BIG DATA ANALYTICS

Lekha A
Computer Applications

BIG DATA ANALYTICS

Introduction to Hadoop Eco System

Introduction

Lekha A
Computer Applications

BIG DATA ANALYTICS

How big is Big Data?

-   

- How much data do these organizations deal with?

BIG DATA ANALYTICS

How big is Big Data?

	Facebook	Instagram	Youtube
Current Storage	> 300 petabytes (10^{15})	~ 500 terabytes (10^{12})	one exabyte (1,000,000 terabytes)
Processed per day	4 PB		440,000 terabytes 19 minutes per day on YouTube.
Users per month	3.049 billion	1 billion (69 million in India)	2.7 billion monthly active users
Daily users	2.09 billion active users	63% active users	122 million
Likes / Stories per day	2.7 billion (4 million per minute)	500 million	3.7 million per day (videos)

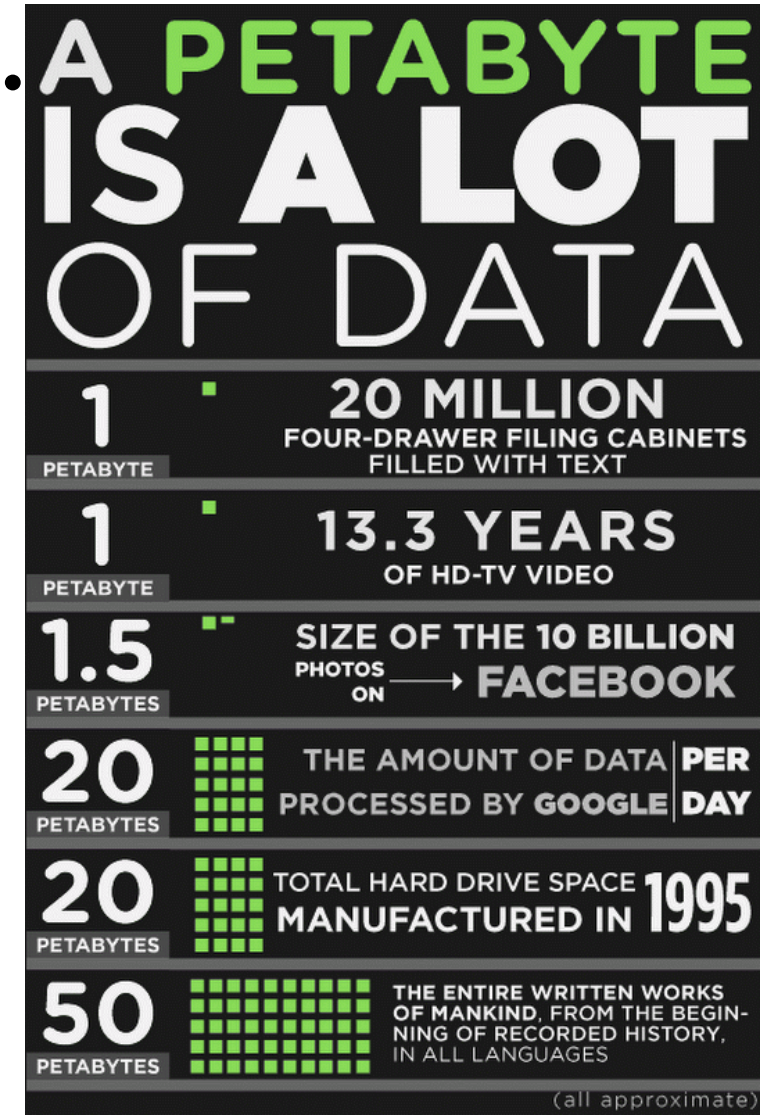
BIG DATA ANALYTICS

How big is Big Data?

	Facebook	Instagram	Youtube
Photos uploaded per day	350 million	46,740 photos per minute	
Time spent per day	33 minutes	< 25 years 32 minutes > 25 24 minutes	47.5 minutes per day
Content	136000 Photos per minute 8 billion – Videos 15% of all content	73.5% - images 13.7% - video 12.7% - carousels	> 114 million active YouTube channels - 321,000 channels have 100,000 subscribers or more.

BIG DATA ANALYTICS

How big is Big Data?

- 

Big Data System requirements

Raw Data

Store Data

Extract useful information

Process

????

