International Journal of Recent Engineering Science (IJRES),
ISSN: 2349-7157, *volume 2 Issue 5 September to October 2015*

A Gentle Introduction to Hadoop Platforms

Rabi Prasad Padhy[1], Deepti Panigrahy[2]
[1(]*Technical SME, IBM India Pvt. Ltd., Bangalore, Karnataka, India)*
[2(]*R & D Engineer, Redknee Technologies, Bangalore, Karnataka, India)*

Abstract: *Big fish, eats small fish: Big enterprises eat small enterprises-law of nature cloned into law of economics and technology. A further proof of this law is the popularity of Big Data Hadoop Platforms because it meets the needs of many organizations for flexible data analysis capabilities with an unmatched price-performance curve. This is the era of big data and an increasing number of companies are using to analyze structured and unstructured data due to features like scalability, cost effectiveness, flexibility and fault tolerance. Currently Hadoop is in boom stage and there is a WhatsApp-like movement in Big Data Analytics Market. In this research paper we have focused basic architecture of Hadoop, implementation of HDFS file system and MapReduce Algorithm. We have also briefly discussed on various big data computing Hadoop platforms with the advantages and disadvantages of each platform.*

Keywords: *Hadoop, HDFS, Big Data, MapReduce, Unstructured Data;*

I. INTRODUCTION:

Hadoop is a project from the Apache Software Foundation written in Java. It enables management of petabytes of data in thousands of machines. The inspiration comes from Google's MapReduce and Google File System papers. Hadoop's biggest contributor has been the search giant Yahoo [1]. In recent years Hadoop has become a widely used platform and runtime environment for the deployment of Big Data applications. Big data is a collection of large data sets that include different types such as structured, unstructured and semistructured data. This data can be generated from different sources like social media (Facebook, Twitter, YouTube etc.), audios, images, log files, sensor data, stock market exchanges, transactional applications, sensor web etc. are the main contributors of these data [2]. These large data sets are called Big Data. In terms of relational databases, moving and modifying large volumes of unstructured data into the necessary form for Extraction, Transformation, Loading (ETL) can be both costly and time-consuming. To process or analyze this huge amount of data or extracting meaningful information is a challenging task now a days. Big data exceeds the processing capability of traditional database to capture, manage, and process the voluminous amount of data. Falling storage costs and access to better compute power for less money have made the software more attractive as datasets continue to grow, As a result, many companies are rethinking their approach to traditional enterprise storage and architecture to leverage big data [3]. Hadoop is best suited for Processing unstructured data, Complex parallel information processing, Large Data Sets/Files, Machine Learning Critical fault tolerant data processing, Reports not needed in real time, Queries that cannot be expressed by SQL and Data processing Jobs needs to be faster. These are the key reasons why Hadoop platforms so attractive [4].

II. OVERVIEW OF HADOOP PLATFORM 1.0

Hadoop 1.0 popularized MapReduce programming for batch jobs and demonstrated the potential value of large scale, distributed processing. MapReduce, as implemented in Hadoop 1.0, can be I/O intensive, not suitable for interactive analysis [5]. In Hadoop 1.0, a single Namenode managed the entire namespace for a Hadoop cluster. [6]. The basic architecture of Hadoop 1.0 version we have shown in figure 1.

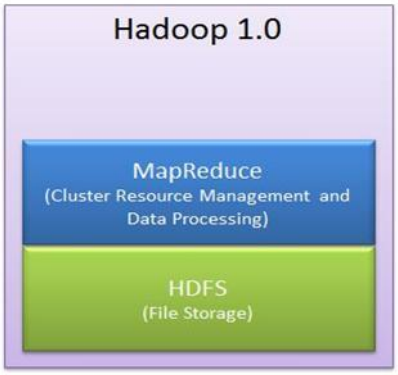


Fig 1. Basic components of Hadoop Platform 1.0

III. ISSUES & LIMITATIONS OF HADOOP 1.0

Issue of Availability: Hadoop 1.0 Architecture had only one single point of availability i.e. the Job Tracker, so in case if the Job tracker fails then all the jobs will have to restart [7].

Issue of Scalability: The Job Tracker runs on a single machine performing various tasks such as Monitoring, Job Scheduling, Task Scheduling and Resource Management. In spite of the presence of several machines (Data Nodes), they were not being utilized in an efficient manner, thereby limiting the scalability of the system.

Cascading Failure Issue: When the number of nodes is greater than 4000 in a cluster, some kind of fickleness is observed Hadoop 1.0 architecture.

Multi-Tenancy Issue: The major issue with Hadoop MapReduce that paved way for the advent of Hadoop YARN was multi-tenancy. With the increase in the size of clusters in Hadoop systems, the clusters can be employed for a wide range of models.

IV. HADOOP 2.0 (YARN) AND ITS COMPONENTS

YARN (Yet Another Resource Negotiator) is a new component added in Hadoop 2.0. YARN is backward compatible existing MapReduce job can run on Hadoop 2.0 without any change. In Hadoop 2.0 a new layer called YARN has been introduced between HDFS and MapReduce. YARN is responsible for doing Cluster Resource Management i.e (Memory, CPU etc) managing the resources of the Hadoop Clusters [8]. No more JobTracker and TaskTracker needed in Hadoop 2.0. We have given the high-level architecture of hadoop 2.0 version in figure 2.

In Hadoop 2.0, the Job Tracker in YARN mainly depends on 3 important components.

1. Resource Manager Component: This component is considered as the negotiator of all the resources in the cluster. Resource Manager is further categorized into an Application Manager that will manage all the user jobs with the cluster and a pluggable scheduler. This is a relentless YARN service that is designed for receiving and running the applications on the Hadoop Cluster. In Hadoop 2.0, a MapReduce job will be considered as an application [12].

