Journal of Harbin Engineering University
ISSN: 1006-7043
Vol 44 No. 7
July 2023

A Novel Coordinated Hybrid Model for Multi Paradigm Database

Manbir Singh Punia[1], Kamal Malik[2] and Vikash Kumar Garg3
[1]Research Scholar, Department of Computer Science, CT University Ludhiana, Punjab
[2] Professor, Department of Computer Science, CT University, Ludhiana,
[3]Professor, Department of Computer Science Engineering, SLIET, Longowal, India

Abstract-NOSQL document databases emerged as an alternative to relational databases for managing large volumes of data. NOSQL document databases are ensures big data storage and high-quality query performance and essentials, when the data scheme does not fit into the scheme of relational databases. They store their data in the form of documents and can handle unstructured, semi-structured, and structured data. As a result of the vast number of data that is being obtained from digital users by various businesses and organizations, the advancement of technology has resulted in the building of massive data repositories, which are collectively referred to as "big data." through the extraction of hidden data from sizable datasets, a technology known as data mining is utilized to discover particular patterns and rules. The term "big data" refers to the total amount of data adjoining each stage of the human lifecycle. It asserts the idea that any sector that performs tasks can be listed and recorded in a series of data that is expanding extraordinarily quickly. The production of a large quantity of data at a speedy rate is the main issue. Due to the variety of different types of data, companies must also store and process it efficiently (structured data, semi-structured data, and unstructured data). This work evaluates the top open-source NoSQL document databases: pig, hive, NoSQL, and MongoDB, which has become a standard for nosql database evaluation. Test the architecture with different tools in terms of time required for read operation, scan and update on various different factors. The performance and scale-up of document databases are assessed using different workloads with a different number of records and threads, where the runtime is measured for each database. In the experimental evaluation, it was concluded that MongoDB is the database with the best runtime, except for the workload composed by scan operations.

Keywords-Big Data, HDFS, NoSQL, shading key-value databases, document databases.

INTRODUCTION

NoSQL databases, often known as non-relational or not just SQL databases, have become popular as a relational data-bases replacement for handling large volumes of data [1]. To solve the volume, diversity, and velocity challenges related to big data, NoSQL offers high performance, flexibility, availability, data replication, and scalability [2]. The four different types of NoSQL databases are key-value, column-oriented, document, and graph databases [3]. The adaptability of NoSQL text databases, which have recently gained a lot of popularity, in terms of handling schema is one of its main advantages. The top NoSQL text databases, according to the DB-Engine Ranking 2023, are Pig, Hive, NoSQL, and MongoDB. MongoDB, utilized by companies like Vodafone, Bosch, SAP, CISCO, eBay, Google, Sage, and others, is the most well-liked NoSQL database. NoSQL document databases are essential when the data structure cannot be accommodated by a relational database. They are a great way to process a large amount of data since they make it simple to modify schemas and fields. In addition to identifying the optimum text database, this experimental project intends to assist users and researchers in choosing the optimal database for their needs. The effectiveness of several databases is evaluated using the Yahoo! Cloud Serving Benchmark (YCSB). The YCSB, an open-source benchmarking tool developed in 2010, has replaced other benchmarks as the standard measure for evaluating NoSQL databases utilizing CRUD operations. With an emphasis on performance and flexibility, the YCSB benchmarking tool compares several systems using the same workloads [4]. In YCSB, there are six common workloads, and adds two more to test these three databases. These workloads consist of reading, inserting, updating, scanning, and read-modify-write operations. The aim is to investigate the scaling behavior of the database engine by using a single node and vary-

ing the number of threads and records. It evaluates the three databases using a regular, limited personal computer rather than a specialized machine. When runtime was taken into account, MongoDB performed the best in the tests. However, MongoDB has the worst performance in the workload comprised of scan operations. The data base which shows the best scaling up when the number of threads varied and the best runtime. When the scanning process was carried out however is Pig. The world's environment is altered by new media advancements. As a result of this development, new technologies and a wide range of datasets are produced. In the last century, a devastating flow of data occurred as a result of the continuous increase in data volumes in a variety of industries, including banking, education, healthcare, social media, and education. This is due to the fact that each and every organization uses the internet to achieve perfection in their work. In just a few milliseconds, a lot of data is created by the internet. Many businesses were confronted with the performance issue as a result of this volume of data, which presented difficulties in data storage, analysis, and processing. Any organization's financial and technical feasibility is impacted by its subpar performance. Structured data are stored, processed, and analyzed with traditional systems. However, the volumes generated by social media and other forms are in a variety of formats that traditional systems are unable to handle. This is the sole reason that traditional systems are ideal for handling these large volumes of structured, semi-structured, and unstructured data, which are referred to as "big data."

The term "Big Data" [3] refers to the total volume of data pertaining to each stage of a human life. It asserts the idea that every industry that performs tasks can be listed and recorded in a group of data that is expanding at an unexpected rate. Information can be broken down into nearly as many distinct categories as there are human hairs. The way data is stored and retrieved has changed as a result of the Internet's design and invention. On CDs and computers, the large files with pages have been restored. Data management has been almost completely transformed by digital data storage. This crucial position in globalization has been achieved in a very short amount of time, with data in the millions of zeta bytes. This means that more data is stored than can be processed or stored. The main issue is how to process this large amount of data effectively and quickly. The big data ought to be processed without being altered or thrown away. The primary issue is the rapid generation of large amounts of data. Because the data comes in a variety of formats (structured data, semi-structured data, and unstructured data), it is also difficult for businesses to store and process it efficiently. As a result, these kinds of data cannot be handled by conventional systems [2]. These are significant issues in every organization. Professional researchers offer numerous solutions to these kinds of issues. Additionally, researchers offer a variety of optimized methods for this kind of data. Hadoop is, therefore, one option. In 2005, Doug Cutting and Mike Caferella developed Hadoop, which Doug gave the name of his son's toy elephant. Hadoop [2] is made to only deal with big data. It distributes data among nodes, each of which contains 64 MB or 128 MB of data, giving it sufficient storage space. It indicates that the data is stored using the appropriate block sizes. Hadoop does not waste space by storing data, and it is also useful for processing large amounts of data quickly because any remaining space in the node is occupied by another type of data. The built-in redundancy of the framework is one of Hadoop's most important mechanisms. Not only is the data redundantly stored across the cluster in multiple locations, but the training model is set up so that failures are expected and automatically determined by running portions of the program on various cluster servers. The goal of Hadoop is to use servers that are frequently and universally available in a very large cluster. Each server will have a set of inexpensive internal disk drives [2]. Map Reduce tries to assign workloads to these servers where the data that needs to be processed has been accumulated in order to improve performance. The term for this is "data locality." Databases now need to be able to handle large volumes of concurrent operations, scale horizontally, and offer storage platforms that are more efficient as a result of the development of the internet and the cloud. Therefore, Hadoop is utilized to optimize large data storage volumes for this purpose. The database querying functionality is provided by Hive, which is built on top of Hadoop and makes use of HDFS. The per-

formance of databases is also evaluated by comparing Hadoop and conventional systems. The data, which came from the internet, will be rearranged based on how platforms are used and moved. One of the most widely used Big Data tools, Sqoop (SQL-to-Hadoop) converts data from a non-Hadoop data store into a format that can be easily accessed and utilized by Big Data Hadoop before uploading it into HDFS. The term "ETL," which stands for "Extract, Transform, and Load," best describes this procedure. While it is essential to load data into Big Data Hadoop in order to use Map Reduce, it is just as important to move data out of Big Data Hadoop and into an external data source so that it can be used more effectively in other applications. While it is essential to load data into Big Data Hadoop in order to use Map Reduce, it is just as important to move data out of Big Data Hadoop and into an external data source so that it can be used more effectively in other applications.

BIG DATA

The simplest definition of "Big Data" is a large amount of data stored in various formats. There's more to it than that, though. "Big Data" is the relationship between a system's processing speed and the size of its data. Every day, a large amount of data is created, and it's growing at an even faster rate. The situation is made even worse by the creation of social media and digital media. "Big data" refers to a collection of data sets that are so large and complex that conventional database management tools or data processing applications are unable to process them [1]. There are difficulties associated with the collection, storage, search, sharing, transfer, analysis, and visualization of these data.

BIG DATA'S IMPORTANCE

Real-time data processing has made big data as a tradition to beneficial field. As a result, using big data has many advantages, which are outlined below:

[1]. Recognizing Customers: A lot of information is used to figure out how people act and what they like. It is used by telecom companies to predict customer stir, auto insurers to determine how well their customers drive, and numerous other businesses to predict customer behavior [2] [3].

[2]. Recognizing the Business Procedure: Large amounts of data can be used to comprehend and enhance business operations. Business procedures are continually being improved through the use of big data analytics. Predictions based on social media data, general trends, weather forecasts, and assisting retailers in optimizing stock levels are just a few examples of its applications [1][3].

[3]. Improvements to Performance: Businesses, the government, and individuals can all make use of big data. The fitness band is a good illustration of this, as it enables the user to track and receive fitness data, thereby enhancing the user's fitness levels.

[4]. Enhance Medical Care: Predicting disease patterns with the help of big data analytics makes it easier to find treatments. Using past data analysis, epidemics can be avoided in the future and pre-emptive measures can be taken in advance.

[5]. Enhance Performance in Sports: The use of extensive data analysis has begun in numerous games. Big data analytics can help athletes perform better in a number of ways, including video analytics to track players, sensor technology, and other applications for the IBM Slam Tracker tool in tennis tournaments.

[6]. Enhance Research and Science: The use of extensive data analysis has begun in numerous games. Big data analytics can help athletes perform better in a number of ways, including video analytics to track players, sensor technology, and other applications for the IBM Slam Tracker tool in tennis tournaments. For new technologies like e-commerce applications, the need for faster and more diverse use of large data sets became increasingly pressing over time. Programmers required relational databases, which were more adaptable than SQL databases. That alternative was replaced by NoSQL. Although NoSQL offered an alternative to SQL, it never completely replaced SQL databases. For instance, the positions of retail order manager at a company. Customer, order, and product data would be managed separately by each table in a relational model, and they would be linked together by a unique, common key like a Customer ID or an Order ID. This is great for quickly storing and retrieving data, but it uses a lot of memory. SQL databases can only scale vertically, not horizontally, when you want to add more memory. This means that the

amount of memory you can add is limited by the hardware you have. Vertical scaling ultimately limits your business's capacity for data storage and retrieval as a result.

Because they are non-relational, NoSQL databases do not require connecting tables. Horizontal scaling is made simpler by their high availability and built-in shading features. The workload can be divided across two or more servers, allowing businesses to scale their data horizontally if a single database server is insufficient to store all of their data or handle all of their queries.

HADOOP

Hadoop is an architecture framework that makes it easier for other applications to work with structured and unstructured data by presenting it in a more defined and manageable form. For handling large amounts of data, Hadoop is now widely used. It depends on how the guide and diminishing capabilities concepts are utilized in the guide decrease structure. Even though the Hadoop framework is written in Java, developers can use any language to run their own custom programs. Hadoop can be used to handle data quickly and expand its versatility. As a tool for effectively processing data for a variety of purposes, Hadoop has gained popularity. HDFS, Map Reduce, HBase, Hive, Pig, Spark, and Flume are among the Hadoop services. This simplifies data processing. A Hadoop cluster can be scaled up to hundreds or thousands of nodes to process larger data sets. Apache's Hadoop is an open-source framework for storing, processing, and analysing massive amounts of data. Hadoop, which is not online analytical processing (OLAP), is written in Java. It can be used for offline or batch processing. Facebook, Google, Twitter, LinkedIn, and a lot of other companies use it. In addition, it can be scaled up by simply adding more cluster nodes.

The first Hadoop module

❖ HDFS: Distributed File System for Hadoop. Google released its paper GFS, which served as the foundation for the development of HDFS. It says that the distributed architecture will break the files into blocks and store them in nodes.

❖ Yarn: Job scheduling and cluster management are handled by yet another Resource Negotiator.

❖ Reduce Map: This framework makes it easier for Java programs to use key-value pairs in parallel computations of data. The Map task transforms the input data into a data set that can be calculated using Key value pairs. Reduce task consumes map task output before delivering the desired outcome from reducer.

❖ Common to Hadoop: These Java libraries are utilized by other Hadoop modules and are used to start Hadoop.

Hadoop Architecture.

❖ The Processing/Computation layer (Map Reduce) and

❖ The Storage layers (Hadoop Distributed File System) are at the heart of the Hadoop architecture.

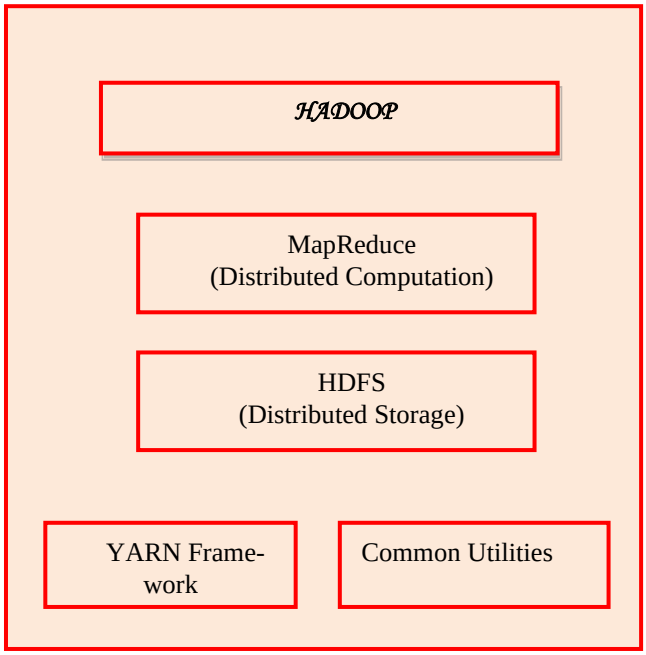


Figure 1: Hadoop Architecture [1]

Map Reduce

Map Reduce is a parallel programming model developed by Google for the efficient, fault-tolerant processing of large amounts of data (multi-terabyte data sets) on large clusters (thousands of nodes) of common hardware. Hadoop, an open-source framework developed by Apache, is the platform on which Map Reduce runs.

Hadoop Distributed File System (HDFS)

The Hadoop Distributed File System (HDFS) is a distributed file system that is built to run on common hardware and is based on the Google File System (GFS). It is a lot like other distributed file systems that are already in use. However, there are significant distinctions from other distributed file systems. It is made to be used on cheap hardware and is very fault-tolerant. It is suitable for applications with large datasets and provides access to application data at a high throughput. The Hadoop framework includes the following two modules in addition to the two core components mentioned earlier:

- **Hadoop Common -** these are Java libraries and utilities that are required by other Hadoop modules.
- **Hadoop YARN** is a framework for scheduling jobs and managing cluster resources.