Store massive amounts of data

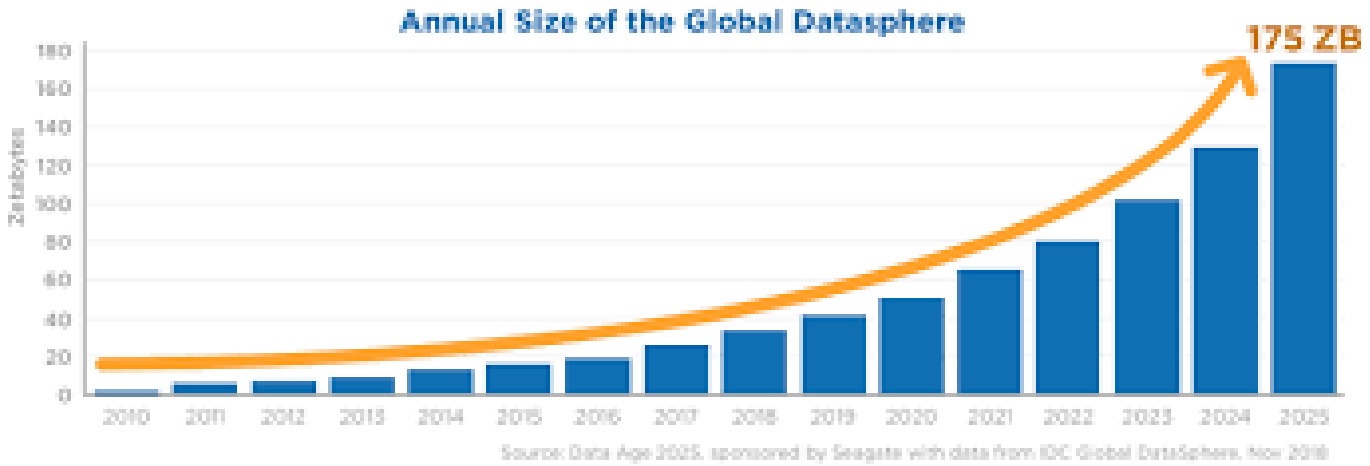
Process it in a timely manner

BIG DATA ANALYTICS

Big Data System Requirements

- Store Data??

Figure 1 - Annual Size of the Global Datasphere



- The infrastructure needs to keep up with the growing size of data

Big Data System requirements

Store	Process	Accommodate changing needs
Store massive amounts of data	Process it in a timely manner	Scale easily as data grows

- Distributed Computing Frameworks like Hadoop were developed for exactly this.

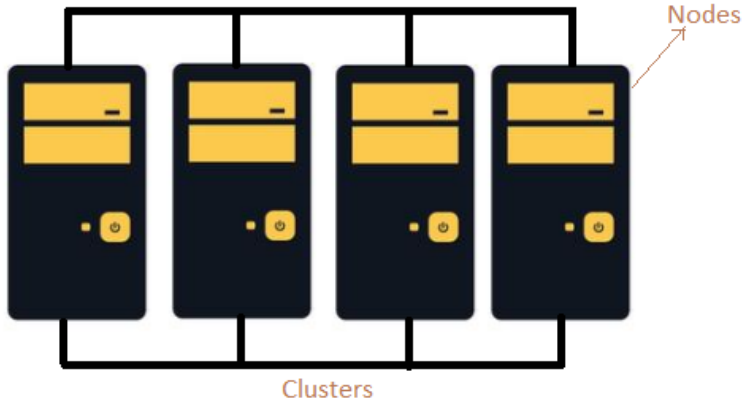
BIG DATA ANALYTICS

Big Data System requirements

Monolithic

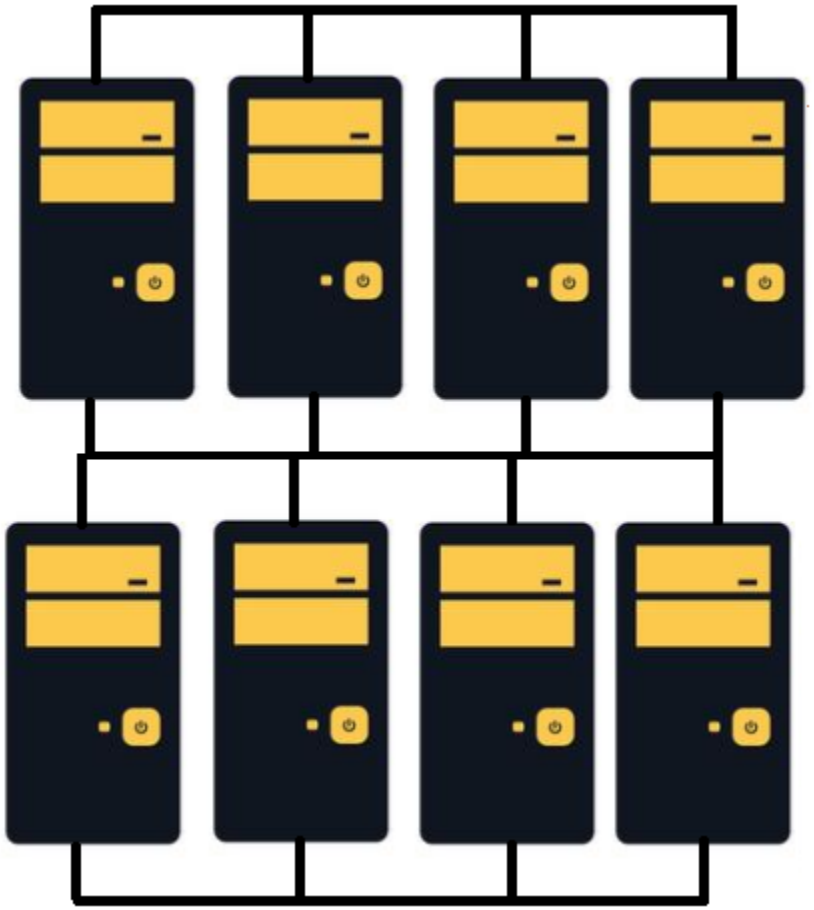
- Super Computer
- A single powerful server
- 2x Expense
- < 2x Performance

