A

Seminar Report on

Data management with Hadoop

Submitted as a partial fulfilment of

DIPLOMA IN INFORMATION TECHNOLOGY ENGINEERING

Submitted By

Miss. Pawar Sejal Ravindra (2201967)

PRN No. - 2030408246063

DEPARTMENT OF INFORMATION TECHNOLOGY ENGINEERING

DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY'S

INSTITUTE OF PETROCHEMICAL ENGINEERING

Lonere, Tal. Mangaon, Dist. Raigad, Maharashtra-402103

Academic Year: 2022-2023

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE OF PETROCHEMICAL ENGINEERING

Lonere, Tal. Mangaon, Dist. Raigad, Maharashtra-402103

CERTIFICATE

This is to certify that the Seminar report entitled

Hadoop-Data Management with Hadoop

Submitted by

Miss.Pawar Sejal Ravindra (2201967)

PRN No. - 2030408246063

is a bonafide work of student of final year Diploma in Information Technology submitted in partial fulfilment for the award of Diploma in Information Technology as prescribed by Dr. Babasaheb Ambedkar Technological University's, Institute of Petrochemical Engineering, Lonere during the academic year 2022-2023.

Prof. S. S. Kamble
Internal Guide

Prof. K. R. Korpe
Seminar Coordinator

Prof. S. M. Gaikwad
Head of Department

Place: Lonere

Date:

Acknowledgement

It is indeed a matter of great pleasure & privilege to be able to present this Seminar on **"Hadoop-Data management technique"** under the valuable guidance of **Prof. S. S. Kamble mam**, thanks for her valuable guidance, advice and constant aspiration to my work.

I have made this report file on the topic Hadoop; I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic. My efforts and wholehearted co-corporation of each and every one has ended on a successful note. I express my sincere gratitude to **Prof. S. S. Kamble mam**, who assisting me throughout the preparation of this topic. I thank her for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it

I would also like to thank to Project Co-ordinator **Prof. K. R. Korpe sir,** my Institute as well as department for giving the opportunity to present this seminar, it will indeed help me to improve my speaking skills and I will be more confident while presenting for the industrial purpose after completion of the studies.

Thank you!

Miss.Pawar Sejal Ravindra

Department of Information Technology

Dr. Babasaheb Ambedkar Technological University's

Institute of Petrochemical Engineering, Lonere

ABSTRACT

We live in on-demand, on-command Digital universe with data prolife ringby Institutions, Individuals and Machines at a very high rate. This data is categories as “Big Data” due to its sheer Volume, Variety, Velocity and Veracit.

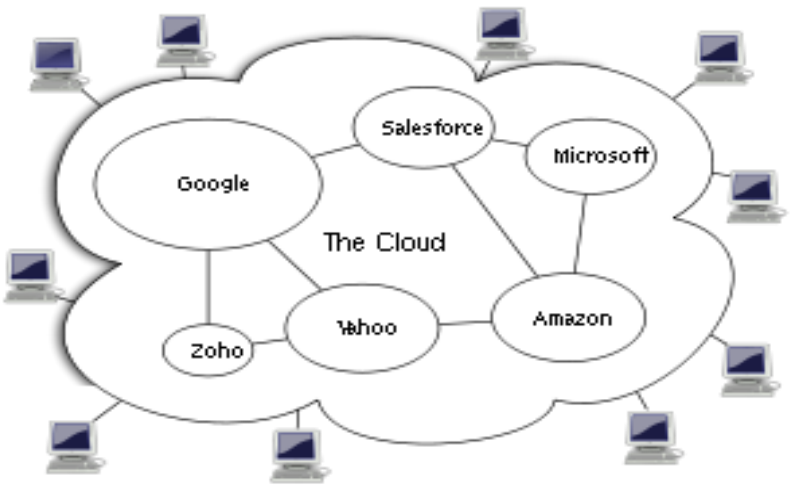
Why do you even need to take care of storing and processing this data? To what purpose? The response is that we need these data to make better and more computational decisions in whatever area we’re working in. Company forecasting is not anything new. It has also been prepared in the past, but with limited data. Businesses must use this data, too ahead of the competition, and then make better decisions. These decisions vary from the presumption of customer desires to the avoidance of fraud well in advance. Professionals in every field can find the reasons for the analysis of this data.

Index

1.INTRODUCTION

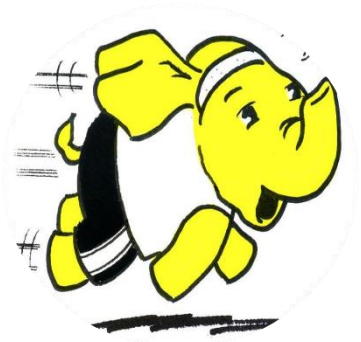
Computing in its purest form, has changed hands multiple times. First, from near the beginning mainframes were predicted to be the future of computing. Indeed mainframes and large scale machines were built and used, and in some circumstances are used similarly today. The trend, however, turned from bigger and more expensive, to smaller and more affordable commodity PCs and servers.

Most of our data is stored on local networks with servers that may be clustered and sharing storage. This approach has had time to be developed into stable architecture, and provide decent redundancy when deployed right. A newer emerging technology, cloud computing, has shown up demanding attention and quickly is changing the direction of the technology landscape. Whether it is Google's unique and scalable Google File System, or Amazon's robust Amazon S3 cloud storage model, it is clear that cloud computing has arrived with much to be gleaned from.



Cloud computing is a style of computing in which dynamically scalable and often virtualize resources are provided as a service over the Internet. Users need not have knowledge of, expertise in, or control over the technology infrastructure in the "cloud" that supports them.

2.Hadoop Technology



HADOOP is Storage System and Analytics Program Based on Apache.

HADOOP is Open Source Framework that Allow you to stored large amount of data in distributed Environment so you can process it Parallelly.

It is implemented by Java.means the language has been used for creating this technology is Hadoop.

Hadoop consists of four main modules:

- Hadoop Distributed File System (HDFS) – A distributed file system that runs on standard or low-end hardware. HDFS provides better data throughput than traditional file systems, in addition to high fault tolerance and native support of large datasets.

- Yet Another Resource Negotiator (YARN) – Manages and monitors cluster nodes and resource usage. It schedules jobs and tasks.

- MapReduce – A framework that helps programs do the parallel computation on data. The map task takes input data and converts it into a dataset that can be computed in key value pairs. The output of the map task is consumed by reduce tasks to aggregate output and provide the desired result.

- Hadoop Common – Provides common Java libraries that can be used across all modules.

3.Big Data

Big data is a collection of large datasets that cannot be processed using traditional computing techniques.

It is not a single technique or a tool, rather it has become a complete subject, which involves various tools, techniques and frameworks.

Hadoop. Definition. Big Data refers to a large volume of both structured and unstructured data. Hadoop is a framework to handle and process this large volume of Big data.

4.What Comes Under Big Data?

Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

- **Black Box Data** − It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.

- **Social Media Data** − Social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.

- **Stock Exchange Data** − The stock exchange data holds information about the 'buy' and 'sell' decisions made on a share of different companies made by the customers.

- **Power Grid Data** − The power grid data holds information consumed by a particular node with respect to a base station.

- **Transport Data** − Transport data includes model, capacity, distance and availability of a vehicle.

- **Search Engine Data** − Search engines retrieve lots of data from different databases.

Thus Big Data includes huge volume, high velocity, and extensible variety of data. The data in it will be of three types.

- **Structured data** − Relational data.
- **Semi Structured data** − XML data.
- **Unstructured data** − Word, PDF, Text, Media Logs.

5.Applications of Hadoop

Banking and security:-

Since the transactions in the banks occur at a faster rate so the financial services are heading towards the use of big data and Hadoop in order to avoid fraud in a much standard fashion. Hadoop can figure out the customer's behaviour and filter it if there is in case any sign of fraud.

Social Media and entertainment:-

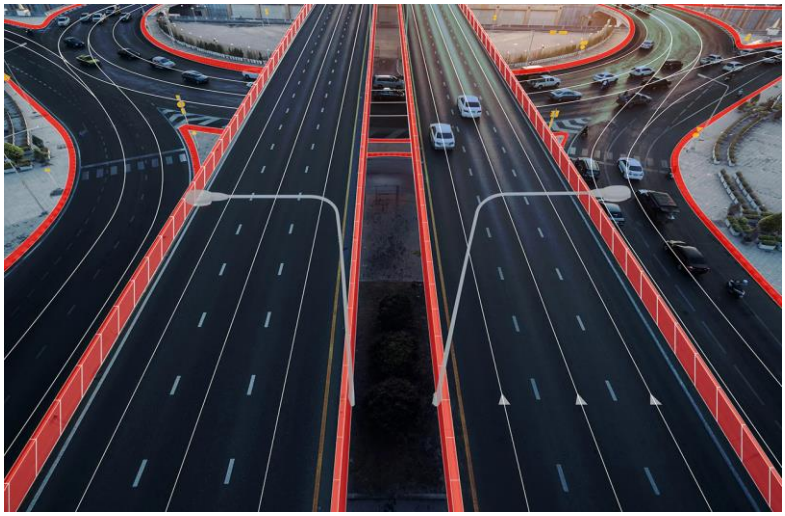
Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data.

Healthcare Provider:-

Hadoop **makes data storage less expensive and more available**:This includes physicians' notes, medical reports, lab results, X-ray, MRI images, vitals and financial data among others. Hadoop provides doctors and researchers the opportunity to find insights from data sets that were earlier impossible to handle.

