqoop

Sqoop is a program that connects Hadoop to relational databases and can move large amounts of data between them. With Sqoop, may bring in data from other sources and load it into HDFS, HBASE, or HIVE. It enables parallel data transfer and data import/export to external relational databases. Importing data updates from Hadoop and relational databases is made possible by using simple SQL queries and multi-run tasks.

7) ZooKeeper

Zookeeper provides Hadoop-based operations services, which are a high-performance application coordination solution. Data synchronization, group services, and configuration information are all handled by this centralized service, which is utilized by dispersed applications. Due to its design, Zookeeper is able to provide high availability through the use of multiple services. Multiple distribution procedures can work together thanks to a shared hierarchical name space of registers, or znodes.

8) Avro

Avro is a method for serialising data that offers a compact binary data format, large data structures, and a container file for persistent data storage. Data reading and writing are supported via schemas. Java Script Open Notation (JSON) is used to describe data types and protocols. Client programs and services, as well as Hadoop nodes, communicate with one another over wire format.

9) Tez

This generalised data-flow programming framework provides a flexible and powerful engine for batch and interactive data processing using an arbitrary directed acyclic graph (DAG). Tez is replacing Hadoop MapReduce as the primary execution engine in several commercial applications and frameworks within the Hadoop ecosystem, including Hive and Pig.

10) Spark

Apache Hadoop is an efficient and versatile computer engine. It offers a straightforward and expressive programming model that can handle large-scale graph computation, stream processing, machine learning, and a wide range of other applications. Quickly analyse and analyse massive amounts of data using Apache Spark, a computational engine based on ML. Hadoop map-reduction with in-memory processing is 100 times slower than Spark, thanks to Spark's sophisticated analytical engine and its integration with Hadoop.

G. Applications of Hadoop in Big Data Analytics

The Hadoop framework provides distributed storage (HDFS) and batch processing (MapReduce), enabling data mining and analytics on massive amounts of unstructured data. Important uses include:

1) Classification Analysis

Hadoop is used to perform large-scale classification based on distributed machine learning algorithms. It has extensive use in spam detection, medical diagnosis, fraud detection, as well as customer segmentation where large volumes of historical data are processed in parallel.

2) Cluster Analysis

Hadoop allows users to distribute cluster massive data sets in order to find groups with shared attributes. It is especially effective when it comes to customer behavior analysis, market segmentation, and social media analytics, when large user data need to be clustered effectively.

3) Evolution Analysis

Hadoop processes time series data and historical data to establish trends and patterns with time. Its applications include the stock market prediction, financial risk modeling and long-term business forecasting as well as using large archived datasets.

4) Outlier Analysis

Hadoop is used in detecting anomalies in transactional and medical data of large scale. Its application has been common in fraud detection in banks, cybersecurity, and health care analytics to detect abnormal trends in large repositories of data.

III. AI-Big Data Analytics Using Spark

Presented below are the primary components of Spark as well as the Apache Spark project. The article explains how Apache Spark is the next big thing in big data analytics engines, following in the footsteps of Hadoop's MapReduce. There are a few industry contributions and case studies that also summarise.

A. Architecture of Apache Spark

Apache Spark is designed to process data spread among cluster environments using a master-worker architecture concept. A Spark application starts a driver program, the main management hub in terms of the job scheduling and resources management. The user code is translated by the driver into a DAG of stages [13]. On the internal level, Spark has two important scheduling structures: DAGScheduler which breaks jobs into stages depending on the data dependencies and keeps the lineage of RDDs, and TaskScheduler which allocates particular tasks of the current stage to the existing worker nodes to be executed. Spark can have several cluster deployment configurations, such as standalone mode, and integration with cluster managers, e.g. Mesos and YARN. Spark also deploys executors on every worker node and they run tasks and cache intermediate data either on memory or disk [14]. This in-memory computation model allows processing at a very high speed on iterative point and real-time analytics unlike traditional processing frameworks that operate on disk. Figure 4 displays the layered architecture of Apache Spark which describes its parts in terms of storage to an application layer. Storage systems, HDFS and HBase are included in the bottom layer and are followed by cluster managers, Spark Standalone and YARN.