A distributed file system for Hadoop is the Hadoop Distributed File System (HDFS). The architecture is master/slave. A single Name Node serves as the master in this architecture, while a number of Data Nodes serve as slaves. Name Node and DataNode are both able to run on common computers. HDFS is developed using the Java programming language. Therefore, the NameNode and DataNode software can be easily run on any machine that supports Java.

Name Node

- It is a single HDFS cluster master server.
- Because it is only one node, it might be the cause of a single point failure.
- It performs operations like opening, renaming, and closing files to manage the file system namespace.
- It makes the system's architecture easier to understand.

DataNode

- There are a number of Data Nodes in the HDFS cluster.
- There are multiple data blocks in each DataNode.
- Data is stored in these data blocks.
- DataNode is in charge of responding to read and write requests from clients of the file system.
- On request from the NameNode, it creates, deletes, and replicates blocks.

Job Tracker

- Job Tracker's job is to process the data using NameNode and accept MapReduce jobs from clients.
- NameNode responds by providing Job Tracker with metadata.

Task Tracker

- It serves as Job Tracker's slave node.
- It applies the code to the file after receiving the task and code from Job Tracker. Another name for this process is "Mappers."

MapReduce Layer

When the client application submits the MapReduce job to Job Tracker, the MapReduce layer is created. The request is forwarded to the appropriate Task Trackers by the Job Tracker in response. The TaskTracker occasionally malfunctions or crashes. That portion of the job is rescheduled in such a circumstance.

The Hadoop Distributed File System, or HDFS, is Two parts make up Hadoop: HDFS is used for data storage. For better execution and adaptation to internal failure, it divides the information into blocks and distributes it across numerous hubs. The architecture of HDFS is shown in Figure 2. It is organized as follows:

- **The first name node:** It is responsible for opening, renaming, and closing files and directories. Data Nodes are also mapped to data blocks.
- **Node with secondary name:** In the event of failure, it stores a copy of the NameNode for recovery, though some data may be lost because the copy may not be recent.
-

ode of data: It is used to handle incoming I/O requests from clients. It is utilized for block replication, data creation and deletion, and other tasks.

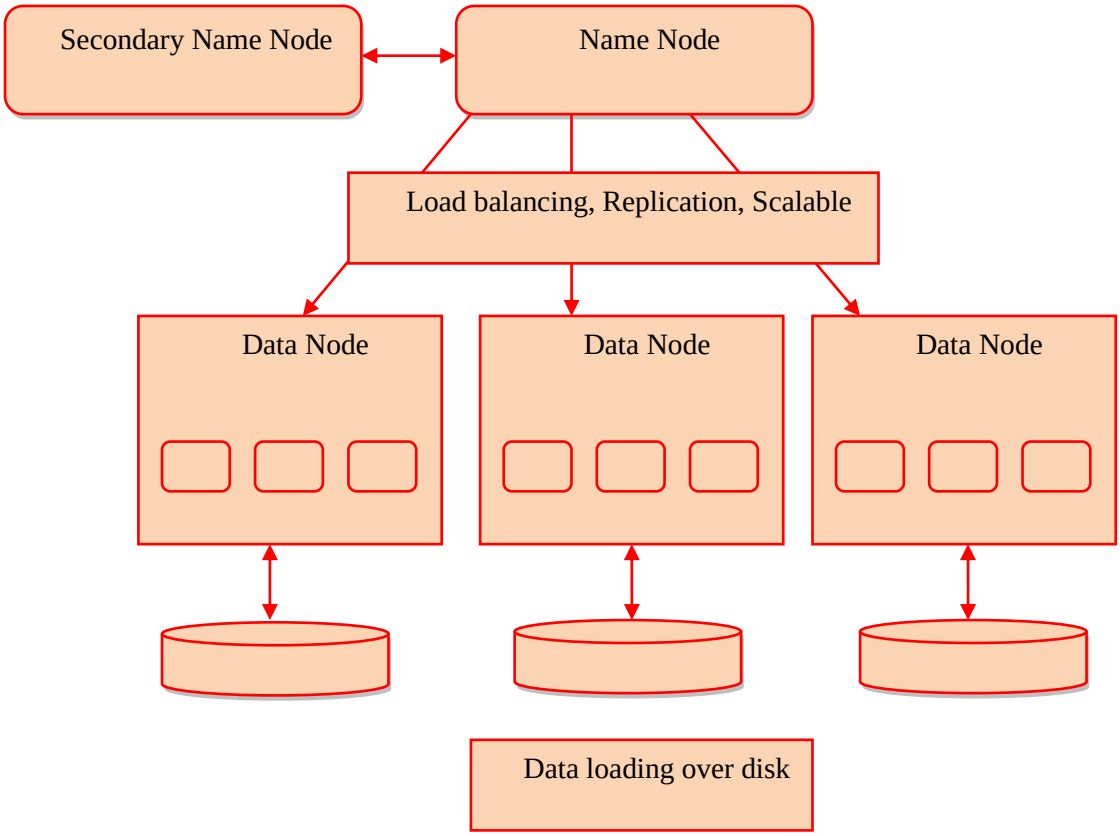


Figure 2: Showing Architecture of HDFS [2]

1) The Map Reduce System: The second component of Hadoop is the Map Reduce. Map Reduce is the heart of Hadoop. It is a Java-based programming technique for distributed computing. Map Reduce incorporates parallelism [1-2]. A Job Tracker is in charge of scheduling and managing all jobs. Each task must be initiated by a Task Tracker, and progress must be reported to the Job Tracker. This paradigm has two responsibilities:

a) Locator: It splits elements of one set of data into (key, value) pairs to create a new set of data.

b) Decreaser: Using the map output as its input, it reduces the set of data tuples. The reduce task comes after the map job.

The map and reduce tasks are sent to the appropriate cluster servers by Hadoop.

Hadoop Operation Modes

❖ ocal/Standalone Mode: After downloading Hadoop onto your system, it is configured in a standalone mode and can be run as a single Java process. This mode allows you to operate your Hadoop cluster in one of the three supported modes.

❖ Pseudo Distributed Mode: A single machine-based distributed simulation Every Hadoop daemon, including HDFS, yarn, MapReduce, and others, will run independently as a Java process. This mode is helpful for growth.

❖ Fully Distributed Mode: In this mode, two or more machines form a cluster and are fully distributed.

SQOOP WORKING

In general, Sqoop's operations are usually simple to perform. Sqoop processed user commands through the command-line interface. Sqoop can also interact with the user in other ways by using Java APIs. Sqoop handles a command from the user when it receives it, and then the command is further processed. The only way Sqoop can import and export data is through user commands; it is unable to create a data aggregation.

The following is how the tool Sqoop works: It begins by parsing the user-supplied arguments through the command-line interface before passing those arguments on to a subsequent stage, where

arguments are induced for a Map-only job. The map issues the command to release multiple mappers based on the number specified by the user in the command line interface when it receives arguments. Based on a key that the user specifies in the command-line interface, each mapper's task receives a portion of the imported data when these jobs are assigned to the Import command. Sqoop makes use of the parallel processing method, in which all mappers are given equal access to the data, in order to improve the process's efficiency. After that, each mapper employs the Java database connection model to establish a unique connection to the database and then fetches individual portions of the Sqoop-assigned data. Depending on the command line arguments, the data is written to HDFS, HBase, or Hive following retrieval. Consequently, the Sqoop import procedure has been completed. The exporting of data process in Sqoop is comparable; a set of files from the Hadoop distributed system can be transferred back to the relational database management system with the help of the Sqoop export tool. During the import process, records are the files that are used as input. A user's job is mapped into a Map Task when it is submitted, bringing the data files from the Hadoop data storage. After that, these files can be exported to any structured data repository that is based on a relational database management system like MySQL, SQL Server, Oracle, or another similar system. NoSQL has solved the most challenging problem with RDBMSs—vertical scaling—by introducing horizontal database scaling. Data structures like key-value pairs, document stores, column stores, and graph stores are used in NoSQL databases. The data structure to use is determined by the problem that needs to be solved.

NoSQL database types include:

❖ Key-Value storage associative arrays are the fundamental data model of key-value store toNoSQL [11] databases. Each element in an associative array is a key-value pair. Key acts as an index for the value and represents the value. One requirement for keys is that each named key in an array must always be unique and cannot be duplicated. Numerous data models are implemented as extensions to this model because it is the easiest to comprehend.
Ex. Dynamo 2, Aerospike, ArangoDB, Couch base

❖ **Store of documents: -** Document-based data collection constitutes this data model. This database stores all of its data in document format. The JSON (Java Script object Notation) format is used to store data in the document [12]. Each document in the database has a single, unique ID that is used to identify the document. Keys are used to search a database for data. This kind of database is made up of a collection that contains numerous documents. Collections represent tables, whereas documents represent records in RDBMS [4].
Ex. CosmoDB 3, CouchDB, and MongoDB

❖ **Store graph: -** The majority of the data that can be represented by the graph are stored in this database. Similar to a relation in a RDBMS, it has nodes and edges that keep data connected. Edges are used to link related data together.Ex. Virtuoso 4, Neo4J, MarkLogic, and OrientDB.

❖ Column storage RDBMS and this database are almost identical. Column-based storage is used to store data. Similar to tables, which store data in rows and columns, it is very similar.Ex. Cassandra, Druid, HBase, Vertical, and SAP HANA are just a few of the popular NoSQL applications [11].

Other appealing features include horizontal scaling, replication, and de-normalization. When the system asks for more storage, horizontal scaling means that it can be achieved by simply adding a server to the cluster. Rather than being stored on a single machine, data in horizontal scaling are stored on multiple machines, which is also known as distributed architecture [13]. Replication refers to the addition of additional servers to the cluster that will host the same data and can be referred to as storage backups. It contributes to an increase in data availability while also increasing redundancy. It helps with issues with data loss. Denormalization is the process of getting rid of dependency and redundant data in tables. Constraints are used to implement the complicated idea of normalization in SQL databases.

NoSQL

A database that offers a method for storing and retrieving data is known as NoSQL, which originally stood for non-relational or non-SQL. This data is modelled in a different way than in relational databases, which use tabular relations. These kinds of databases first appeared in the late 1960s, but they didn't get the name "NoSQL" until they became increasingly popular in the early 21st century. Real-time web applications and big data make use of

NoSQL databases, and their use is growing. Because they may support query languages that are similar to SQL, some NoSQL systems are also referred to as "Not only SQL." The design of a NoSQL database is straightforward, horizontal scaling to clusters of machines is made simpler, and availability control is more precise. Because the data structures used by NoSQL databases differ from those used by relational databases by default, NoSQL databases can perform some operations more quickly. A given NoSQL database's suitability depends on the problem it should solve. NoSQL databases' data structures are sometimes regarded as more adaptable than relational database tables. Consistency is sacrificed by many NoSQL stores for speed, availability, and partition tolerance. The use of low-level query languages, the absence of standardized interfaces, and significant prior investments in existing relational databases are all obstacles to the widespread adoption of NoSQL stores. True ACID (Atomicity, Consistency, Isolation, and Durability) transactions are missing from the majority of NoSQL stores; however, a few databases, including Mark Logic, Aerospike, FairCom c-treeACE, Google Spanner (which is technically a NewSQL database), Symas LMDB, and OrientDB, have incorporated them into the core of their designs. The majority of NoSQL databases provide a concept called "eventual consistency," which states that changes made to the database are sent to all nodes. As a result, queries for data may not immediately return updated data or may result in reading data that isn't accurate, a problem called "stale reads." Additionally, lost writes and other forms of data loss may occur in some NoSQL systems. To prevent data loss, some NoSQL systems offer concepts like write-ahead logging. Consistency of the data presents even huge problems when dealing with distributed transaction processing across multiple databases. This is challenging for relational and NoSQL databases alike. Referential integrity constraints cannot span databases, even in modern relational databases. For distributed transaction processing, very few systems maintain both X/Open XA standards and ACID transactions.

Characteristics of NoSQL:

- ❖ It doesn't use the relational database model at all.
- ❖ It never provides a table with records with a flat fixed column.
- ❖ It lacks schema.
- ❖ It creates a zero-share environment.
- ❖ It can be scaled.
- ❖ Hardware costs little.
- ❖ It speeds up performance.
- ❖ An example: Cassandra, MongoDB.

Benefits of NoSQL:

Working with NoSQL databases like Cas-Sandra and MongoDB has numerous benefits. High availability and high scalability are the primary benefits.

- ❖ **High scalability:** for horizontal scaling, NoSQL databases use sharding. Sharding is the process of partitioning data and distributing it across multiple machines while maintaining the data's order. When a machine is vertically scaled, more machines are added to handle the data, whereas when a machine is horizontally scaled, more machines are added to handle the data. Vertical scaling is difficult to implement, whereas horizontal scaling is simple. MongoDB, Cassandra, and other examples of horizontal scaling databases are examples. Due to its scalability, NoSQL can handle a lot of data. As the data grows, NoSQL scales itself to handle it efficiently.
- ❖ **High availability:** Because data replicates itself to the previous consistent state in the event of a failure, the auto replication feature of NoSQL databases is highly available.

Negative aspects of NoSQL:

The disadvantages of NoSQL are as follows:

1. **Narrow focus:** Because it is primarily intended for storage but offers very little functionality, NoSQL databases have a very narrow focus. In the area of Transaction Management, relational databases are superior to NoSQL.
2. **Open-source:** NoSQL is a database that is free to use. There is currently no reliable NoSQL standard. To put it another way, two database systems are probably not equal.
3. **Problem with management:** The goal of big data tools is to make it as easy as possible to manage a lot of data. But it's not that simple. The complexity of NoSQL's data management is much higher than that of a traditional database. Particularly, NoSQL has a bad reputation for being hard to set up and even harder to keep up with on a daily basis.

4. **The GUI mode** tools necessary to access the database are not readily available on the market.
5. **Backup:** Some NoSQL databases, like MongoDB, have major problems with backup. There is no consistent method for backing up data in MongoDB.
6. **Documents with a large file size are stored in JSON format** by some database systems like CouchDB and MongoDB. This indicates that documents are quite large (Big Data, network bandwidth, speed), and since descriptive key names increase document size, they actually impair.

NoSQL database types include:

The following are examples of NoSQL databases, as well as the database system that belongs to that category:

- Databases for graphs: Neo4j
- Neptune from Amazon Store of key value: Redis, Memcached, and Coherence
- Tabular: Big Table, HBase, and Accumulo
- Document-based: MongoDB, CouchDB, and Cloudant

When is NoSQL appropriate?

