**Advance Database Management System  
  
  
Hadoop and its Ecosystem**

**Meghashree Maddihally Nagoji**

TABLE OF CONTENTS

[ABSTRACT 3](#_Toc88688508)

[1. Hadoop Distributed File System (HDFS) - by Meghashree Maddihally Nagoji 4](#_Toc88688509)

[Architecture 4](#_Toc88688510)

[NameNodes 4](#_Toc88688511)

[DataNodes 5](#_Toc88688512)

[HDFS Client 5](#_Toc88688513)

[CheckPointNode 6](#_Toc88688514)

[BackupNode 6](#_Toc88688515)

[Data Replication 6](#_Toc88688516)

[2. HIVE - by Aneesh Antony 8](#_Toc88688517)

[What is HIVE? 8](#_Toc88688518)

[How Does HIVE Work? 8](#_Toc88688519)

[HIVE Architecture 9](#_Toc88688520)

[Major Components of HIVE Architecture 9](#_Toc88688521)

[Major Benefits of HIVE 10](#_Toc88688522)

[Real-Life HIVE Use Cases 10](#_Toc88688523)

[3. MAHOUT - by Siddharth Malhotra 11](#_Toc88688524)

[What is Mahout? 11](#_Toc88688525)

[Features of Mahout 12](#_Toc88688526)

[Applications of Mahout 13](#_Toc88688527)

[4. SPARK - by Aditya Madhavi 14](#_Toc88688528)

[What is Spark? 14](#_Toc88688529)

[Description of features 14](#_Toc88688530)

[Applications of Apache Spark 15](#_Toc88688531)

[When to Avoid Using Spark 16](#_Toc88688532)

[5. Zookeeper - by Veshrut Malik 17](#_Toc88688533)

[What is Zookeeper? 17](#_Toc88688534)

[Zookeeper Guarantees: 17](#_Toc88688535)

[Ordering guarantees for watch notifications 17](#_Toc88688536)

[Zookeeper Implementation: 18](#_Toc88688537)

[CONCLUSION 20](#_Toc88688538)

[REFERENCES 21](#_Toc88688539)

# ABSTRACT

Apache Hadoop being an open-source framework which was intended to make interactions with big data to be easy. However, what is a big data? It is a term given to big data sets which takes a long time to process and generally cannot be efficiently processed with traditional methodologies such as RDBMS. Hadoop has made its place in companies and industries that need to work on large databases and large datasets which are very sensitive and needs to be handled efficiently and correctly with a high robust rate. Hadoop is a framework that enables processing of large data sets which reside in the form of clusters. Being a framework, Hadoop is made up of several modules that are supported by a large ecosystem of technologies. The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size.

Hadoop Ecosystem is a platform which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions. There are four major elements of Hadoop: HDFS, MapReduce, YARN, Hadoop Common. Most of the tools or solutions are used to supplement or support these major elements. All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc.  Hadoop has its own filesystem which replicates data to multiple nodes to ensure if one node holding data goes down, there are at least 2 other nodes from which to retrieve that piece of information.  This protects the data availability from node failure, something which is critical when there are many nodes in a cluster. We explain mainly about HDFS – architecture, different nodes, clients and data replications. Then HIVE and its architecture, major components and major benefits and real-life use cases. Then MAHOUT, its features and applications. Then Spark, its features, applications and when to avoid using it. Then lastly, Zookeeper, guarantees, ordering guarantees and implementations.

# Hadoop Distributed File System (HDFS)

With enormous amount of data coming in every day, storage is a challenging task. To tackle this issue, data is stored across multiple machines and such a filesystem is called distributed filesystem. One of the most efficient, fault-tolerant, and reliable filesystems is Hadoop Distributed File System which runs on commodity hardware. It is the primary data storage module in Hadoop Ecosystem.

### Architecture

HDFS follows a master-slave architecture.

### NameNodes

NameNode is the Master Node in the Hadoop File System responsible for managing the file system namespace. There can be only a single NameNode in the HDFS cluster.