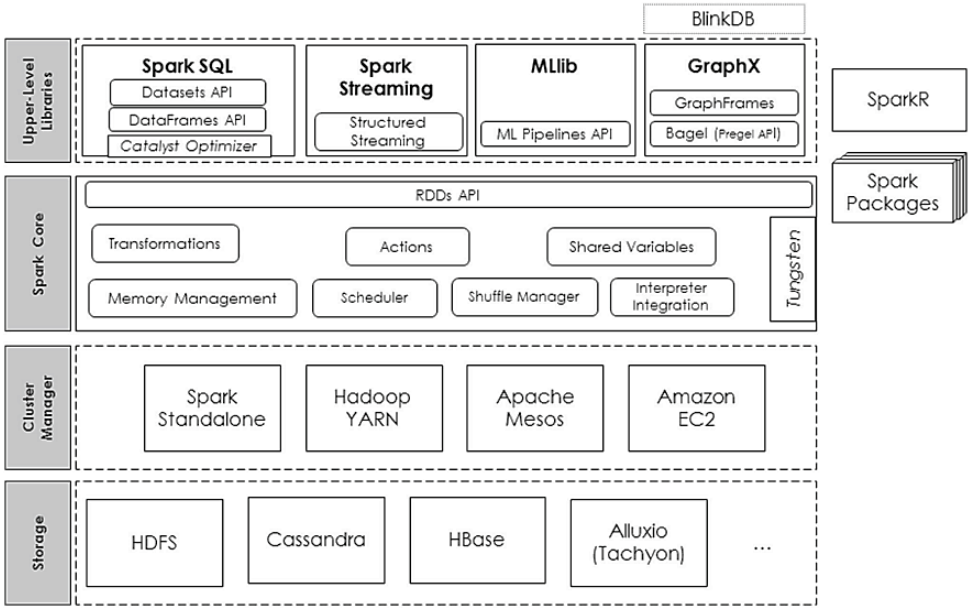

Fig. 4. Architecture of Spark.

On top of Spark Core (RDDs, scheduling, memory management), more advanced analytics are presented by higher level libraries e.g. Spark SQL, Spark Streaming, Spark MLlib and Spark GraphX.

B. Ecosystem of Spark

Developers utilising Scala, Java, or Python can construct streaming jobs utilising the Spark Streaming module, which provides language-integrated APIs for stream processing. Among Spark Streaming's features are those that facilitate recovery from operator states and lost work. A number of sources can have their data read by it, such as HDFS, Kafka, Flume, Twitter, and ZeroMQ. A library that provides machine learning capabilities is Spark, and it's named MLib [15]. Connect this library to Hadoop workflow, and it works with R and Python libraries as well as other sources including HBase and local files [16]. Spark's GraphX Library offers features that facilitate graph parallel processing and graph data visualisation. The underlying data structure of GraphX is the Resilient Distributed Dataset. It's meant to describe items related to subjects or resources (as illustrated in figure 5. In addition, GraphX offers a library of graph algorithms that can be utilised according to the requirements.

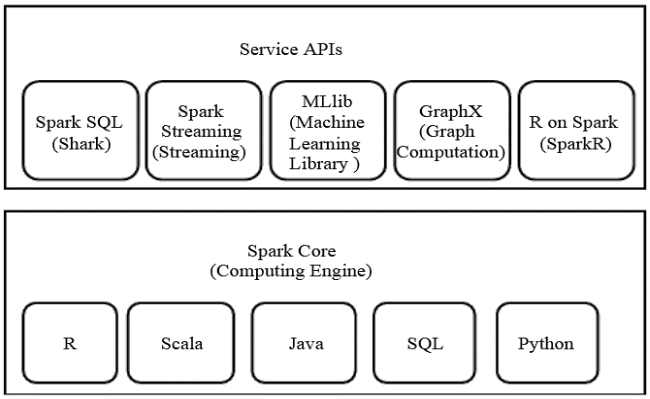

Fig. 5. Apache Spark Ecosystem

There are a few key parts of the Apache Spark ecosystem:

- **Spark SQL:** Formerly known as Shark. One such distributed framework is Spark SQL, which can handle both structured and semi-structured data [17]. Supporting data formats including Hive tables, JSON, and Parquet, it enables the development of analytical and interactive applications.
- **Spark Streaming:** The users can process data streams as they happen. Streaming analysis is made possible by Spark streaming, which enhances Apache Spark's fast scheduling capabilities by merging data into small batches. IoT sensors Twitter, Amazon Kinesis, and Apache Kafka are among the data sources that contribute micro batches, which are then transformed.
- **MLlib:** Machine learning is made easier and more scalable with its rapid delivery of high-quality algorithms. Among the many machine learning techniques at disposal are clustering, regression, classification, and linear algebra [18]. An optimisation approach for generic gradient descent is just one example of the lower-level machine learning primitives made available in its library. Data import and model evaluation are two of its other features. This tool works with Java, Scala, and Python.
- **GraphX:** Scalable graph-structured data generation, manipulation, transformation, and execution are made possible by this graph computation engine. Directed graphs can be easily created with the help of several Spark RDD APIs.
- **Spark Core:** Apache Spark's core functionality serves as a foundation upon which further features are developed. Scala, Java, and Python are just a few of the languages it supports, and its large range of APIs makes development a breeze. Spark Core uses in-memory computing to overcome MapReduce's slowness and provide speed.
- **SparkR:** Data scientists loves this R package since it integrates Spark into the R environment. In the same way that R's DataFrame is the most fundamental data structure for processing data, SparkR's DataFrame is the building block of SparkR. It is capable of selecting, filtering, and aggregating data from big datasets, among other things.

C. Apache Spark in Emerging Technologies

Emerging technologies in big data analytics include Apache Spark:

1) Fog Computing: Fog computing requires extremely low

Apache Spark provides low-latency, parallel machine learning computation as well as advanced graph analytics methods. Utilising Spark streaming, MLlib, and Apache Kafka, an apparatus is able to identify monetary transactions suspicious [19]. An individual's spending habits can be categorized by obtaining their credit card transactions. In addition to Kafka and Spark's real-time streaming, additional models can be trained to anticipate any card transaction anomalies. Because of its superior speed compared to MapReduce, which offers interactive analysis tools like Hive and Pig, Spark can also be utilised for interactive analysis.

2) Machine Learning: Apache spark has a highly powerful

Apache Spark's MLlib package is great for ML applications because it contains many machine learning techniques. One such example is Spark's built-in SVM. One type of ML techniques that can help with classification and regression is support vector machines, or SVMs. The SGD optimiser is the sole option for optimising SVMin Spark. A different machine learning technique known as XGBoost (Extreme Gradient Boosting) is also available in Spark [20]. By integrating XGBoost into the Apache Spark-based data processing system, this approach allows users to construct a unified pipeline [21]. DL is another way Spark uses machine learning. Apache Spark is one of the easiest ways to build distributed ledger technology (DL), which is a smart solution because DL is computationally expensive. Here are a few examples of how DL can be used in Apache Spark:

- Python Spark, Keras, and Distributed DL for Elephas
- Yahoo Inc. uses TensorFlowOnSpark
- CERN uses Distributed Keras
- Qubole uses tutorial Keras and Spark

Also, an open-source tool named DL Pipelines provides high-level APIs for running scalable DL in Python with Apache Spark.

D. Applications of Spark in Big Data Analytics.

There are some applications are as follows:

1) Distributed Scalable Machine Learning.

The MLlib of the Spark allows large-scale training of classification, regression, clustering and recommendation models on distributed data.

2) Streaming Analytics and Real-Time AI.

Spark streaming is used to detect anomalies in real-time, detect fraud, and predictive monitoring with the help of high-velocity data streams.

3) Deep Learning Integration

Spark is also compatible with deep learning systems (e.g. TensorFlow, PyTorch) to facilitate neural network training on big data systems in a distributed manner.

4) Natural Language Processing (NLP)

Spark is utilized to analyze sentiment, topic model, document classification, and chatbot through the analysis of large volumes of text.

5) Social Network Intelligence and Graph Analytics.

Spark allows community detection, influence analysis, and recommendation systems applications, all based on graphs, but at scale, using GraphX.