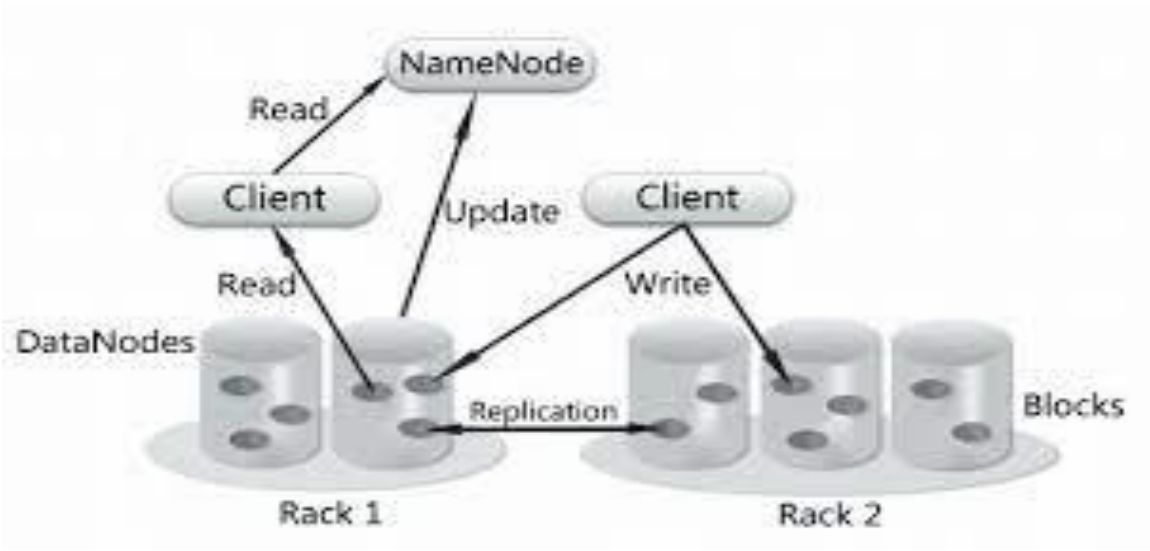
Education:-

Transportation:-



6. Hadoop Approach

Hadoop is designed to efficiently process large volumes of information by connecting many commodity computers together to work in parallel. The theoretical 1000-CPU machine described earlier would cost a very large amount of money, far more than 1,000 single-CPU or 250 quad-core machines. Hadoop will tie these smaller and more reasonably priced machines together into a single cost-effective compute cluster.



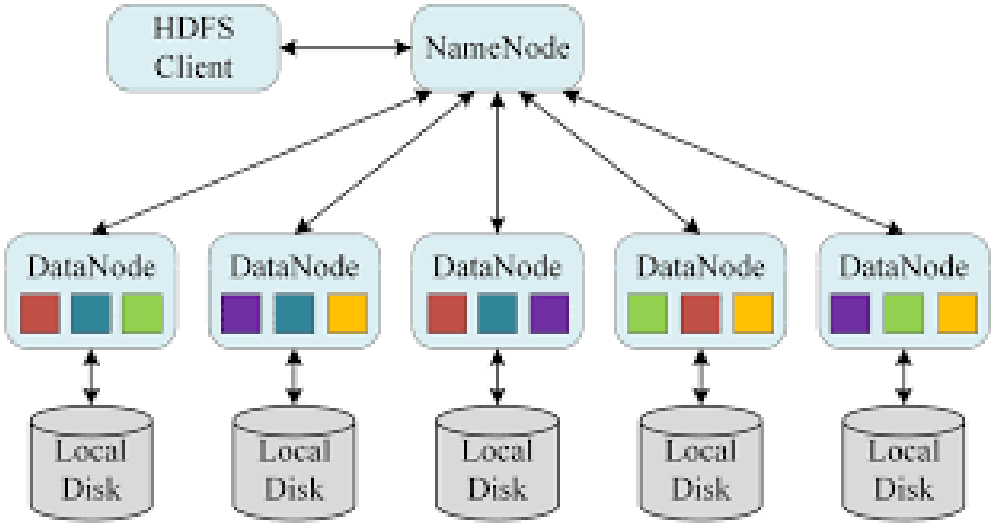
Performing computation on large volumes of data has been done before, usually in a distributed setting. What makes Hadoop unique is its simplified programming model which allows the user to quickly write and test distributed systems, and its efficient, automatic distribution of data and work across machines and in turn utilizing the underlying parallelism of the CPU cores.

In the new Hadoop Approach, instead of fetching the data on local machines we send the query to the data. Obviously, the query to process the data will not be as huge as the data itself. Moreover, at the server, the query is divided into several parts. All these parts process the data simultaneously.

7.Why we used Hadoop?

Hadoop cluster system:-

Apache Hadoop is an open source, Java-based, software framework and parallel data processing engine. It enables big data analytics processing tasks to be broken down into smaller tasks that can be performed in parallel by using an algorithm (like the MapReduce algorithm), and distributing them across a Hadoop cluster. A Hadoop cluster is a collection of computers, known as nodes, that are networked together to perform these kinds of parallel computations on big data sets.

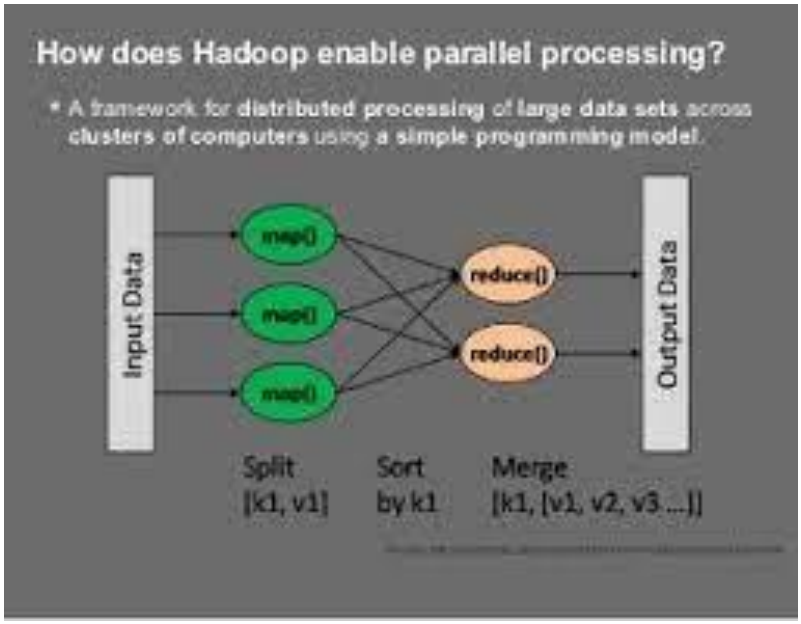


Unlike other computer clusters, Hadoop clusters are designed specifically to store and analyze mass amounts of structured and unstructured data in a distributed computing environment. Further distinguishing Hadoop ecosystems from other computer clusters are their unique structure and architecture. Hadoop clusters consist of a network of connected master and slave nodes that utilize high availability, low-cost commodity hardware. The ability to linearly scale and quickly add or subtract nodes as volume demands makes them well-suited to big data analytics jobs with data sets highly variable in size.

Platform for Massively Scalable Applications:-

Hadoop's scalability comes from the fact that the map and reduce operations can be run in parallel across several machines by breaking the input into smaller chunks. This is mainly handled by the machine playing the role of Job Tracker node in the cluster.

Enable parallel data processing:-

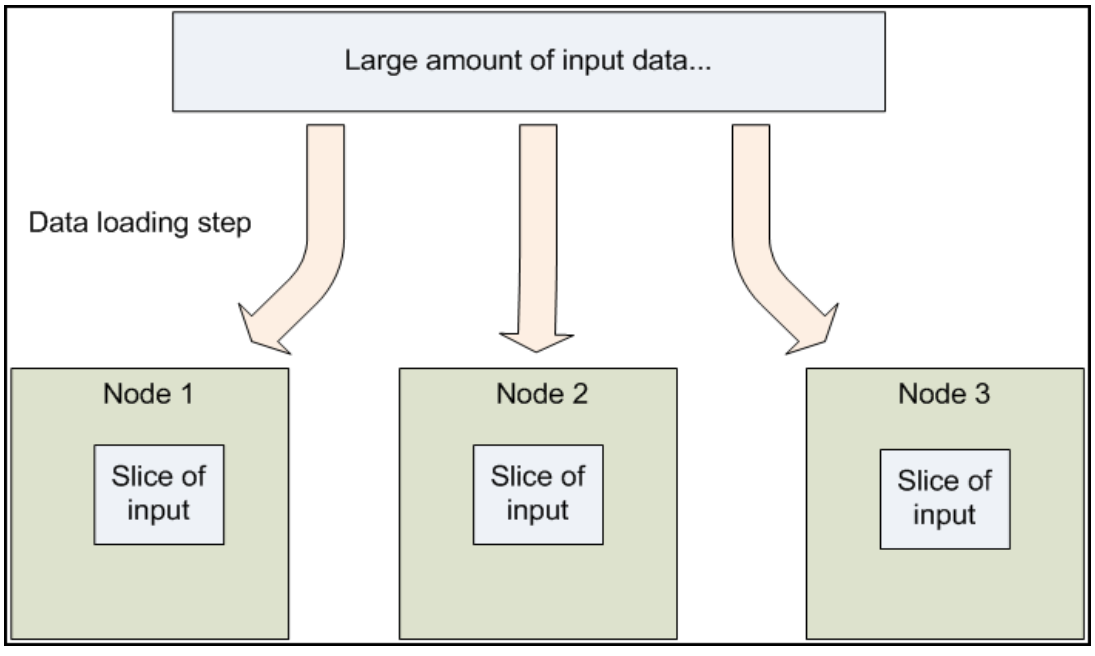


Hadoop allows parallel and distributed processing. Each feature selector can be divided into subtasks and the subtasks can then be processed in parallel. Multiple feature selectors can also be processed simultaneously (in parallel) allowing multiple feature selectors to be compared.

8.Data Distribution

In a Hadoop cluster, data is distributed to all the nodes of the cluster as it is being loaded in. The Hadoop Distributed File System (HDFS) will split large data files into chunks which are managed by different nodes in the cluster. In addition to this each chunk is replicated across several machines, so that a single machine failure does not result in any data being unavailable. An active monitoring system then re-replicates the data in response to system failures which can result in partial storage. Even though the file chunks are replicated and distributed across several machines, they form a single namespace, so their contents are universally accessible.