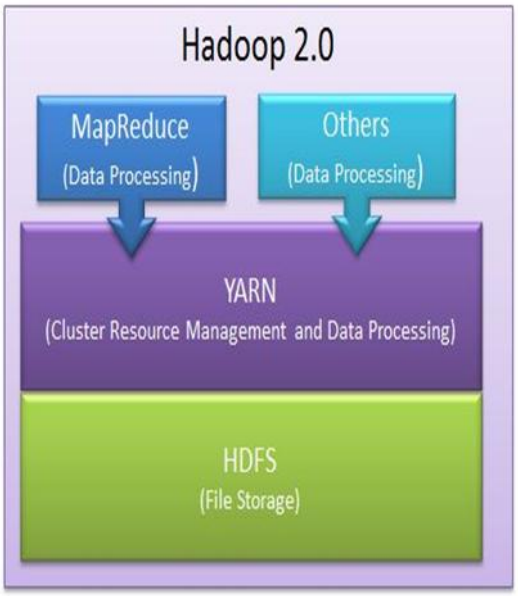


Fig 2. Basic components of Hadoop Platform 1.0

2. Node Manager Component: This is the job history server component of YARN which will furnish the information about all the completed jobs. The Node Manager keeps a track of all the users' jobs and their workflow on any particular given node.

3. Application Master Component: (User Job Life Cycle Manager): This is the component where the job actually resides and the Application Master component is responsible for managing each and every Map Reduce job and is concluded once the job completes processing.

Advantage of YARN

- There are no more fixed map-reduce slots. YARN provides central resource manager. With YARN, now we can run multiple applications in Hadoop, all sharing a common resource.
- Yarn can even run application that do not follow MapReduce model.
- YARN decouples MapReduce resource management and scheduling capabilities from the data processing component, enabling Hadoop to support more varied processing approaches and a broader array of applications. For example, Hadoop clusters can now run interactive querying and streaming data applications simultaneously with MapReduce batch jobs [12].

V. HADOOP ECOSYSTEM

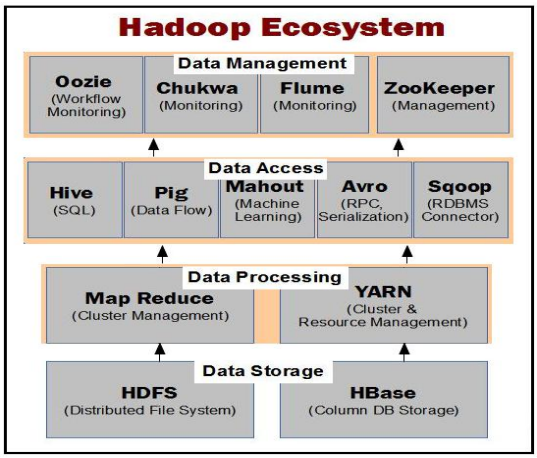


Fig 3. Different Layers of Hadoop Ecosystem

1. Data Storage Layer: This is where the data is stored in a distributed file system, consist of HDFS and HBase ColumnDB Storage.

a. **HDFS (Hadoop Distributed File System):** It is a Java based file system providing scalable and reliable data storage, designed to span large clusters of commodity servers
b. **HBase:** A non-relational (NoSQL) columnar database running on top of HDFS

2. Data Processing Layer: It is the layer where the scheduling, resource management and cluster management to be calculated here. YARN job scheduling and cluster resource management with MapReduce are located in this layer.

a. **MapReduce:** A programming model for large scale data processing.

YARN: The resource management and processing components in Hadoop 2.

3. Data Access Layer: This is the layer where the request from Management layer was sent to Data Processing Layer.

a. **Hive:** A data warehouse infrastructure
b. built on Hadoop, providing a mechanism to project structure onto the data and query it using SQL like language – HiveQL
c. **Pig:** It allows writing complex MapReduce jobs using a scripting language – PigLatin
d. **Mahout:** A Scalable machine learning and data mining library
e. **Avro:** Data serialization system.
f. **Sqoop:** Tool for transferring bulk data between Hadoop and structured databases e.g. RDBMS

4. Management Layer: This is the layer that meets the user. User access the system through this layer which has the components like: Chukwa, ZooKeeper.

- **Chukwa:** A data collection system for managing large distributed system.
- **ZooKeeper:** High-performance coordination service for distributed applications.
- **Flume:** Service for integrating large amounts of streaming data (e.g. logs) into HDFS.
- **Oozie:** Java web application used to schedule Hadoop jobs.

VI. HADOOP ARCHITECTURE

Hadoop cluster has three components:

1. Client Node: Client machines are responsible for loading data into the cluster, submitting MapReduce jobs and viewing the results of the job once complete. The client applications access the file system via the HDFS client. HDFS supports the basic operations e.g. read, write and delete files along with and operations to create and delete directories. The client's reference these files and directories by their paths in the namespace. When a client application reads a file, the HDFS client first checks the NameNode for the list of DataNodes which host the replicas of the blocks of the file [13]. This list is sorted by the network topology distance from the client location. The client then contacts the DataNode directly and requests to transfer the desired block. When a client writes, it first seeks the DataNode from the NameNode. The client then organizes a pipeline from node-to-node and starts sending the data. Once the initial block is filled,

client requests for new DataNodes. These DataNodes are to be chosen to host replicas of the next block. A fresh pipeline is then organized, and the client sends further bytes of the file. Choice of DataNodes for every single block is different.

2. Master Node: It consists of NameNode and Secondary namenode which manages the file system namespace operations like opening, closing, and renaming files and directories and determines the mapping of blocks to DataNodes along with regulating access to files by clients [9].