IV. Challenges and future directions in Hadoop and Spark-Based Big Data Systems.

Big Data systems driven by AI and built on Apache Hadoop and Spark have the following challenges, with some possible solutions, described below:

A. Challenges in Hadoop and Spark-Based Big Data Systems

Hadoop and Spark-Based AI Big Data Systems Face the Following Obstacles:

1) Scalability and Performance Issue

Systems based on Hadoop and Spark are necessary for managing massive volumes of diverse data created by social media, enterprise applications, sensors, and logs at high velocity. Scalability is also a requirement because the volume of data keep increasing with time. Vertical scaling fails to work correctly, and horizontal scaling between distributed clusters is needed. Nonetheless, disk I/O latency, network latency, and overhead of data movement are the sources of performance bottlenecks[22]. The concept of computation to data (through HDFS and MapReduce) in Hadoop decreases the amount of congestion on the network but fails to remove data delays due to the disk-based processing. Spark is enhanced with in-memory computing, but memory setup and JVM garbage collection cost are the major factors that influence efficiency. Ineffective RDD operations and data skew, as well as unbalanced partitions can further lower the performance of parallel processing. Streaming analytics brings forth more latency requirements by necessitating high throughput storage and real time processing layers. Scalability Heterogeneous schema management, incomplete streaming data, and high ingestion rates have continued to be a thorn in the side of scalability

2) Data security and privacy

Big data platforms combine information of several and heterogeneous distributed sources, exposing them to security vulnerabilities. In distributed Hadoop and Spark clusters, it is complicated to assure safe data acquisition, transmission, storage, and processing. The insider danger is one of many, along with data breaches and unauthorized access. The availability, accuracy, and conformity of data with applicable regulations depend on effective data governance [23], [24]. Data provenance management is made more challenging by the size and velocity of big data systems.

3) Resource management and fault tolerance

Distributed environments consist of a fundamental challenge of efficient resource allocation. CPU, memory, disk and network bandwidth have to be managed optimally in Hadoop clusters in order to prevent bottlenecks. The in-memory processing model in Spark needs to have a close attention to the memory partitioning between the caching of the RDDs and the execution of the tasks. Poor set-up may result in over collection of garbage and poor performance. Hadoop and Spark are different in terms of fault tolerance. Hadoop uses data replication and checkpointing to make it reliable at the cost of storage overhead. Spark provides precomputation of lost RDD partitions by using lineage, so that it lowers storage costs, but it requires the stability of drivers and the presence of lineage metadata[25]. The problem of node failures, network failures, distorted data partitions, and workload imbalance is hard to handle.

4) Infrastructure cost and complexity

Installation and operations of Hadoop and Spark environment involve high infrastructure cost. Distributed clusters require scalable storage systems, high speed networking and powerful compute nodes. The streaming applications also add more money to the budget because of the storage requirement of low latency as well as real-time processing engines. Another issue is the complexity in operations. Integration of a variety of data formats, schema mapping, data cleaning, transformation and data governance all add to administration overheads. The interoperability between tools in the Hadoop and Spark ecosystem is a complex task that requires expert knowledge. In addition, extensive visualization, modeling, and analytics can take a significant amount of hardware acceleration, distributed processing, increasing capital and operations costs.

B. Future Perspectives

The future directions in Hadoop and Spark-Based Big Data Systems are described in below:

1) Potential areas for further research and exploration

Determine whether there is a need for more study to improve understanding of Hadoop and Spark within the framework of Big Data Analytics. Discussed about new developments in areas like distributed computing, machine learning, and streaming analytics that might affect these frameworks' functionality and performance [26]. Draw attention to the places that need more research, like ways to improve data processing efficiency, fault tolerance systems, or resource allocation.

2) Emerging trends and technologies in Big Data Analytics

The future of Big Data Analytics, Spark, and Hadoop could be affected by new technical developments and trends, therefore it's important to be informed. Serverless computing, edge computing, and cloud-based analytics platforms are bringing about a paradigm shift in Big Data Analytics [27], [28]. Find ways these new technologies could fit into the present-day Hadoop and Spark environments.

3) Opportunities for enhancing Hadoop and Spark for improved performance

Discuss possible ways to make the Hadoop and Spark tools better to fix the problems and issues they currently have [29]. Determine where the two frameworks can be improved in terms of performance, scalability, and use of resources. Improving Hadoop and Spark's performance could be as simple as using distributed caching, hardware accelerators, or dynamic workload management.