Data is conceptually **record-oriented** in the Hadoop programming framework. Individual input files are broken into lines or into other formats specific to the application logic. Each process running on a node in the cluster then processes a subset of these records. The Hadoop framework then schedules these processes in proximity to the location of data/records using knowledge from the distributed file system. Since files are spread across the distributed file system as chunks, each compute process running on a node operates on a subset of the data. Which data operated on by a node is chosen based on its locality to the node: most data is read from the local disk straight into the CPU, alleviating strain on network bandwidth and preventing unnecessary network transfers. This strategy of **moving computation to the data**, instead of moving the data to the computation allows Hadoop to achieve high data locality which in turn results in high performance.

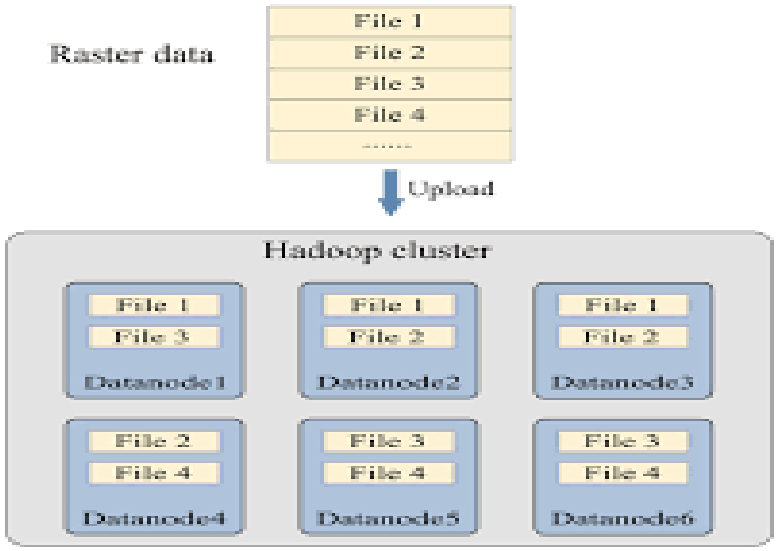


In a

Hadoop cluster, data is distributed to all the nodes of the cluster as it is being loaded in. The Hadoop Distributed File System (HDFS) will split large data files into chunks which are managed by different single machine failure does not result in any data being unavailable. An active monitoring system then re-replicates the data in response to system failures which can result in partial storage. Even though the file chunks are replicated and distributed across several machines, they form a single namespace, so their contents are universally accessible.

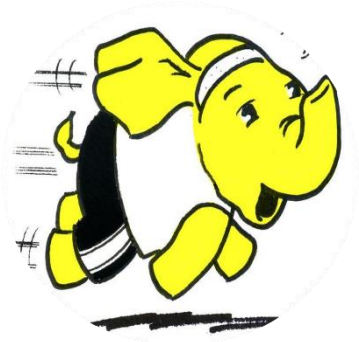
Data is conceptually record-oriented in the Hadoop programming framework. Individual input files are broken into lines or into other formats specific to the application logic. Each process running on a node in the cluster then processes a subset of these records. The Hadoop framework then schedules these processes in proximity to the location of data/records using knowledge from the distributed file system. Since files are spread across the distributed file system as chunks, each compute process running on a node operates on a subset of the data. Which data operated on by a node is chosen based on its locality to the node: most data is read from the local disk straight into the CPU, alleviating strain on network bandwidth and preventing unnecessary network transfers. This strategy of moving computation to the data, instead of moving the data to the computation allows Hadoop to achieve high data locality which in turn results in high performance.

Hadoop is a distributed file system which follows Master Slave Architecture for Data distribution. In this architecture there is a cluster which consists of one single Name node(Master node) and Data nodes (slave nodes).Here the Name node and Data node will be working to distribute the file in a structural way with limited memory byte in each nodes.Name node has the function to manage and maintain the Data nodes. It records the Meta data of the actual data . Meta data is something which keep the location of the block ,the size of the block. Name node also responsible to records if any modification made to the files . For example: If any file is deleted , the Name node will immediately record it.



Data nodes are the slave nodes which is responsible to store the actual data ,where data will be stored in different Data nodes. It also sends the heartbeat a kind of report to Name node periodically to make sure that the Data nodes are working properly or not. In every 3 secs the process will repeat by default.

9.Features of Hadoop



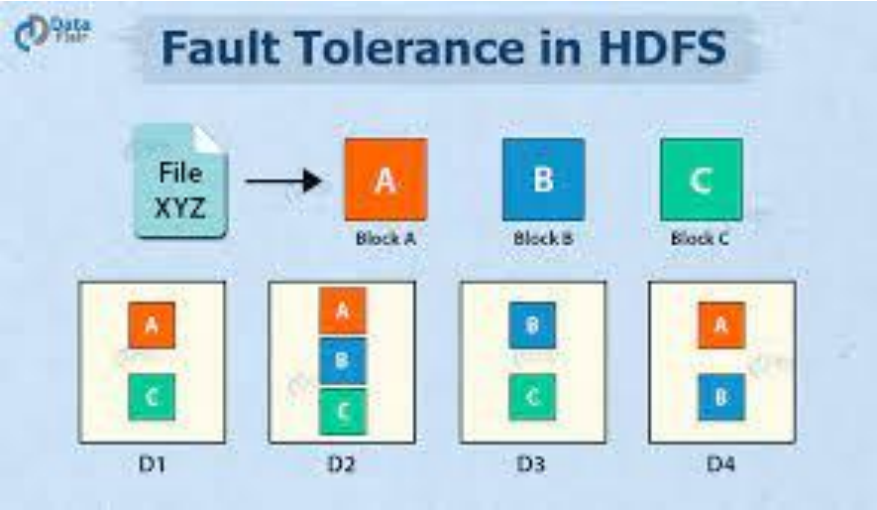
Open Source

Hadoop is open-source, which means it is free to use. Since it is an open-source project the source-code is available online for anyone to understand it or make some modifications as per their industry requirement.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

Fault Tolerance

Hadoop uses commodity hardware(inexpensive systems) which can be crashed at any moment. In Hadoop data is replicated on various DataNodes in a Hadoop cluster which ensures the availability of data if somehow any of your systems got crashed. You can read all of the data from a single machine if this machine faces a technical issue data can also be read from other nodes in a Hadoop cluster because the data is copied or replicated by default. By default, Hadoop makes 3 copies of each file block and stored it into different nodes. This replication factor is configurable and can be changed by changing the replication property in the hdfs-site.xml file.



TheThe objective of creating a fault-tolerant system is to prevent disruptions arising from a single point of failure, ensuring the high availability and business continuity of mission-critical applications or systems.Fault-tolerant systems use backup components that automatically take the place of failed components, ensuring no loss of service. These include:Hardware systems that are backed up by identical or equivalent systems. For example, a server can be made fault tolerant by using an identical server running in parallel, with all operations mirrored to the backup server.Software systems that arebacked up by other software instances. For example, a database with customer information can be continuously replicated to another machine.

If the primary datagoes down, operations can be automatically redirected to the second database.Power sources that are made fault tolerant using alternative sources. For example, many organizations have power generators that can take over in case main line electricity fails.In similar fashion, any system or component which is a single point of failure can be made fault tolerant using redundancy.

Fault tolerance can play a role in a disaster recovery strategy. For example, fault-tolerant systems with backup components in the cloud can restore mission-critical systems quickly, even if a natural or human-induced disaster destroys on-premise IT infrastructure. objective of creating a fault-tolerant system is to prevent disruptions arising from a single point of failure, ensuring the high availability and business continuity of mission-critical applications or systems.

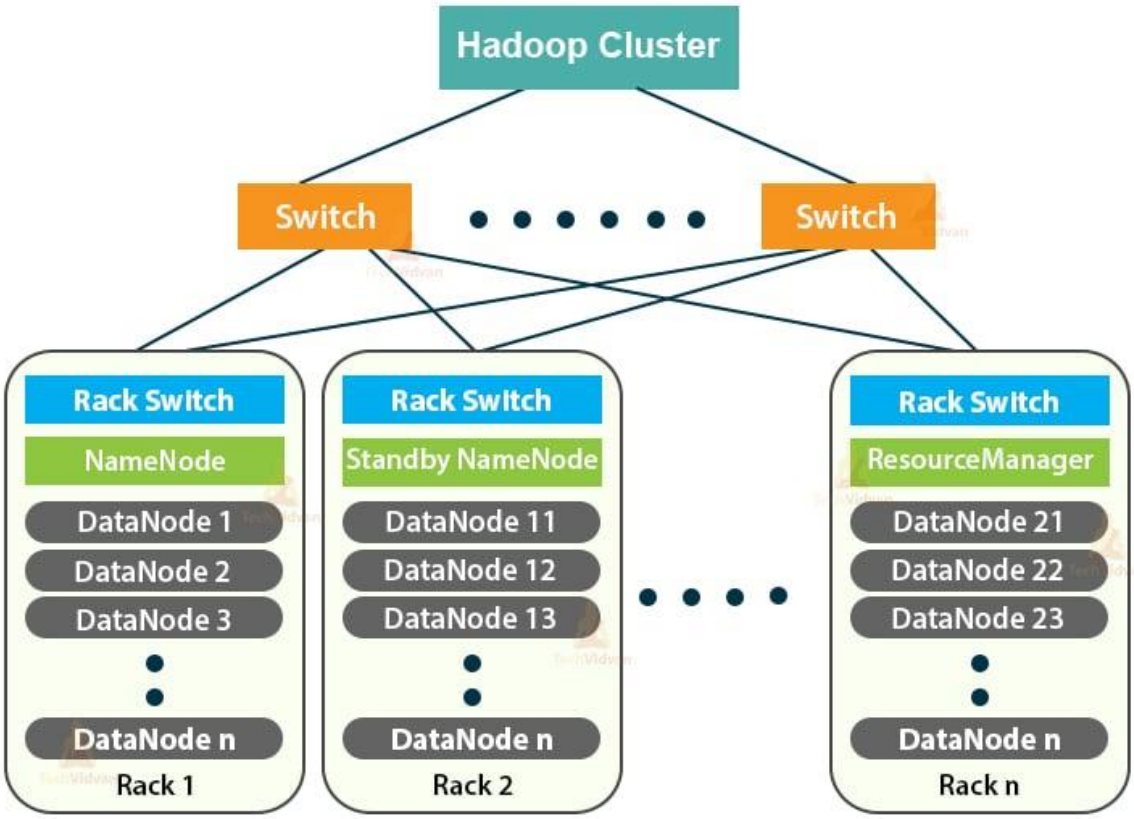
Fault-tolerant systems use backup components that automatically take the place of failed components, ensuring no loss of service. These include:Hardware systems that are backed up by identical or equivalent systems. For example, a server can be made fault tolerant

by using an identical server running in parallel, with all operations mirrored to the backup server.Software systems that are backed up by other software instances. For example, a database with customer information can be continuously replicated to another machine.

