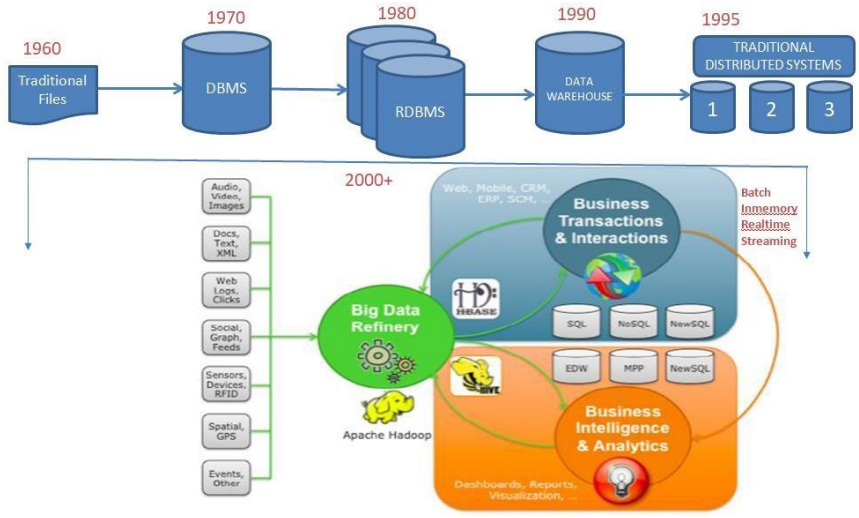
BigData, Hadoop and HDFS

Evolution of Data:



What is Big data?

Big data is a phrase or methodology that describes the way of aquire, cure, process, store and Analyze the humongous volume of hetrogeneous data that flows in different frequency which is limited to handled by the traditional systems with in the stipulated period of time.

Big data is simply the large sets of data that businesses and other parties put together to serve specific goals and operations. Big data can include many different kinds of data in many different kinds of formats. As a rule, it's raw and unsorted until it is put through various kinds of tools and handlers.

Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time, extremely handles large data sets that may be analysed computationally to reveal patterns, trends, and associations, especially relating to human behaviour and interactions.

Types Of Big Data:

Broadly classified into man made and machine made data such as click stream event logs, device logs, photos, videos, historical datawarehouse data, geo satellite data etc. Shared nothing architecture.

Big data sourced from web data, social media, click stream data, man-made data, machine/device data etc.

Web log click stream data eg. Online banking transactions comparison between traditional and bigdata.

BigData Statistics

Common data sets

•40 Zetabytes of data exist in the digital universe today.

•Data is doubling every 2 years, 90% of world's data volume is created in past 2 years.

•Facebook stores, accesses, and analyzes 30+ Petabytes of user generated data.

•Walmart handles more than 1 million customer transactions every hour, which is imported into databases estimated to contain more than 2.5 petabytes of data.

•More than 5 billion people are calling, texting, tweeting and browsing on mobile phones worldwide.

•Decoding the human genome originally took 10 years to process; now it can be achieved in one week.

•The largest AT&T database boasts titles including the largest volume of data in one unique database (312 terabytes) and the second largest number of rows in a unique database (1.9 trillion), which comprises AT&T's extensive calling records.

The Rapid Growth of Unstructured Data

- YouTube users upload 48 hours of new video every minute of the day.
- Every person in the US tweeting three tweets per minute for 26,976 years.
- Every person in the world having more than 215m high-resolution MRI scans a day.
- More than 200bn HD movies – which would take a person 47m years to watch?

The Market and The Marketers' Challenge with Big Data

• Big data is a top business priority and drives enormous opportunity for business improvement. Wikibon's own study projects that big data will be a $50 billion business by 2017.

•140,000 to 190,000. Too few people with deep analytical skills to fill the demand of Big Data jobs in the U.S. by 2018.

Big Data & Real Business Issues

•According to estimates, the volume of business data worldwide, across all companies, doubles every 1.2 years.

•Poor data can cost businesses 20%–35% of their operating revenue.

•Bad data or poor data quality costs US businesses $600 billion annually.

•According to execs, the influx of data is putting a strain on IT infrastructure. 55 percent of respondents reporting a slowdown of IT systems and 47 percent citing data security problems, according to a global survey from Avanade.

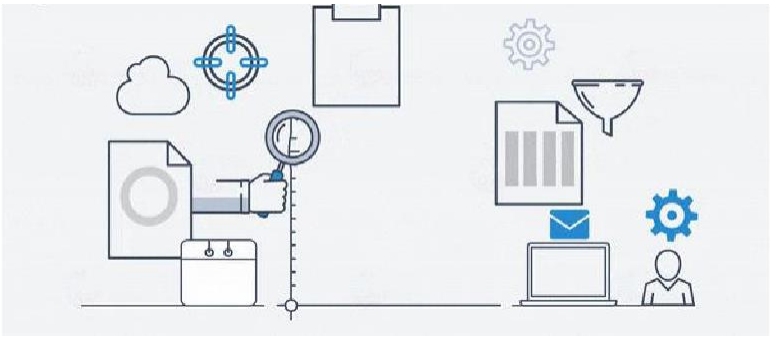
•In that same survey, by a small but noticeable margin, executives at small companies (fewer than 1,000 employees) are nearly 10 percent more likely to view data as a strategic differentiator than their counterparts at large enterprises.

•Three-quarters of decision-makers (76 per cent) surveyed anticipate significant impacts in the domain of storage systems as a result of the "Big Data" phenomenon.

Bigdata is the center of attraction for all the technologies storage, processing and business value derivation needs. With the current/traditional systems industry was concentrating of implementing digital provisions and solutions for their business starting from application development programming to help clients interact with the business providing better UI experience, datawarehouse solution to convert, store, process and produce business indicators to promote the business, data storage solutions provided by db companies to store and retrieve RDBMS data efficiently. Now bigdata is in high demand due to all traditional systems are incapable of handling humongous volume of data being sent from different mediums of different format that has to be processed in a stipulated period of time and brings a real insight and business output from those data including the search of new business oppurtunities such as customer retention, scaling the business etc..