V. LITERATURE REVIEW

Data processing frameworks such as Apache Spark, Hadoop, Hive, and Pig are the subject of the research presented in Table I, which includes comparisons and analyses of these frameworks. The papers look at methods of performance optimization, parameter tuning, machine learning integration, and distributed deep learning abilities in different fields of application. All in all, the results indicate that Spark has better performance in in-memory processing and mentions scaling issues and opportunities further optimization research.

Tang *et al.* (2020) purpose to conduct an exhaustive analysis of several optimisation methods with respect to Spark's generalisability and performance enhancement. Spark is introduced together with its computing system and programming style. This section discusses Spark's advantages and disadvantages. The literature on solving techniques is investigated and categorized.

In addition, provide a number of Spark-compatible data management and processing platforms, methods for machine learning, and apps. Last but not least, go over some of the rough spots in using Spark for massive in-memory data processing [30].

Ahmed *et al.* (2020) evaluates Hadoop and Spark on a lab-installed cluster, finds the most important parameters under different conditions (such as resource usage, input splits, and shuffle), and compares their performance. To fine-tune these settings, they resorted to a trial-and-error approach after a battery of experiments. Choose WordCount and TeraSort as two workloads to compare and contrast the frameworks. Time-to-execution, throughput, and speedup are the three pillars on which performance measurements rest. It was found that the amount of input data and the accuracy of parameter selection are critical to the system's performance [31].

Cheng, Zhang and Ye (2019) Potential applications of cloud computing frameworks such as Apache Spark and Apache Hadoop in agricultural big data analysis have been investigated. To anticipate the yield of multiple linear regression, the author used Spark MLlib and developed big data analysis apps in both frameworks. The approach is applicable to real-world agricultural parks. The two frameworks were tested and evaluated to determine how well they handled large volumes of farm data. Spark performs better than Hadoop in the experiments, and the model gets better prediction results overall [32].

Ahmad *et al.* (2019) Used Big Data tools such as Apache Hive and Apache Pig to analyse different ECG datasets. When comparing Apache Pig and Apache Hive, both used to analyse ECG data, the results showed that Apache Pig was more efficient and straightforward, yielding faster results with less effort. "Big Data Analytics" describes the steps used to examine datasets that are very huge, dynamic, diverse, and multiplex. Decisions cannot be made using traditional methods of data extraction from complex and multi-structured information [33].

Sharma and Kaur (2019) shown the extensive study on various tools related to Big Data processing and has done extensive comparison on MapReduce Vs Spark. The frameworks have been studied on real time datasets and finally compared in terms of processing time. Spark showing the remarkable improvement over MapReduce. The processing began with Hadoop's MapReduce Framework, however it has numerous drawbacks because it uses multiple disc processing processes [34]

Ahn, Kim and You (2018) assess Apache Spark's speed and scalability, a widely used tool for managing massive datasets. They assess not only Spark's performance in a distributed cluster setting, but also that of TensorFlowOnSpark, an exciting new distributed deep learning framework for efficient handling of massive data. The theory behind Spark on YARN is that it offers a robust foundation for distributed ML and DL, which can scale effectively through parallel processing of algorithms and data [35].

TABLE I. SUMMARY OF RECENT STUDIES ON AI-BIG DATA ANALYTICS USING HADOOP AND SPARK.

Authors	Study On	Approach	Key Findings	Limitations	Future Work
Tang *et al.* (2020)	Optimization techniques for improving generality and performance of Apache Spark	Comprehensive review of Spark programming model, system architecture, classification of optimization techniques, and discussion of data management systems and ML applications supported by Spark	Identified various optimization strategies enhancing Spark's scalability and in-memory performance; highlighted Spark's advantages over traditional systems	Mainly review-based; lacks experimental validation of proposed classifications; limited real-world benchmarking	Explore adaptive optimization mechanisms and address challenges in large-scale in-memory processing and heterogeneous environments
Ahmed *et al.* (2020)	Analysing Hadoop and Spark's respective performance	Experimental cluster implementation; tuning parameters (resource utilization, input splits, shuffle) using trial-and-error; workloads: WordCount and TeraSort; metrics: execution time, throughput, speedup	Spark typically beats Hadoop under optimised configurations, however performance is quite sensitive to input size and proper parameter tuning.	Trial-and-error tuning may not generalize; limited workloads used for evaluation	Develop automated parameter tuning frameworks and extend comparison to diverse real-world big data workloads
Cheng, Zhang and Ye (2019)	Investigating Spark and Hadoop for Agricultural Big Data	Agricultural big data applications were developed, including yield prediction using Spark MLlib and multiple linear	Spark showed better comprehensive performance than Hadoop; yield prediction model achieved	Focus limited to agricultural datasets; limited ML models evaluated	Apply advanced ML/DL models and test scalability across broader agricultural