- When it is necessary to store and retrieve a large amount of data.
- The connection between the data you store is not particularly significant
- The data is not structured and changes over time.
- At database level
- Constraints and Joins are not required.

In order to handle the data, you need to scale the database frequently because the data keeps growing.

NoSQL vs. SQL

When discussing NoSQL, the distinction between NoSQL and SQL Structured query language (SQL) is frequently made. Understanding the history of SQL, a programming language used to retrieve specific data from a database, may be helpful in clarifying the distinction between NoSQL and SQL.

Companies used a hierarchical database system with data tables that had a tree-like structure prior to the introduction of relational databases. Users were able to organize huge amounts of data thanks to these early database management systems (DBMS). However, they were difficult to use, often exclusive to a single application, and limited in their ability to uncover data. Relational database management systems, which organize data into tables, eventually emerged as a result of these limitations. SQL provided an interface for working with relational data, making it possible for analysts to join tables by merging fields that are similar. For new technologies like e-commerce applications, the need for faster and more diverse use of large data sets became increasingly pressing over time. Programmers required relational databases, which were more adaptable than SQL databases. That alternative was replaced by NoSQL. Although NoSQL offered an alternative to SQL, it never completely replaced SQL databases. Take, for instance, the position of retail order manager at a company. Customer, order, and product data would be managed separately by each table in a relational model, and they would be linked together by a unique, common key like a Customer ID or an Order ID. This is great for quickly storing and retrieving data, but it uses a lot of memory. SQL databases can only scale vertically, not horizontally, when you want to add more memory. This means that the amount of memory you can add is limited by the hardware you have. Vertical scaling ultimately limits your business's capacity for data storage and retrieval as a result. Because they are non-relational, NoSQL databases do not require connecting tables. Horizontal scaling is made simpler by their high availability and built-in sharding features. The workload can be divided across two or more servers, allowing businesses to scale their data horizontally if a single database server is insufficient to store all of their data or handle all of their queries.

RELATED WORK

NoSQL document databases have grown in acceptance and use across numerous industries. The effectiveness of NoSQL databases, including some NoSQL document databases, has been compared and evaluated in numerous studies. Only a small number of scholarly articles have been published, and the field of NoSQL databases study is currently developing. As a result, there aren't many studies that compare and assess the performance of solely NoSQL document databases. Pig, Hive, and MongoDB, three NoSQL databases, were compared in [5]. In this study, YCSB was used to do a performance evaluation with just three workloads and

100% insert, 100% read, and 100% update, respectively. They used just one and four threads and changed the number of records by 10,000, 100,000, and 1,000,000. Ten runs were completed for each workload, and the average of these numbers served as the experiment's conclusion. The tests were run in an Ubuntu Linux system. The authors came to the conclusion that while dealing with a load of 10,000 records, NOSQL and MongoDB perform similarly. However, MongoDB outperforms NOSQL as the workload increases. It performs additional types of operations than the study that was reported, including read-modify-write, scanning, and other operations. There were also variations in the amount of threads and records used. 100,000, 1,000, and 10,000,000 records were used. This will used one, three, and six threads to vary the amount of threads. The tests were run on a Windows system, and each workload was run three times. The average of these numbers served as the experiment's final finding. Lidong Wang and others [1] have looked into machine learning, which is a way for artificial intelligence to find knowledge so that smart decisions can be made. The creation of value and scientific discoveries are greatly influenced by big data. The main technologies of big data, machine learning methods, and some big data applications are all discussed in this paper. The difficulties of applying machine learning to big data are discussed. In big data machine learning, new approaches and technological advancements are also presented. JunfeiQiu and co. 2] have discovered that big data is without a doubt expanding rapidly across all areas of science and engineering. Even though these enormous amounts of data have a lot of potential, they need new ways of thinking and new ways to learn in order to fully understand them. The most recent developments in research on machine learning for big data processing are reviewed in this paper. Representation learning, deep learning, distributed and parallel learning, transfer learning, active learning, and kernel-based learning are just a few of the promising machine learning techniques that they discussed. The challenges and potential solutions of machine learning for big data are the primary topics of discussion and analysis in the following section. After that, they look into how big data processing signal processing and machine learning are closely related. In conclusion, this paper will discuss a number of pending issues and research trends. Saikat Das et al.[3] to describe data that is so large, fast, or complex that it is difficult or impossible to process it using conventional methods. Due to the extensive use of data, the concept of big data gained traction at the beginning of the 2000s. In the past, big data used 3Vs, but now it uses 5Vs, which stand for volume, velocity, variety, veracity, and value. Despite the fact that these massive amounts of data have a lot of potential. The artificial intelligence technique of discovering knowledge for making intelligent decisions is known as machine learning. The main technologies of big data (case studies) and some applications of machine learning in big data are introduced in this paper. In their research, Junfei-Qiuet al.[4] discovered that machine learning has widespread implementations and applications in numerous facets of our lives. However, as the big data era approaches, some conventional machine learning methods are unable to meet the demands of real-time data processing. As a result, machine learning must rethink itself in light of big data. They review recent research on machine learning for big data processing in this article. First, a discussion of big data is given, and then the new features of machine learning in the big data context are looked at. After that, they offer a machine learning-based reference framework for dealing with big data. Finally, a number of open research questions and challenges are addressed.This paper by Alexandra L'Heureuxet al.[5] compiles, summarizes, and organizes Big Data-based machine learning challenges. In contrast to other studies that look at challenges, this one focuses on the relationship between challenges and the Big Data Vs or dimensions that caused the problem: quantity, speed, variety, or veracity. In addition, emerging machine learning approaches and methods are discussed in terms of their ability to deal with a variety of obstacles, with the ultimate goal of assisting practitioners in selecting suitable solutions for their use cases. The issues and solutions are then arranged in a matrix at the end. This paper provides a perspective on the subject, identifies research gaps and opportunities, and provides a solid foundation and encouragement for further machine learning with Big Data research through this process. SuryabhanPratap Singh et al.[6] All areas of computer research are increasingly interested in the topic of big data. Innovative approaches to learning that address a variety of issues are specifically communicated through learning techniques, which

highlight crucial opportunities and transformative potential in a number of thought domains. The most recent advancements in big data machine learning are examined in this paper. In addition, it talk about novel approaches to machine learning, with a focus on recently proposed learning methods like representation learning, transfer learning, deep learning, and active learning. Specifically from the point of view of big data, a comparison is made between the proposed solutions and the existing machine learning tools, as well as an analysis and discussion of the issues they raise. According to Lidong Wang et al.[7], machine learning is an artificial intelligence technique for acquiring knowledge for autonomous decision-making. The creation of value and scientific discoveries are greatly influenced by big data. The main Big Data technologies, machine learning methods, and some Big Data applications are all discussed in this paper. The difficulties associated with applying machine learning to Big Data are discussed. There are also some brand-new approaches and technological advancements for machine learning in Big Data. Using a key-word search, Isaac Kofi Ntiet al.[8] present a comprehensive mini-literature review of ML in BDA; There were a total of 1512 articles that had been published. Based on the study's proposed novel taxonomy, 140 articles were selected for consideration. According to the findings of the study, ensemble learning techniques, decision trees, deep neural networks, artificial neural networks, support vector machines, and artificial neural networks account for 15%, 14%, and 11% of BDA applications, respectively. The opportunities for future research, challenges, and related application fields are all described in detail. Mehdi Assefiet al.[9] The research community has utilized artificial intelligence, and machine learning in particular, in numerous ways to transform a variety of diverse and even heterogeneous data sources into high-quality facts and knowledge, providing premier capabilities for accurate pattern discovery. However, it takes a lot of logical and physical resources, such as data file space, CPU, and memory, and it is computationally expensive to apply machine learning strategies to large and complex datasets. As the daily amount of data generated exceeds a quintillion bytes, an advanced platform for effective big data analytics is becoming increasingly important. Regression, classification, dimension reduction, clustering, and rule extraction are just a few of the machine learning techniques supported by Apache Spark MLlib, one of the most well-known platforms for big data analysis. As an open-source, distributed, scalable, and platform-independent machine learning library, the expanding body of Apache Spark MLlib 2.0 is the subject of our computational investigation in this contribution. In particular, they conduct a number of real-world machine learning experiments to investigate the platform's qualitative and quantitative characteristics. Furthermore, they offer suggestions for future work and draw attention to current trends in big data machine learning research. Shao Chunziet al.[10] use Zhuhai College of Jilin University as the subject of their empirical research on the application of big data to online language learning. The software EVIEWS is used to analyze important factors that affect online learning and help students predicts their final grades. This allows students to benefit from learning supervision and feedback and improves learning efficiency. There are also suggestions for overcoming the current challenges associated with integrating online learning and big data. Carson K. Leung et al.[11] The current technological era sees the generation and collection of enormous amounts of big data from a vast array of rich data sources. In the sense that some of these big data are precise while others are imprecise and uncertain, their veracity can vary. There is valuable knowledge and useful information embedded in these big data that can be discovered. Health-care and epidemiological data, such as patient records from epidemics like the corona virus disease 2019 (COVID-19), are examples of big data. Data science methods like machine learning, data mining, and online analytical processing (OLAP) enable researchers, epidemiologists, and policymakers to gain a deeper comprehension of the disease, which may inspire them to devise strategies for its detection, management, and prevention. For the purpose of processing and evaluating COVID-19 epidemiological data, they present a machine learning and big data analytical tool in this paper. To be more specific, the tool makes excellent use of taxonomy and OLAP to transform a few particular attributes into a few generalized attributes that are necessary for efficient big data analytics. ManojMuniswamaiahet al.[12] have investigated the fact that, in order to run machine learning algorithms for analysis, data scientists must utilize data from a variety of sources. The results of data sci-

ence depend on the quality of the data that has been extracted. A federated query processing framework that extracts data from multiple data sources and stores the resulting datasets in a common in-memory data format is the focus of this study. Without having to convert the data into their native data format, this makes it easier for data scientists to use various data engines to carry out their analysis and execute machine learning algorithms.MilindBhandarkar et al.[13] have researched the design philosophy of Hadoop and have written about how to create Hadoop applications and higher-level application frameworks that can crunch several terabytes of data using anywhere from four to 4,000 computers. They will talk about common issues that come up when trying to get the most out of the performance of a Hadoop application. Additionally, they will talk about a number of Hadoop-based frameworks and applications that improve application performance and programmer productivity. According to GurjitsinghBhathalet al.[14], the low cost, scalable data processing capabilities, and high fault tolerance of big data technology—also known as Apache Hadoop are driving its rapid growth. Organizations began utilizing this technology following the release of Hadoop 1.0, but encountered some difficulties. At this point, the expectations of various organizations looking for an alternative to Apache Hadoop were not met by Hadoop Framework in time. As a result, Hadoop vendors improved the enterprise-ready open source core Apache Hadoop release. A single product was presented by Hadoop Distributions: an added apache repository, new tools, security, and central administration so that the organizations didn't have to spend time putting all of these essentials together into a single piece of functionality. Cloud era CDH, Horton works HDP, and MapR M5 are the Hadoop distributions that are currently gaining the most traction in the market. This paper compares and contrasts the different Hadoop vendor distributions based on their functionality and contribution to the big data market. VaishaliSontakkeet al.[15] In this paper, They propose an optimized HPMR (Hadoop Map Reduce) model that balances CPU and I/O system performance while maximizing task memory utilization. Similar to any other Hadoop model, HPMR optimizes all three phases-maps, shuffle, and reduce-in addition to the three phases of Hadoop. In addition, HPMR uses dynamic terminology to optimize the memory model, and dual operation is used to optimize input and output. Also, it uses the WordCount application on Wikipedia data with sizes of 128 Mb, 256 Mb, 512 Mb, 1 GB, and 2 GB to see how well our model worked. Our model outperforms the previous one by nearly 30%, according to the comparison analysis. This project uses the Hadoop Map Reduce framework on AWS, a cloud platform, to analyze YouTube data, according to Prathyusha Rani Merla and others [16]. On AWS (Amazon Web Services), a private cloud, a Hadoop multi-node cluster is set up. I have created EC2 instances in AWS with one name node and five data nodes. The HDFS (Hadoop Distributed File System) stores the video statistics that are obtained from the API, and the MapReduce system processes the data. According to ChitreshVerma et al.[17], Big Data is a substantial dataset that presents the characteristics of variety, velocity, and volume in an OR relationship. If it is not accessible for strategic analysis and application, Big Data as a large dataset is of no use. The technological landscape is full of hardware and software solutions that make it possible to capture, store, and then analyze Big Data. One of them is Hadoop and the technology that goes with it. The software framework for processing large amounts of data is called Hadoop. There are four main modules in it. HadoopMapReduce, YARN, Hadoop Common, and Hadoop Distributed File System (HDFS) are these modules. Under Job Tracker's direction, Hadoop Map Reduce breaks down large problems into smaller sub problems. A Big Data representation for grade analytics in an educational setting is proposed in this paper. The research and the experiments can be carried out using R or AWS, Amazon's cloud infrastructure. Qi Wang et al. [18] In this paper, They propose a new scheduling algorithm for Hadoop YARN called PFT. This algorithm uses a multi-level queue, a time factor, a job urgency factor, and a domain resource ratio to effectively trade off fairness and performance while also reducing the duration of MapReduce jobs. In Hadoop YARN, They implement PFT as a pluggable scheduler. According to the findings of the experiments, PFT can reduce the duration of MapReduce jobs by 34.53 percent, increase CPU utilization by 34.93%, and increase memory utilization by 38.98%.

According to Alexey Siretskiyet al.[19], Hadoop is a useful e-Science framework for scalable distributed data analysis. Next-generation sequencing in mo-

lecular biology generates a lot of data and necessitates adaptable frameworks for building analysis pipelines. By adding functionality to count genes mapped by short reads and massively parallel versions of short read quality assessment, they bring the well-known HTSeq package into Hadoop. They evaluate the components' performance in two distinct execution environments with the help of the Hadoop-streaming library, which enables them to run on both standard Linux systems and Hadoop. Hadoop cluster in a private cloud and a single node in a computational cluster. They demonstrate the improved runtime performance of our developed methods by comparing the implementations with Apache Pig. To make user interaction simpler, they also inject components into the Cloud gene graphical platform.

The Apache Hadoop project's core component is the Hadoop Distributed File System (HDFS), according to Talluri Lakshmi Siva Rama Krishna et al.[20]. The relevant data is stored at the nodes in HDFS, where the computation takes place. MapReduce, a parallel computational paradigm, were also implemented by Hadoop. By taking into account both small and large files, they have measured the performance of read and write operations in HDFS in this paper. They have utilized a five-node Hadoop cluster for performance evaluation. According to the findings, HDFS performs admirably for files that are larger than the default block size, but it performs poorly for files that are smaller than the default block size.