Data Generators

Analytics



IOT

Cloud

- **Bigdata & Analytics** is one of the top trending technology, Bigdata, Cloud computing, Enterprise mobility, DevOps, IOT.
- **Enterprise mobility** - The term refers not only to mobile workers and mobile devices, but also to the mobility of corporate data. An employee may upload a corporate presentation from his or her desktop PC to a cloud storage service, then access it from a personal iPad to show at a client site, for example. Enterprise mobility can improve employee productivity, but it also creates security risks.Enterprise mobility management products, such as data loss prevention technologies, are available to help IT departments address these risks. A strong acceptable use policy for employees can also contribute to a successful enterprise mobility strategy.
- **Internet of Things**: Monitor kids, old parents, motion sensor to track any unwanted movement around or inside house etc.
- **Cloud Computing**

Characteristics:

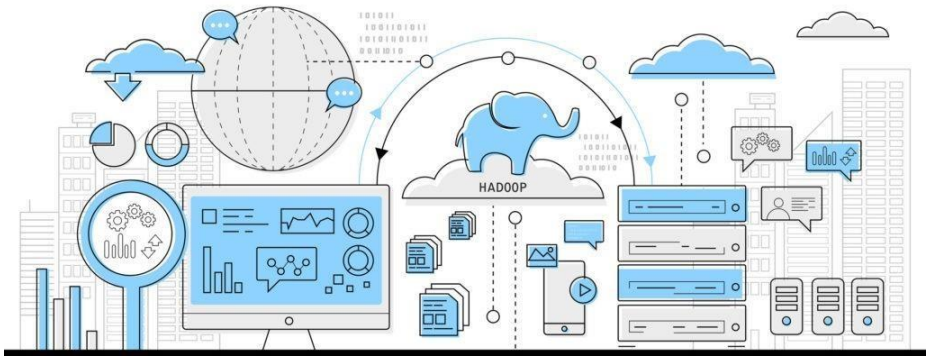
- **Volume** – huge volume of machine and man made data such as logs, click events, media files and historical warehouse data.

  Eg . FB, Historical data (Bank example, Airlines example) etc.

- **Velocity** – Speed at which data is generated, captured, processed and stored.
  Eg. Tweets, click stream events, GPS Geospatial data etc.

- **Variety** – not only structured, it can accommodate un structured, semi structured, media, logs etc.
  Eg. STB logs, logs generated from different devices.
- **Veracity** – Trustworthy of the data we receive, how much true data we receive. Data accuracy is based on the veracity of source data that we receive at the very lower granular level. **Eg.** Website click steam logs.
- **Value** – ROI can be derived base on utilizing the data stored and bringing business insights from it.
  Eg. Click stream logs ROI derived based on the interest shown by the customer in the webpage (positive and negative scores).
- **Variable** – data varies from time to time wrt size, type that is unpredictable.
  Eg. A company started with only deals (Snapdeal) then enter into retail marketing and business grown in all media.

Big data tools (other than hadoop):



Spark – In memory computing provided with Low latency SQL solution standalone and Hadoop, provides machine learning libraries and graph database solution in a single place.

MongoDB – Document oriented DB, stores JSON (Java script object notation data).

TD Aster – Teradata uses mapreduce for the MPP store and process.

Cloud MapReduce -Initially developed at Accenture Technology Labs, Cloud Mapreduce can be broadly defined as Mapreduce Implementation on Amazon Cloud OS. When compared with other open source implementation, it is completely different architecture than others.

Amazon EMR

Microsoft HD Insight

IBM Big Insight

Disco - Disco can be broadly defined as an open-source and lightweight framework for distributed computing, which is based on the MapReduce paradigm. Due to Python, it is easy and powerful to use.

Misco - Misco can be broadly defined as distributed computing framework, which has been especially designed for mobile devices. Highly portable, Misco is being 100% implemented in Python and should be able to run on any system, which supports Python.

What is hadoop:

Hadoop is a open source Apache software stack that runs on cluster of commodity hardware that provides distributed storage and processing of large data sets.It is highly scalable, fault tolerant, data locality, highly distributed coordination, distributed MPP for heterogeneous data and heterogeneous software frameworks, low cost commodity hardware, high availability, highly secured, open source, resilient etc.

Evolution of Hadoop (spider to elephant)

Year	Evolution
1997	Doug Cutting working in lucene search engine.
2001	Doug Lucene to ASF and Mike Caferella 'web crawler' to index www , at that time 1.7 million web sites was there. Doug and Caferella joined and dubbed web crawler and named as NUTCH and tested in 1GB/1TB server, indexed about 100 webs /sec, and capable of maximum indexing of 100million. Scalability is an issue, tried with 4 nodes, but didn't achieved the expected performance.
2003	GFS white paper released by google, that exactly addressed the issues faced by Doug and Caferella.
2004	Google released another paper on MR for processing solutions. for processing solutions. Doug and Caferella made NDFS build based on GFS.
2005	Doug and Caferella wrote MR on top of NDFS.
2006	Doug made a subproject of lucene using MR and HDFS and named it as Hadoop – the name of his son's yellow elephant toy.

	Yahoo faced the same processing and scalability issue, employed Doug to implement hadoop in yahoo.
2007	Budding social media giants started their own projects on top of hadoop, such as Facebook released (hive, cassantra), linkedin – kafka, twitter – storm etc.
2008	Employees from google, facebook, yahoo and Berkleydb started 1st hadoop distribution called Cloudera.
2009	Yahoo sorts 1 PB in 16 hours using 3600 nodes cluster.
2011	Yahoo sponsored to its own hadoop company namely Hortonworks.
2012	Arun murthy and other open source community users proposed YARN.
2013	Hadoop 1.1.2 and Hadoop 2.0.3 alpha. Ambari, Cassandra, Mahout have been added

Hadoop – Features

Handle Huge volume of data

If the Data is too large to store and process in one computer, Hadoop can solve the problem by dividing the data and store it into cluster of several nodes and it is ready to process in a shorter period of time. Data is visible to the client in a unified way.Distributed parallel read and processing enhance performance

Highly Scalable:

Hadoop scales linearly. Due to linear scale, a Hadoop Cluster can be added with tens, hundreds, or even thousands of servers at any time with minimal effort.