		regression. Experimental performance was also compared.	improved accuracy		datasets
Ahmad *et al.* (2019)	Utilising Hive and Pig for ECG Big Data Performance Analysis	Analysis of electrocardiogram (ECG) datasets using Apache Hive and Apache Pig; performance-based assessment	Apache Pig provided faster and more systematic results compared to Hive	Limited to ECG datasets; does not compare with Spark or newer frameworks	Extend analysis to real-time streaming frameworks and integrate Spark-based analytics for healthcare big data
Sharma and Kaur (2019)	Comparison of MapReduce vs Spark	Comparative study on real-time datasets; performance evaluation based on processing time	Spark significantly improves processing speed over MapReduce due to in-memory computation; MapReduce suffers from disk I/O overhead	Limited dataset diversity; lacks deep configuration tuning analysis	Investigate hybrid frameworks and optimize memory management for large-scale real-time analytics
Ahn, Kim and You (2018)	Performance evaluation of Apache Spark and TensorFlowOnSpark	Assessing Spark on YARN and TensorFlowOnSpark's performance for distributed deep learning; conducting experimental evaluations in distributed cluster environments	Spark on YARN ensures scalability and strong performance for distributed ML/DL tasks; TensorFlowOnSpark effectively supports deep learning workloads	Focused mainly on cluster-based setup; limited evaluation of heterogeneous cloud environments	Explore optimization in multi-cloud and GPU-ac

VI. Conclusion and Future Work

Hadoop and Spark, the backbone of Big Data analytics, have made it possible to handle massive data quantities and make well-informed decisions. For handling massive amounts of diverse data, Hadoop's dependable distributed storage solution, HDFS, and scalable batch processing, MapReduce, are ideal. Spark supplements such features with in-memory computation, DAG-based scheduling and rich libraries supporting real-time analytics, machine learning, graphs processing and streaming applications. These frameworks combined allow processing structured and unstructured data efficiently in the healthcare industry, finances, cybersecurity, and industry. Non-etheless, obstacles like bottlenecks of scalability, security issues of data, complexity of resource management and cost of infrastructure are still present. These problems would have to be addressed through optimized workload management, enhanced fault tolerance, and improved governance mechanisms. Hadoop and Spark, used together, provide the foundation of present AI-Big Data ecosystems; they facilitate the delivery of distributed computing systems' scalable, adaptable, and high-performance analytics.

Future research should aim to combine Hadoop and Spark with cloud-native architectures, edge computing, and hardware accelerators. Scalability, performance, and real-time AI in the next-generation Big Data systems can be further enhanced by improving adaptive resource allocation, secure multi-tenant frameworks and low-latency streaming analytics.

References

[1] Z. H. Munim, M. Dushenko, V. J. Jimenez, M. H. Shakil, and M. Imset, "Big data and artificial intelligence in the maritime industry: a bibliometric review and future research directions," *Marit. Policy Manag.*, vol. 47, no. 5, pp. 577–597, Jul. 2020, doi: 10.1080/03088839.2020.1788731.

[2] D. P. and Acharjya and K. A. P, "A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 2, pp. 1899–1908, Dec. 2016, doi: 10.22214/ijraset.2022.48294.

[3] H. K. Omar and A. K. Jumaa, "Big Data Analysis Using Apache Spark MLlib and Hadoop HDFS with Scala and Java," *Kurdistan J. Appl. Res.*, vol. 4, no. 1, pp. 7–14, May 2019, doi: 10.24017/science.2019.1.2.