If the primary database goes down, operations can be automatically redirected to the second database.Power sources that are made fault tolerant using alternative sources. For example, many organizations have power generators that can take over in case main line electricity fails.In similar fashion, any system or component which is a single point of failure can be made fault tolerant using redundancy.

Fault tolerance can play a role in a disaster recovery strategy. For example, fault-tolerant systems with backup components in the cloud can restore mission-critical systems quickly, even if a natural or human-induced disaster destroys on-premise IT infrastructure.

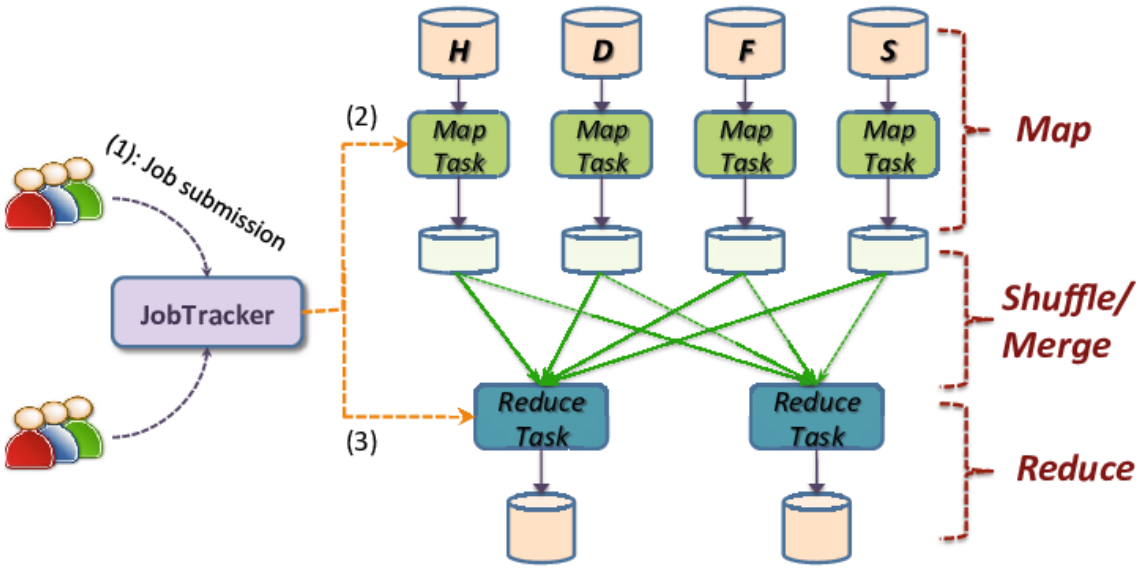
Highly Scalable cluster



- Hadoop clusters can boost the processing speed of many big data analytics jobs, given their ability to break down large computational tasks into smaller tasks that can be run in a parallel, distributed fashion.
- Hadoop clusters are easily scalable and can quickly add nodes to increase throughput, and maintain processing speed, when faced with increasing data blocks.
- The use of low cost, high availability commodity hardware makes Hadoop clusters relatively easy and inexpensive to set up and maintain.
- Hadoop clusters replicate a data set across the distributed file system, making them resilient to data loss and cluster failure.
- Hadoop clusters make it possible to integrate and leverage data from multiple different source systems and data formats.
- It is possible to deploy Hadoop using a single-node installation, for evaluation purposes.

Faster in Data Processing

Hadoop is lightning fast because of data locality – move computation to data rather than moving the data, as it is easier and make processing lightning fast. The Same algorithm is available for all the nodes in the cluster to process on chunks of data stored in them.



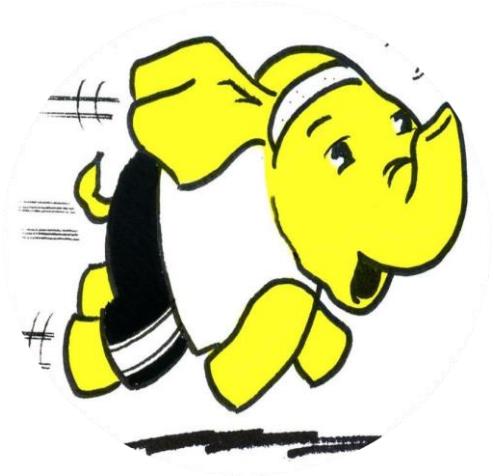
With the exponential growth of the World Wide Web over the years, the data being generated also grew exponentially. This led to a massive amount of data being created and it was being difficult to process and store this humungous amount of data with the traditional relational database systems.

Also, the data created was not only in the structured form but also in the unstructured format like videos, images, etc. This kind of data cannot be processed by relational databases. To counter these issues, Hadoop came into existence.
Before we dive into the data processing of Hadoop, let us have an overview of Hadoop and its components. Apache Hadoop is a framework that allows the storing and processing of huge quantities of data in a swift and efficient manner. It can be used to store huge quantities of structured and unstructured data .

Easy to use and low cost

Easy to use:-

Hadoop is easy to use since the developers need not worry about any of the processing work since it is managed by the Hadoop itself. Hadoop ecosystem is also very large comes up with lots of tools like Hive, Pig, Spark, HBase, Mahout, etc.



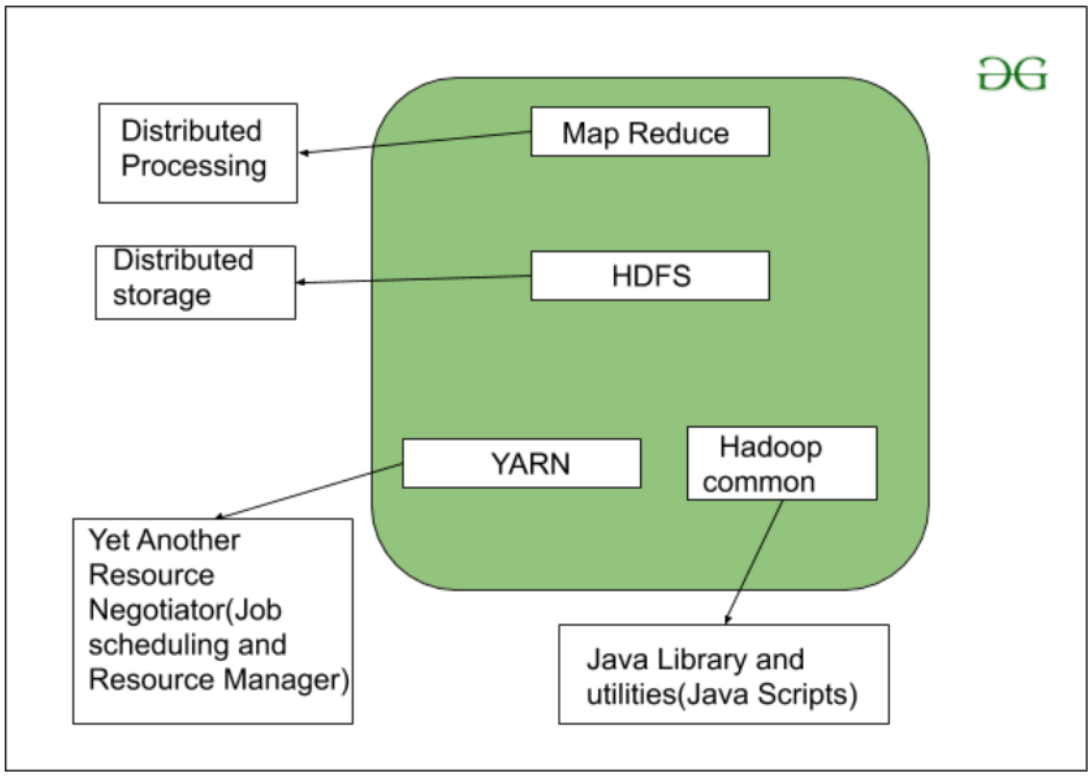
Low cost:-

Hadoop is open-source and uses cost-effective commodity hardware which provides a cost-efficient model, unlike traditional Relational databases that require expensive hardware and high-end processors to deal with Big Data. The problem with traditional Relational databases is that storing the Massive volume of data is not cost-effective, so the company's started to remove the Raw data. which may not result in the correct scenario of their business. Means Hadoop provides us 2 main benefits with the cost one is it's open-source means free to use and the other is that it uses commodity hardware which is also

10.Hadoop Architecture

As we all know Hadoop is a framework written in Java that utilizes a large cluster of commodity hardware to maintain and store big size data. Hadoop works on MapReduce Programming Algorithm that was introduced by Google. Today lots of Big Brand Companies are using Hadoop in their Organization to deal with big data, eg. Facebook, Yahoo, Netflix, eBay, etc. The Hadoop Architecture Mainly consists of 4 components.