Flexible:

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. This means businesses can use Hadoop to derive valuable business insights from data sources such as social media, email conversations or

clickstream data. In addition, Hadoop can be used for a wide variety of purposes, such as log processing, recommendation systems, data warehousing, market campaign analysis and fraud detection.

Data locality - Move computation rather than data

Data blocks reside on each nodes locally. In hadoop, there is no central system to hold the data as like traditional systems that stores data in SAN that bring the data through network to the program running node, process and store back the result to the SAN for client's access, hadoop runs the program on the node where data resides and coordinate all nodes to produce the output and produce to the client.

Highly Reliable

Data is replicated across multiple nodes (replication factor is configurable) and if a node goes down, the required data can be read from another node which has the copy of that data. And it also ensures that the replication factor is maintained, even if a node goes down, by replicating the data to other available nodes.

Integerated

Data integrity will be maintained by comparing the checksum value of the data at the time to data movement between data nodes and clients.

Cost Effectiveness

Commodity hardware, no need of high end servers.Open source, no licence or reneval fee.

Fault Tolerant

Hadoop has built-in fault tolerance. If program fails in one node can be handled automatically by executing the same piece of program in some other node has the same replica data, hence hadoop is highly reliable.

When to Use Hadoop

Hadoop can be used in various scenarios including some of the following:

- Your data may expect to grow exponentially in huge volume.

- If you don’t want to spend much on licensing cost and hardware, since hadoop is open source in commodity hardware.
- If you wanted to store raw data of its own format ie schema less storage system.
- You need to store heterogeneous historical and live data that is available to process at the same time.
- Need to bring all features such as acquisition, ETL, retention, data mining, analytics and machine learning under single roof.
- Need more granular data for the true analytics for business intensive applications such as banking, scientific domains.

When Not to Use Hadoop

There are few scenarios in which Hadoop is not the right fit currently.

- POS systems with low latency data processing with huge number of small files and perform frequent transactions.
- If needed to replace the complete traditional datawarehouse - Hadoop is complementary approach to a traditional data warehouse, not a replacement for it..
- For the mission critical business intensive systems where frequent upgradation or frequent change in application is a risky process to perform trial runs.

Why Hadoop - Comparing Hadoop with Traditional Distributed systems

1. Huge dependency on network and bandwidth need to port raw data from storage disk to san to app server and store the processed data back to SAN, 2 way data transfer occurs.
2. Partial failure is not easy to handle.
3. Not easily scalable.
4. Maintaining coordination and synchronisation is not an easy task.
5. Most of the processing effort spent on data transfer and not on processing it.

Components of Hadoop

1. Hadoop Common: contains libraries and utilities needed by other Hadoop modules.
2. Hadoop Distributed File System (HDFS): a distributed file-system that stores data on the commodity machines, providing very high aggregate bandwidth across the cluster.
3. Hadoop MapReduce: a programming model for large scale data processing.
4. Hadoop YARN: a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications.

Characteristics

1. Batch processing
2. Streaming data access – WORM – Write Once Read Many.
3. Low volume of huge files.

Hadoop Distributed File System (HDFS)

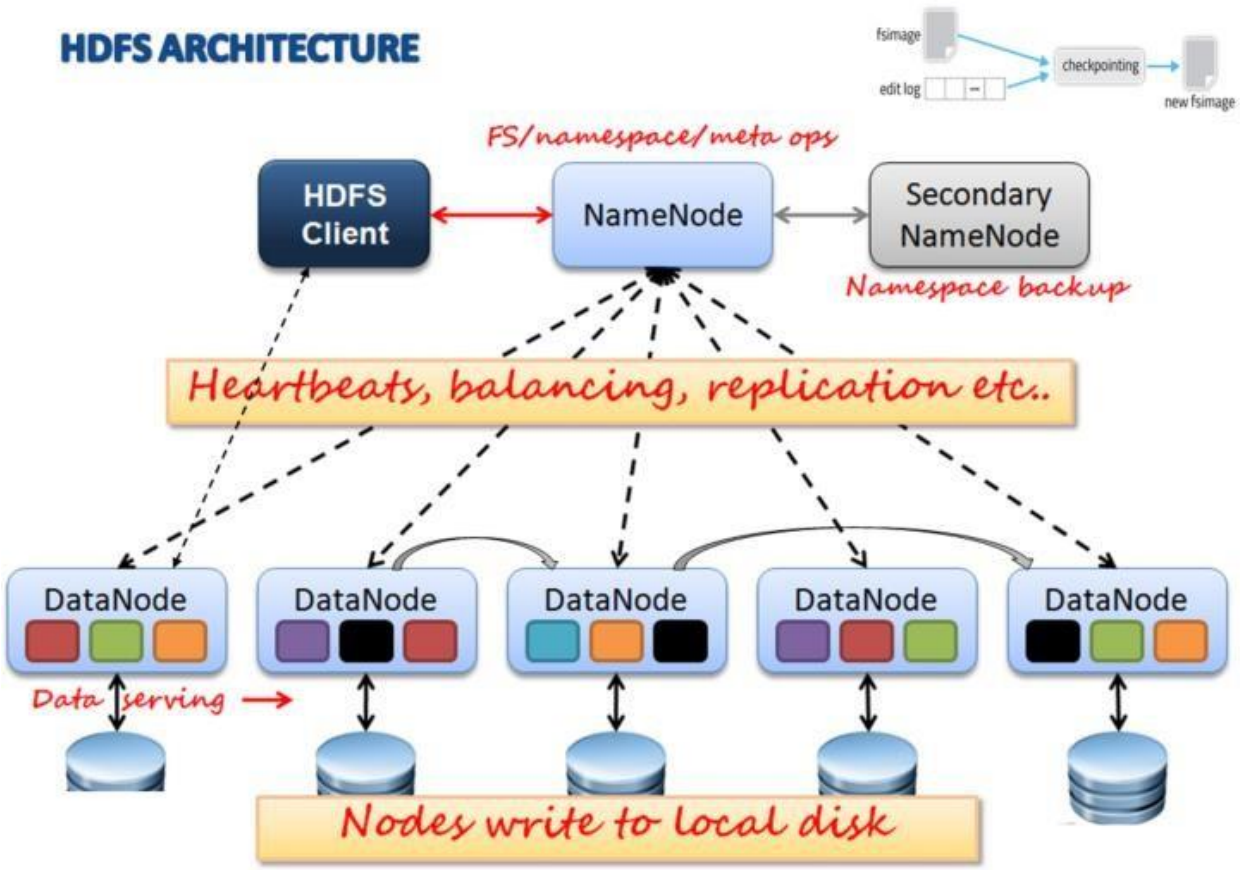
HDFS is a distributed file system that manages and provides high-throughput distributed access to data. HDFS creates multiple replicas of each data block and distributes them on computers throughout a cluster to enable resilient access hence improve fault tolerance.