[4] M. A. Mukhdoomi, "Data Processing Framework Using Apache and Spark Technologies in Big Data," *Int. J. Res. Publ. Rev.*, vol. 2, no. 7, pp. 616–626, 2021.

[5] J. Z. Zhang, P. R. Srivastava, D. Sharma, and P. Eachempati, "Big data analytics and machine learning: A retrospective overview and bibliometric analysis," *Expert Syst. Appl.*, vol. 184, p. 115561, Dec. 2021, doi: 10.1016/j.eswa.2021.115561.

[6] H. Luan *et al.*, "Challenges and Future Directions of Big Data and Artificial Intelligence in Education," 2020, *Frontiers Media SA*. doi: 10.3389/fpsyg.2020.580820.

[7] H. Jasim Hadi, A. Hameed Shnain, S. Hadishaheed, and A. Haji Ahmad, "Big Data and Five V'S Characteristics," *Int. J. Adv. Electron. Comput. Sci.*, vol. 2, no. 1, pp. 2393–2835, 2015.

[8] B. Saraladevi, N. Pazhaniraja, P. V. Paul, M. S. S. Basha, and P. Dhavachelvan, "Big Data and Hadoop-a Study in Security Perspective," *Procedia Comput. Sci.*, vol. 50, pp. 596–601, 2015, doi: 10.1016/j.procs.2015.04.091.

[9] S. Kumari and D. N. Bhardwaj, "A Review paper on Big Data," *Int. J. Mob. Comput. Appl.*, vol. 6, no. 2, pp. 1–3, May 2019, doi: 10.14445/23939141/IJMCA-V6I2P101.

[10] U. A. Korat and A. Alimohammad, "A Reconfigurable Hardware Architecture for Principal Component Analysis," *Circuits, Syst. Signal Process.*, vol. 38, no. 5, pp. 2097–2113, 2019, doi: 10.1007/s00034-018-0953-y.

[11] R. Beakta, "Big Data And Hadoop: A Review Paper," *RIEECE*, vol. 2, no. 2, 2015.

[12] Ishwarappa and J. Anuradha, "A brief introduction on big data 5Vs characteristics and hadoop technology," *Procedia Comput. Sci.*, vol. 48, no. C, pp. 319–324, 2015, doi: 10.1016/j.procs.2015.04.188.

[13] S. Tang, B. He, C. Yu, Y. Li, and K. Li, "A Survey on Spark Ecosystem: Big Data Processing Infrastructure, Machine Learning, and Applications," *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, Nov. 2018, doi: 10.1109/TKDE.2020.2975652.

[14] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on Apache Spark," *Int. J. Data Sci. Anal.*, vol. 1, no. 3–4, pp. 145–164, Nov. 2016, doi: 10.1007/s41060-016-0027-9.

[15] S. Sudheer, "AI-Based Data Governance: Empowering Trust and Compliance in Complex Data Ecosystems," *Int. J. Comput. Math. Ideas*, vol. 13, no. 1, 2021.

[16] V. F. Pinto, S. Kini, and I. Mcluren Dsouza, "A Review Document on Apache Spark for Big Data Analytics with Case Studies," *Int. J. Comput. Sci. Trends Technol.*, vol. 5, no. 5, pp. 70–73, 2013.

[17] E. Shaikh, I. Mohiuddin, Y. Alufaisan, and I. Nahvi, "Apache Spark: A Big Data Processing Engine," in *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, IEEE, Nov. 2019, pp. 1–6. doi: 10.1109/MENACOMM46666.2019.8988541.

[18] S. Jonnalagadda, P. Srikanth, K. Thumati,] Sri, H. Nallamala, and A. Professors, "A Review Study of Apache Spark in Big Data Processing," *Int. J. Comput. Sci. Trends Technol.*, vol. 4, no. 3, pp. 93–98, 2013.

[19] M. Ali Mohamed, I. M. El-henawy, and A. Salah, "Usages of Spark Framework with Different Machine Learning Algorithms," *Comput. Intell. Neurosci.*, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/1896953.