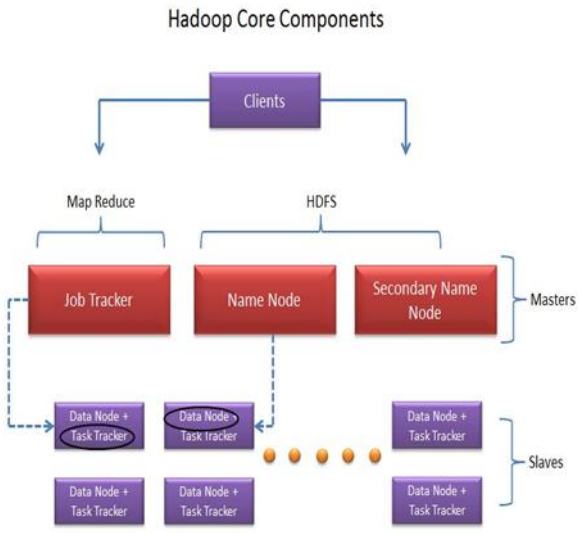
JobTracker (Master process): In Hadoop, The JobTracker (a java process) is responsible for monitoring the job, managing the MapReduce phase, managing the retries in case of errors. The primary functions of JobTracker are resource management, tracking resource availability, and task process cycle. JobTracker identifies the TaskTracker to perform certain tasks and monitors the progress and status of a task. JobTracker is a single point of failure for the MapReduce process [13].

3. Slaves Node: DataNodes which are responsible for serving read and write requests from the file system's clients along with perform block creation, deletion, and replication upon instruction from the Master (NameNode).

TaskTracker (Slave process): The TaskTrackers (Java process) are running on the different DataNodes. Each TaskTracker executes the tasks of the job on the locally stored data. TaskTracker is the slave daemon process that performs a task assigned by JobTracker. TaskTracker sends heartbeat messages to JobTracker periodically to notify about the free slots and sends the status to JobTracker about the task and checks if any task has to be performed.

The architecture of HDFS is shown in Figure 4. HDFS uses the Master / Slave architecture. HDFS mainly consists of the following components: Client, NameNode, Secondary NameNode and DataNode. These components are described separately.

Fig 4. Hadoop Cluster Core Components.



VII. HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

Hadoop Distributed File System (HDFS) is a Java-based file system that provides scalable and reliable data storage that is designed to span large clusters of commodity servers. Files and directories are represented on the NameNode by inodes. Inodes record attributes like permissions, modification and access times, or namespace and disk space quotas. The HDFS file content is split into large blocks (typically 128 megabytes), and each block of the file is independently replicated at multiple DataNodes. A block is the smallest unit of data that can be stored or retrieved from the disk. HDFS stores all the data in terms of block. The file is split into multiple blocks based on the block size of the cluster. If the file was a 128 MB file or 256 MB file, it will have 4 MB blocks; which means the file will go into 4 blocks. If the replication factor is 1, there will be four blocks.The namenode is basically managing the complete file system [13]. The blocks are stored on the local file system on the datanodes. The Namenode actively monitors the number of replicas of a block. When a replica of a block is lost due to a DataNode failure or disk failure, the NameNode creates another replica of the block. The NameNode maintains the namespace tree and the mapping of blocks to DataNodes, holding the entire namespace image in RAM. The NameNode does not directly send requests to DataNodes. It sends instructions to the DataNodes by replying to heartbeats sent by those DataNodes. The

instructions include commands to: replicate blocks to other nodes, remove local block replicas, re-register and send an immediate block report, or shutdown the node [14].

Advantages of HDFS block:

- HDFS blocks are of fixed size, so it is very easy to calculate the number of blocks that can be stored on a disk.
- HDFS block concept simplifies the storage of the datanodes. The datanodes doesn't need to concern about the blocks metadata data like file permissions etc. The namenode maintains the metadata of all the blocks.
- If the size of the file is less than the HDFS block size, then the file does not occupy the complete block storage.
- As the file is chunked into blocks, it is easy to store a file that is larger than the disk size as the data blocks are distributed and stored on multiple nodes in a hadoop cluster.
- Blocks are easy to replicate between the data nodes and thus provide fault tolerance and high availability. Hadoop framework replicates each block across multiple nodes (default replication factor is 3). In case of any node failure or block corruption, the same block can be read from another node.

Read Operation In HDFS:

Data read request is served by HDFS, NameNode and DataNode. Let's call reader as a 'client'. Below figure 5 depicts file read operation in Hadoop [13].

1. Client initiates read request by calling 'open()' method of FileSystem object.
2. This object connects to namenode using RPC (Remote Procedure Call) and gets metadata (location of blocks) information.
3. Addresses of the DataNodes having copy of that block, is returned back.
4. Once addresses of DataNodes are received, an object of type FSDataInputStream is returned to the client. FSDataInputStream contains DFSInputStream which takes care of interactions with DataNode and NameNode. In step 4 shown in above diagram, client invokes 'read()' method which causes DFSInputStream to establish a connection with the first DataNode with the first block of file.
5. Data is read in the form of streams wherein client invokes 'read()' method repeatedly. This process of read() operation continues till it reaches end of block.
6. Once end of block is reached, DFSInputStream closes the connection and moves on to locate the next DataNode for the next block
7. Once client has done with the reading, it calls close() method.

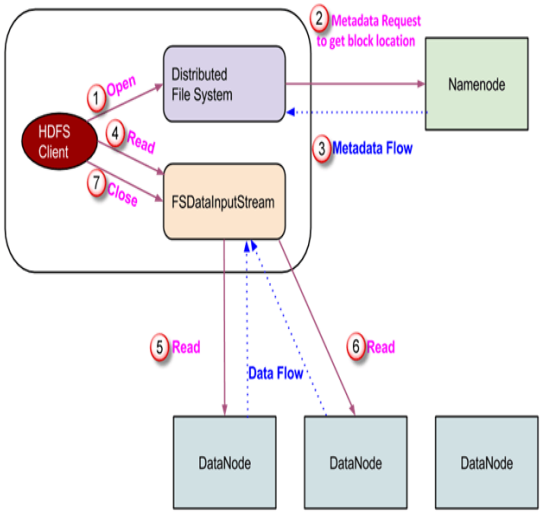


Fig 5. Read Operation In HDFS

Write Operation In HDFS: In this section, we will understand how data is written into HDFS through files [14].

1. Client initiates write operation by calling 'create()' method of DistributedFileSystem object which creates a new file shown in fig 6.
2. DistributedFileSystem object connects to the NameNode using RPC call and initiates new file creation. NameNode verifies that the file (which is being created) does not exist already and client has correct permissions to create new file. If that is not the case then IOException is thrown to client. Otherwise a new record for the file is created by the NameNode.