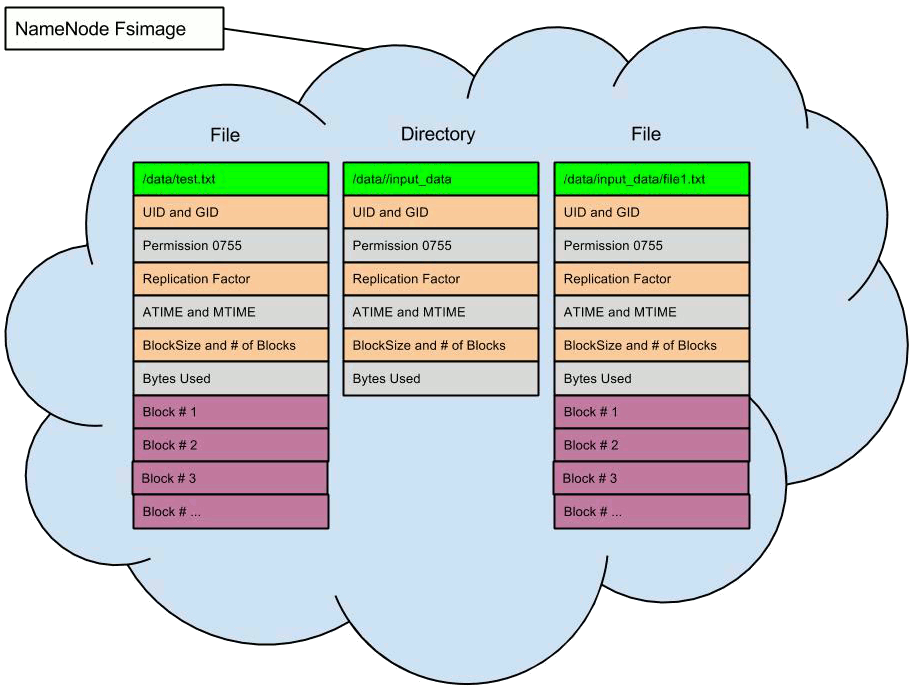
Why HDFS

Where ever you create directories in hdfs will be displayed commonly in all nodes in the cluster as a single filesystem unlike creating directories in each node separately.

Architecture

HDFS Architecture





NameNode is the master node or DFS Master of the system. It maintains the name system (directories and files) and manages the blocks which are present on the DataNodes.

Role of NameNode:

1. Name node holds the metadata for HDFS in memory and in disk.
2. Metadata in Memory serves the purpose of instant access to the metadata.
3. Files (fsimage and editlog) serves only when the cluster is restarted due to failure which will be referencing these files.
4. Controls read/write access to files, check the the existence of the directory/files before read/write.
5. Manages blocks, replication and re replication.

Metadata Components:

fsimage – Stores the inode details like modification time, access time, access permission, replication. **editlogs** – This keeps tracking of each and every change that is being done on HDFS. (Like adding a new file, deleting a file, moving it between folders..etc).

Any change that we do to HDFS will be tracked in edit logs, not in the fsimage. So the edit logs file will keep on growing whereas the fsimage size remains the same. This won't have any impact unless or until we restart the cluster. When we restart the cluster, the fsimage needs to get loaded in to the main memory. Since all changes are present in the edilogs not in the fsimage, hadoop will try to write editlogs changes to fsimage (this takes some time depends on the size of the editlogs file).

If you have not restarted your cluster for months together and if you are about to do it, then there will be a vast down time as editlogs size would have grown like anything.

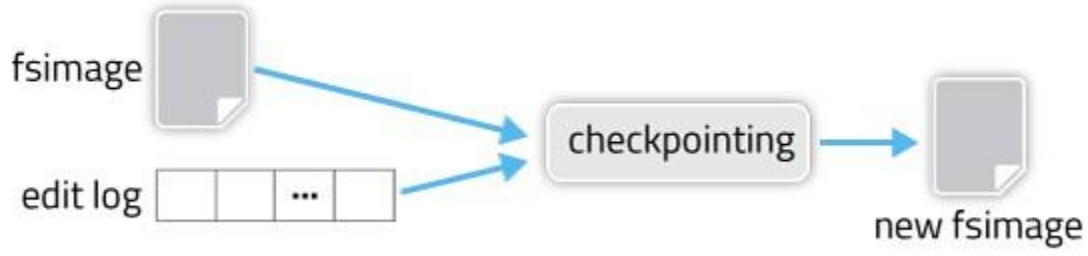
The namenode maintains the entire metadata in RAM, which helps clients receive quick responses to read requests. Therefore, it is important to run namenode from a machine that has lots of RAM at its disposal. The higher the number of files in HDFS, the higher the consumption of RAM. The namenode daemon also maintains a persistent checkpoint of the metadata in a file stored on the disk called thefsimage file.

Whenever a file is placed/deleted/updated in the cluster, an entry of this action is updated in a file called the edits logfile. After updating the edits log, the metadata present in-memory is also updated accordingly.

It is important to note that the fsimage file is not updated for every write operation.

Namenode - Checkpoint

When the NameNode starts up, it reads the FsImage and EditLog from disk, applies all the transactions from the EditLog to the in-memory representation of the FsImage, and flushes out this new version into a new FsImage on disk. It can then truncate the old EditLog because its transactions have been applied to the persistent FsImage. This process is called a **checkpoint**.