[20] A. H. Ali, M. Z. Abdullah, S. N. Abdul-wahab, and M. Alsajr, "A Brief Review of Big Data Analytics Based on Machine Learning," *Iraqi J. Comput. Sci. Math.*, vol. 1, no. 2, Jul. 2020, doi: 10.52866/ijcsm.2020.01.01.002.

[21] S. Achouche, U. B. Yalamanchi, and N. Raveendran, "Method, Apparatus, and Computer-Readable Medium for Dynamic Binding of Tasks in a Data Exchange," Sep. 2021

[22] A. Z. Abualkishik, "Hadoop and big data challenges," *J. Theor. Appl. Inf. Technol.*, vol. 97, no. 12, pp. 3488–3500, 2019.

[23] T. Tekdogan and A. Cakmak, "Benchmarking Apache Spark and Hadoop MapReduce on Big Data Classification," in *2021 5th International Conference on Cloud and Big Data Computing (ICCBDC)*, New York, NY, USA: ACM, Aug. 2021, pp. 15–20. doi: 10.1145/3481646.3481649.

[24] S. B. Venkata Naga, K. C. Sunkara, S. Thangavel, and R. Sundaram, "Secure and Scalable Data Replication Strategies in Distributed Storage Networks," *Int. J. AI, BigData, Comput. Manag. Stud.*, vol. 2, no. 2, pp. 18–27, Jun. 2021, doi: 10.63282/3050-9416.IJAIBDCMS-V2I2P103.

[25] R. Hossen *et al.*, "BDPS: An Efficient Spark-Based Big Data Processing Scheme for Cloud Fog-IoT Orchestration," *Information*, vol. 12, no. 12, p. 517, Dec. 2021, doi: 10.3390/info12120517.

[26] T. Hussain, A. Sanga, and S. Mongia, "Big Data Hadoop Tools and Technologies: A Review," *SSRN Electron. J.*, vol. 13, no. 14, pp. 574–578, 2019, doi: 10.2139/ssrn.3462554.

[27] V. Belov, A. Tatarintsev, and E. Nikulchev, "Choosing a Data Storage Format in the Apache Hadoop System Based on Experimental Evaluation Using Apache Spark," *Symmetry (Basel).*, vol. 13, no. 2, p. 195, Jan. 2021, doi: 10.3390/sym13020195.

[28] S. Garg, "Predictive Analytics and Auto Remediation using Artificial Inteligence and Machine learning in Cloud Computing Operations," *Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci.*, vol. 7, no. 2, 2019.

[29] D. Patel and R. Tandon, "Recent advances in distributed systems: Addressing latency, consistency and scalability in modern applications," *Int. J. Res. Anal. Rev.*, vol. 8, no. 2, pp. 1–7, 2025.

[30] S. Tang, B. He, C. Yu, Y. Li, and K. Li, "A Survey on Spark Ecosystem: Big Data Processing Infrastructure, Machine Learning, and Applications," in *IEEE Transactions on Knowledge and Data Engineering*, 2020, pp. 1–1. doi:

10.1109/TKDE.2020.2975652.

[31] N. Ahmed, A. L. C. Barczak, T. Susnjak, and M. A. Rashid, "A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench," *J. Big Data*, vol. 7, no. 1, p. 110, Dec. 2020, doi: 10.1186/s40537-020-00388-5.

[32] Y. Cheng, Q. Zhang, and Z. Ye, "Research on the Application of Agricultural Big Data Processing with Hadoop and Spark," in *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, IEEE, Mar. 2019, pp. 274–278. doi: 10.1109/ICAICA.2019.8873519.

[33] M. Ahmad, S. Kanwal, M. Cheema, and M. A. Habib, "Performance Analysis of ECG Big Data using Apache Hive and Apache Pig," in *2019 8th International Conference on Information and Communication Technologies (ICICT)*, IEEE, Nov. 2019, pp. 2–7. doi: 10.1109/ICICT47744.2019.9001287.

[34] M. Sharma and J. Kaur, "A Comparative Study of Big Data Processing: Hadoop vs. Spark," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2019, pp. 1073–1077.

[35] H. Y. Ahn, H. Kim, and W. You, "Performance Study of Distributed Big Data Analysis in YARN Cluster," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE, Oct. 2018, pp. 1261–1266. doi: 10.1109/ICTC.2018.8539474.