- MapReduce
- HDFS(Hadoop distributed File System)
- YARN(Yet Another Resource Framework)
- Common Utilities or Hadoop Common

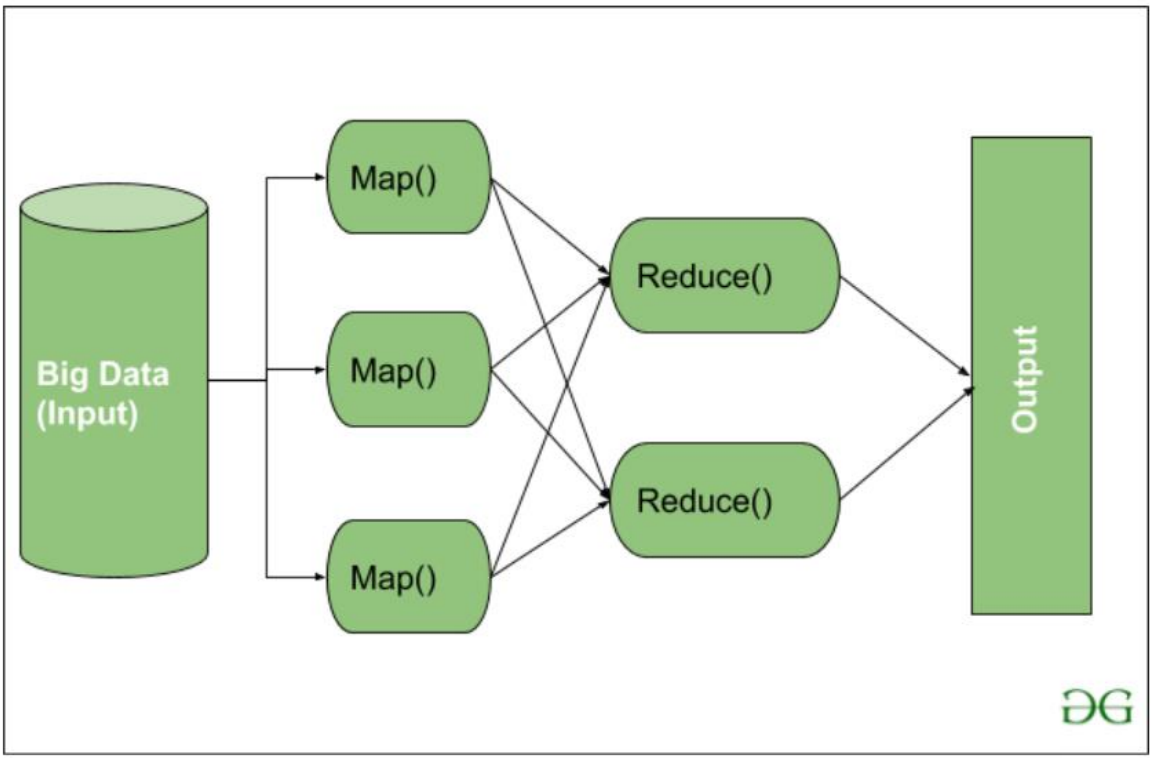


Let's understand the role of each one of this component in detail.

1. MapReduce

MapReduce nothing but just like an Algorithm or a data structure that is based on the YARN framework. The major feature of MapReduce is to perform the distributed

processing in parallel in a Hadoop cluster which Makes Hadoop working so fast. When you are dealing with Big Data, serial processing is no more of any use. MapReduce has mainly 2 tasks which are divided phase-wise:
In first phase, **Map** is utilized and in next phase **Reduce** is utilized.



Here, we can see that the *Input* is provided to the Map() function then it's *output* is used as an input to the Reduce function and after that, we receive our final output. Let's understand What this Map() and Reduce() does.
As we can see that an Input is provided to the Map(), now as we are using Big Data. The Input is a set of Data. The Map() function here breaks this DataBlocks into **Tuples** that are nothing but a key-value pair. These key-value pairs are now sent as input to the Reduce(). The Reduce() function then combines this broken Tuples or key-value pair based on its Key value and form set of Tuples, and perform some operation like sorting, summation type job, etc. which is then sent to the final Output Node. Finally, the Output is Obtained.
The data processing is always done in Reducer depending upon the business requirement of that industry. This is How First Map() and then Reduce is utilized one by one.

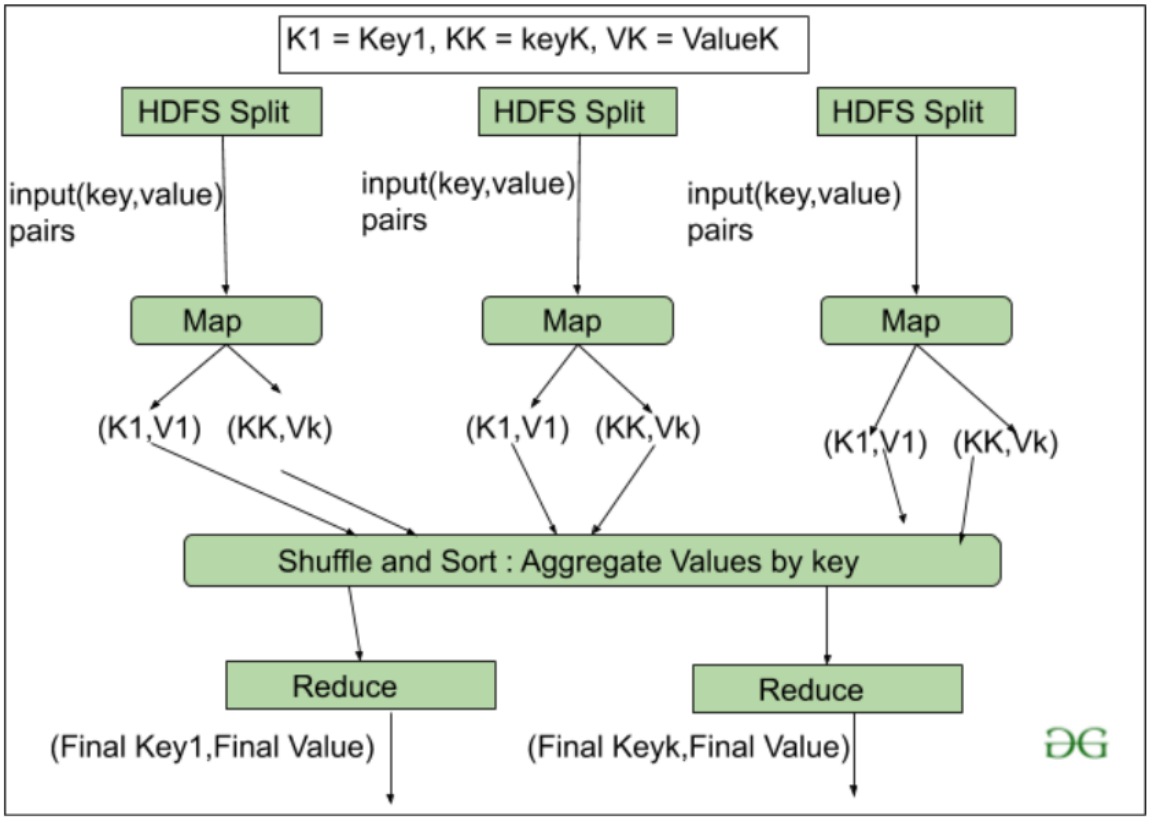
Let's understand the *Map Task* and *Reduce Task* in detail.
Map Task:

- **RecordReader** The purpose of *recordreader* is to break the records. It is responsible for providing key-value pairs in a Map() function. The key is actually is its locational information and value is the data associated with it.

- **Map:** A map is nothing but a user-defined function whose work is to process the Tuples obtained from record reader. The Map() function either does not generate any key-value pair or generate multiple pairs of these tuples.
- **Combiner:** Combiner is used for grouping the data in the Map workflow. It is similar to a Local reducer. The intermediate key-value that are generated in the Map is combined with the help of this combiner. Using a combiner is not necessary as it is optional.
- **Partitionar:** Partitional is responsible for fetching key-value pairs generated in the Mapper Phases. The partitioner generates the shards corresponding to each reducer. Hashcode of each key is also fetched by this partition. Then partitioner performs it's(Hashcode) modulus with the number of reducers(*key.hashcode()%(number of reducers)).*

Reduce Task

- **Shuffle and Sort:** The Task of Reducer starts with this step, the process in which the Mapper generates the intermediate key-value and transfers them to the Reducer task is known as *Shuffling*. Using the Shuffling process the system can sort the data using its key value.
  Once some of the Mapping tasks are done Shuffling begins that is why it is a faster process and does not wait for the completion of the task performed by Mapper.
- **Reduce:** The main function or task of the Reduce is to gather the Tuple generated from Map and then perform some sorting and aggregation sort of process on those key-value depending on its key element.
- **OutputFormat:** Once all the operations are performed, the key-value pairs are written into the file with the help of record writer, each record in a new line, and the key and value in a space-separated manner.



2. HDFS

HDFS(Hadoop Distributed File System) is utilized for storage permission is a Hadoop cluster. It mainly designed for working on commodity Hardware devices(inexpensive devices), working on a distributed file system design. HDFS is designed in such a way that it believes more in storing the data in a large chunk of blocks rather than storing small data blocks.

HDFS in Hadoop provides Fault-tolerance and High availability to the storage layer and the other devices present in that Hadoop cluster. Data storage Nodes in HDFS.