Note: It is possible to archive the FSImage and Editlogs old images if we need to rollback to any older good state.

Safemode

On startup, the NameNode enters a special state called Safemode. Replication of data blocks does not occur when the NameNode is in the Safemode state. The NameNode receives Heartbeat and Blockreport messages from the DataNodes. A Blockreport contains the list of data blocks that a DataNode is hosting. Each block has a specified minimum number of replicas. A block is considered

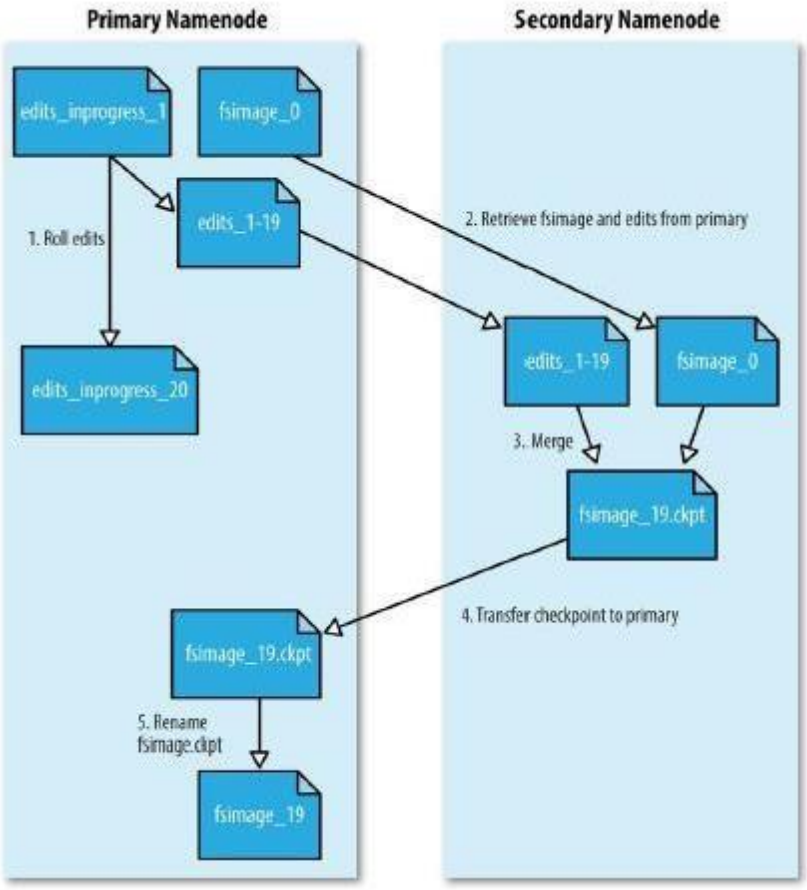
safely replicated when the minimum number of replicas of that data block has checked in with the NameNode. After a configurable percentage of safely replicated data blocks checks in with the NameNode, the NameNode exits the Safemode state. It then determines the list of data blocks (if any) that still have fewer than the specified number of replicas. The NameNode then replicates these blocks to other DataNodes.

Secondary Name Node:

1. Secondary name node act as a backup node in the case of name node crashed and the fsimage/editlog is lost completely in namenode. With the Admin interaction the FSImage from Secondary name node can be copied to a new name node and bring the name node up and running using the fsimage copied from SNN.
2. The secondary name node is responsible for performing periodic housekeeping functions for the *NameNode*. It only creates checkpoints of the filesystem present in the *NameNode*.
3. The *Secondary NameNode* is not a failover node for the *NameNode*.
4. Checkpoint configurations are
   fs.checkpoint.period – 3600 Seconds.
   fs.checkpoint.size – 64 MB

Secondary Name Node – Checkpoint

1. The secondary asks the primary to roll its in-progress edits file, so new edits go to a new file.
2. The secondary retrieves the latest FsImage and edits files from the primary namenode.
3. The secondary loads FsImage applies each transaction from edits into memory, then creates a new merged FsImage file.
4. The secondary sends the new FsImage back to the primary, and the primary saves it as a temporary .ckpt file.
5. The primary renames the temporary FsImage file to make it available.



Why is Checkpointing Important?

A typical edit ranges from 10s to 100s of bytes, but over time enough edits can accumulate to become unwieldy. A couple of problems can arise from these large edit logs. In extreme cases, it can fill up all the available disk capacity on a node, but more subtly, a large edit log can substantially delay NameNode startup as the NameNode reapplies all the edits. This is where checkpointing comes in.

Checkpointing is a process that takes an fsimage and edit log and compacts them into a new fsimage. This way, instead of replaying a potentially unbounded edit log, the NameNode can load the final in-memory state directly from the fsimage. This is a far more efficient operation and reduces NameNode startup time.**C**heckpointing creates a new fsimage from an old fsimage and edit log.

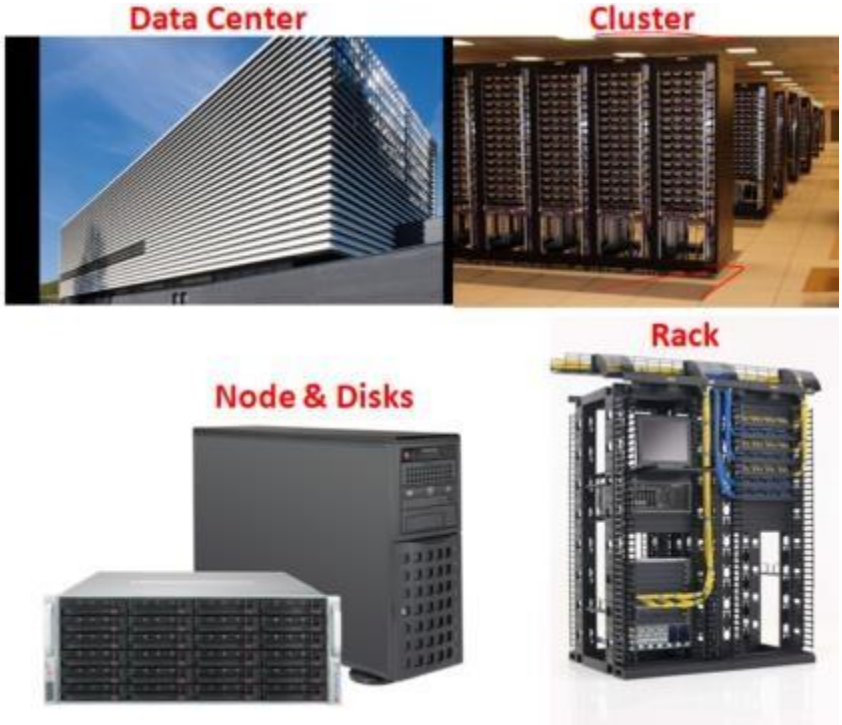
However, creating a new fsimage is an I/O- and CPU-intensive operation, sometimes taking minutes to perform. During a checkpoint, the namesystem also needs to restrict concurrent access from other users. So, rather than pausing the active NameNode to perform a checkpoint, HDFS defers it to either the SecondaryNameNode or Standby NameNode, depending on whether NameNode high-availability is configured.