Distributed



- Cluster of machines
- Many small and cheap computers come together to act as a single entity
- Such a system can scale linearly

Building the system



2x Nodes

2x Storage

~ 2x Speed

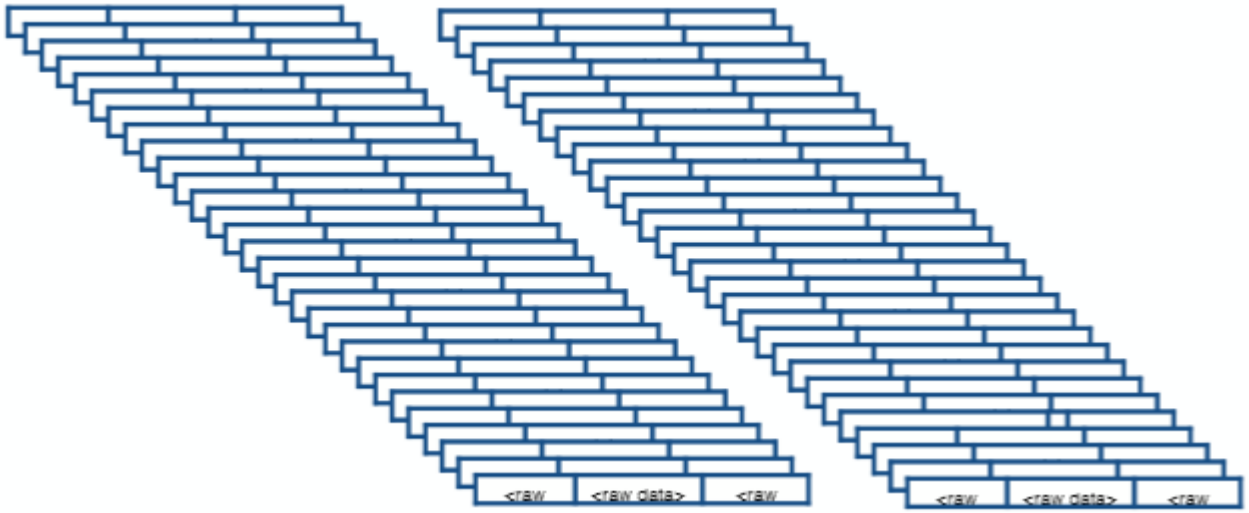
Example – Server Farms

- Companies like Facebook, Google, Amazon are building vast server farms
- These farms have 100s of 1000s of servers working in tandem to process complex data
- All of these servers need to be coordinated by a single piece of software.

BIG DATA ANALYTICS

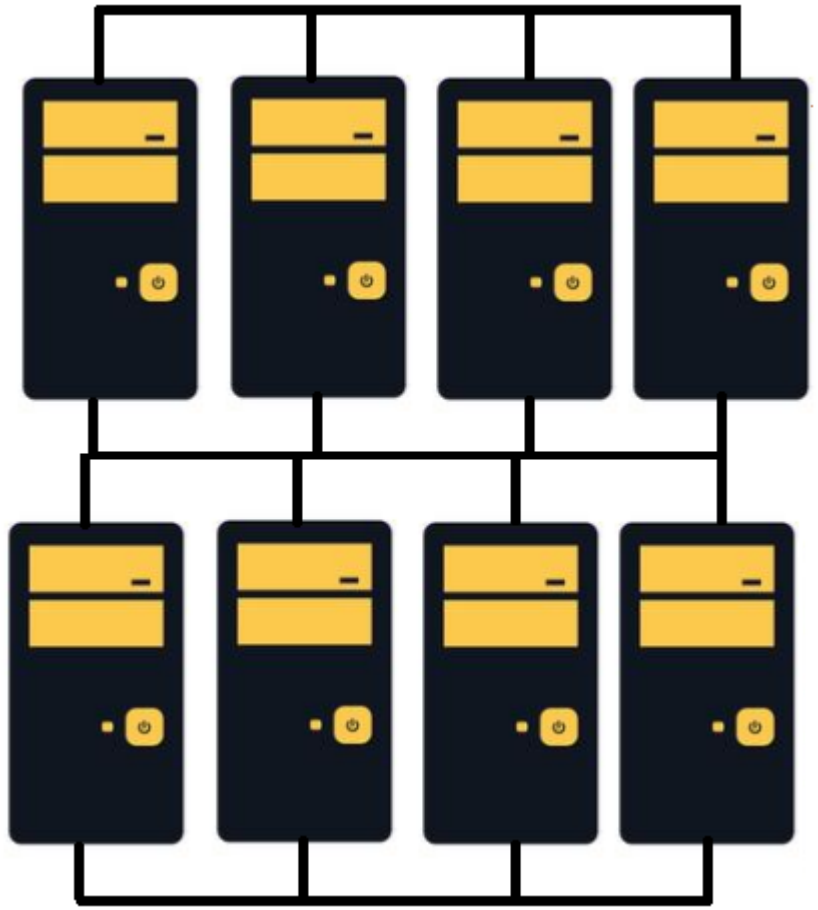
Example – Server Farms

- Single Co-ordinating System – Step 1
  - Store millions of records on multiple machines