NameNodes are the Master nodes responsible for works such as reading, writing, processing etc. These nodes give instructions to the DataNode and can also perform duties such as creation, deletion, and replication. The image shows the NameNode in the HDFS cluster.

Diagram

Description automatically generated

### DataNodes

There are multiple data nodes in the HDFS architecture. There is usually one per node in the cluster. These generally manage the storage attached to the nodes that they run on. Usually, a file is split into multiple blocks and each of the blocks are stored in a set of DataNodes. The main responsibility is for serving the read and write requests from the file system’s client. The DataNode also performs, upon the instruction of the NameNode, creation, deletion and replication of the blocks. The image above shows the DataNode in the HDFS cluster.

Diagram

Description automatically generated

### HDFS Client

HDFS client is a code library that exports the functionality and interface of the HDFS file system, which is also the user application access to the file system. On the same line of conventional file system, the client can perform read, write and deletion of files, also it can perform creation and deletion of directories. Namespace gives the path of the files and directories. When an application reads a file, first we need to know about the NameNode and then the DataNode. The client organizes the pipeline from node-to-node and sends the data. Unlike the traditional system, HDFS client exposes the location of a file blocks, this allows applications like MapReduce to improve the read performance.

Diagram

Description automatically generated

### CheckPointNode

A NameNode apart from its usual functionality, it also works as either a CheckpointNode or a BackupNode. This role is specified at the node start up. Creating a CheckPointNode lets a NameNode truncate the tail of the journal when the new checkpoint is upload to the NameNode.

### BackupNode

Similar to a CheckPointNode, the BackupNode is capable of creating periodic checkpoints, but in addition it maintains an in-memory, up-to-date image of the file system namespaces that is always synchronous with the state of the NameNode.

### Data Replication

HDFS is designed to reliably storage of large files in a large cluster across multiple machines. The storage is done as a sequence of blocks, where all the blocks except the last are of the same size. There are block replications done for fault tolerance. Block size and the replication factor are configurable per file. Files have restrictions like write-once and only one writer. NameNode takes care of all decisions done regarding the replication of blocks. It receives Heartbeat and Blockreport from each of its cluster through the DataNodes.

Diagram

Description automatically generated

A Rack is a collection of machines (30-40 in Hadoop) that are stored in the same physical location. There are multiple racks in a Hadoop Cluster and all the racks are interconnected through a switch as shown below.

Diagram

Description automatically generated

# HIVE

### What is HIVE?

Hive is a fault-tolerant, distributed data warehouse system for data analysis. It is also used for querying of large data systems in the open-source Hadoop platform. Hive allows users to read, write, and manage petabytes of data using SQL. It converts SQL queries into MapReduce jobs which in turn helps in easy execution and processing of extremely large volumes of data.

Hive Queries can be run from a command line interface in any available database management system, example from a JAVA JDBC or ODBC applications. HIVE is better suited for data warehousing tasks such as ETL reporting, data analysis which majorly involves read access to the data on the database.

### How Does HIVE Work?

The below diagram depicts HIVE workflow:

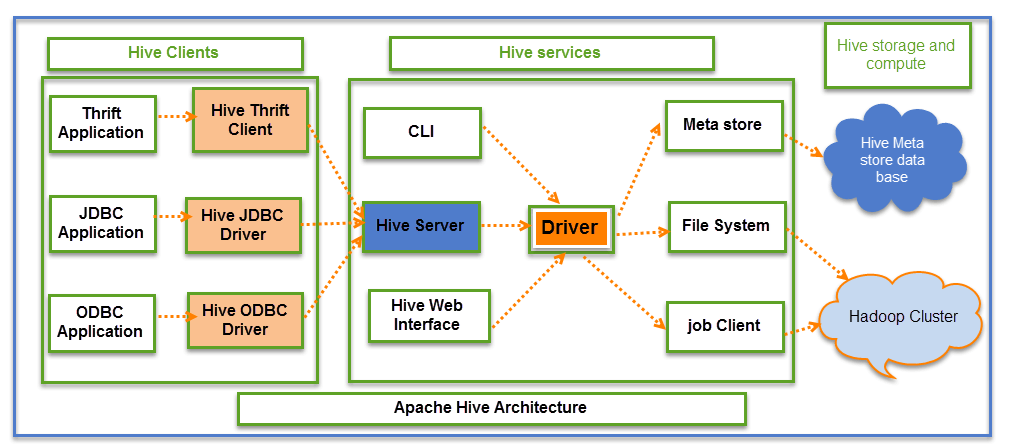


The HIVE interaction with Hadoop environment is as follows:

1. Query Execution – Hive sends command line query to Driver for execution
2. Compiler - Driver with the help of compiler parses the query and check for syntax and logic
3. Metadata Request – Compiler then requests data
4. Receive Response – Compiler receives response for the request
5. Compiler Response – Compiler checks the data and responds to Driver
6. Execution – Driver sends the complied result to execution engine
7. Map Reduce Job – Execution engine runs the map reduce job
8. Results are fetched – Results from Data nodes are received
9. Driver – Resultant data is received by driver
10. Interface – Driver displays the data in the HIVE interface.

### HIVE Architecture

The below diagram depicts the architectural framework of HIVE:



### Major Components of HIVE Architecture

1. Metastore – It is the storage responsible to store metadata, which consists of tables, its location and schemas
2. Driver – It is the controller which works with the internal components and the external interface
3. Compiler – It has the task of converting HiveQL to MapReduce Input
4. Executor – It executes tasks after compiling and optimizing
5. Interface – It interacts with the user

### Major Benefits of HIVE

1. Fast in Use – It handles heavy data by using batch processing
2. Familiar to Use – SQL is known interface, so it is accessible to non-programmers as well.
3. Scalable as per needs – Distribution is easier

### Real-Life HIVE Use Cases

1. Airbnb - Airbnb uses Amazon EMR to run Apache Hive on a S3 data lake.
2. Vanguard - Vanguard uses Amazon EMR to run Apache Hive on a S3 data lake.

# MAHOUT

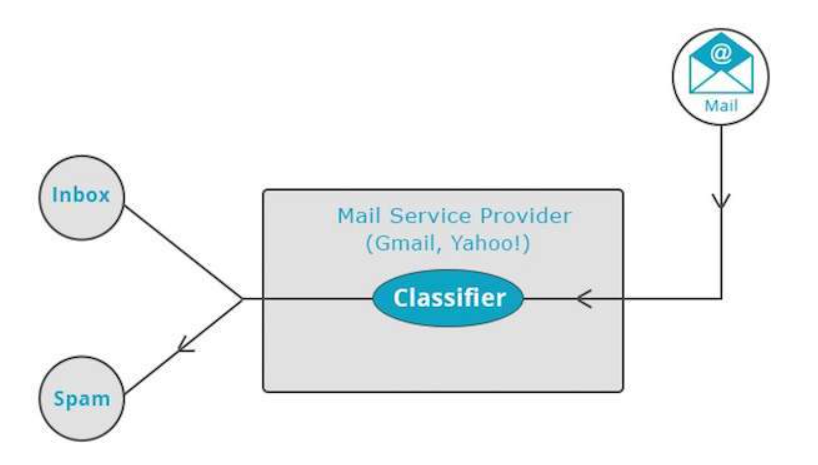
### What is Mahout?

A mahout is one who drives an elephant as its master. The name comes from its close association with Apache Hadoop which uses an elephant as its logo.