DataNode

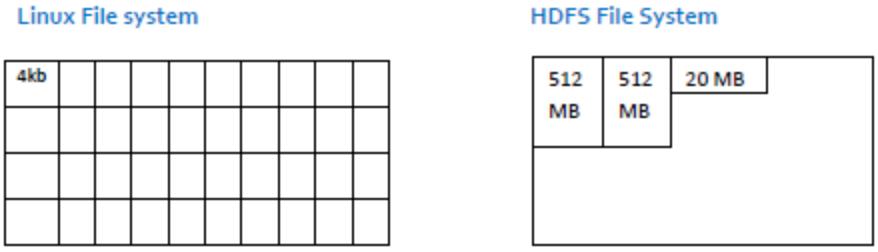
1. Data Nodesare the slaves which are deployed on each machine and provide the actual storage.
2. They are responsible for serving read and write requests for the clients.

3. All datanodes send a heartbeat message to the namenode every 3 seconds to say that they are alive. If the namenode does not receive a heartbeat from a particular data node for 10 minutes, then it considers that data node to be dead/out of service and initiates replication of blocks which were hosted on that data node to be hosted on some other data node.
4. The data nodes can talk to each other to rebalance by move and copy data around and keep the replication high.
5. When the datanode stores a block of information, it maintains a checksum for it as well. The data nodes update the namenode with the block information periodically and before updating verify the checksums. If the checksum is incorrect for a particular block i.e. there is a ***disk level corruption for that block,*** it skips that block while reporting the block information to the namenode. In this way, namenode is aware of the disk level corruption on that datanode and takes steps accordingly.
6. **Blockreport** - The DataNode stores HDFS data in files in its local file system. The DataNode has no knowledge about HDFS files. It stores each block of HDFS data in a separate file in its local file system. The DataNode does not create all files in the same directory. Instead, it uses a heuristic to determine the optimal number of files per directory and creates subdirectories appropriately. It is not optimal to create all local files in the same directory because the local file system might not be able to efficiently support a huge number of files in a single directory. When a DataNode starts up, it scans through its local file system, generates a list of all HDFS data blocks that correspond to each of these local files and sends this report to the NameNode.

Terminologies



Cluster: A cluster is a group of servers and other resources that act like a single system and enable high availability and, in some cases, load balancing and parallel processing.
Rack: A computer rack (commonly called a rack) is a metal frame used to hold various hardware devices such as servers, hard disk drives, modems and other electronic equipment
Node: A single machine in cluster
Daemon: process which runs in background and has no controlling terminal.
Block: A block is the smallest unit of data that can be stored or retrieved from the disk. File systems deal with the data stored in blocks.



Rack Awareness: Replica Placement : Assuming the replication factor is 3; When a file is written from a data node (say R1N1), Hadoop attempts to save the first replica in same data node (R1N1). Second replica is written into another node (R2N2) in a different rack (R2). Third replica is written into another node (R2N1) in the same rack (R2) where the second replica was saved.

Hadoop Version 1 - Limitation wrt Storage

1. Name Node is single point of Failure- If name node is down the whole cluster is down, inorder to overcome this High Availability is brought.
2. Name Node scalability is not possible – Only one name node can be maintained to support all data nodes, no scalability was possible hence no isolation of name node.

HDFS High Availability

Overview

Before Hadoop 1.X version, the NameNode was a single point of failure (SPOF) in an HDFS cluster. Each cluster had a single NameNode, and if that machine or process became unavailable, the cluster as a whole would be unavailable until the NameNode was either restarted or brought up on a separate machine. The Secondary NameNode did not provide failover capability.

This reduced the total availability of the HDFS cluster in two major ways:

1. In the case of an unplanned event such as a machine crash, the cluster would be unavailable until an operator restarted the NameNode.

2. Planned maintenance events such as software or hardware upgrades on the NameNode machine would result in periods of cluster downtime.
   The HDFS HA feature addresses the above problems by providing the option of running two NameNodes in the same cluster, in an Active/Passive configuration. These are referred to as the Active NameNode and the Standby NameNode. Unlike the Secondary NameNode, the Standby NameNode is hot standby, allowing a fast failover to a new NameNode in the case that a machine crashes, or a graceful administrator-initiated failover for the purpose of planned maintenance. You cannot have more than two NameNodes.

   Architecture
   In a typical HA cluster, two separate machines are configured as NameNodes. At any point in time, one of the NameNodes is in an Active state, and the other is in a Standby state. The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough state to provide a fast failover if necessary.

   The namenode daemon is a single point of failure in Hadoop 1.x, which means that if the node hosting the namenode daemon fails, the filesystem becomes unusable. To handle this, the administrator has to configure the namenode to write the fsimage file to the local disk as well as a remote disk on the network. This backup on the remote disk can be used to restore the namenode on a freshly installed server. Newer versions of Apache Hadoop (2.x) now support **High Availability** (**HA**), which deploys two namenodes in an active/passive configuration, wherein if the active namenode fails, the control falls onto the passive namenode, making it active. This configuration reduces the downtime in case of a namenode failure.