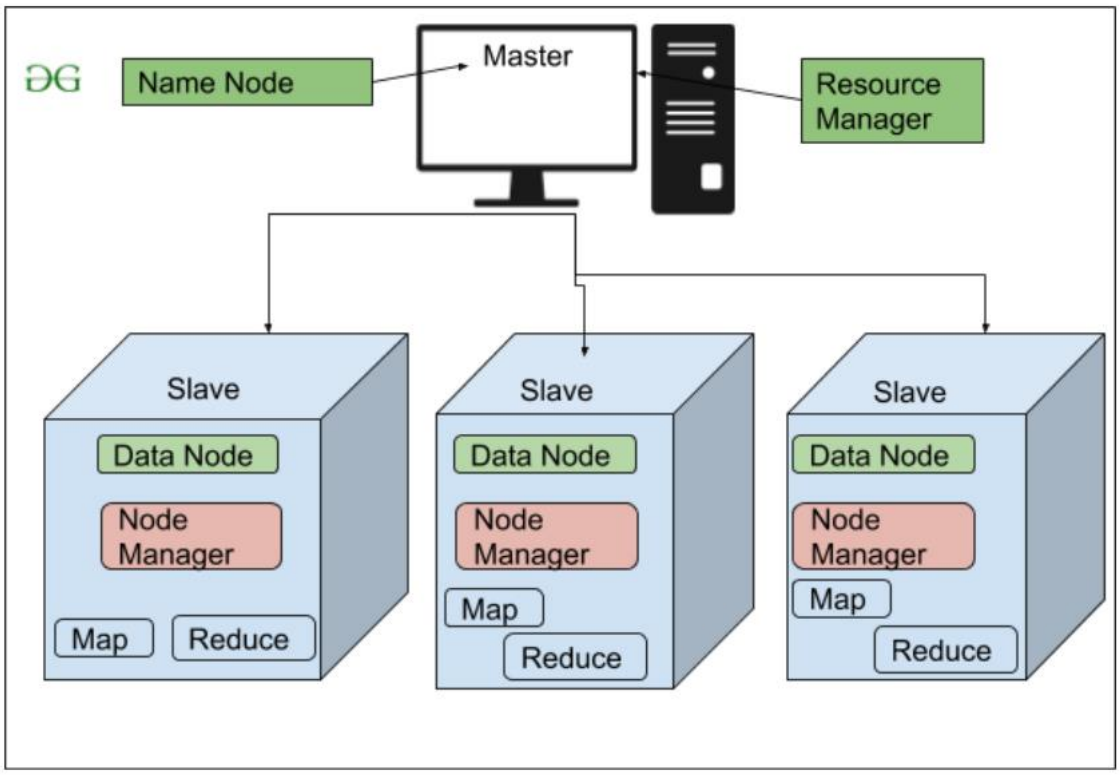
- NameNode(Master)
- DataNode(Slave)

NameNode: NameNode works as a Master in a Hadoop cluster that guides the Datanode(Slaves). Namenode is mainly used for storing the Metadata i.e. the data about the data. Meta Data can be the transaction logs that keep track of the user's activity in a Hadoop cluster.

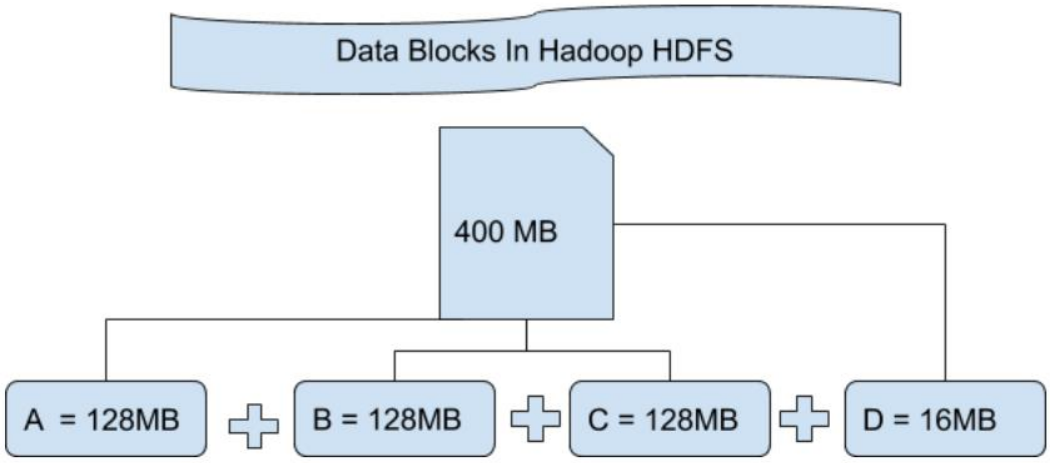
Meta Data can also be the name of the file, size, and the information about the location(Block number, Block ids) of Datanode that Namenode stores to find the closest DataNode for Faster Communication. Namenode instructs the DataNodes with the operation like delete, create, Replicate, etc.

DataNode: DataNodes works as a Slave DataNodes are mainly utilized for storing the data in a Hadoop cluster, the number of DataNodes can be from 1 to 500 or even more than that. The more number of DataNode, the Hadoop cluster will be able to store more data. So it is advised that the DataNode should have High storing capacity to store a large number of file blocks.

High Level Architecture Of Hadoop



File Block In HDFS: Data in HDFS is always stored in terms of blocks. So the single block of data is divided into multiple blocks of size 128MB which is default and you can also change it manually.



Let's understand this concept of breaking down of file in blocks with an example. Suppose you have uploaded a file of 400MB to your HDFS then what happens is this file got divided into blocks of 128MB+128MB+128MB+16MB = 400MB size. Means 4 blocks are created each of 128MB except the last one. Hadoop doesn't know or it doesn't care about what data is stored in these blocks so it considers the final file blocks as a partial record as it does not have any idea regarding it. In the Linux file system, the size of a file block is about 4KB which is very much less than the default size of file blocks in the Hadoop file system. As we all know Hadoop is mainly configured for storing the large size data which is in petabyte, this is what makes Hadoop file system different from other file systems as it can be scaled, nowadays file blocks of 128MB to 256MB are considered in Hadoop.

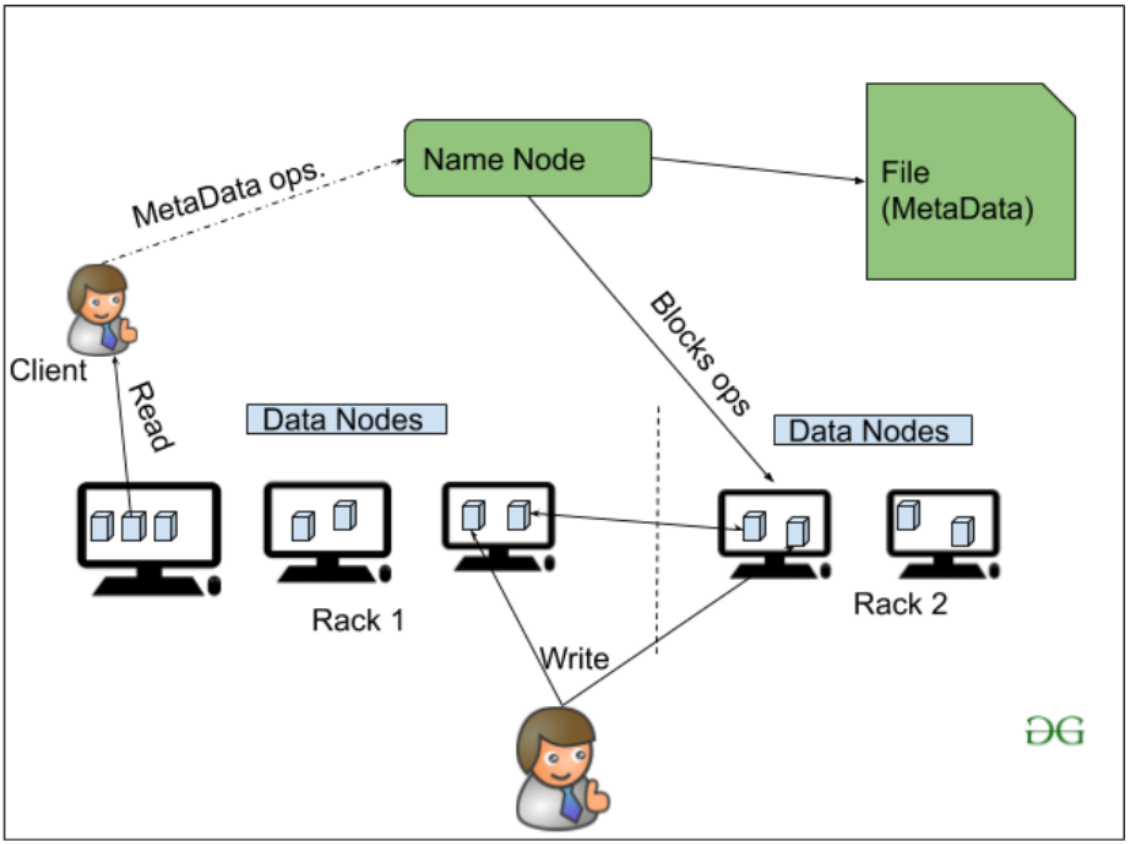
Replication In HDFS Replication ensures the availability of the data. Replication is making a copy of something and the number of times you make a copy of that particular thing can be expressed as it's Replication Factor. As we have seen in File blocks that the HDFS stores the data in the form of various blocks at the same time Hadoop is also configured to make a copy of those file blocks.
By default, the Replication Factor for Hadoop is set to 3 which can be configured means you can change it manually as per your requirement like in above example we have made 4 file blocks which means that 3 Replica or copy of each file block is made means total of 4×3 = 12 blocks are made for the backup purpose.

This is because for running Hadoop we are using commodity hardware (inexpensive system hardware) which can be crashed at any time. We are not using the supercomputer for our Hadoop setup. That is why we need such a feature in HDFS which can make copies of that file blocks for backup purposes, this is known as fault tolerance.

Now one thing we also need to notice that after making so many replica's of our file blocks we are wasting so much of our storage but for the big brand organization the data is very much important than the storage so nobody cares for this extra storage. You can configure the Replication factor in your *hdfs-site.xml* file.

Rack Awareness The rack is nothing but just the physical collection of nodes in our Hadoop cluster (maybe 30 to 40). A large Hadoop cluster is consists of so many Racks . with the help of this Racks information Namenode chooses the closest Datanode to achieve the maximum performance while performing the read/write information which reduces the Network Traffic.