3. Once new record in NameNode is created, an object of type FSDataOutputStream is returned to the client. Client uses it to write data into the HDFS. Data write method is invoked.
4. FSDataOutputStream contains DFSOutputStream object which looks after communication with DataNodes and NameNode. While client continues writing data, DFSOutputStream continues creating packets with this data. These packets are enqueued into a queue which is called as DataQueue.
5. There is one more component called DataStreamer which consumes this DataQueue. DataStreamer also asks NameNode for allocation of new blocks thereby picking desirable DataNodes to be used for replication.
6. The process of replication starts by creating a pipeline using DataNodes. In our case, we have chosen replication level of 3 and hence there are 3 DataNodes in the pipeline.
7. The DataStreamer pours packets into the first DataNode in the pipeline.
8. Every DataNode in a pipeline stores packet received by it and forwards the same to the second DataNode in pipeline.
9. Another queue, 'Ack Queue' is maintained by DFSOutputStream to store packets which are waiting for acknowledgement from DataNodes.
10. Once acknowledgement for a packet in queue is received from all DataNodes in the pipeline, it is removed from the 'Ack Queue'. In the event of any DataNode failure, packets from this queue are used to reinitiate the operation.
11. After client is done with the writing data, it calls close() method (Step 9 in the diagram) Call to close(), results into flushing remaining data packets to the pipeline followed by waiting for acknowledgement.
12. Once final acknowledgement is received, NameNode is contacted to tell it that the file write operation is complete.

Advantage & Disadvantage of HDFS:

Advantages

- Very large files
- Streaming data access
- Commodity hardware

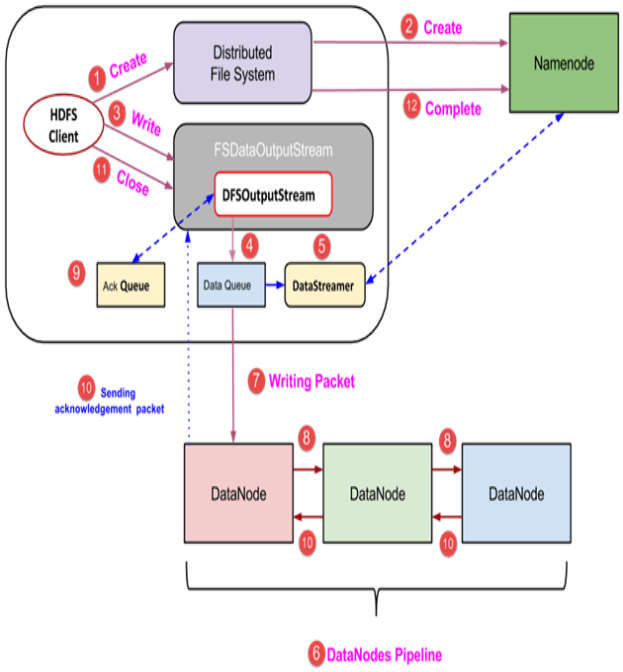


Fig 6. Read Operation In HDFS

Disadvantages

- Low-latency data access
- Lots of small files
- Multiple writers, arbitrary file modifications

VIII. MAPREDUCE

MapReduce is a framework for parallel processing of massive data sets. A job to be performed using the MapReduce framework has to be specified as two phases: the map phase as specified by a Map function (also called mapper) takes key/value pairs as input, possibly performs some computation on this input, and produces intermediate results in the form of key/value pairs; and the reduce phase which processes these results as specified by a Reduce function (also called reducer). The data from the map phase are shuffled, i.e., exchanged and merge-sorted, to the machines performing the reduce phase [15]. It should be noted that the shuffle phase can itself be more time-consuming than the two others depending on network bandwidth availability and other resources. In big data, we want to break a large data set into many smaller pieces and process them in parallel with the same algorithm. With the HDFS, the files are already divided into bite-sized pieces. MapReduce helps in processing all the pieces of this data. MapReduce jobs are complex and involve multiple steps; some steps are

performed by Hadoop with default behavior and can be overridden if needed. The below are the mandatory steps performed in MapReduce in sequence:

Mapper: In MapReduce, Mapper code should have a logic, which can be independent of other block data. Mapper logic should leverage all the parallel steps possible in the algorithm. Input to Mapper is set in the Driver program of a particular Input Format type and file(s) on which the Mapper process has to run. The output of Mapper will be a map <key, value>, key and value set in Mapper output is not saved in HDFS, but an intermediate file is created in the OS space path and that file is read and shuffle and sorting takes place [16].

Shuffle and sorting (Combine): Shuffle and sort are intermediate steps in MapReduce between Mapper and Reducer, which is handled by Hadoop and can be overridden if required. The Shuffle process aggregates all the Mapper output by grouping key values of the Mapper output and the value will be appended in a list of values. So, the Shuffle output format will be a map <key, List<list of values>>.

Reducer: Reducer is the aggregator process where data after shuffle and sort, is sent to Reducer where we have <key, List<list of values>>, and Reducer will process on the list of values. Each key could be sent to a different Reducer. Reducer can set the value, and that will be consolidated in the final output of a MapReduce job and the value will be saved in HDFS as the final output.

Let's see an example of above steps. Consider we have one file splitted into two nodes and content of files are:

1. file1 content : "hello world hello moon"
2. file2 content : "goodbye world goodnight moon"

Now the result of MapReduce after each step will be as below:

```
MAP
First map:                  Second map:
< hello, 1 >                < goodbye, 1 >
< world, 1 >                < world, 1 >
< hello, 1 >                < goodnight, 1 >
< moon, 1 >                 < moon, 1 >

COMBINE
First map:                  Second map:
< moon, 1 >                 < goodbye, 1 >
< world, 1 >                < world, 1 >
< hello, 2 >                < goodnight, 1 >
                            < moon, 1 >
REDUCE
< goodbye, 1 >
< goodnight, 1 >
< moon, 2 >
< world, 2 >
< hello, 2 >
```

Advantages & Disadvantages of MapReduce

Advantages

- The MapReduce model is simple but expressive. A programmer defines a job with only the Map and Reduce functions, and does not have to specify the physical distribution of the job across the nodes.
- MapReduce does not have any dependency on the data model and schema. A programmer can work with unstructured data more easily than they do with a conventional DBMS.
- MapReduce is independent from the underlying storage layers. Thus, it can work with different storage layers such as BigTable and various others.
- It is highly fault tolerant and also highly scalable.