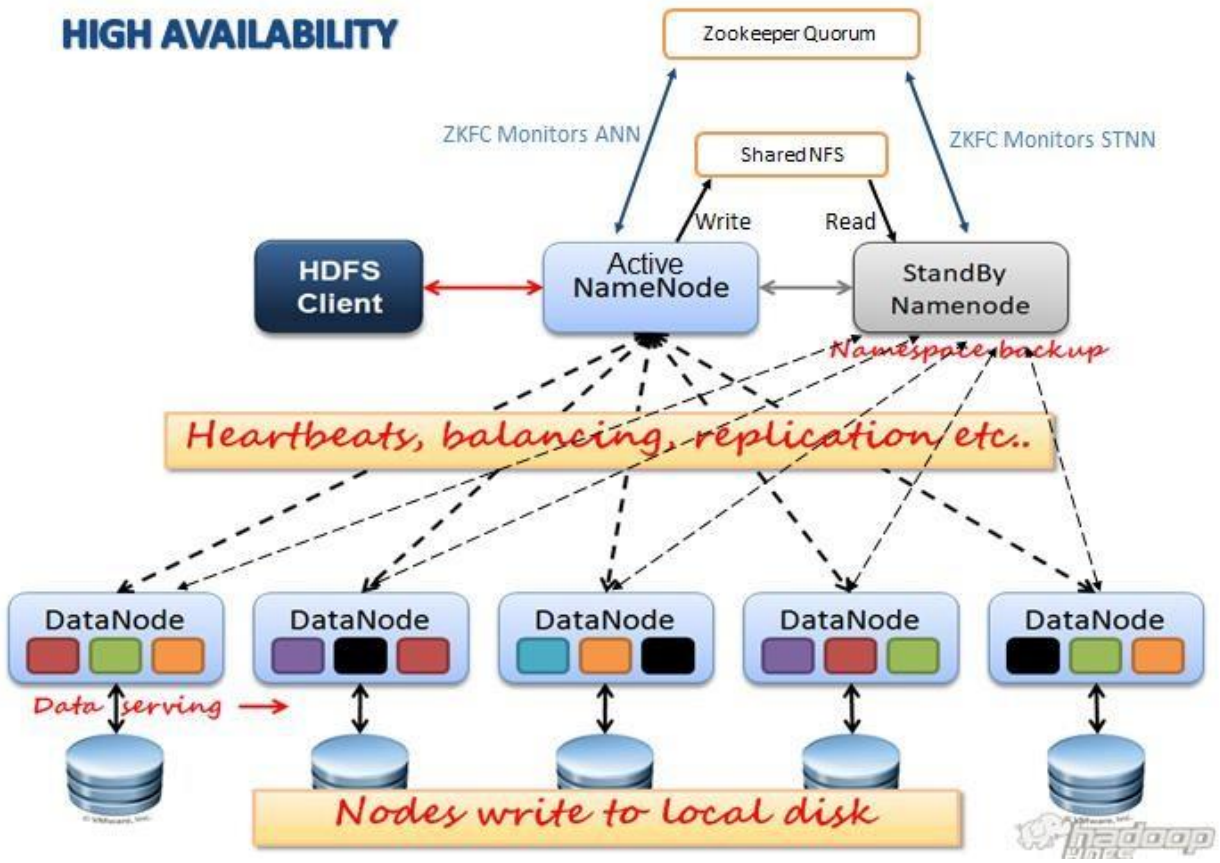
   In a typical HA cluster, two separate machines are configured as NameNodes. At any point in time, exactly one of the NameNodes is in an *Active* state, and the other is in a *Standby*state. The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough state to provide a fast failover if necessary.

   In order for the Standby node to keep its state synchronized with the Active node, the current implementation requires that the two nodes both have access to a directory on a shared storage device (eg an NFS mount from a NAS). This restriction will likely be relaxed in future versions.

   When any namespace modification is performed by the Active node, it durably logs a record of the modification to an edit log file stored in the shared directory. The Standby node is constantly watching this directory for edits, and as it sees the edits, it applies them to its own namespace. In the event of a failover, the Standby will ensure that it has read all of the edits from the shared storage before promoting itself to the Active state. This ensures that the namespace state is fully synchronized before a failover occurs.

In order to provide a fast failover, it is also necessary that the Standby node have up-to-date information regarding the location of blocks in the cluster. In order to achieve this, the DataNodes are configured with the location of both NameNodes, and send block location information and heartbeats to both.

It is vital for the correct operation of an HA cluster that only one of the NameNodes be Active at a time. Otherwise, the namespace state would quickly diverge between the two, risking data loss or other incorrect results. In order to ensure this property and prevent the so-called "split-brain scenario," the administrator must configure at least one ***fencing*** *method* for the shared storage which will be taken care by the Zookeper service (Zookeper failover controller) which manages the liveliness and health of the active and standby namenodes by retaining inuse.lock file in the active name node. During a failover, if it cannot be verified that the previous Active node has relinquished its Active state, the fencing process is responsible for cutting off the previous Active's access to the shared edits storage. This prevents it from making any further edits to the namespace, allowing the new Active to safely proceed with failover.



In order to provide a fast failover, it is also necessary that the Standby node have up-to-date information of the location of blocks in your cluster. Data Nodes are configured with the location of both the Name Nodes and send block location information and heartbeats to both Name Node machines.