Apache Mahout is an open-source project that is primarily used for creating scalable machine learning algorithms. It implements popular machine learning techniques such as:

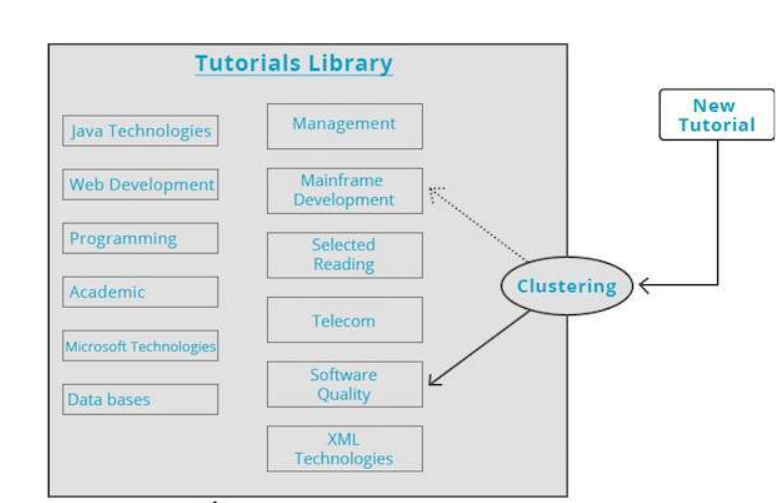
* **Classification**

Mahout was specifically designed for serving as a recommendation engine, employing what is known as a *collaborative filtering* algorithm. Mahout combines the wealth of clustering and classification algorithms at its disposal to produce more precise recommendations based on input data.



* **Clustering**

Clustering is a form of *unsupervised* learning — where the labels for data points are unknown ahead of time and must be inferred from the data without human input (the *supervised* part).



### Features of Mahout

The primitive features of Apache Mahout are listed below.

1. The algorithms of Mahout are written on top of Hadoop, so it works well in distributed environment. Mahout uses the Apache Hadoop library to scale effectively in the cloud.
2. Includes several Map Reduce enabled clustering implementations such as k-means, fuzzy k-means, Canopy, Dirichlet, and Mean-Shift.
3. Supports Distributed Naive Bayes and Complementary Naive Bayes classification implementations.
4. Includes matrix and vector libraries.

### Applications of Mahout

1. Companies such as Adobe, Face book, LinkedIn, Foursquare, Twitter, and Yahoo use Mahout internally.
2. Foursquare helps you in finding out places, food, and entertainment available in a particular area. It uses the recommender engine of Mahout.
3. Twitter uses Mahout for user interest modelling.
4. Yahoo! uses Mahout for pattern mining.

# 4. SPARK

### What is Spark?

Apache Spark is a robust big data processing platform that uses a hybrid framework. A hybrid framework supports batch and stream processing capabilities. In spite of using many of the same principles as Hadoop's MapReduce engine, Spark outperforms MapReduce engine in terms of performance. For example, given the same batch processing workload, spark may be faster than MapReduce due to Spark's "full in-memory computation" feature, as opposed to MapReduce's traditional read from and write to disk. Spark can be used independently or in conjunction with Hadoop to replace the MapReduce engine.

1. Spark Batch Processing Model: The in-memory computation is the most significant advantage of Spark over MapReduce. Spark only interacts with the disk for two purposes: loading data into memory and storing the final results in memory. All intermediate results are stored in memory. Because of this in-memory processing, Spark is significantly faster than its competitor batch processing framework Hadoop. Spark employs Resilient Distributed Datasets to support the in-memory computation feature (RDD). RDD is a read-only data structure that is kept in memory to allow Spark to function as a fault - tolerant framework without actually writing to disk after each operation.
2. Spark Stream Processing Model: In addition to batch processing, Spark supports stream processing via micro-batches. Data streams are treated as a group of very small batches in micro-batching, which would then be handled as regular tasks by the Spark batch engine. Even though this micro-batching procedure appears to work well, it may result in some performance differences when compared to true stream processing frameworks.

### Description of features

Apache Spark has several distinguishing characteristics. The following are some of these characteristics:

• Speed: Apache Spark is a tool which can be used for running Spark applications in Apache Hadoop cluster. Apache Spark is 100 times faster than Apache Hadoop and 10 times faster than disk access. Spark employs the concept of a Resilient Distributed Dataset (RDD) to allow it to precisely store data within memory.

• Usability: Spark allows users to quickly write applications in a variety of programming languages, including Java, Scala, R, and Python. This allows programmers to develop and run their applications in languages they are familiar with, making it easier to develop parallel applications.