Example – Server Farms

- Single Co-ordinating System – Step 2
  - Run processes on all these machines to crunch data



Single Coordinating Software

- 


To solve distributed storage

Used by Nutch distributed framework of Apache


To solve distributed computing

BIG DATA ANALYTICS

Single Coordinating Software

- A file system to manage the storage of data
- A framework to process data across multiple servers
- Programming module for parallel processing

Hadoop 2.0 in 2013

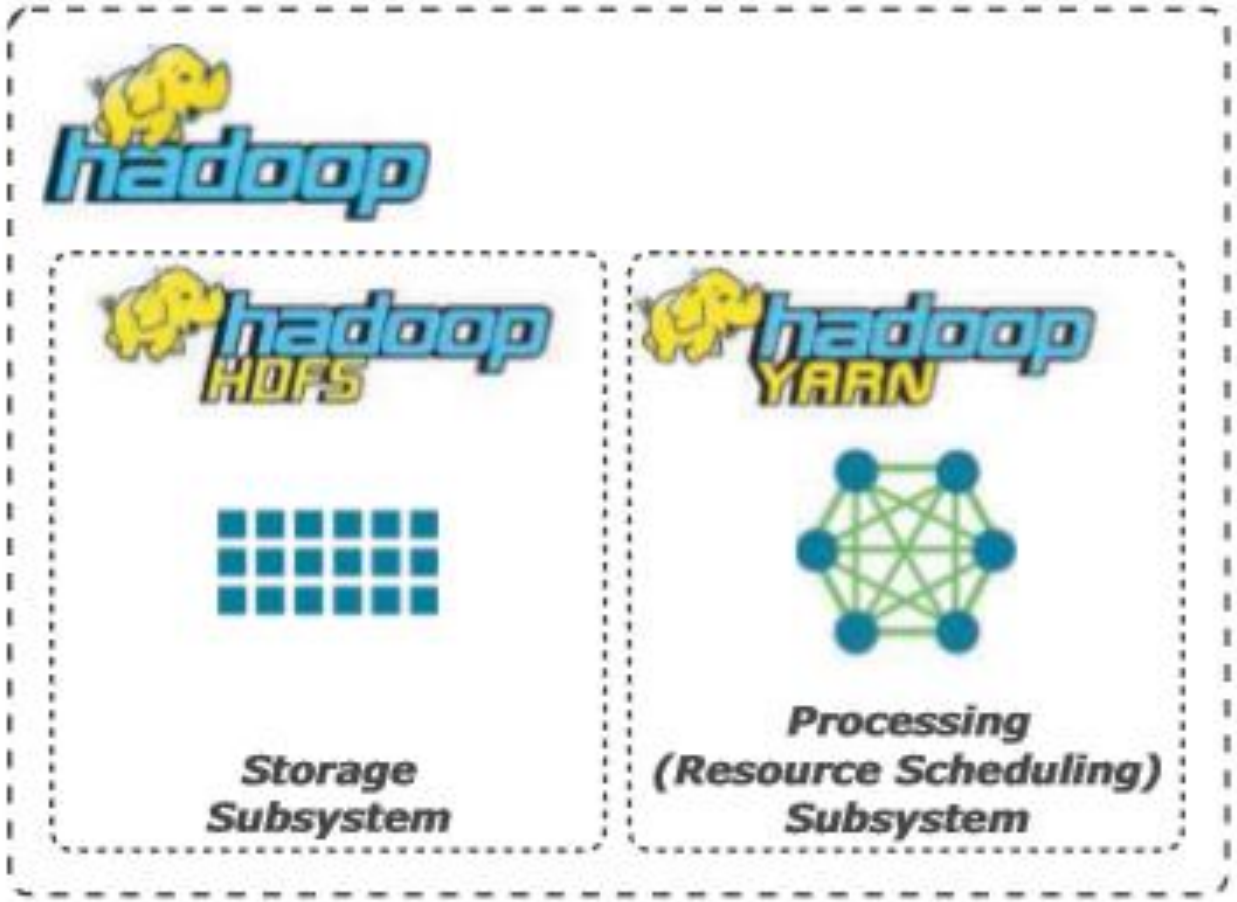
MapReduce was broken into two separate parts