Disadvantages

- MapReduce by itself does not provide support for high-level languages like SQL in DBMS nor for any query optimization technique.
- MapReduce also does not support schemas. Thus, the framework parses each data record at reading input, leading to performance degradation.
- MapReduce offers ease of use with a simple abstraction, but in a fixed dataflow. Hence, several complex algorithms are difficult to implement with only map and reduce functions. Additionally, algorithms that require multiple inputs are not well supported since the MapReduce data flow is designed to read a single input and produce a single output.
- Since the primary goals of MapReduce are scalability and fault tolerance, its operations are not optimized for efficient I/O. Also, map and reduce are both blocking operations. A transition to the next stage cannot be done until all the tasks of the previous stage are completed. There is no support for pipelined execution. Additionally, MapReduce does not have execution plans and does not optimize plans as DBMS does in order to minimize data transfer across nodes. Thus, its performance is often poorer than that of DBMS. The

MapReduce framework also suffers from a latency problem due to batch processing. All the inputs for a job must be prepared in advance for processing.

Key features of Hadoop

- **Distributed:** The data is split into manageable blocks which are stored on different nodes in a network.
- **Scalable:** To increase cluster capacity, add more nodes.
- **Fault-tolerant:** Automatically restarts failed jobs, node failures are inevitable. Replicant copies of each data block are kept on different nodes to avoid data loss upon failure of a node.
- **Open source:** The Hadoop project is managed by the Apache Software Foundation
- **low cost:** Hadoop runs on commodity hardware
- **Resilience:** With built-in fault tolerance, e.g. multiple copies of data replicated on cluster nodes, and with high availability HDFS in version 2.0, Hadoop provides cost-effective resilience to faults and data loss.
- **Flexible:** Hadoop is schema-less, and is capable of absorbing any type of structured or unstructured data from any number of sources. Multiple source data can be joined and aggregated in arbitrary ways enabling deeper analysis that cannot be provided by any one system.

Hadoop use cases for different industries are:

Social Media Engagement and Clickstream Analysis (Web Industry): A clickstream is the recording of the parts of the screen a computer user clicks on web while browsing or using another software application. Clickstream analysis is useful for web activity analysis, and customer behavior software testing, market research, and even for analyzing employee productivity.

Content Optimization and Engagement (Media Industry): Content required to be optimized for rendering on different devices supporting different content formats. Media companies require large amount of content to be processed in different formats. Also content engagement models need to be mapped for feedback and enhancements.

Network Analytics and Mediation (Telecommunication Industry): Telecommunication companies generate a large amount of data in the form of usage transaction data, network performance data, cell-site information device level data and other forms of back office data. The real time analytics plays a critical role in reducing the OPEX and enhancing the user experience.

Targeting and Product Recommendation (Retail Industry): The retail companies and e-Commerce companies model the data from different sources to target customers and provide product recommendations based on end user's profile and usage patterns.

Risk Analysis, Fraud Monitoring and Capital Market Analysis (BFSI Industry): Banking and finance sectors have large sets of structured and unstructured data generated by different sources like trading pattern in capital markets, consumer behavior for banking services etc. Financial institutions use big data to perform Risk Analysis, Fraud Monitoring and Tracking, Capital Market Analysis, converged data management etc.

Key Adopters of Hadoop

The early adopters of Hadoop are the web giants like Facebook, Yahoo, Google, LinkedIn, Twitter etc. In Table 1 we have given the classification of companies in Big Data & Hadoop Space.

Facebook: Facebook uses Hadoop – Hive and HBase for data warehousing (over 300 PB in aggregate and over 600 TB daily data inflows) and real-time application, serving up dynamic pages customized for each of its over 1.2 billion users.
Yahoo: Yahoo uses Hadoop and Pig for data processing and analytics, web search, email antispam and ad serving with more than 100,000

CPUs in over 40,000 servers running Hadoop with 170 PB of storage.

Google: Google used MapReduce to create its web index from crawl data and also uses Hadoop clusters on its cloud platform with Google Compute Engine (GCE).

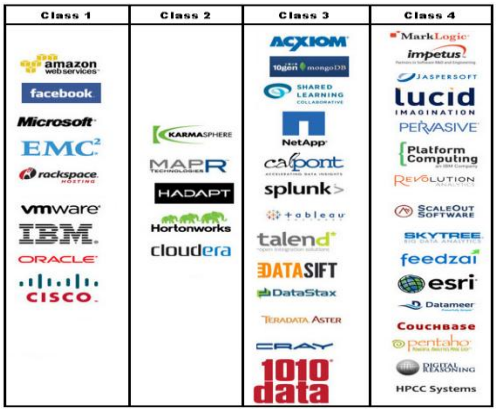
LinkedIn: LinkedIn uses Hadoop for data storage and analytics driving personalized recommendations like "People you may know" and ad targeting.

Twitter: Twitter uses Hadoop – Pig and HBase for data visualization, social graph analysis and machine learning.

Classification of Companies in Big Data & Hadoop Space

- **Class 1:** Companies who have adopted Hadoop as Big Data Strategy. Class 1 companies are using Class 2 companies are partner to excel in Big Data space.
- **Class 2:** Companies who have taken Hadoop and productized it.
- **Class 3:** These companies are creating products which are adding value to overall Hadoop eco-system.
- **Class 4:** These companies are consuming or providing Hadoop based services to other companies on smaller scale compared to class 1 and class 2 companies.