• Advanced analytics: Sparks favours SQL queries, data streaming, and more complicated analytics like machine learning and graph algorithms over simple map and reduce operations.

• Runs everywhere: Apache Spark can run on a variety of platforms, including Apache Hadoop YARN, Mesos, EC2, Kubernetes, and in the cloud via the Apache Spark standalone cluster mode. It can retrieve data from HDFS, Cassandra, HBase, and other databases.

• In-memory computing: In-memory cluster computation enables Spark to run iterative machine learning algorithms while also facilitating bilateral querying and data streaming analysis at lightning-fast speeds. Spark stores data in the RAM of the servers so that it can be accessed quickly.

• Real-time stream processing: Spark streaming understands real-time stream processing as well as other configurations, resulting in spark streaming being simple, fault tolerant, and unidirectional.

### Applications of Apache Spark

• Healthcare: Spark is being used in the healthcare sector because it provides a thorough analysis of patient information as well as prior medical data. This assists in identifying which patients are likely to experience medical complications soon, thereby avoiding hospital re-admission and lowering costs for healthcare providers and patients alike, as it is now possible to utilize home care for the identified patient. Additionally, Spark is also used in genomic sequencing because it can minimize the computation needed to process genome data, which previously would take weeks or months to organize all the chemical compounds with genes. MyFitnessPal is a company which makes use of Spark.

• E-commerce: In the e-commerce sector, Spark is being used to discover insights into the real time transactions that are sent to a streaming clustering algorithm like K-means clustering algorithm or alternating least squares. It also improves client recommendations based on current patterns. Spark is used in e-commerce by organizations like Alibaba and eBay.

• Entertainment: In the gaming industry, Apache Spark aids in the recognition of patterns from real-time in-game events and then responding to them to generate profitable business opportunities such as selective advertising, player retention, and the automatic switching of gaming levels based on difficulty. Additionally, video sharing services like as Pinterest, Netflix, and Yahoo employ Spark in conjunction with MongoDB. These websites serve up relevant adverts to their visitors based on the videos they've watched, shared, and browsed.

### When to Avoid Using Spark

Even though Apache Spark's in-memory capabilities are diverse, they aren't necessarily the ideal fit for all use scenarios. Spark, in particular, was not intended to be used in a multi-user context. Spark users must determine whether the amount of RAM available to them is sufficient for a dataset. Adding more users complicates things even more because users will have to coordinate memory use in order to execute many projects at the same time. Because of its inability to manage this level of concurrency, users should consider using an alternative engine for large batch workloads, such as Apache Hive.

# Zookeeper

### What is Zookeeper?

Zookeeper is a distributed, open-source coordination service for distributed applications. It exposes a simple set of primitives that distributed applications can build upon to implement higher level services for synchronization, configuration maintenance, and groups and naming.

### Zookeeper Guarantees:

**Linearizable writes**

All writes are serializable and respect precedence

Order of writes: w1 < w2 < w3 < w4 < w5

However, different zookeeper servers may have committed only a subset of the global order.

**FIFO Client Order**

All requests from a client are executed in the order they came in. This holds true even for clients that initiate asynchronous operations.

If a client issues the following sequence of operations:

1. setData(“/lorem/dipsum/”, “value1”, 1)
2. getData(“/lorem/dipsum/”, false) ---> guaranteed to return version greater than or equal to “value1”

The operations will execute in the order A < B. This means that the read operation at B is guaranteed to see a version of the system that contains operation A, and all write operations that occur before A in the global ordering.

Note: Since only writes need to be linearizable, read operations can be served from local store of each server without coordinating with other servers. This allows the system to scale well for read-heavy workloads

### Ordering guarantees for watch notifications

A client will see a watch event for a znode it is watching before seeing the new data that corresponds to that event.

The order of watch events from ZooKeeper corresponds to the order of the updates as seen by the ZooKeeper service.

Consider the sequence of write operations, w1 < w2, modifying nodes z1 and z2 respectively.