The Apache Software Foundation has released Apache Hadoop 3.0.0-alpha3, according to Rohit G. Masuret al.[21]. The focus of this research paper is on comparing the new version of Hadoop to the older version, Apache Hadoop 2.7.3. On the two versions of Apache Hadoop that are installed on distinct isolated Virtual Machines, various benchmark tests that are included in the Apache Hadoop distribution are used to conduct the performance analysis. This paper provides an in-depth account of the test runs' outcomes.

Yinwei Li et al.[22]'s goal in this article is to create an effective big data processing platform by utilizing the Sqoop data synchronization technology, Hive, flume data collection technology, and Hadoop big data storage architecture. Map Reduce programming is used to implement the traditional data mining algorithm, and the Hadoop platform's implementation of the data mining algorithm is studied primarily for its execution efficiency and scalability. In order to test and validate its effect on the Hadoop platform, they select the data clustering task in data mining as an example and write a Map Reduce version of it. The use of Hadoop distributed systems for data mining tasks has a good acceleration ratio and efficiency, and the extended performance analysis of computing power also demonstrates that it has great potential. This conclusion is reached through comparative experiments of various cluster sizes and data sizes.

This article by Ankit Shah et al.[23] describes how data locality in Hadoop maps data blocks to processes in the same node. However, when dealing with Big Data, it is frequently necessary to map data blocks to processes in multiple nodes. Hadoop has the capability to copy the data block where mappers are running to deal with this. Due to I/O delays or network congestions, this results in significant performance degradation, particularly on heterogeneous clusters. By dividing the total number of nodes into two categories like: homogeneous versus heterogeneous, or nodes with high performance versus those with low performance. They are able to place data blocks precisely where they want our data to be placed for processing thanks to this policy, which helps to achieve better load rearrangement among the nodes.

Yinan Tang et al.[24] propose the OE-Hadoop, a modified Hadoop built by co-designing Hadoop with a hybrid optical and electrical data center network, as a way to boost the performance of Hadoop applications. They change MapReduce jobs' pipeline-based replication to optical multicast for Hadoop. They construct a reconfigurable optical multicast system for the DCN architecture to accommodate multicast traffic. In the data center, a software-defined networking controller is used to adjust the DCN architecture and exchange data with the application layer. A new algorithm for scheduling multicast requests is presented and implemented in the controller to speed up MapReduce jobs. In order to evaluate the control overhead and demonstrate the OEHadoop's viability, They construct a modest prototype. Our multicast requests scheduling algorithm outperforms related state-of-the-art solutions in a simulation conducted at the scale of real DCN. Additionally, it demonstrates that the average speed of MapReduce jobs in our Hadoop is approximately two times that of native Hadoop.

Yuanyuan Wu et al.[25] came up with Hadoop-EDF, a distributed signal processing tool that uses Hadoop Map Reduce to load EDF data in parallel. Since EDF data can be processed in parallel, Hadoop-EDF makes use of a robust data partition algorithm. Utilizing two datasets from the National Sleep Research Resource and carrying out experiments on Amazon Web Service clusters, they assess the scalability and performance of Hadoop-EDF. On a 20-node cluster, Hadoop-EDF performed approximately 26 times faster than the sequential processing of 200 small-size files and 47 times faster than the sequential processing of 200 large-size files, respectively. The findings indicate that large EDF files can be processed more efficiently with Hadoop-EDF.

Apache Hadoop randomly distributes the imported data on data nodes in order to improve the efficiency of parallel and distributed computing, according to Jen-Chun Hsu et al.[26]. The general analysis of data benefits from this mechanism. Similar to Apache Sqoop, each table is divided into four parts and distributed at random on data nodes. However, this method of placing data still raises concerns regarding the performance of the database. For better data placement, a correlation-aware Sqoop method called CA Sqoop is proposed in this paper. By gathering related data as close to each other as possible, the network's cost of data transformation can be reduced, and database utilization performance can be improved. For improved data locality and query efficiency, the CA Sqoop also takes into account the size and correlation of the tables. Data locality in CA Sqoop is two times better than in the original Apache Sqoop, according to simulation results.

According to P.V. SaiCharanet al.[27], the most common issue in today's digital age is being surrounded by e-commerce and social networking sites that offer accurate and timely recommendations based on user interests. It will be simpler to provide recommendations to those users based on their previous activity on that particular website when they have user-related data. However, a cold start problem occurs when a user or item is unfamiliar with the website and there is no information about them. One of the distributed big data processing frameworks known as Hadoop offers a practical solution to these cold start issues in real-time situations. In this case, they built a model using Sqoop and Hive to address and resolve specific cold start issues in recommendation systems.

S. Thilagavathiet al.[28] use digitized India to connect different countries to the fast Internet. As a result, it is used to reduce wrongdoing, manual labor, and documentation while also creating work opportunities. People are having a lot of problems these days when they forget to bring their driver's license. The proposed system also combines the driver's license with the data skew to reduce this problem. The MapReduce Counters can be used to connect the driving license and data skew information points of interest. As a result, it accumulated during the Map and Reduce stages. It is used to make an apparatus that deals with the process of getting a permit by using unique, recognizable evidence that is related to each person. It encourages the customer to move between locations without a permit. As a result, the proposed framework will enable widespread digitization of information for quick and easy access throughout India. Sqoop is a device that is expected to exchange data with social databases and Hadoop. It imports and processes data using MapReduce, which facilitates parallel tasks and also allows for adjustment of non-basic disappointment. As a result of concurrent activities, the amount of time needed to exchange information is significantly reduced.

When They hear the term "BIG DATA" used by Manika Manwalet al.[29], many assumptions and questions arise, such as "when did this term Big Data come into the picture?" Which is it? What is required from it? How does it help? Data has always been stored, from the ancient to the modern era. There are numerous events that can be considered. Numerous organizations, including Twitter, Facebook, LinkedIn, and others, produce an enormous amount of data today. As a result, Big Data enables the storage, management, and processing of a vast amount of data in a variety of formats. By accounting for the properties of volume, value, variety, veracity, and velocity, big data ensure that the knowledge and accuracy derived from the data are generated rapidly and provide organizations, researchers, and consumers with convenience. The introduction of this study explains what Big Data is, as well as its characteristics and classification. The architecture of Hadoop and its components, such as HDFS, Map Reduce, Pig, Hive, HBase, and Sqoop, are then discussed in this study. A global valuation is provided in the section that concludes.

According to AleksandarTunji and colleagues [30], the data ingestion procedure appears straightforward. However, the ingestion process is complicated by the presence of structured, semi-structured, and unstructured data, which can come from a variety of database systems. In most cases, simply storing everything is not sufficient. Users must be able to quickly access and manipulate data when it is stored. In the big data ecosystem, numerous ingestion-specific solutions are available. An actual system for importing data into a Hive database from MSSQL, MySQL, and Postgres will be described in this paper. The process begins with the creation of tables containing the relevant metadata, continues with the ingestion procedure, and concludes with a description of the automated procedure. The Cloud era Hue web user interface and the open-source Sqoop implementation will be discussed.

ZhaiWeixinet al.[31]'s paper is a first look at how the non-sql databases HBase and GeoSOT grids can be used to manage spatial data in the big data era of spatial databases. The traditional approaches to overcoming the challenge focus primarily on expanding downward; additionally, it costs a lot and takes up a lot of storage space and time. This study combines the global subdivision grid and the distributed database HBase for data management. The grid's geocodes can be used as the keyvalue in HBase and can show a spatial object's position. The design shifts from an object-oriented database to a geography-oriented database, taking into account more geographical characteristics. In addition, a comparison experiment based on our theory shows that, when dealing with large amounts of data, our new method outperforms geographical queries.

This paper presents the DONSL (data server of non-SQL-query) architecture, which employs Web-tier object modeling to address the aforementioned issues. Through simplified use of query logic property, this architecture ensures transaction processing and performance between Web-tier and DBMS. Additionally, this new conceptual framework simplifies tier and eliminates DAO and entity, resolving enterprise site implementation issues.

A system for automatic crime reporting and immediate response is proposed in this paper by KefilweMkhwanazi et al.[33]. Based on system integration, the system combines Raspberry Pi, Microsoft IoT, a mobile application, and a web application. Not only does an automatic crime reporting and immediate response system guarantee the safety and secrecy of informants, but it also prevents cases and reports from being deleted or removed and ensures the integrity of information. Informers who want to report crimes anonymously or by providing their information via mobile phones could greatly benefit from the automatic crime reporting system. It also helps the police provide adequate security and reduces the amount of manual work involved in reporting crimes. The non-Structured Query Language (non-SQL) database used by this system speeds up the retrieval of video, audio, and image evidence from crimes. The quality of the photos and videos used as evidence never deteriorates; they are always accurate to 97 percent when automatically detecting a location and to 99.9 percent when retrieving and saving them to the database.

According to Haller Piroskaet al.[34], smart environments need the ability to store and analyze a large amount of data generated by a variety of nodes, including high-resolution cameras and sensors. Current video monitoring systems store the data streams in files and cannot be synchronized with external events, add markers, or search the content of internal data. Events and multimedia streams, for example, can be saved, searched, and replayed using the presented distributed framework. Data persistency is provided by a document-oriented database in the proposed storage layer. Using the framework that has been implemented, a large number of performance tests are presented.

The mixed method of extracting OLAP cubes from NoSQL data sources and creating a standard data entry system for both NoSQL and SQL data sources has been proposed by BrozolalMondolet al.[35]. There are four stages in our proposed algorithm: MapReduce (LSHMR) includes shingling, chunking, minhashing, and locality-sensitive hashing. Upon entering NoSQL directories, each stage uses an efficient procedure. In comparison to other algorithms, our proposed method and algorithm exhibit productivity gains of over 75%.

Mohammed Al-Katebet al.[36] discuss the improvements made at the infrastructure and algorithmic levels to make analytical workloads scale while paying close attention to the quality of service guarantees provided by various technologies. Before discussing the shift toward distributed analytics over non-SQL infrastructures, they begin with an overview of traditional centralized analyti-

cal methods. Systems that integrate analytical functionality inside, above or adjacent to SQL engines are in contrast to these methods. They also look at how the virtualization capabilities of Cloud platforms make it easier and less expensive for end users to apply these new analytical methods to their data. They end with a vision for the near future and the lessons learned.

Zhiwei Liang et al.'s [37] secure XHTML-structured web solution for high-quality EHRs includes ICD-11 MMS-compliant coding and terming input, electronically signed legally binding input, sensitive masking input encrypted with a public key, and de-identified data exchanges for research. Such a system is advantageous because it is adaptable to application requirements and rigorous in its collection of genuine, high-quality data for scientific research and decision-making; and can be easily incorporated into a distributed system. Traditional medicine data will be integrated into the entire EHR system, generating real-world evidence for the role and efficacy of TCM in patient care and collecting broad and comprehensive healthcare data.

METHODOLOGY

The system architecture's methodological approach is based on horizontal scaling and will operate in a multi-persistence environment. As depicted in the figure3, horizontal scalability is also referred to as scaling-out-upgrade by adding new machines. A new machine is added to the distributed system as each machine runs out of capacity. The capacity of the existing machines is not increased when horizontal scaling is used; instead, the cluster gets more nodes. Cassandra, MongoDB, HBase [4], and Dynamo are among the NOSQL systems with horizontal scalability. Newer relational SQL systems also offer horizontal scalability to compete with NoSQL [5] systems. For instance, Facebook stores user data primarily in a MySQL database that is shared across over 4,000 MySQL server instances. As depicted in the figure, horizontally scalable systems are more cost-effective than vertically scalable systems.

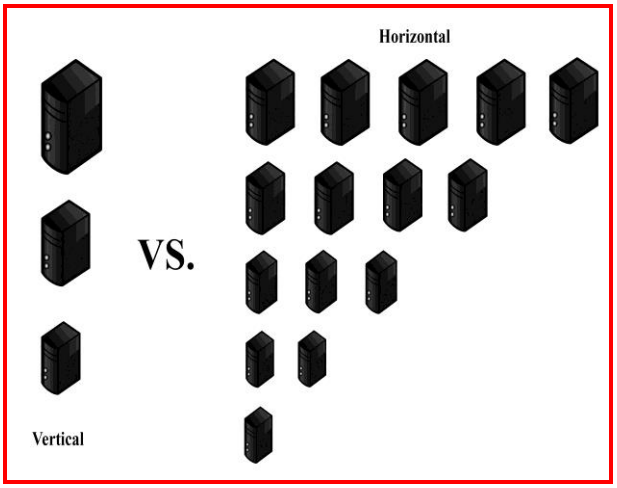


Figure 3: Vertical Scalability vs. Horizontal Scalability [3]

As depicted in Figure 3, vertical scalability is also referred to as scaling-up-upgrade because it involves increasing the capacity of existing machines by replacing them with more powerful boxes. By increasing the RAM, processing power, and storage disks of individual nodes in the system, this can be accomplished. SQL-based relational system of the past favor is on vertical scaling.

Vertical scalability, on the other hand, has the drawback of increasing hardware costs exponentially.

Figure 4 shows that all IoT devices that are edge nodes are labeled "E," "Sub Fog Server," "Center for Server," or "main Fog Server," and "Cloud Server" are labeled "C." The following explains how the diagram above works: -

End Nodes: Data is processed and transmitted directly to the devices by edge nodes. Without considering the significance of the data, edge nodes transmit all sensed, captured, or generated data. In the diagram above, the "E" edge computer or edge node is connected to the device's network to process data and send it to the cloud in real time. Local training of the edge devices is the local collection and processing of data by edge computers. The processed data is sent to the Sub Fog Server "F" for further processing after the local aggregation is implied on the data collected by the edge nodes.