HDFS Architecture



3. YARN(Yet Another Resource Negotiator)

YARN is a Framework on which MapReduce works. YARN performs 2 operations that are Job scheduling and Resource Management. The Purpose of Job schedular is to divide a big task into small jobs so that each job can be assigned to various slaves in a Hadoop cluster and Processing can be Maximized. Job Scheduler also keeps track of which job is important, which job has more priority, dependencies between the jobs and all the other information like job timing, etc. And the use of Resource Manager is to manage all the resources that are made available for running a Hadoop cluster.

Features of YARN

- Multi-Tenancy
- Scalability

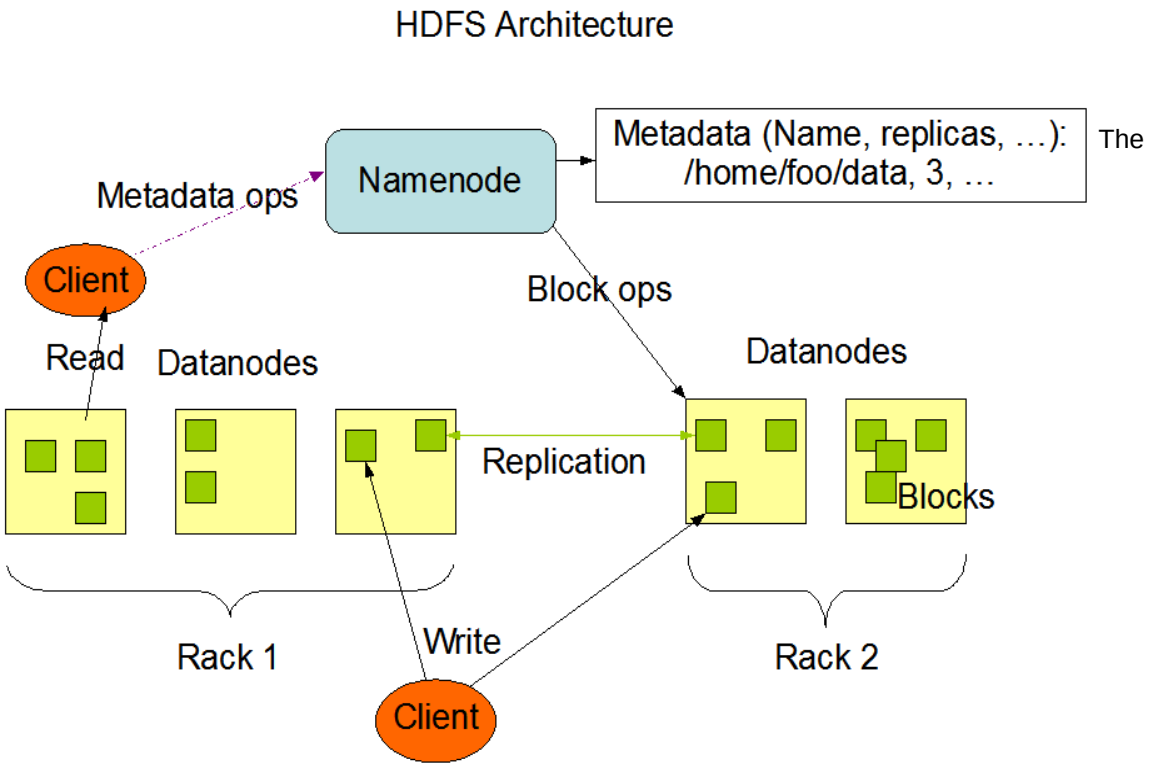
- Cluster-Utilization
- Compatibility

4. Hadoop common or Common Utilities

Hadoop common or Common utilities are nothing but our java library and java files or we can say the java scripts that we need for all the other components present in a Hadoop cluster. these utilities are used by HDFS, YARN, and MapReduce for running the cluster. Hadoop Common verify that Hardware failure in a Hadoop cluster is common so it needs to be solved automatically in software by Hadoop Framework.

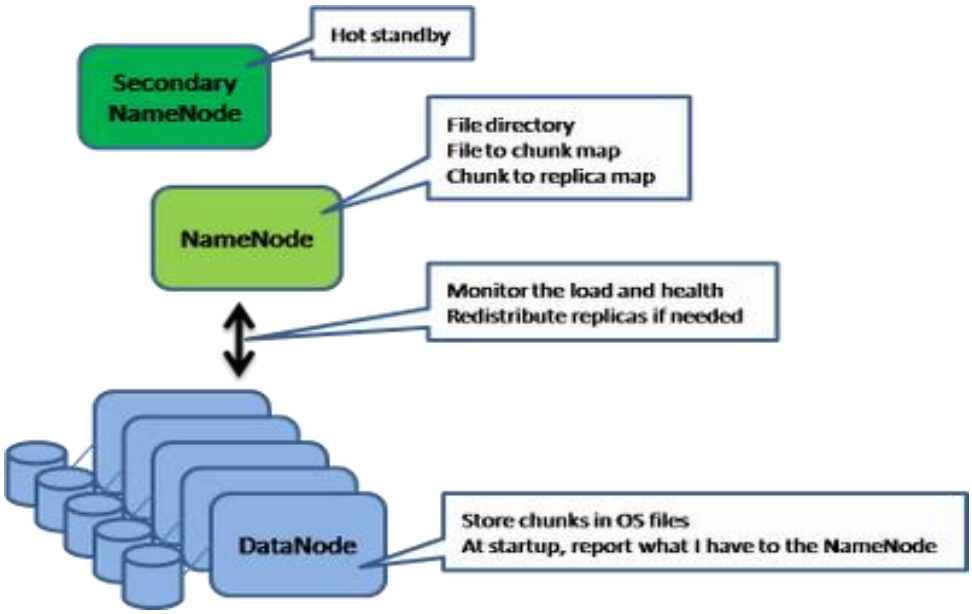
11.Namenodes and Data nodes

A HDFS cluster has two types of node operating in a master-worker pattern: a namenode (the master) and a number of datanodes (workers). The namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. This information is stored persistently on the local disk in the form of two files: the namespace image and the edit log. The namenode also knows the datanodes on which all the blocks for a given file are located, however, it does not store block locations persistently, since this information is reconstructed from datanodes when the system starts. A client accesses the filesystem on behalf of the user by communicating with the namenode and datanodes.



client presents a POSIX-like filesystem interface, so the user code does not need to know about the namenode and datanode to function. Datanodes are the work horses of the filesystem. They store and retrieve blocks when they are told to (by clients or the namenode), and they report back to the namenode periodically with lists of blocks that they are storing. Without the namenode, the filesystem cannot be used. In fact, if the machine running the namenode were obliterated, all the files on the filesystem would be lost since there would be no way of knowing

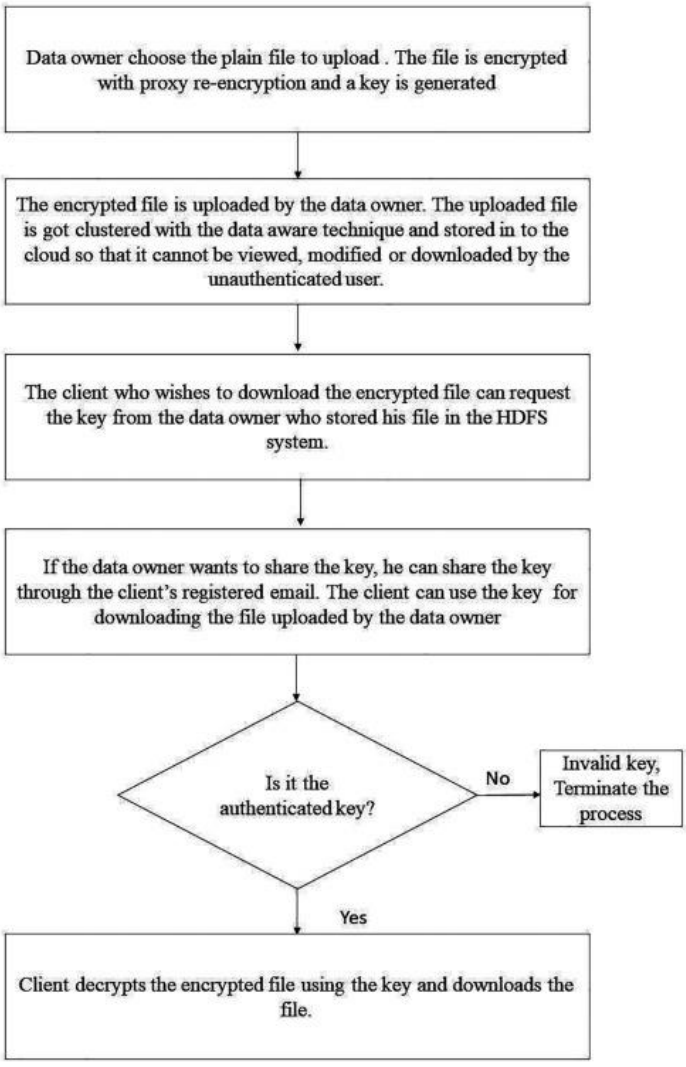
how to reconstruct the files from the blocks on the datanodes. For this reason, it is important to make the namenode resilient to failure, and Hadoop provides two mechanisms for this.



The first way is to back up the files that make up the persistent state of the filesystem metadata. Hadoop can be configured so that the namenode writes its persistent state to multiple filesystems. These writes are synchronous and atomic. The usual configuration Choice is to write to local disk as well as a remote NFS mount. It is also possible to run a secondary namenode, which despite its name does not act as a namenode. Its main role is to periodically merge the namespace image with the edit log to prevent the edit log from becoming too large. The secondary namenode usually runs on a separate physical machine, since it requires plenty of CPU and as much memory as the namenode to perform the merge. It keeps a copy of the merged namespace image, which can be used in the event of the namenode failing. However, the state of the secondary namenode lags that of the primary, so in the event of total failure of the primary data, loss is almost guaranteed. The usual course of action in this case is to copy the namenode's metadata files that are on NFS to the secondary and run it as the new primary.