If a client is watching both z1 and z2 before the writes, then it is guaranteed to receive the notification for w1 before w2

Consider this sequence of write operations, w1 < w2, modifying nodes z1 and z2 respectively and setting it to value v1, v2.

If a client is watching z1, then it is guaranteed to see the notification for w1 before it can read the new value of w2.

i.e the client can never see the following order of notification and reads:  
 (read of z2 which returns v2) < (notification for z1 about w1)

**Liveness**If a majority of Zookeeper servers are active and communicating the service will be available

**Durability**

If the Zookeeper service responds successfully to a change request, that change persists across any number of failures as long as a quorum of servers is eventually able to recover

### Zookeeper Implementation:

**Atomic Broadcast**

ZooKeeper uses ZAB for atomic broadcast.

It uses simple majority quorums to decide on a proposal.

Zab guarantees that changes broadcast by a leader are delivered in the order they were sent and all changes from previous leaders are delivered to an established leader before it broadcasts its own changes

The transaction created by the request processor is broadcast to the followers.

**Replicated Database**

Each server maintains an in-memory copy of the znode hierarchy that is stored in the replicated database.

When a follower receives a transaction from the leader through atomic broadcast it applies it to the database.

All updates are logged to the disk before it is applied to the in-memory database.

If the server crashes and recovers, it must reconstruct the replicated database and replaying all transactions could take really long.

ZooKeeper takes periodic ‘fuzzy’ snapshots of the znode hierarchy.

It’s ‘fuzzy’ because the hierarchy isn’t locked while the snapshot is taken and so different znodes could be at different versions of the system.

{SetDataTxn, /foo, f2, 2}

{SetDataTxn, /goo, g2, 2}

{SetDataTxn, /foo, f3, 3}

Snapshot:  
{goo=g1, foo=f3}

This represents an incorrect state because foo has moved forward without goo even though the order of write operations requires goo to be updated to g2 before f3

Fortunately, transactions are idempotent and as long as we replay the transactions in order on the snapshot, a correct state of the replicated database will be restored.

# CONCLUSION

We learnt mainly about HDFS – architecture, different nodes, clients and data replications. Hadoop is a powerful system developed as a platform to support an enormous quantity of varied data applications. It provides an interface to process both structured and complex data thus facilitating heterogeneous data consolidation. Then HIVE and its architecture, major components and major benefits and real-life use cases. Hive performance consists of Start-up Overhead and Context-Switch Overhead. If we believe that start-up overhead is systematic offset of small standard deviation between queries, context-switch overhead affects the performance of Hive between queries. Then MAHOUT, its features and applications. Mahout is a young, open source, scalable machine learning library from Apache, and this book is a practical guide to using Mahout to solve real problems with machine learning techniques. In particular, soon explore recommender engines, clustering, and classification. If you’re a researcher familiar with machine learning theory and you’re looking for a practical how-to guide, or you’re a developer looking to quickly learn best practices from practitioners, this book is for you. Then Spark, its features, applications and when to avoid using it. The Spark ecosystem is growing day by day with new features being added. Also, the industry adoption is also increasing at a very fast pace as Spark has a lot to offer to the Big Data and Data Science world. Then lastly, Zookeeper, guarantees, ordering guarantees and implementations. Zookeeper as a technology is actually simple, but its features are powerful. Arguably, it would be difficult, if not impossible, to create resilient, fault-tolerant distributed Hadoop applications without it.

# REFERENCES

1. <http://svn.apache.org/repos/asf/hadoop/common/tags/release-0.16.3/docs/hdfs_design.pdf>
2. <https://ieeexplore.ieee.org/document/5496972>
3. <https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html>
4. <https://storageconference.us/2010/Papers/MSST/Shvachko.pdf>
5. <https://ycode.org/papers/ds-hdfs.html>
6. <https://mahout.apache.org>
7. https://www.geeksforgeeks.org/hadoop-ecosystem
8. <https://horizonguides.com/guides>
9. <https://en.wikipedia.org/wiki/Apache_Spark>
10. <https://www.researchgate.net/publication/339176824_Apache_Spark_A_Big_Data_Processing_Engine>