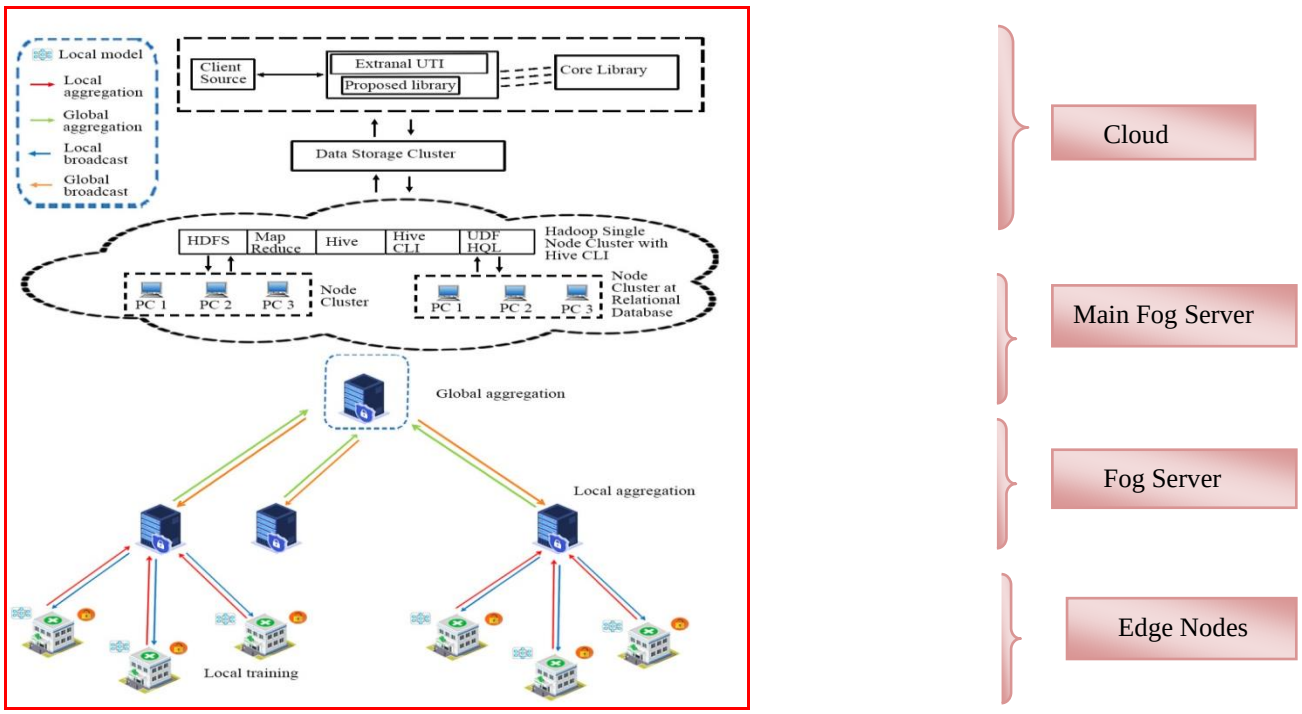


Figure 4: Proposed Architecture of MPI model [4]

Local Education: By setting the parameters for collecting the necessary data from the various resources, local training is carried out at the edge devices marked "E." Local training that takes place close to or at the user's or data source's physical location. Physical, local computational resources are used for machine vision-related computing.

Aggregation, both local and global: Local and global aggregations are handled by cloud C and fog node F, respectively. C' is able to obtain processed aggregation results from edge devices that provide processed data services in addition to global results. A centralized authority only aggregates the model's local parameters; the new global parameters are then computed and sent back to edge devices.

At each round, variable node "C" is considered a global aggregator, while "F," the local aggregator, completes each local aggregation following the completion of other local aggregations to save the central authority unnecessary work.

Server of Fog: Data is collected at the edge, and unprocessed data is sent to the cloud. The solution for this issue is fog computing. Fog networking and fog computing are other names for fog computing. It is a decentralized computing infrastructure in which applications and data storage reside in the cloud and data source. Fog Computing is a local architecture for the efficient use of edge devices for computation, storage, and communication. The term "edge device" refers to any device that regulates data flow between two data sources, such as a router, switch hub, integrated access device, multiplex, or gateway.

A "bridge" or "intimidator" between hardware remote servers is Fog Server "FS." Cloud computing and the internet of things are closely related to fog, a distributed network environment. The cloud-based work and IOT data producer are expanded by fog. Fog's security is excellent. All of the IOT processes safeguard fog node "FS." It processes some data locally rather than sending it to the cloud, saving network bandwidth. Additionally, it lessens the need for latency. It aids in quick decision-making. Fog nodes can be set up in difficult places, like on the roof of a building in the middle of the ocean, along a railway line, or in faraway places.

Cluster of Storage: The distributed network system is supported by FS, which also functions as a Storage Cluster. The storage cluster is made up of the sub-servers of Centre FS. The data from various physical or edge devices is combined and sent to FS by these Storage clusters. Each storage cluster has the ability to collect data from multiple physical devices of the same data type.

Server Cloud: Cloud server "C" in the preceding figure 4 is a virtual server that is also known as a

computer server. It is used to provide remote access to its resources via a network. Servers are a crucial component of cloud technology. The ability to virtually access and utilize actual network resources from faraway locations is made possible by Cloud Server.

Map Reducer for HDFS: The Distributed Parallel Processing File System (HDFS) is used to store massive clusters of multiple machines' large files. MapReduce, on the other hand, is a software framework that uses data from multiple clusters to build applications that process a lot of data simultaneously on many clusters.

Hive: The Hadoop warehouse infrastructure tool Hive is used to process structured data. It is built on top of Hadoop to simplify querying and analyzing Big Data.

Hive CLI: The Hive CLI is an old tool that was used for two main things. The first is that it was a command-line tool for Hive Server and the second is that it was a thick client for SQL on Hadoop.

UDF: a function that is defined by the user of a program or environment, in a situation where it is common practice to assume that functions are integrated into the program or environment. Most of the time, UDFs are written to meet the needs of their creator.

HQL: HiveQL, a query language similar to SQL, is used to write Hive queries. You don't need to know Java or MapReduce to use HiveQL to query the data after you've defined the structure.

The flow of work of proposed Architecture of MPI model is explained below:

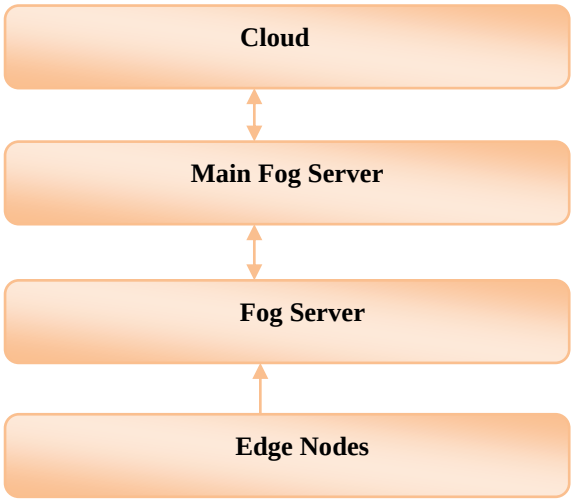


Figure 5: Work Flow of Proposed Architecture of MPI model [5]

From the above flow of work the edge node send data to fog server. There is bidirectional connection between main fog server and fog server, same as main fog server to cloud during the transmission of data.

MongoDB

In 2009, MongoDB Inc. created the open-source C++ document-oriented database known as MongoDB. Web, mobile, and Internet of Things apps can all use MongoDB. MongoDB uses JSON-formatted documents but stores them in BSON format. It also includes its own query language called MongoDB. The well-known operating systems Linux, macOS, Windows, and Solaris are all compatible with MongoDB, and it supports 17 programming languages [11].

There are three software editions available from MongoDB: Atlas, Enterprise Advanced, and Community Edition. MongoDB comes with a GUI (graphical user interface), MongoDB Compass, Mongo Express, a web application, and mongo, a shell. Downloading MongoDB is possible at https://www.mongodb.com (accessed on March 9, 2023).

Collections and documents can be modified and removed. Also included are data exploration and manipulation, query creation, pipeline aggregation, document visualization, index creation, and the development of schema validation rules. It offers a dashboard with a graphic representation of the server's present condition [8].

When a BSON document in MongoDB is larger than 16 MB, the distributed storage engine GridFS is used to store the huge binary documents. Figure 6 shows that these papers are broken into many 255 KB pieces or portions, with the exception of the final piece, which occupies the required amount of space. Two collections are used by the GridFS engine. Document metadata are stored in the second collection, while document fragments are kept in the first.

GridFS is likewise utilized when they just need to get a section or a piece of the record without stacking the entire report into memory [9].

The primary qualities and benefits of MongoDB are:

- It gives ordering, impromptu questions, Muck activities, and MapReduce;
- It is Corrosive agreeable;

- It gives local drivers to programming dialects and structures;
- It upholds ace slave replication;
- It upholds level, vertical, and layered scaling;
- It utilizes MVCC.

A portion of the constraints of MongoDB are as per the following:

- Information can without much of a stretch be killed unintentionally levy to the absence of relations;
- Ordering takes up a lot of Slam;
- It simply upholds triggers on MongoDB Map book.

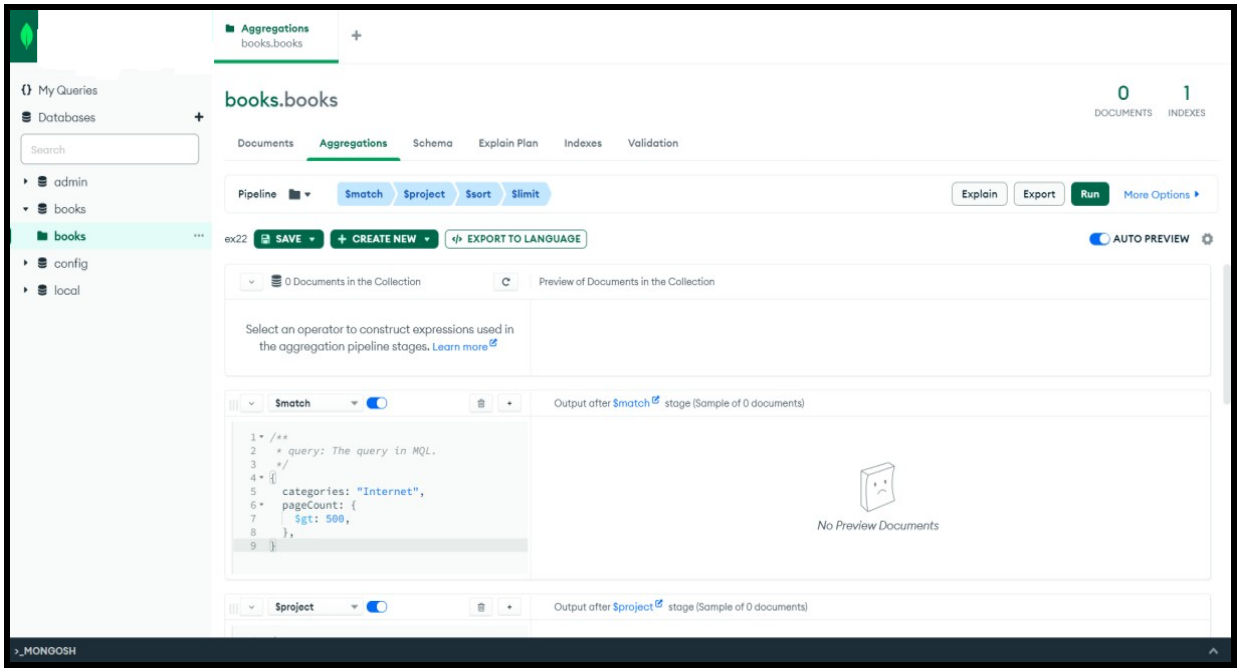


Figure 6: illustrates the MongoDB Compass interface [6]

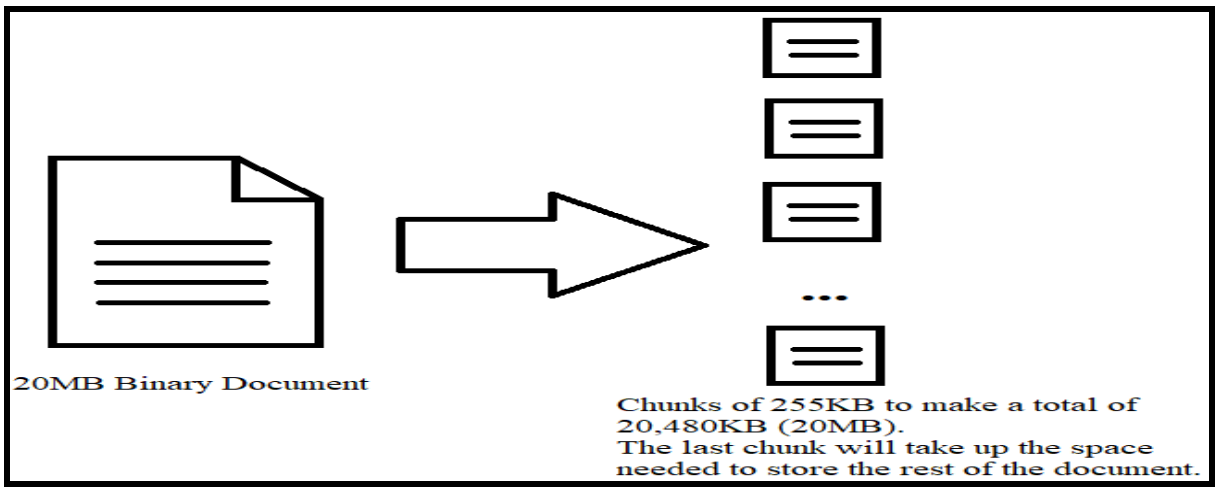


Figure 7: HDFS mechanism (based on [7]

Yahoo! Cloud Serving Benchmark (YCSB)

The YCSB benchmark performs an essential role in the process of developing and testing data management systems. The YCSB has become the *de facto* standard bench- mark to compare and measure the performance of different NoSQL databases under various workloads. The YCSB architecture is illustrated in Figure 8, which consists of the YCSB Client (composed of workload executor), the client threads, and the statistics

module. YCSB Client is a Java application that generates the data to be loaded into the target databases and the operations representing the workloads. In basic operations, the workload executor triggers several client threads. Each thread performs a series of operations that make calls to the database interface layer to load the database, known as the load phase, and to execute the workloads, known as the transaction phase. The database interface layer converts simple requests, such as read requests, from client threads into database calls. Threads measure latency and throughput are achieved by their operations and pass these measurements to the statistics module. At the end of the workload execution, the statistics module aggregates these measurements and returns the execution time, the average latency, the latency in percentage, and a histogram or a time series of latencies [5,8]. The client receives two properties (name/value pairs) that define the operation to be performed [9]: Workload properties: define the workload; for example, the combination of reading and writing, the distribution to be used, and the size and number of fields in a record; Runtime properties: define the specific properties of a workload execution; for example, the database interface layer uses the properties used to initialize this layer, such as the database service hostname and the number of client threads. In this work, the standard workloads provided by YCSB, called YCSB Core Workloads are used, which are described in Table 1. The standard workloads are from workload W1 to workload W8. It defines two new workloads, W7 and W8, which are similar to workloads W2 and W3. However, workloads W2 and W3 focus on the read operation, and workloads W7 and W8 focus on the update operations.