Table 1. Classification of Companies in Big Data & Hadoop Space



IX. HADOOP PLATFORMS

There are two types of Hadoop Distributions i.e Open Source & Commercial. Open Source Hadoop Distributions examples are Apache Hadoop, Hortonworks Hadoop [18], [19]. The Commercial Hadoop Distributions examples are cloudera, Amazon EMR, Microsoft HDinsight [9], [10].

Apache Hadoop: Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware [9]. All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are commonplace and thus should be automatically handled in software by the framework. Figure 7 is all about the basic architecture view of Apache Hadoop components.

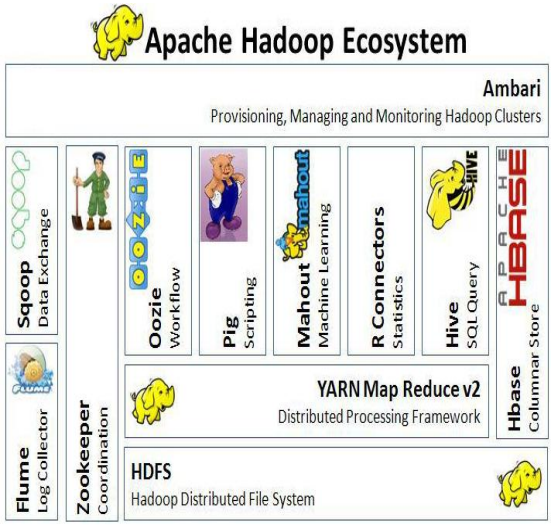


Fig 7. Basic Architecture of Apache Hadoop

Cloudera Hadoop: Cloudera Inc. was founded by big data geniuses from Facebook, Google, Oracle and Yahoo in 2008. It was the first company to develop and distribute Apache Hadoop-based software and still has the largest user base with most number of clients. Although the core of the distribution is based on Apache Hadoop, it also provides a proprietary Cloudera Management Suite to automate the installation process and provide other services to enhance convenience of users which include reducing deployment time, displaying real time nodes' count, etc. Cloudera CDH can be run on windows server, HDP is available as a native component on the windows server [10]. A Windows-based Hadoop cluster can be deployed on Windows Azure through HDInsight Service. Besides the core Hadoop platform (HDFS,

MapReduce, Hadoop Commons), CDH integrates 10 open source projects including HBase, Mahout, Pig, ZooKeeper, and others shown in figure 8.

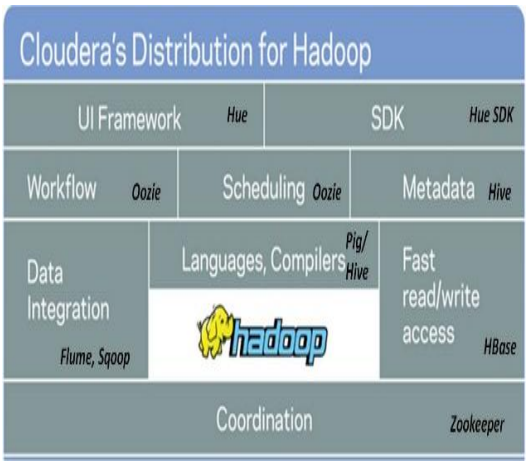


Fig 8. High-level Architecture of Cloudera Hadoop

Hortonworks: Hortonworks, one of the leading vendors of Hadoop provides open source platform based on Apache Hadoop for analysing, storing and managing big data. It is the only commercial vendor to distribute complete open source Apache Hadoop without additional proprietary software [11]. It distribution HDP2.0 can be directly downloaded from their website free of cost and is easy to install. The components of Hortonworks has been given in figure 9.

Amazon Elastic MapReduce (Amazon EMR): Amazon Elastic MapReduce (EMR) is Amazon's packaged Hadoop offering. Rather than building Hadoop deployments manually on EC2 (Elastic Compute Cloud) clusters, users can spin up fully configured Hadoop installations using simple invocation commands, either through the AWS Web Console or through command-line tools [18]. Several of the popular Hadoop tools are available as options, including Hive, Pig, and HBase. Amazon EMR removes most of the cumbersome details of Hadoop, while taking care of provisioning of Hadoop, running the job flow, terminating the job flow, moving the data between Amazon EC2 and Amazon S3, and optimizing Hadoop shown in figure 10. We can also run other popular distributed frameworks such as Spark and Presto in Amazon EMR, and interact with data in other AWS data stores such as Amazon S3 and Amazon DynamoDB. Amazon EMR securely and reliably handles big data use cases, including log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, and bioinformatics.

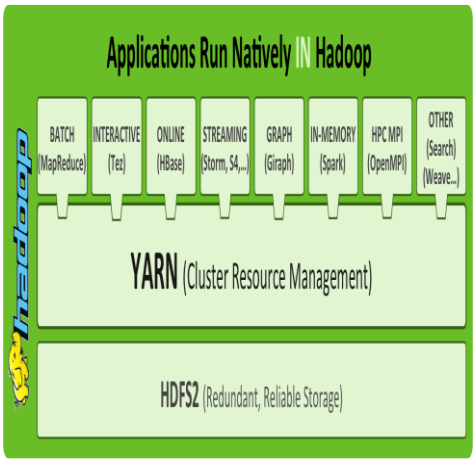


Fig 9. The Hortonworks Hadoop architecture

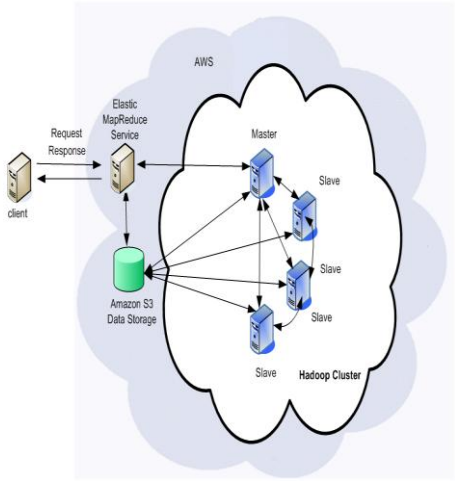


Fig 10. High-level Architecture of Amazon EMR

MapR: MapR is a complete enterprise-grade distribution for Apache Hadoop. It engineered to improve Hadoop's reliability, performance, and ease of use. The MapR distribution provides a full Hadoop stack that includes the MapR File System (MapR-FS), MapReduce, a complete Hadoop ecosystem, and the MapR Control System user interface which shown in figure 11. We can use MapR with Apache Hadoop, HDFS, and MapReduce APIs [19]. The major differences to CDH and HDP is that MapR uses their proprietary file system MapR-FS instead of HDFS.