A framework to define a data processing task

A framework to run the data processing task

BIG DATA ANALYTICS

Hadoop Core System



Hadoop Core System

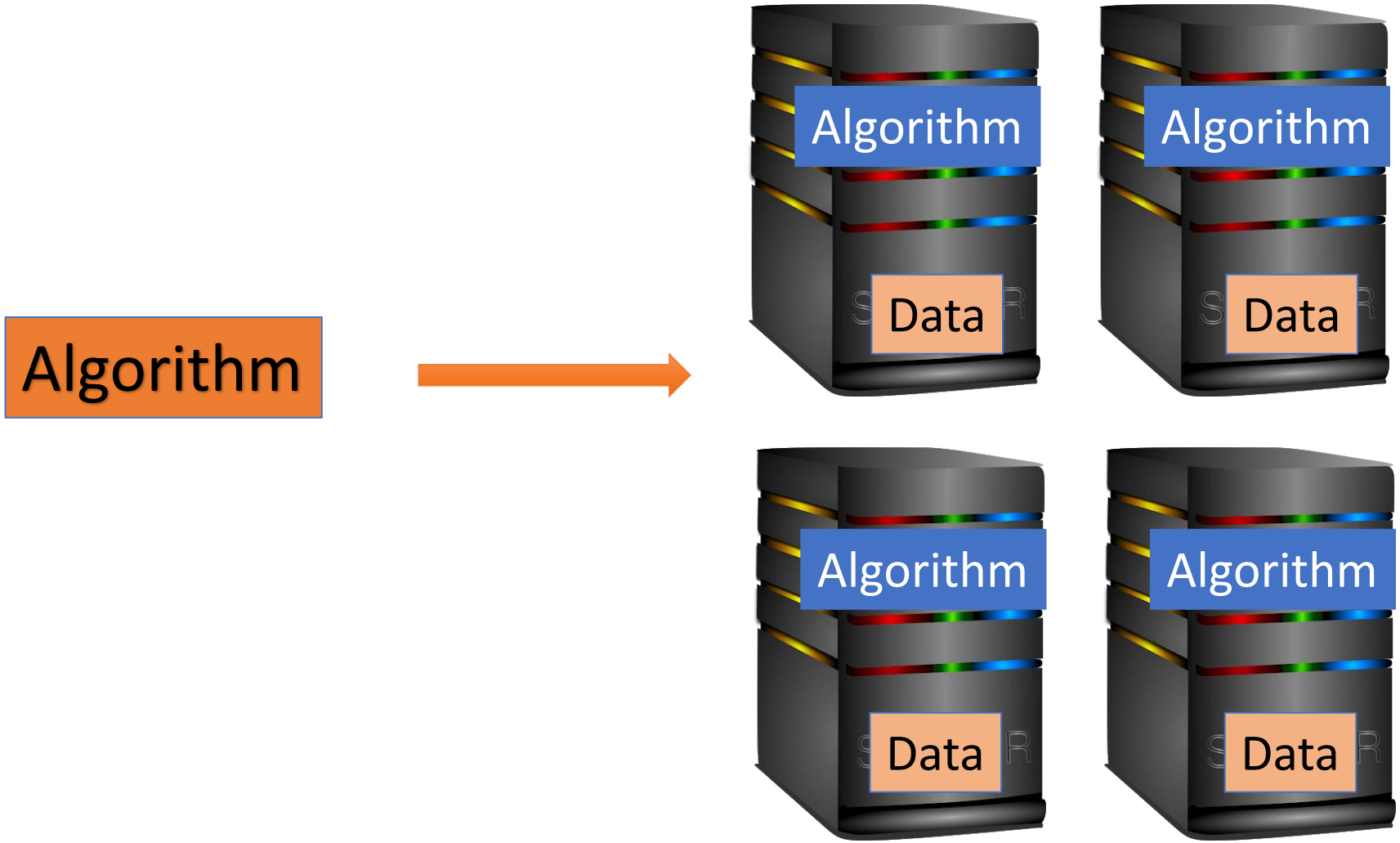
- Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library.
- Hadoop has its origins in Apache Nutch, an open source web-search engine, itself a part of the Lucene project.

Fig - https://www.computerweekly.com/news/450420002/Doug-Cutting-father-of-Hadoop-talks-about-big-data-tech-evolution

Hadoop

- Hadoop is a data storage and processing platform, based upon a central concept: data locality.
- Data locality
  - Processing of data where it resides by bringing the computation to the data, rather than the typical pattern of requesting data from its location (for example, a database management system) and sending this data to a remote processing system or host.

-


Algorithm
Algorithm
Data
Algorithm
Data
Algorithm
Data
Algorithm
Data

- Hadoop is schema-less with respect to its write operations (it is what's known as a schema-on-read system).
  - This means that it can store and process a wide range of data, from unstructured text documents, to semi-structured JSON (JavaScript Object Notation) or XML documents, to well structured extracts from relational database systems.