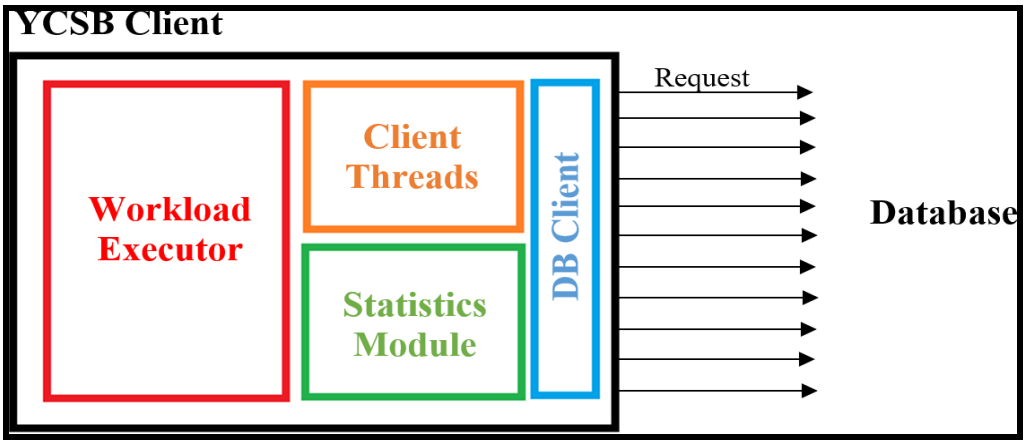


Figure 8: YCSB Client Architecture.

Table1: Workloads used to evaluate the NoSQL databases.

Workloads	Operations	Distribution	Records	Threads	Data Size
W1-Update Heavy	Read: 50% Update: 50%	Zipfian	100,000	1	Field size = 500 bytes
			1,000,000	3	Field number = 20
			10,000,000	6	500 bytes × 20 = 10 KB
W2-Read Mostly	Read: 95% Update: 5%	Zipfian	100,000	1	Field size = 500 bytes
			1,000,000	3	Field number = 20
			10,000,000	6	500 bytes × 20 = 10 KB
W3-Read Only	Read: 100%	Zipfian	100,000	1	Field size = 500 bytes
			1,000,000	3	Field number = 20
			10,000,00	6	500 bytes × 20 = 10

			0		KB
W4-Read Latest	Read: 95%	Latest	100,000	1	Field size = 500 bytes
	Insert: 5%		1,000,000	3	Field number = 20
			10,000,000	6	500 bytes × 20 = 10 KB
W5-Short Ranges	Scan: 95%	Zipfian	100,000	1	Field size = 500 bytes
	Insert: 5%	Uniform	1,000,000	3	Field number = 20
			10,000,000	6	500 bytes × 20 = 10 KB
W6-Read–Modify–Write	Read: 50%	Zipfian	100,000	1	Field size = 500 bytes
	Read–Modify–Write: 50%		1,000,000	3	Field number = 20
			10,000,000	6	500 bytes × 20 = 10 KB
W7-Update Mostly	Update: 95%	Zipfian	100,000	1	Field size = 500 bytes
	Read: 5%		1,000,000	3	Field number = 20
			10,000,000	6	500 bytes × 20 = 10 KB
W8-Update Only	Update: 100%	Zipfian	100,000	1	Field size = 500 bytes
			1,000,000	3	Field number = 20
			10,000,000	6	500 bytes × 20 = 10 KB

In Table 1, it can see the dispersion of activities and the sort of appropriation utilized. In all responsibilities, the information size was generally 10 KB since they characterized various twenty fields, and each field has 500 bytes. The quantities of records were characterized by specifying a ten-fold increment, beginning with 100,000, 1 million, and 10 million records. The quantity of strings also changed between 1, 3, and 6. While running a responsibility, the tasks, dispersion, and information size boundaries generally stay consistent. The boundaries that differ are the quantity of records and the quantity of strings. For instance, a responsibility with 100,000 records and one string, or 100,000 records and three strings, or 100,000 records and six strings, and so on, for the leftover records. For every responsibility, three executions were performed, and the average of these runtime values was utilized as the eventual outcome of the trial. The trial arrangement for the tests was a HP Structure PC 15t-eg000 computer with 476 GB of SSD, 15.4 GB of memory, an AMD Ryzen 5 4500U processor, with Radeon Illustrations, 2375 MHz, and six centers with six coherent processors in a Windows 10 environment. The test plan was intended to investigate the way the information bases behave when they increase and their parallelism limit. To this end, they utilized a sensible number of records to recreate huge amounts of information (100,000, 1 million, and 10 million records). As a result of PC asset restrictions, they didn't utilize in excess of 10 million records, which possess roughly 95 GB on disk. As indicated by our ten-fold increase, on the off chance that they added the 100 million records, they would involve roughly 953 GB on the circle, and our PC can uphold just 476 GB. They involved strings to break down the data set's capacity for parallelism. The climate utilized upholds just six centers, so they change the quantity of strings to one, three, and six.

EXPERIMENTAL RESULTS

Objective:

- To Test the architecture with different tools in terms of time required for read operation, scan and update on various different factors.
- To compare Novel Coordinated Hybrid Model for Multi Paradigm Database like Pig, Hive, NOSQL and MongoDB.
- To calculate run time of paradigm with 3 threads on eight workloads and compare it with each other's on different records.

Setup Requirement:

5 Virtual Machines or Server with the following configurations

Minimum Ram	**8GB**
Minimum HDD	25 GB
Minimum Cores of CPU	2
Internet Connectivity	Yes
Network Adapter	Bridged

OS Used:

- CentOS 6.4
- Windows 10(For Web Utility Setup)

Technologies Used:

- XAMP
- MySQL Community Edition
- PHP version 7

Tools Used:

- VM Workstation 10
- ATOM IDE
- HeidiSQL
- MySQL Workbench

Dataset Details*:*

Internet Source	**kaggle.com/datasets**
Name	NYS Jail Population By Country: Beginning 1997
Actual Source	New York State Open Data
Number of Columns	11
Number of Records	1.30k

The accompanying subsections present the outcomes got by applying YCSB on NoSQL document data sets: by Pig, MongoDB, Hive and NOSQL.. In the examinations are utilized the responsibilities recently characterized in Table 1, with 100,000, 1 million, and 10 million records, and the quantity of strings changes between one, three, and six. They finish up this part by examining the outcomes got with the three data sets. Every one of the outcomes got by applying YCSB on NoSQL record data sets: Pig, Hive, NOSQL and MongoDB are openly accessible at GitHub: https://github.com/isofiaC/Performance-Assessment of NoSQL-Archive Data sets (got to on 5 January 2023). Pig Table 2 presents the runtime of Pig, for all jobs, with 100,000 records, a million records, and 10,000,000 records. Because of the absence of room, just the running season of the pig with three strings is introduced. Utilizing pig, responsibility W5, with 100,000 records and a million records, is the responsibility with the longest runtime on the grounds that it is generally made out of sweep operation. Then again, with 10,000,000 records, the jobs with the longest execution time wereW6. Responsibility is made out of perused and update tasks, and responsibility W6 is made out of perused change compose activities. For all records utilized, responsibility W3 had the briefest runtime in light of the fact that this responsibility is made exclusively out of understood tasks.

Table 2: Runtime of Pig with 3 threads

Pig	**100,000 Records**	**1,000,000 Records**	**10,000,000 Records**
Workload W1	11.27 s	1057.88 s	87,950.68 s
Workload W2	10.26 s	698.35 s	34,780.45 s
Workload W3	7.75 s	90.35 s	3543.44 s
Workload W4	13.41 s	221.97 s	30,849.66 s
Workload W5	472.18 s	3398.56 s	35,048.55 s
Workload W6	44.15 s	2932.35 s	48,426.55 s
Workload W7	31.64 s	534.55 s	7915.99 s
Workload W8	21.89 s	85.99 s	1168.98 s
Total	**612.55 s**	**9020 s**	**249684.3 s**

Table 3.*Runtime of Pig*

Pig	100,000 Records	1,000,000 Records	10,000,000 Records
1 Thread	1074.67 s	19,961.38 s	485,121.62 s
3 Threads	587.03 s	9735.38 s	343,383.46 s
6 Threads	265.28 s	7086.60 s	249,349.99 s
Total	265.28 s	7086.6 s	249,349.99 s

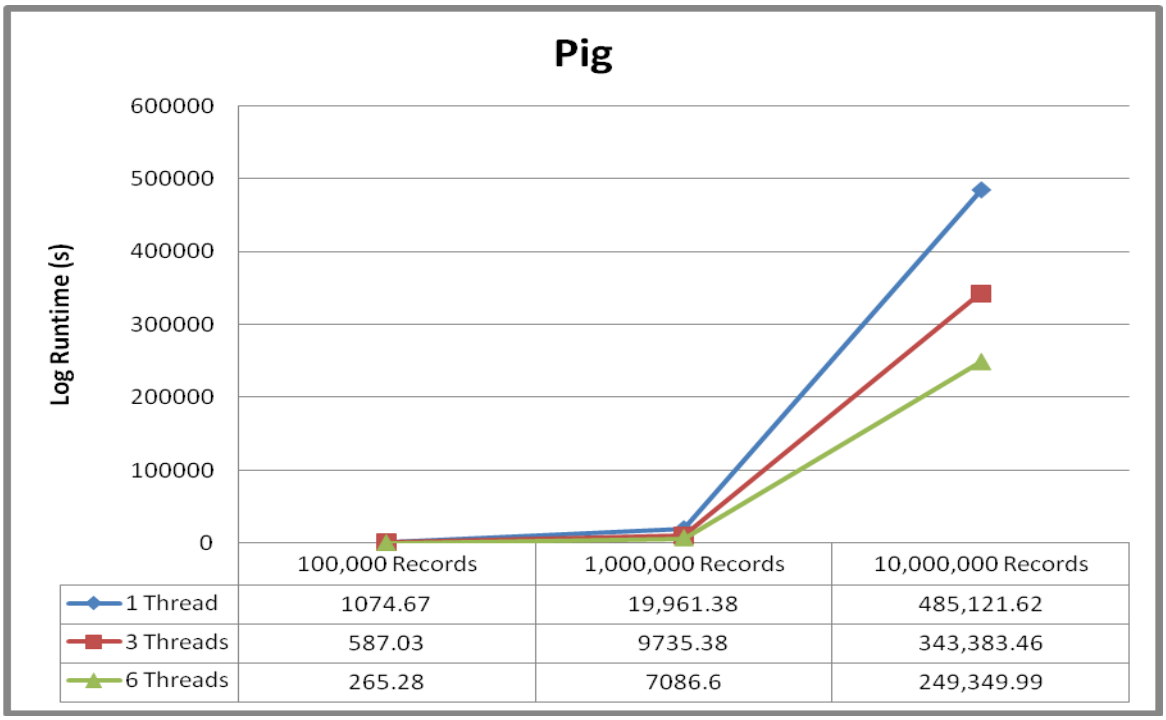


Figure 9: Data Analysis Using Pig

Table 4. Runtime of Hive with 3 threads.

Hive	100,000 Records	1,000,000 Records	10,000,000 Records
Workload W1	11.68 s	1477.85 s	28,614.15 s
Workload W2	15.83 s	980.45 s	37,547.66 s
Workload W3	16.20 s	255.45 s	5240.16 s
Workload W4	20.00 s	296.47 s	9131.68 s
Workload W5	251.33 s	3716.88 s	56,268.57 s
Workload W6	32.57 s	935.88 s	9878.77 s
Workload W7	36.53 s	998.58 s	21,053.57 s
Workload W8	32.35 s	457.87 s	8089.89 s
Total	**416.49**	**9119.43**	**175824.45**

Table 5. Runtime of Hive

Hive	100,000 Records	1,000,000 Records	10,000,000 Records
1 Thread	943.39 s	15,243.94 s	323,048.99 s
3 Threads	424.28 s	8087.15 s	231,580.94 s
6 Threads	353.088 s	4927.95 s	176,021.71 s

Total	1720.758 s	28259.04 s	730651.64 s

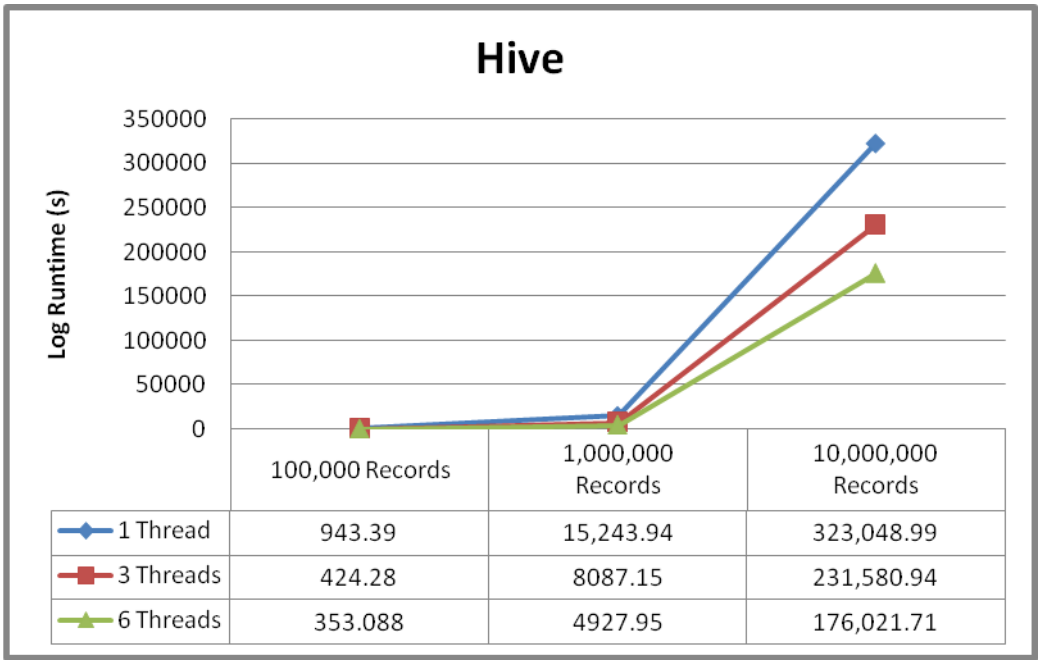


Figure 10: Data Analysis Using Hive

Table 6. Runtime of NOSSQL with 3 threads

NOSQL	**100,000 Records**	**1,000,000 Records**	**10,000,000 Records**
Workload W1	11.77 s	1377.85 s	19,523.25 s
Workload W2	10.85 s	880.44 s	29,526.43 s
Workload W3	15.25 s	185.66 s	4240.16 s
Workload W4	16.00 s	244.44 s	8811.54 s
Workload W5	152.35 s	2916.88 s	52,244.57 s
Workload W6	21.55 s	824.87 s	8548.55 s
Workload W7	25.43 s	790.68 s	16,153.43 s
Workload W8	21.36 s	375.78 s	7248.44 s
Total	**274.56 s**	**7596.6 s**	**146296.37 s**