Shared Storage Using NFS

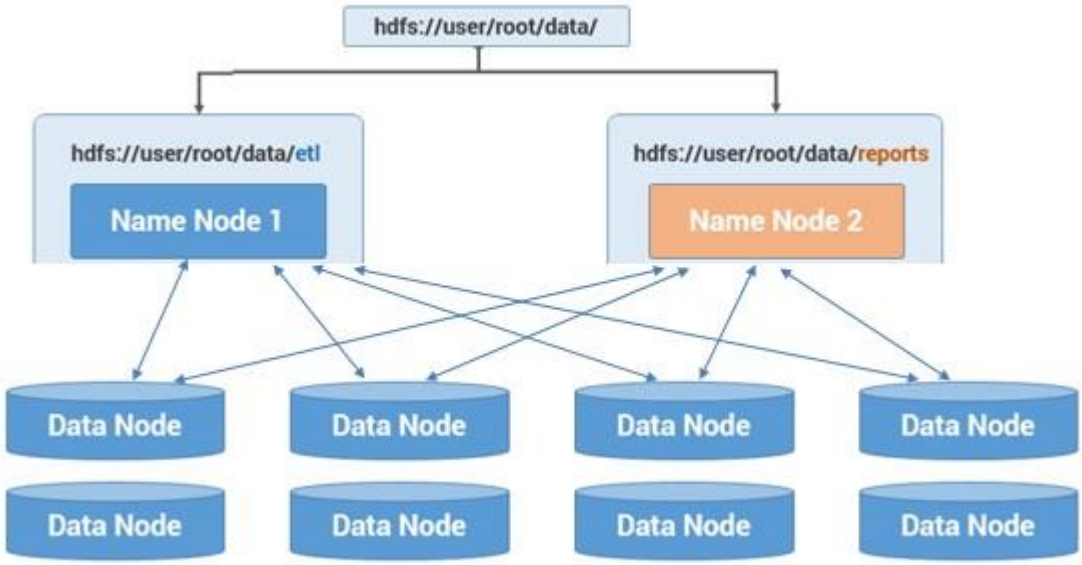
In order for the Standby node to keep its state synchronized with the Active node, this implementation requires that the two nodes both have access to a directory ong a shared storage device (for example, an NFS mount from a NAS).

Federation

In order to scale the name service horizontally, federation uses multiple independent Namenodes/namespaces. The Namenodes are federated, that is, the Namenodes are independent and don't require coordination with each other. The datanodes are used as common storage for blocks by all the Namenodes. Each datanode registers with all the Namenodes in the cluster. Datanodes send periodic heartbeats and block reports and handles commands from the Namenodes.

Block Pool

A Block Pool is a set of blocks that belong to a single namespace. Datanodes store blocks for all the block pools in the cluster. It is managed independently of other block pools. This allows a namespace to generate Block IDs for new blocks without the need for coordination with the other namespaces. The failure of a Namenode does not prevent the datanode from serving other Namenodes in the cluster.



A Namespace and its block pool together are called Namespace Volume. It is a self-contained unit of management. When a Namenode/namespace is deleted, the corresponding block pool at the datanodes is deleted. Each namespace volume is upgraded as a unit, during cluster upgrade.

ClusterIDA new identifier **ClusterID** is added to identify all the nodes in the cluster. When a Namenode is formatted, this identifier is provided or auto generated. This ID should be used for formatting the other Namenodes into the cluster.

Data Integrity

The property **io.bytes.per.checksum** controls for how many bytes the check CRC code is calculated. In the new version of hadoop this property has been changed to **dfs.bytes-per-checksum** . The default value of this is 512

bytes.It should not be larger than dfs.stream-buffer-size

Data nodes calculates the checksum for data it receives and raises CheckSumException if something is wrong

Whenever client reads data from DN , it responds back that checksum has been verfied to be correct and DN also updates the checksum log present in itself keeping info about when was the last time when checksum for the block was verified

Every DN also regularly runs DataBlockScanner which verifies the data stored in it for health.

Whenever a corrupted block is found NN starts replication of it from some healthy block so that required replication factor (3 by default , it means each block has 3 copies in HDFS) can be achieved.

While reading any file from HDFS using FileSystem API , we can also tell that we dont want to verify the checksum (for some reason) during this transfer.

File compression brings two major benefits: it reduces the space needed to store files, and it speeds up data transfer across the network or to or from disk. When dealing with large volumes of data, both of these savings can be significant, so it pays to carefully consider how to use compression in Hadoop.

Common input format

Compression format	Tool	Algorithm	File extension	Splittable
gzip	*gzip*	DEFLATE	.gz	No
bzip2	*bizp2*	bzip2	.bz2	Yes
LZO	*lzop*	LZO	.lzo	Yes if indexed
Snappy	N/A	Snappy	.snappy	No

Some tradeoffs:

All compression algorithms exhibit a space/time trade-off: faster compression and decompression speeds usually come at the expense of smaller space savings. The tools listed in above table typically give some control over this trade-off at compression time by offering nine different options: –1 means optimize for speed and -9 means optimize for space.The different tools have very different compression characteristics. Gzip is a general purpose compressor, and sits in the middle of the space/time trade-off. Bzip2 compresses more effectively than gzip, but is slower. Bzip2's decompression speed is faster than its compression speed, but it is still slower than the other formats. LZO and Snappy, on the other hand, both optimize for speed and are around an order of magnitude faster than gzip, but compress less effectively. Snappy is also significantly faster than LZO for decompression.

gzip:

gzip is naturally supported by Hadoop. gzip is based on the DEFLATE algorithm, which is a combination of LZ77 and Huffman Coding.

bzip2:

bzip2 is a freely available, patent free (see below), high-quality data compressor. It typically compresses files to within 10% to 15% of the best available techniques (the PPM family of statistical compressors), whilst being around twice as fast at compression and six times faster at decompression.

LZO:

The LZO compression format is composed of many smaller (~256K) blocks of compressed data, allowing jobs to be split along block boundaries. Moreover, it was designed with speed in mind: it decompresses about twice as fast as gzip, meaning it's fast enough to keep up with hard drive read speeds. It doesn't compress quite as well as gzip

— expect files that are on the order of 50% larger than their gzipped version. But that is still 20-50% of the size of the files without any compression at all, which means that IO-bound jobs complete the map phase about four times faster.

Snappy:

Snappy is a compression/decompression library. It does not aim for maximum compression, or compatibility with any other compression library; instead, it aims for very high speeds and reasonable compression. For instance, compared to the fastest mode of zlib, Snappy is an order of magnitude faster for most inputs, but the resulting compressed files are anywhere from 20% to 100% bigger. On a single core of a Core i7 processor in 64-bit mode, Snappy compresses at about 250 MB/sec or more and decompresses at about 500 MB/sec or more. Snappy is widely used inside Google, in everything from BigTable and MapReduce to our internal RPC systems.