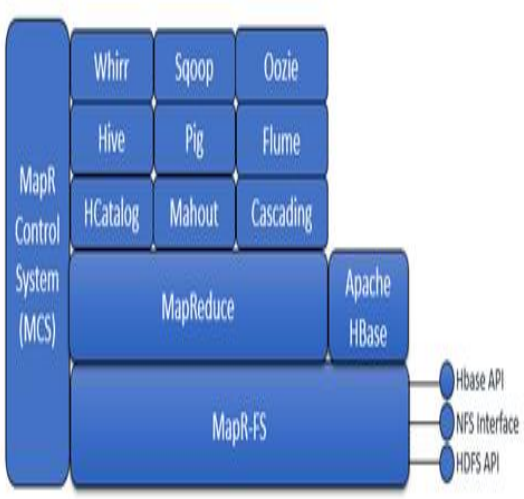


Fig 11. High-level view of the MapR Hadoop Distribution

Microsoft HDInsight: HDInsight is a cloud implementation on Microsoft Azure of the rapidly expanding Apache Hadoop technology stack that is the go-to solution for big data analysis [20]. It includes implementations of Storm, HBase, Pig, Hive, Sqoop, Oozie, Ambari, and so on which shown in figure 12. HDInsight also integrates with business intelligence (BI) tools such as Excel, SQL Server Analysis Services, and SQL Server Reporting Services. Windows Azure HDInsight Service uses Azure Blob Storage as the default file system (or we can store it in the native Hadoop Distributed File System (HDFS) file system that is local to the compute nodes but would lose the data if we deleted our cluster). There is a thin layer over Azure Blob Storage that exposes it as an HDFS file system called Windows Azure Storage-Blob or WASB. Azure HDInsight deploys and provisions Apache Hadoop clusters in the cloud, providing a software framework designed to manage, analyze, and report on big data with high reliability and availability. HDInsight uses the Hortonworks Data Platform (HDP) Hadoop distribution.

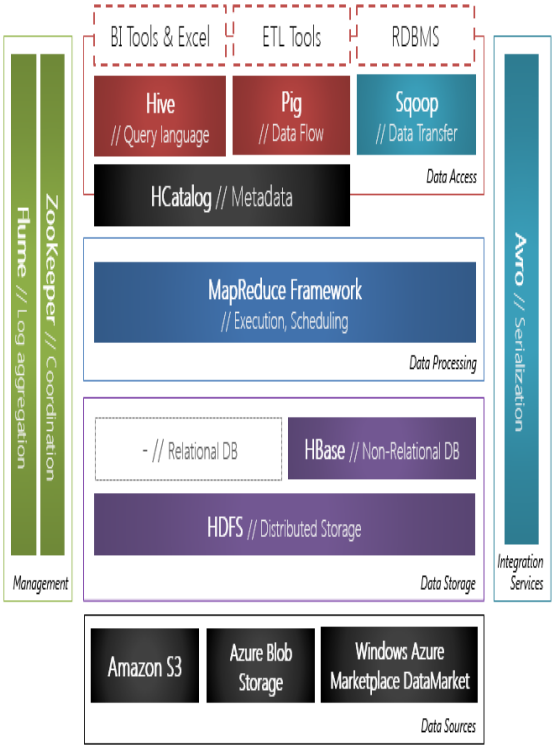


Fig 12. High-level view of the Microsoft HDInsight

X. HADOOP ARCHITECTURE COMPARISON

Hadoop introduced a new way to simplify the analysis of large data sets, and in a very short time reshaped the big data market. In fact, today Hadoop is often synonymous with the term big data. Since Hadoop is an open source project, a number of vendors have developed their own distributions, adding new functionality or improving the code base. In table 2 we have given in details of the Hadoop platform comparison.

XI. CHALLENGES OF ADOPTING HADOOP

Hadoop adoption has grown and companies increasingly rely on Hadoop, a few shortcomings have limited its potential business value [22].

- Mixed workloads and multi-tenancy environments cause jobs to fight for resources: Hadoop schedulers still based on pre-allocating resources when a job starts. The problem is that jobs use a varying mix of different hardware resources during the

course of their lifetime [21]. In addition, some hardware resources (such as disk I/O and network) aren't limited in standard Hadoop. Both of these factors lead to competition for resources that ideally should be arbitrated at runtime, resulting in work not being completed in time or at all.

Table 2. Comparison of various Hadoop Distributions

	Hortonworks	Cloudera	MapR
Performance and Scalability			
Data Ingest	Batch	Batch	Batch and streaming writes
Metadata Architecture	Centralized	Centralized	Distributed
HBase Performance	Latency spikes	Latency spikes	Consistent low latency
NoSQL Applications	Mainly batch applications	Mainly batch applications	Batch and online/real-time applications
Dependability			
High Availability	Single failure recovery	Single failure recovery	Self healing across multiple failures
MapReduce HA	Restart jobs	Restart jobs	Continuous without restart
Upgrading	Planned downtime	Rolling upgrades	Rolling upgrades
Replication	Data	Data	Data + metadata
Snapshots	Consistent only for closed files	Consistent only for closed files	Point-in-time consistency for all files and tables
Disaster Recovery	No	File copy scheduling (BDR)	Mirroring
Manageability			
Management Tools	Ambari	Cloudera Manager	MapR Control System
Volume Support	No	No	Yes
Heat map, Alarms, Alerts	Yes	Yes	Yes
Integration with REST API	Yes	Yes	Yes
Data and Job Placement Control	No	No	Yes
Data Access			
File System Access	HDFS, read-only NFS	HDFS, read-only NFS	HDFS, read/write NFS (POSIX)
File I/O	Append only	Append only	Read/write
Security: ACLs	Yes	Yes	Yes
Wire-level Authentication	Kerberos	Kerberos	Kerberos, Native

- Troubleshooting is difficult and can take hours: Although there are a multitude of tools that allow users to monitor their clusters, administrators are often left with an incomplete view of the factors affecting cluster health. It difficult to isolate the root cause of problems and drives a lot of inefficient behavior, such as guess-and-check restarting and asking users about jobs they submitted. As cluster size grows and businesses increasingly rely on Hadoop, such methods will become unsustainable. Also need for advanced systems administration and analyst capabilities when working with Hadoop.