Table 7. Runtime of NOSSQL

NOSQL	**100,000 Records**	**1,000,000 Records**	**10,000,000 Records**
1 Thread	643.37 s	13,233.93 s	282,056.98 s
3 Threads	354.29 s	7298.17 s	182,580.95 s
6 Threads	283.29 s	3847.76 s	96,072.73 s
Total	**1280.95 s**	**24379.86 s**	**560710.66 s**

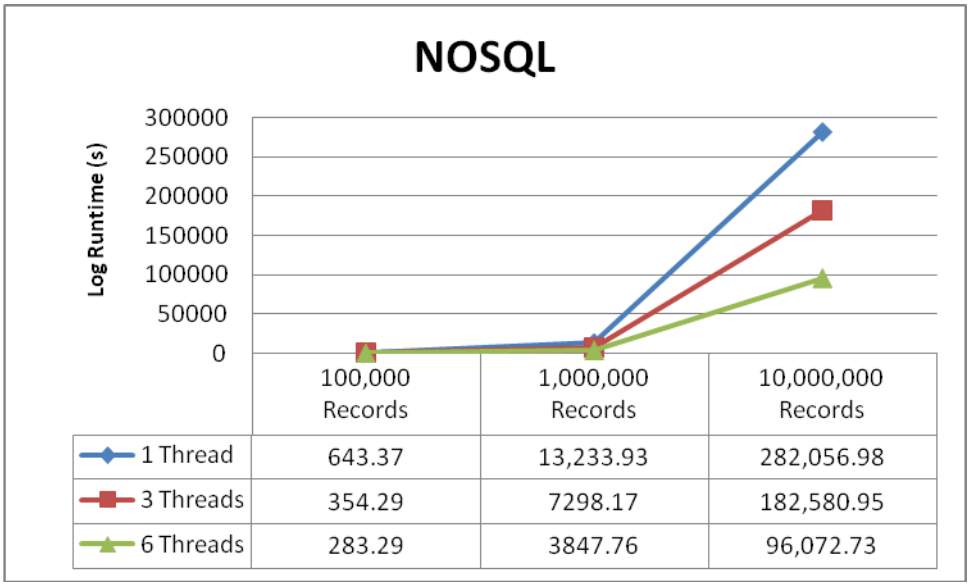


Figure 11: Data Analysis Using NOSSQL

Table 8 presents the MongoDB runtime for all jobs with every one of the records utilized and with three strings. Because of the absence of room, the runtime results with one and six strings are accessible at: https://github.com/isofiaC/Execution Assessment of-NoSQL-Archive Data sets (got to on 5 January 2023).

Table 8.*Runtime of MongoDB with 3 threads.*

MongoDB	**100,000 Records**	**1,000,000 Records**	**10,000,000 Records**
Workload W1	11.85 s	198.22 s	3750.77 s
Workload W2	6.16 s	102.27 s	1544.41 s
Workload W3	5.66 s	101.44 s	1505.88 s
Workload W4	7.35 s	78.55 s	1125.34 s
Workload W5	68.72 s	3592.81 s	136,320.55 s
Workload W6	11.59 s	186.45 s	3786.87 s
Workload W7	11.25 s	481.55 s	5775.99 s
Workload W8	11.21 s	351.98 s	6854.57 s
Total	**133.79 s**	**5093.27 s**	**160664.38 s**

In MongoDB, responsibility E has the most horrendously terrible runtime result, taking into account every one of the records utilized. This responsibility is mostly made out of sweep tasks, and since MongoDB performs many circle gets to, this causes an expansion in runtime.

Then again, with 100,000 records, responsibility C has the best runtime in light of the fact that this responsibility is com-presented exclusively of understood activities. Nonetheless, with a million records, and 10,000,000 records, responsibility W4 had the best runtime results. This responsibility is made out of 95% read activities, and simply 5% supplement tasks[38-48]. The responsibilities W2, W3, and W4 are fundamentally made out of understood tasks, which are quicker than different activities. This makes responsibilities W2, W3, and W4 show better runtime, in any event, fluctuating the quantity of records. Table 7 shows the runtime results for every one of the responsibilities utilized in MongoDB. From these outcomes, inferring that was conceivable: Taking into account 100,000 records, it confirms that the runtime diminishes more than divided while expanding from one to three strings. The equivalent doesn't occur while expanding from three to six strings, as it drops just 20% of

the runtime, addressing unfortunate increasing. This 20% can likewise be made sense of by the way that the PC involved just backings six strings and uses them for different cycles that run at the same time;

Taking into account a million records, they check that the runtime was diminished by half when it changes from one to three strings. At the point when it was changed from three to six strings, the abatement in runtime was far more terrible, decreasing by just 10%. This can happen in light of the fact that there is a lot handling, consequently utilizing the plate all the more habitually, which dials back the runtime; Yet again taking into account 10,000,000 records, that's what they confirm, its presentation has worsened. This is on the grounds that it has more information to process and consumes more PC assets. While expanding it from one to three strings, the runtime just increased by 30%. It is more awful when it goes from three to six strings, in which it just declines by 10%. These outcomes show that MongoDB increases inadequately while utilizing more strings; While moving from 100,000 records to a million records, the runtime was supposed to increment by multiple times; in any case, there was a 47.4 times increment. This addition is a lot higher than Pig, Hive and NOSQL with an increment of 19.3 times and 18.9 times, separately. This development can be made sense of by the constraints of PC equipment and the utilization of PC assets by different cycles running simultaneously; While moving from a million records to 10,000,000 records, the increment expected was multiple times; nonetheless, there was an expansion in a runtime of 28.8 times. This augmentation of multiple times is because of the way that the responsibility is higher, and hence, more circle gets to should be made, which dials back the runtime;

Table 9.Runtime of MongoDB.

MongoDB	**100,000 Records**	**1,000,000 Records**	**10,000,000 Records**
1 Thread	257.32 s	7873.53 s	187,148.38 s
3 Threads	127.83 s	5147.57 s	140,154.84 s
6 Threads	111.43 s	4637.53 s	131,718.76 s
Total	**496.58 s**	**17658.63 s**	**459021.98 s**

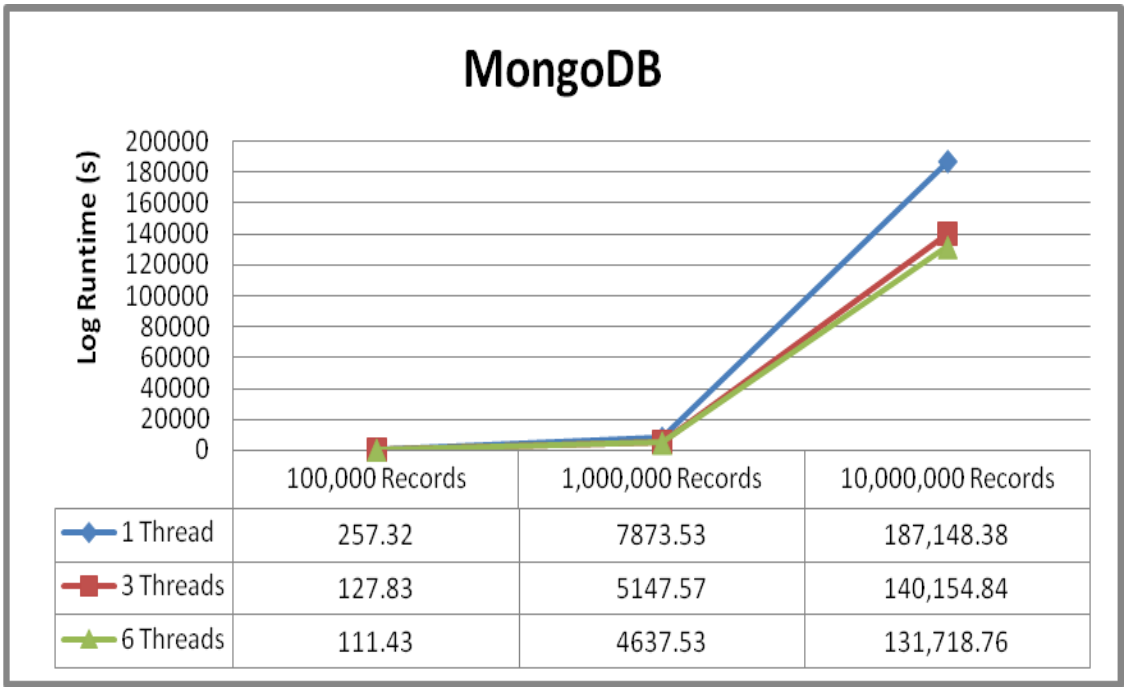


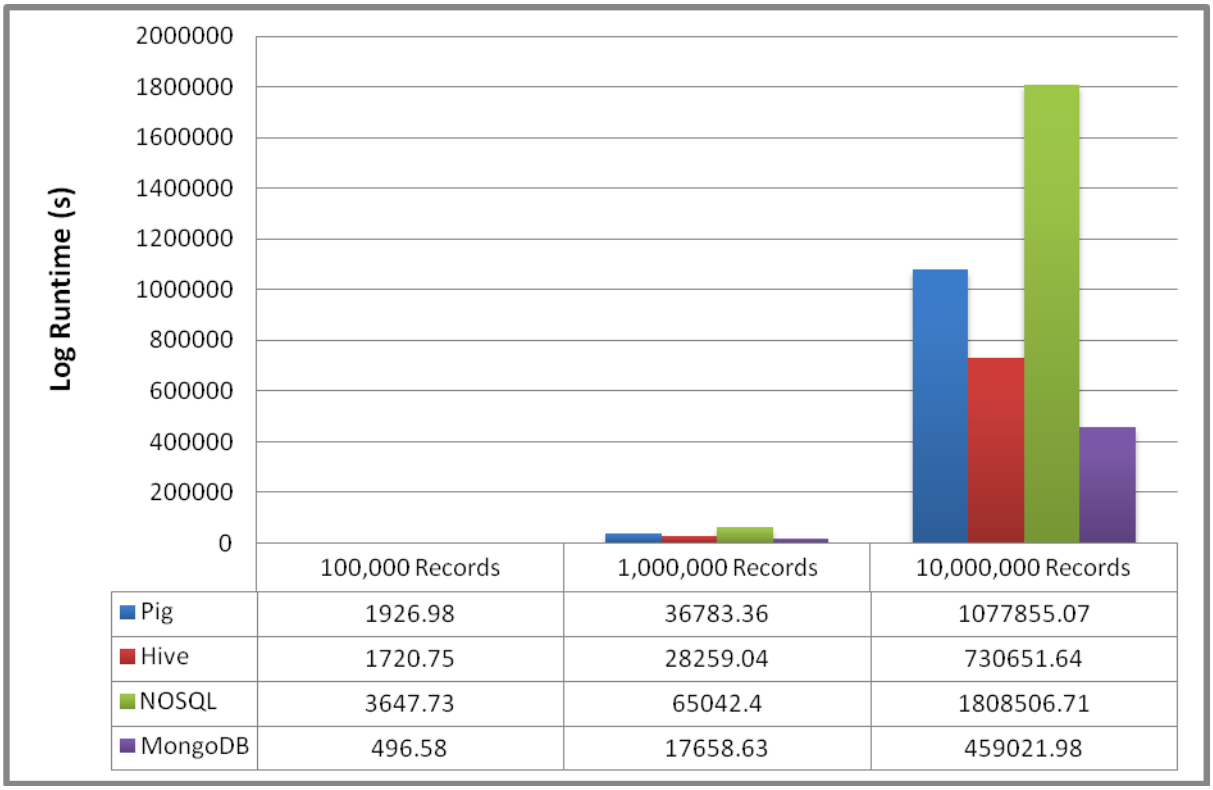
Figure 12: Data Analysis Using MongoDB

DISCUSSION OF THE RESULTS

Figure 13 shows, in logarithmic scale, the all values for 100,000, a million, and 10,000,000 records utilized in the three report data sets. From this, they can presume that: With 100,000 and a million records, MongoDB was the information base that had the most runtime. Nonetheless, while utilizing 10,000,000 records, Pig had the briefest runtime, trailed by Pig, MongoDB, Hive and NOSQL;

MongoDB had the most terrible runtime in responsibility W4. Then again, Pig had the most terrible runtime utilizing responsibilities W1, W5, and W6. The tasks that create these responsibilities are filter in responsibility W5, and read-change write in responsibility W6; It was checked that the three archive information bases present great scope up while moving from one to three strings. While moving from three to six strings, MongoDB presents the most obviously terrible scale-up, particularly while expanding the responsibility size;

At times, Pig performed better compared to MongoDB, especially while changing from three to six strings and furthermore while utilizing an enormous number of records, where the distinction between runtime was more noteworthy.



	100,000 Records	1,000,000 Records	10,000,000 Records
Pig	1926.98	36783.36	1077855.07
Hive	1720.75	28259.04	730651.64
NOSQL	3647.73	65042.4	1808506.71
MongoDB	496.58	17658.63	459021.98

Figure 13: Log runtime representation, in seconds, of Pig, Hive, NOSQL and MongoDB.

Another important factor that contributed to this outcome was workload W4, which results are presented in Figure 14, using a logarithmic scale.