12.Data Integrity

It is possible that a block of data fetched from a DataNode arrives corrupted. This corruption can occur because of faults in a storage device, network faults, or buggy software. The HDFS client software implements checksum checking on the contents of HDFS files. When a client creates an HDFS file, it computes a checksum of each block of the file and stores these checksums in a separate hidden file in the same HDFS namespace. When a client retrieves file contents it verifies that the data it received from each Data Node matches the checksum stored in the associated checksum file. If not, then the client can opt to retrieve that block from another Data Node that has a replica of that block.



13.Data Organization

Data Blocks

HDFS is designed to support very large files. Applications that are compatible with HDFS are those that deal with large data sets. These applications write their data only once but they read it one or more times and require these reads to be satisfied at streaming speeds. HDFS supports write-once-read-many semantics on files. A typical block size used by HDFS is 64 MB. Thus, an HDFS file is chopped up into 64 MB chunks, and if possible, each chunk will reside on a different DataNode.

Staging

A client request to create a file does not reach the NameNode immediately. In fact, initially the HDFS client caches the file data into a temporary local file. Application writes are transparently redirected to this temporary local file. When the local file accumulates data worth over one HDFS block size, the client contacts the NameNode. The NameNode inserts the file name into the file system hierarchy and allocates a data block for it. The NameNode responds to the client request with the identity of the DataNode and the destination data block. Then the client flushes the block of data from the local temporary file to the specified DataNode. When a file is closed, the remaining un-flushed data in the temporary local file is transferred to the DataNode. The client then tells the NameNode that the file is closed. At this point, the NameNode commits the file creation operation into a persistent store. If the NameNode dies before the file is closed, the file is lost.

The above approach has been adopted after careful consideration of target applications that run on HDFS. These applications need streaming writes to files. If a client writes to a remote file directly without any client side buffering, the network speed and the congestion in the network impacts throughput considerably. This approach is not without precedent. Earlier distributed file systems, e.g. AFS, have used client side caching to improve performance. A POSIX requirement has been relaxed to achieve higher performance of data uploads.

14.Hadoop Filesystems

Hadoop has an abstract notion of filesystem, of which HDFS is just one implementation. The Java abstract class org.apache.hadoop.fs.FileSystem represents a filesystem in Hadoop, and there are several concrete implementations, which are described in following table.

Local	**file**	**fs.LocalFileSystem**	**A filesystem for a locally connected disk with client-side checksums. Use RawLocalFileSystem for a local filesystem with no checksums.**
HDFS	hdfs	hdfs.DistributedFileSystem	Hadoop's distributed filesystem. HDFS is designed to work efficiently in conjunction with Map-Reduce.
HFTP	hftp	hdfs.HftpFileSystem	A filesystem providing read-only access to HDFS over HTTP. (Despite its name, HFTP has no connection with FTP.) Often used with distcp ("Parallel Copying with
HSFTP	hsftp	Hdfs.HsftpFileSystem	A filesystem providing read-only access to HDFS over HTTPS. (Again, this has no connection with FTP.)
HAR	har	Fs.HarFileSystem	A filesystem layered on another filesystem for archiving files. Hadoop Archives are typically used

			for archiving files in HDFS to reduce the namenode's memory usage.
KFS(Cloud Store)	Kfs	fs.kfs.KosmosFileSystem	CloudStore (formerly Kosmos filesystem) is a distributed filesystem like HDFS or Google's GFS, written in C++.
FTP	ftp	fs.ftp.FtpFileSystem	A filesystem backed by an FTP server.
S3(Native)	s3n	fs.s3native.NativeS3FileSystem	A filesystem backed by Amazon S3.
S3(Block Based)	S3	fs.s3.S3FileSystem A	A filesystem backed by Amazon S3, which stores files in blocks (much like HDFS) to overcome S3's 5 GB file size limit.

15. Hadoop Archives, Using Hadoop Archives

HDFS stores small files inefficiently, since each file is stored in a block, and block metadata is held in memory by the namenode. Thus, a large number of small files can eat up a lot of memory on the namenode. (Note, however, that small files do not take up any more disk space than is required to store the raw contents of the file. For example, a 1 MB file stored with a block size of 128 MB uses 1 MB of disk space, not 128 MB.) Hadoop Archives, or HAR files, are a file archiving facility that packs files into HDFS blocks more efficiently, thereby reducing namenode memory usage while still allowing transparent access to files. In particular, Hadoop Archives can be used as input to MapReduce.

Using Hadoop Archives

A Hadoop Archive is created from a collection of files using the archive tool. The tool runs a MapReduce job to process the input files in parallel, so to run it, you need a MapReduce cluster running to use it.

Limitations

There are a few limitations to be aware of with HAR files. Creating an archive creates a copy of the original files, so you need as much disk space as the files you are archiving to create the archive (although you can delete the originals once you have created the archive). There is currently no support for archive compression, although the files that go into the archive can be compressed (HAR files are like tar files in this respect). Archives are immutable once they have been created. To add or remove files, you must recreate the archive. In practice, this is not a problem for files that don’t change after being written, since they can be archived in batches on a regular basis, such as daily or weekly. As noted earlier, HAR files can be used as input to MapReduce. However, there is no archive-aware InputFormat that can pack multiple files into a single MapReduce split, so processing lots of small files, even in a HAR file, can still be inefficient.

16.ANATOMY OF A MAPREDUCE JOB RUN

• The client, which submits the MapReduce job.

• The jobtracker, which coordinates the job run. The jobtracker is a Java application whose main class is JobTracker.

• The tasktrackers, which run the tasks that the job has been split into. Tasktrackers are Java applications whose main class is TaskTracker.

• The distributed filesystem which is used
for sharing job files between the other entities.

17.Hadoop is now a part of:-

Amazon S3

Amazon S3 (Simple Storage Service) is a data storage service. You are billed monthly for storage and data transfer. Transfer between S3 and AmazonEC2 is free. This makes use of S3 attractive for Hadoop users who run clusters on EC2.

Hadoop provides two filesystems that use S3.

S3 Native FileSystem (URI scheme: s3n)

- A native filesystem for reading and writing regular files on S3. The advantage of this filesystem is that you can access files on S3 that were written with other tools. Conversely, other tools can access files written using Hadoop. The disadvantage is the 5GB limit on file size imposed by S3. For this reason it is not suitable as a replacement for HDFS (which has support for very large files).

S3 Block FileSystem (URI scheme: s3)

- A block-based filesystem backed by S3. Files are stored as blocks, just like they are in HDFS. This permits efficient implementation of renames. This filesystem requires you to dedicate a bucket for the filesystem - you should not use an existing bucket containing files, or write other files to the same bucket. The files stored by this filesystem can be larger than 5GB, but they are not interoperable with other S3 tools.

There are two ways that S3 can be used with Hadoop's Map/Reduce, either as a replacement for HDFS using the S3 block filesystem (i.e. using it as a reliable distributed filesystem with support for very large files) or as a convenient repository for data input to and output from MapReduce, using either S3 filesystem. In the second case HDFS is still used for the Map/Reduce phase. Note also, that by using S3 as an input to MapReduce you lose the data locality optimization, which may be significant.

FACEBOOK

Facebook's engineering team has posted some details on the tools it's using to analyze the huge data sets it collects. One of the main tools it uses is Hadoop that makes it easier to analyze vast amounts of data.

Some interesting tidbits from the post:

- Some of these early projects have matured into publicly released features (like the Facebook Lexicon) or are being used in the background to improve user experience on Facebook (by improving the relevance of search results, for example).
- Facebook has multiple Hadoop clusters deployed now - with the biggest having about 2500 cpu cores and 1 PetaByte of disk space. They are loading over 250 gigabytes of compressed data (over 2 terabytes uncompressed) into the Hadoop file system every day and have hundreds of jobs running each day against these data sets. The list of projects that are using this infrastructure has proliferated - from those generating mundane statistics about site usage, to others being used to fight spam and determine application quality.
- Over time, we have added classic data warehouse features like partitioning, sampling and indexing to this environment. This in-house data warehousing layer over Hadoop is called Hive.

YAHOO!

Yahoo! recently launched the world's largest Apache Hadoop production application. The Yahoo! Search Webmap is a Hadoop application that runs on a more than 10,000 core Linux cluster and produces data that is now used in every Yahoo! Web search query**.**

The Webmap build starts with every Web page crawled by Yahoo! and produces a database of all known Web pages and sites on the internet and a vast array of data about every page and site. This derived data feeds the Machine Learned Ranking algorithms at the heart of Yahoo! Search.

Some Webmap size data:

- Number of links between pages in the index: **roughly 1 trillion links**
- Size of output: **over 300 TB, compressed!**
- Number of cores used to run a single Map-Reduce job: **over 10,000**

- Raw disk used in the production cluster: **over 5 Petabytes**

This process is not new. What is new is the use of Hadoop. Hadoop has allowed us to run the identical processing we ran pre-Hadoop on the same cluster in 66% of the time our previous system took. It does that while simplifying administration.

Conclusion

Indeed ,by this seminar I came to know all the concepts of Hadoop, and how it manage, store and process data in distributed Environment. I have also Understood that Now a days we are using Hadoop Technology very much because it allow us to strong big amount of data easily so the companies bank ,social media have used Hadoop technology. I have also learned the concepts of big data and there Applications.

I have understood Hadoop Architecture and how there components like ex YARN,HDFC,Hadoop common,MapReduce etc works. Thus we need Hadoop technology very much for strong big amount of data

REFERENCES

http://www.cloudera.com/hadoop-training-thinking-at-scale

http://developer.yahoo.com/hadoop/tutorial/module1.html

http://hadoop.apache.org/core/docs/current/api/

http://hadoop.apache.org/core/version_control.html