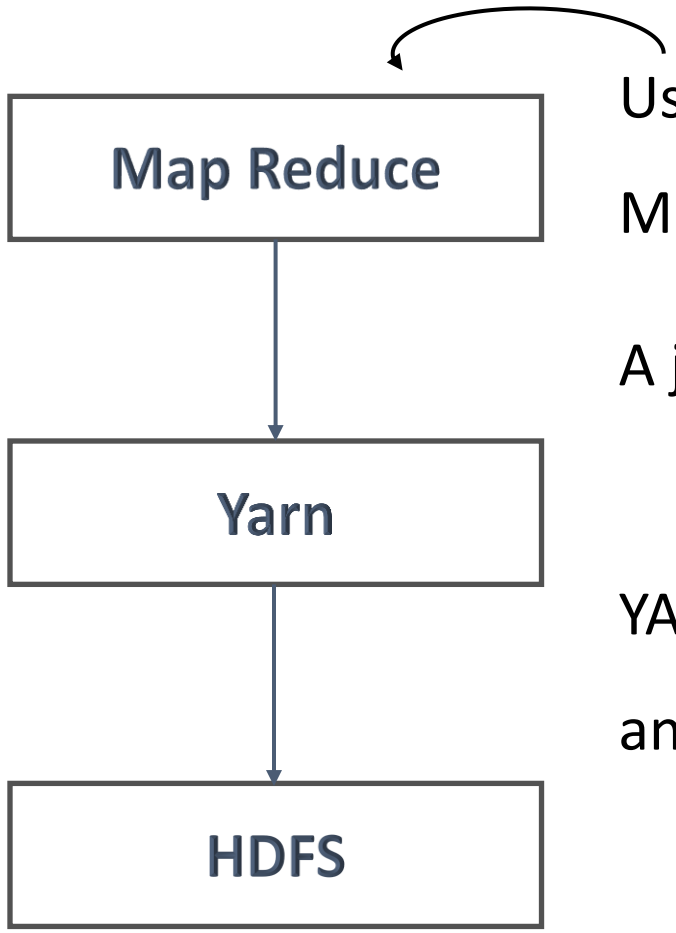
Hadoop

- Hadoop enables large datasets to be processed locally on the nodes of a cluster using a shared nothing approach, where each node can independently process a much smaller subset of the entire dataset without needing to communicate with one another.

Hadoop

- Hadoop is designed to find needles in haystacks.
  - It does so by dividing and conquering a large problem into a set of smaller problems applying the concepts of data locality and shared nothing.

Coordination between Hadoop blocks



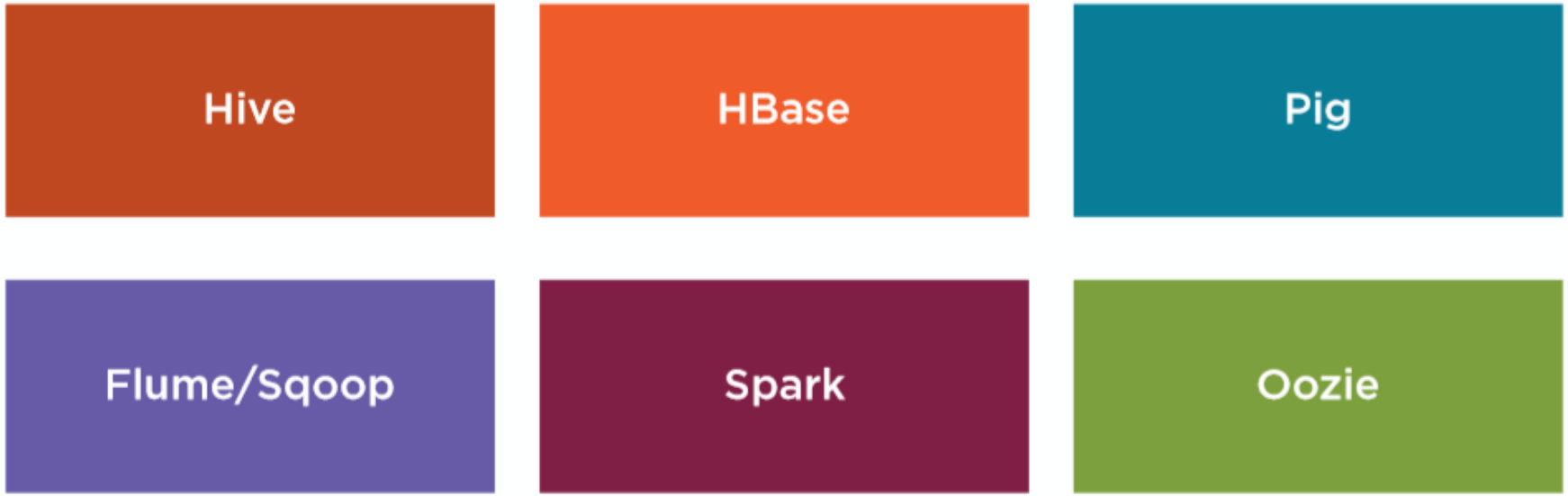
User defines map and reduce tasks using the MapReduce API

A job is triggered on the cluster

YARN figures out where and how to run the job, and stores the result in HDFS

Hadoop Eco System

- An ecosystem of tools have sprung up around this core piece of software
- It is a platform or a suite which provides various services to solve the big data problems.
-



Hadoop Eco System

- Provides an SQL interface to Hadoop
- The bridge to Hadoop for folks who don't have exposure to OOP in Java

- A database management system on top of Hadoop Integrates with the application just like a traditional database

Hadoop Eco System

- A data manipulation language.
- Transforms unstructured data into a structured format
- Query this structured data using interfaces like Hive

- A distributed computing engine used along with Hadoop
- Interactive shell to quickly process datasets
- Has a bunch of built in libraries for machine learning, stream processing, graph processing etc.