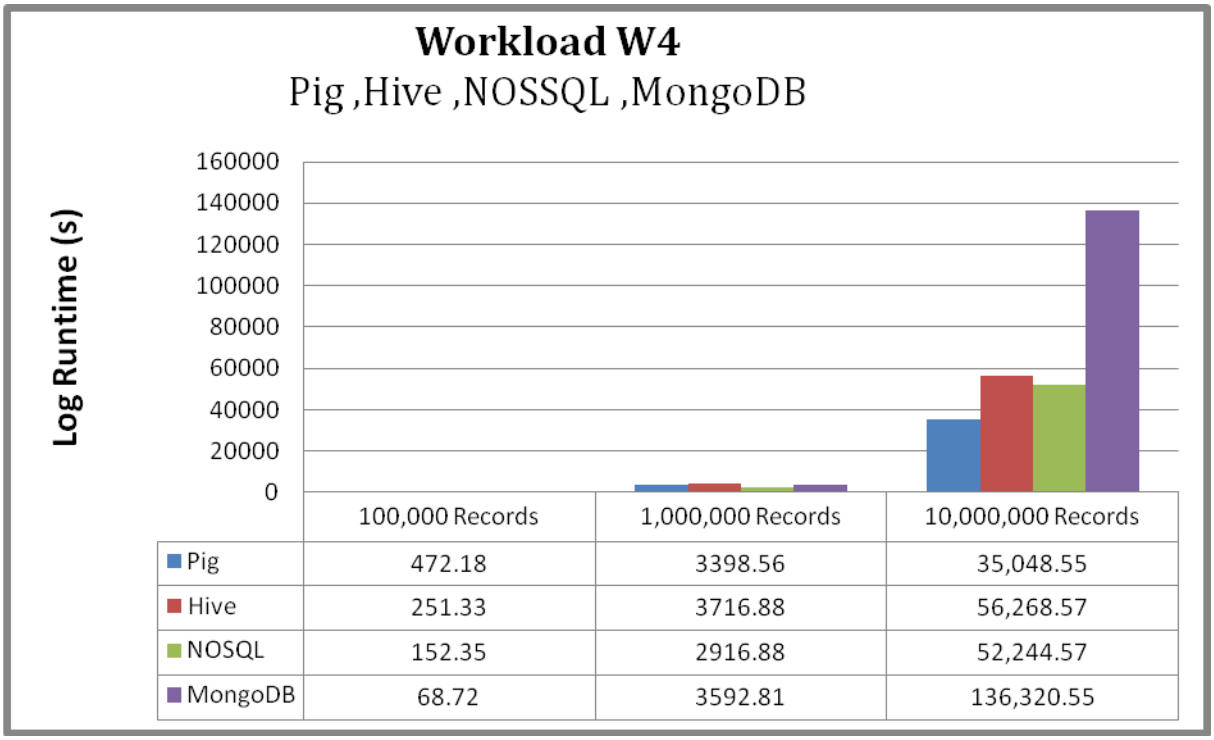


Figure 14: Log runtime representation, in seconds, of workload W4.

CONCLUSION

The field of information mining and examination has gone through a transformation because of the accessibility of enormous information. It has given us a device that permits us to oversee organized, unstructured, and semi-organized information in a way that was beforehand unthinkable. They took a gander at concentrates on information examination in this paper, from conventional information examination to later investigations on large information investigation. The issues that are the focal point of the conversation are execution and results from the enormous information investigation system and stage. NoSQL report data sets store information as records and can deal with unstructured and semi-organized information guaranteeing great question execution. In this paper, they chose the main three NoSQL archive arranged data sets as per the DB-Motors Positioning of 2023 and evaluated their exhibition utilizing the YCSB benchmark. The YCSB benchmark was exceptionally valuable as it permitted us to tentatively concentrate on the effect on the inquiry execution season of utilizing various kinds of activities, various quantities of strings, and different records size. Test the architecture with different tools in terms of time required for read operation, scan and update on various different factors. Contrasted with other logical papers, they utilized a bigger size for each record, 10 KB, and utilized more records, 100,000 records, a million records, and 10,000,000 records. In many papers, the creators didn't utilize it in wide range since they didn't utilize every one of the jobs remembered for YCSB. In the exploratory assessment, they utilized every one of the responsibilities accessible and executed two additional jobs (W7 and W8). Not at all like most works, had they utilized the more usually utilized Windows climate instead of a Ubuntu climate. This work permits that Pig is the data set with the best execution time. Be that as it may, taking into account every one of the tests performed without the output named responsibility, which comprises of sweep tasks, MongoDB is the information base with the best execution time.

In future work, it means to assess more NoSQL archive data sets with greater jobs and contrast them and social data sets in group and cloud conditions.

REFERENCES

[1]. Wang, Lidong, and Cheryl Ann Alexander. "Machine learning in big data." *International Journal of Mathematical, Engineering and Management Sciences* 1, no. 2 (2016): 52-61.

[2]. Ning, L. I., and L. U. O. Wen-juan. "A survey of machine learning algorithms for big data." *Pattern recognition and artificial intelligence* 27, no. 4 (2014): 327-336.

[3]. SaikatDas, SampitaMallick, "Machine Learning in Big Data Analytics International Journal of Engineering Research & Technology (IJERT), Special Issue – 2021, Volume 9, Issue 11, pp. 134-138.

[4]. Qiu, Junfei, and Youming Sun. "A Research on Machine Learning Methods for Big Data Processing." In *4th International Conference on Information Technology and Management Innovation*, pp. 920-928. Atlantis Press, 2015.

[5]. L'heureux, Alexandra, Katarina Grolinger, Hany F. Elyamany, and Miriam AM Capretz. "Machine learning with big data: Challenges and approaches." *Ieee Access* 5 (2017): 7776-7797.

[6]. Singh, SuryabhanPratap, and Umesh Chandra Jaiswal. "Machine learning for big data: a new perspective." *International Journal of Applied Engineering Research* 13, no. 5 (2018): 2753-2762.

[7]. Wang, Lidong, and Cheryl Ann Alexander. "Machine learning in big data." *International Journal of Mathematical, Engineering and Management Sciences* 1, no. 2 (2016): 52-61.

[8]. Nti, Isaac Kofi, Juanita AhiaQuarcoo, Justice Aning, and GodfredKusiFosu. "A mini-review of machine learning in big data analytics: Applications, challenges, and prospects." *Big Data Mining and Analytics* 5, no. 2 (2022): 81-97.

[9]. Assefi, Mehdi, EhsunBehravesh, Guangchi Liu, and Ahmad P. Tafti. "Big data machine learning using apache spark MLlib." In *2017 IEEE international conference on big data (big data)*, pp. 3492-3498. IEEE, 2017.

[10]. Chunzi, Shao, Wang Xuanren, and Liu Ling. "The application of big data analytics in online foreign language learning among college students: Empirical research on monitoring the learning outcomes and predicting final grades." In *2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, IEEE, 2020. , pp. 266-269.

[11]. Leung, Carson K., Yubo Chen, Calvin SH Hoi, Siyuan Shang, and Alfredo Cuzzocrea. "Machine learning and OLAP on big COVID-19 data." In *2020 IEEE International Conference on Big Data (Big Data)*, IEEE, 2020, pp. 5118-5127.

[12]. Muniswamaiah, Manoj, TilakAgerwala, and Charles C. Tappert. "Federated query processing for big data in data science." In *2019 IEEE International Conference on Big Data (Big Data)*, IEEE, 2019, pp. 6145-6147.

[13]. Bhandarkar, Milind. "MapReduce programming with apache Hadoop." In *2010 IEEE International Symposium on Parallel & distributed processing (IPDPS)*, IEEE, 2010, pp. 1-1.

[14]. Singh Bhathal, Gurjit, and Amardeep Singh Dhiman. "Big data solution: Improvised distributions framework of hadoop." In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE, 2018, pp. 35-38.

[15]. Sontakke, Vaishali, and R. B. Dayanand. "Optimization of hadoopmapreduce model in cloud computing environment." In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, IEEE, 2019, pp. 510-515.

[16]. Merla, PrathyushaRani, and Yiheng Liang. "Data analysis using hadoopMapReduce environment." In *2017 IEEE International Conference on Big Data (Big Data)*, IEEE, 2017. ,pp. 4783-4785.

[17]. Verma, Chitresh, and Rajiv Pandey. "Big Data representation for grade analysis through Hadoop framework." In *2016 6th international conference-cloud system and big data engineering (Confluence)*, IEEE, 2016, pp. 312-315.

[18]. Wang, Qi, and Xiaojun Huang. "Pft: a performance-fairness scheduler on hadoop yarn." In *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, 2016, pp. 76-80.

[19]. Siretskiy, Alexey, and Ola Spjuth. "Htseq-hadoop: Extending htseq for massively parallel sequencing data analysis using hadoop." In *2014 IEEE 10th International Conference on e-Science*, vol. 1, IEEE, 2014. , pp. 317-323.

[20]. Krishna, Talluri Lakshmi Siva Rama, ThirumalaisamyRagunathan, and Sudheer Kumar Battula. "Performance evaluation of read and write operations in hadoop distributed file system." In *2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming*, IEEE, 2014. , pp. 110-113.

[21]. Masur, Rohit G., and Suzanne K. McIntosh. "Preliminary performance analysis of Hadoop 3.0. 0-alpha3." In *2017 New York Scientific Data Summit (NYSDS)*, IEEE, 2017, pp. 1-3.

[22]. Li, Yinwei, and Dujuan Zhang. "Hadoop-Based University Ideological and Political Big Data Platform Design and Behavior Pattern Mining." In *2020 International Conference on Advance in*

Ambient Computing and Intelligence (ICAACI), IEEE, 2020. , pp. 47-51.

[23]. Shah, Ankit, and MamtaPadole. "Load balancing through block rearrangement policy for hadoop heterogeneous cluster." In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, 2018. , pp. 230-236.

[24]. Tang, Yinan, HongxiangGuo, Tongtong Yuan, Qi Wu, Xiang Li, Cen Wang, XiongGao, and Jian Wu. "OEHadoop: accelerate Hadoop applications by co-designing Hadoop with data center network." *IEEE Access* 6 (2018): 25849-25860.

[25]. Wu, Yuanyuan, Xiaojin Li, Jinze Liu, and Licong Cui. "Hadoop-EDF: Large-scale distributed processing of electrophysiological signal data in hadoopMapReduce." In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, 2019.,pp. 2265-2271.

[26]. Hsu, Jen-Chun, Ching-Hsien Hsu, Shih Chang Chen, and YehChing Chung. "Correlation aware technique for SQL to NoSQL transformation." In *2014 7th international conference on Ubi-media computing and workshops*, IEEE, 2014, pp. 43-46.

[27]. Charan, PV Sai, P. Ravi Kumar, and P. Mohan Anand. "Addressing cold start problem in recommendation system using custom built hadoop ecosystem." In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, IEEE, 2018, pp. 355-358.

[28]. Thilagavathi, S., B. Akshaya, and S. Vimala. "MODSUM: Mitigation of Data Skew Using Mapper." In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, IEEE, 2018. pp. 128-132.

[29]. Manwal, Manika, and Amit Gupta. "Big data and hadoop—a technological survey." In *2017 International Conference on Emerging Trends in Computing and Communication Technologies (ICETCCT)*, IEEE, 2017, pp. 1-6.

[30]. Tunjić, Aleksandar. "The Automation of the Data Lake Ingestion Process from Various Sources." In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, IEEE, 2019, pp. 1276-1281.

[31]. Weixin, Zhai, Yang Zhe, Wang Lin, Wu Feilong, and Cheng Chengqi. "The non-sql spatial data management model in big data time." In *2015 IEEE international geoscience and remote sensing symposium (IGARSS)*, IEEE, 2015. pp. 4506-4509.

[32]. Cheon, Sang-Ho, Ki-Hyeon Kwon, and Hyung-Jin Choi. "A Framework for Developing Distributed Application with Web-Tier Object Modeling." *The KIPS Transactions: PartD* 11, no. 5 (2004): 1143-1148.

[33]. Mkhwanazi, Kefilwe, Pius AdewaleOwolawi, TemitopeMapayi, and GbolahanAiyetoro. "An automatic crime reporting and immediate response system." In *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, IEEE, 2020.pp. 1-6.

[34]. Piroska, Haller, FarkasGyula, and SzantoIoan-Cosmin. "Data storage for smart environment using non-SQL databases." In *2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing*, IEEE, 2012, pp. 305-308.

[35]. Mondol, Brozolal, and MdAshiqMahmood. "An Efficient Method to Build a Standard Data Entry System by Extracting OLAP Cubes from NoSQL Data Sources." In *2021 5th International Conference on Electrical Information and Communication Technology (EICT)*, IEEE, 2021, pp. 1-6.

[36]. Al-Kateb, Mohammed, Mohamed Y. Eltabakh, Awny Al-Omari, and Paul G. Brown. "Analytics at Scale: Evolution at Infrastructure and Algorithmic Levels." In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, IEEE, 2022. pp. 3217-3220.

[37]. Liang, Zhiwei, Jiangti Kong, NengguiXu, Baoyan Liu, and Ying Lu. "An XHTML Solution for a Secure EDC System for Traditional Chinese Medicine Clinics Based on ICD-11 MMS Database and E-Signature." In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, 2019. , pp. 2634-2641.

[38]. Singha, A.K., Singla, A. and Pandey, R.K., 2016. Study and analysis on biometrics and face recognition methods. EPH-International Journal of Science And Engineering (ISSN: 2454-2016), 2(6), pp.37-41.

[39]. Zubair, S. and Singha, A.K., 2020. Parameter optimization in convolutional neural networks using gradient descent. In Microservices in Big Data Analytics (pp. 87-94). Springer, Singapore.

[40]. Zubair, S. and Singha, A.K., 2021. Network in Sequential Form: Combine Tree Structure Components into Recurrent Neural Network. In IOP Conference Series: Materials Science and Engineering (Vol. 1017, No. 1, p. 012004). IOP Publishing.

[41]. Singha, A.K., Kumar, A. and Kushwaha, P.K., 2018. Patient Cohort Approaches to data science using Biomedical Field. EPH-International Journal of Science And Engineering (ISSN: 2454-2016), 1(1), pp.457-462.
[42]. Singha, A.K., Kumar, A. and Kushwaha, P.K., 2018. Classification of brain tumors using deep Encoder along with regression techniques. EPH-International Journal of Science And Engineering (ISSN: 2454-2016), 1(1), pp.444-449.
[43]. Singha, A.K., Kumar, A. and Kushwaha, P.K., 2018. Speed predication of wind using Artificial neural network. EPH-International Journal of Science And Engineering (ISSN: 2454-2016), 1(1), pp.463-469.
[44]. Singha, A.K., Kumar, A. and Kushwaha, P.K., 2018. Recognition of human layered structure using Gradient decent model. EPH-International Journal of Science And Engineering (ISSN: 2454-2016), 1(1), pp.450-456.
[45]. Singha, A.K. and Zubair, S., 2022. Machine Learning for Hypothesis Space and Inductive Bias: A Review. AIJR Abstracts, p.70.
[46]. Zubair, S., Singha, A. K., Pathak, N., Sharma, N., Urooj, S., & Larguech, S. R. (2023). Performance Enhancement of Adaptive Neural Networks Based on Learning Rate. CMC-COMPUTERS MATERIALS & CONTINUA, 74(1), 2005-2019.
[47]. Singha, A. K., Zubair, S., Malibari, A., Pathak, N., Urooj, S., & Sharma, N. (2023). Design of ANN Based Non-Linear Network Using Interconnection of Parallel Processor. Computer Systems Science & Engineering, 46(3).
[48]. Pathak, N., Siddiqui, S. T., Singha, A. K., Mohamed, H. G., Urooj, S., & Patil, A. R. (2023). Smart Quarantine Environment Privacy through IoT Gadgets Using Blockchain. Intelligent Automation & Soft Computing, 35(3).