- Buying more hardware than needed: To compensate for their lack of control over cluster resources, organizations usually size their clusters based on anticipated peak loads. The goal is to ensure that jobs don't overload the cluster and lead to massively degraded performance, job failures, or worse. However, because of Hadoop's inefficient, up-front allocation of resources, this strategy is expensive and leaves capacity unused much of the time – and can still fail to prevent undesired outcomes as workloads are often unpredictable.

- **Security Challenges:** Originally Hadoop was developed without security in mind,no security model, no authentication of users and services and no data privacy, so anybody could submit arbitrary code to be executed. Although auditing and authorization controls (HDFS file permissions and ACLs) were used in earlier distributions, such access control was easily evaded because any user could impersonate any other use. Organizations face the risk of even further reduced control if Hadoop clusters are deployed in a cloud environment. When used in an enterprise environment, the importance of security becomes paramount. Organizations must protect sensitive customer, partner, and internal information and adhere to an ever-increasing set of compliance requirements. Some of the threats already identified in Hadoop platforms like an

unauthorized user may access an HDFS file via the RPC or via HTTP protocols and could execute arbitrary code or carry out further attacks [22]. An unauthorized client may read/write a data block of a file at a DataNode via the pipeline streaming Data-transfer protocol. An unauthorized client may gain access privileges and may submit a job to a queue or delete or change priority of the job. DataNodes imposed no access control, a unauthorized user could read arbitrary data blocks from Data Nodes, bypassing access control mechanism/restrictions, or writing garbage data to DataNode.

XII. CONCLUSION

It's becoming clear that Hadoop platforms changes the economics and dynamics of Big data analytics due to its scalability, cost effectiveness, flexibility, and built-in fault tolerance. It makes possible the massive parallel computing that today's data analysis requires. Currently, IT organizations and independent users must carefully strategize their approach to dealing with big data to avoid being overrun with data that has no intrinsic value due to the lack of adequate processing tools. To truly realize the promise of Hadoop platforms and its distributed set of resources for big data analysis, businesses and end-users need to expand their approach by relying on the wealth of resources currently available like access to professional training, commercial platform implementation. However, the proper skillset training will be necessary to achieve large-scale data analysis. That's why commercial providers Hadoop Platforms offer such great value to companies. These integrated management features enable the platform to be implemented by a wide range of users at all levels of skill expertise. Organizations can then make appropriate business decisions based on the large amounts of data they accrue by accessing the power of a relatively low-cost, highly scalable infrastructure such as Hadoop to tackle the challenges of big data.

REFERENCE

[1] Padhy, Rabi Prasad. "Big Data Processing with Hadoop-MapReduce in Cloud Systems." International Journal of Cloud Computing and Services Science (IJ-CLOSER) 2, no. 1 (2012): 16-27.
[2] S. Ghemawat, H. Gobioff and S-T. Leung, "The Google file system," Proc. of the 19th symposium on Operating Systems Principles, 2003, pp. 29-43.
[3] Padhy, Rabi Prasad, Manas Ranjan Patra, and Suresh Chandra Satapathy. "RDBMS to NoSQL: Reviewing some next-generation non-relational databases." International Journal of Advanced Engineering Science and Technologies 11.1 (2011): 15-30.
[4] Douglas, Laney. "The Importance of 'Big Data': A Definition". Gartner. Retrieved 21 June 2012.
[5] Apache Hadoop: http://hadoop.apache.org/
[6] Apache Hadoop MapReduce Tutorial: http://hadoop.apache.org/docs/r1.0.4/ mapred_tutorial.html
[7] Hadoop Tutorial, Yahoo Developer Network, http://developer.yahoo.com/hadoop/tutorial Hadoop 2.0 (YARN) and Its Components
[8] Dwivedi, Kalpana, and Sanjay Kumar Dubey. "Analytical review on Hadoop Distributed file system." Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference-. IEEE, 2014.
[9] Apache Hadoop homepage (https://hadoop.apache.org/)
[10] Cloudera homepage (http://www.cloudera.com/content/cloudera/en/home.html)
[11] Hortonworks HDP homepage (http://hortonworks.com/)
[12] Kulkarni, Amogh Pramod, and Mahesh Khandewal. "Survey on Hadoop and Introduction to YARN." International Journal of Emerging Technology and Advanced Engineering 4.05 (2014): 82-87.
[13] K. Shvachko, H. Huang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in 26th IEEE (MSST2010) Symposium on Massive Storage Systems and Technologies, May 2010.
[14] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, Jan. 2008.
[15] T. Gunarathne, T.-L. Wu, J. Qiu, and G. Fox, "Mapreduce in the clouds for science," in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010, pp. 565–572.
[16] Dean J, Ghemawat S.(2008) MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), p.p.07-113.
[17] V. Kalavri and V. Vlassov, "MapReduce: Limitations, Optimizations and Open Issues."
[18] Amazon Elastic MapReduce homepage (http://aws.amazon.com/elasticmapreduce/)
[19] MapR homepage (http://mapr.com/)
[20] Microsoft HDInsight homepage (cloud, http://www.windowsazure.com/)
[21] Kevin T. Smith "Big Data Security : The Evolution of Hadoop's Security Model"
[22] Vinay Shukla s "Hadoop Security: Today and Tomorrow".