Hadoop Eco System

- A tool to schedule workflows on all the Hadoop ecosystem technologies

- Tools to transfer data between other systems and Hadoop

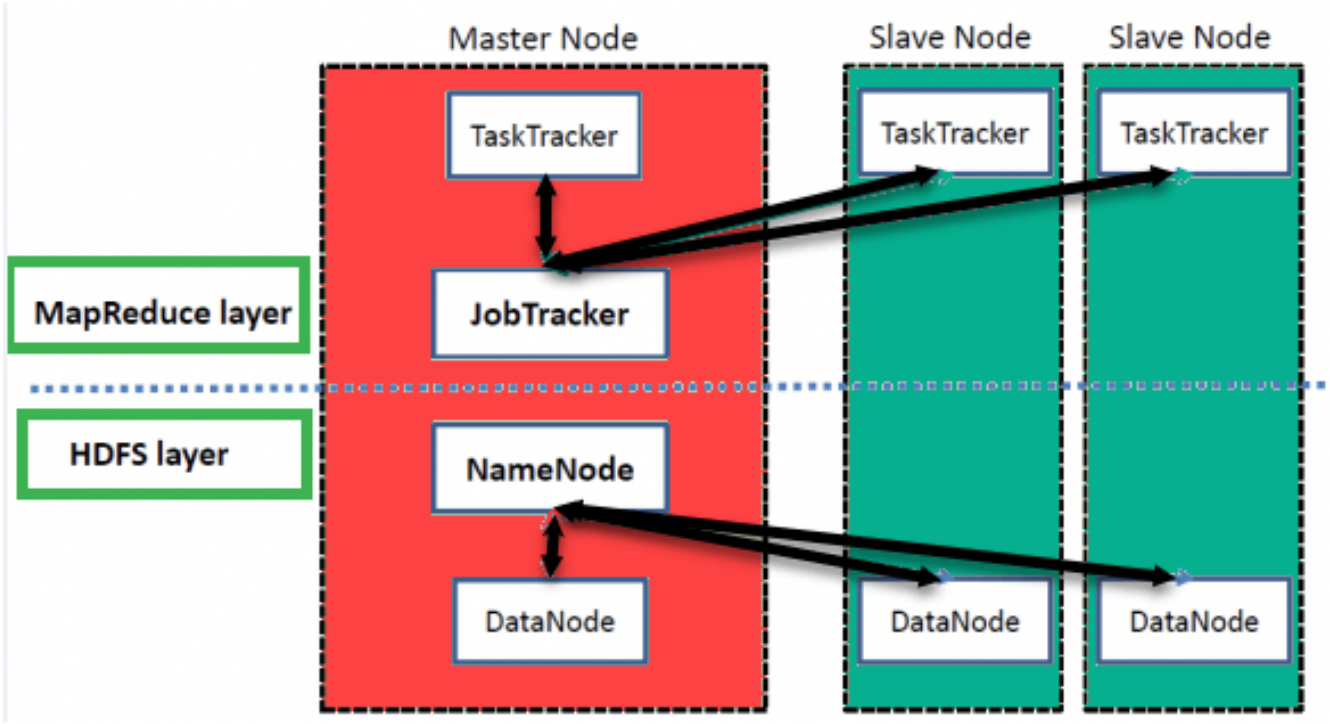
Hadoop Architecture

- The Hadoop architecture is a package of the file system, MapReduce engine and the HDFS (Hadoop Distributed File System).
- The MapReduce engine can be MapReduce/MR1 or YARN/MR2.
- A Hadoop cluster consists of a single master and multiple slave nodes.

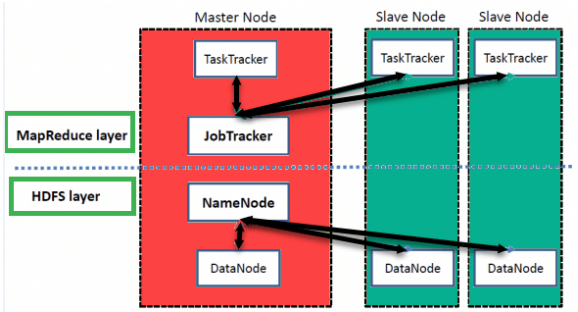
Hadoop Architecture

- The master node includes Job Tracker, Task Tracker, NameNode, and DataNode.

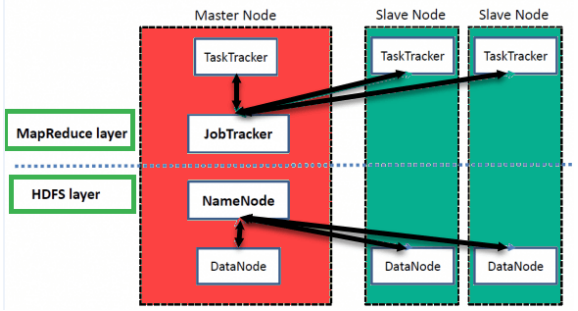
- The slave node includes DataNode and TaskTracker.



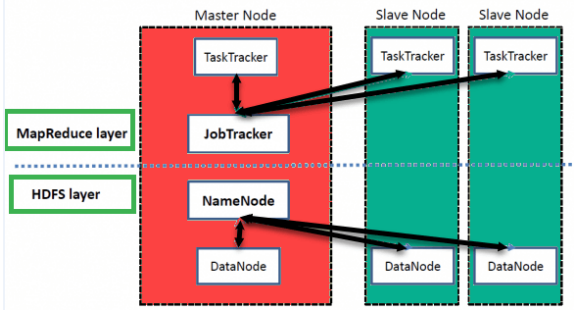
- NameNode
  - It is a single master server exist in the HDFS cluster.
  - As it is a single node, it may become the reason of single point failure.
  - It manages the file system namespace by executing an operation like the opening, renaming and closing the files.
  - It simplifies the architecture of the system.



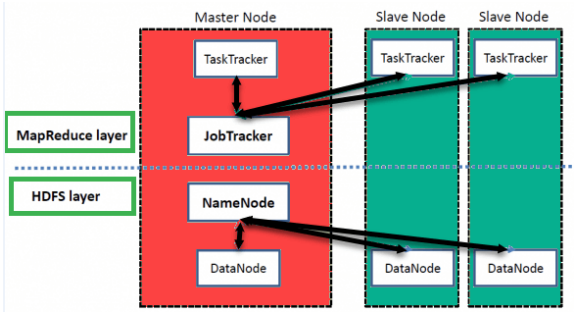
- DataNode
  - The HDFS cluster contains multiple DataNodes.
  - Each DataNode contains multiple data blocks.
  - These data blocks are used to store data.



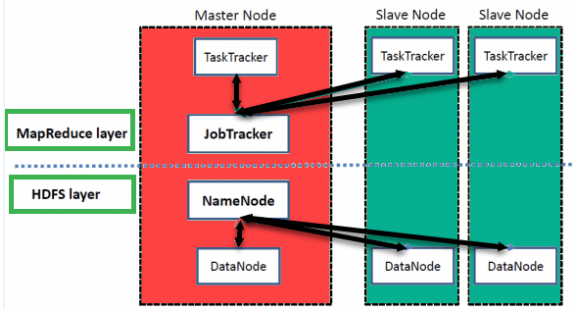
- DataNode
  - It is the responsibility of DataNode to read and write requests from the file system's clients.
  - It performs block creation, deletion, and replication upon instruction from the NameNode.



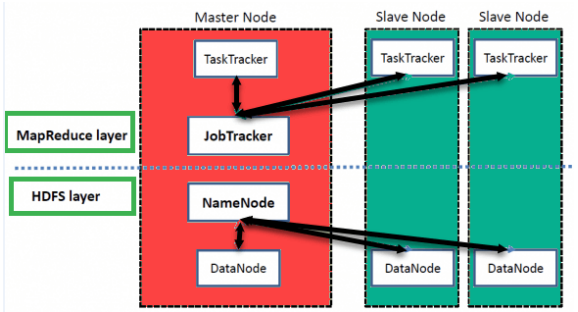
- Job Tracker
  - The role of Job Tracker is to accept the MapReduce jobs from client and process the data by using NameNode.
  - In response, NameNode provides metadata to Job Tracker.



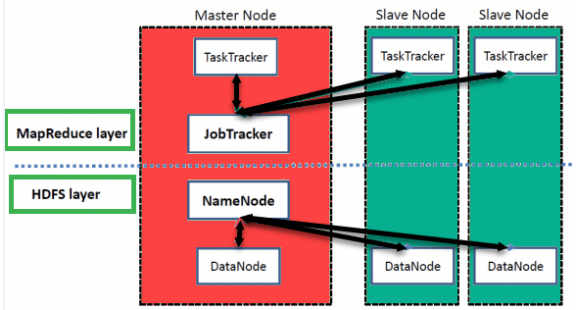
- Master node
  - The master node allows the user to conduct parallel processing of data using Hadoop MapReduce.



- Slave node
  - The slave nodes are the additional machines in the Hadoop cluster which allows the user to store data to conduct complex calculations.
  - All the slave node comes with Task Tracker and a DataNode.
  - This allows the user to synchronize the processes with the NameNode and Job Tracker respectively.



- MapReduce Layer
  - The MapReduce comes into existence when the client application submits the MapReduce job to Job Tracker.
  - In response, the Job Tracker sends the request to the appropriate Task Trackers.
  - Sometimes, the TaskTracker fails or time out.
    - In such a case, that part of the job is rescheduled.



BIG DATA ANALYTICS

Advantages of Hadoop

- 

Advantages of Hadoop

- **Fast**
  - In HDFS the data distributed over the cluster and are mapped which helps in faster retrieval.
  - Even the tools to process the data are often on the same servers, thus reducing the processing time.
  - It is able to process terabytes of data in minutes and Peta bytes in hours.

Advantages of Hadoop

- **Scalable**
  - Hadoop cluster can be extended by just adding nodes in the cluster.
- **Cost Effective**
  - Hadoop is open source and uses commodity hardware to store data so it really cost effective as compared to traditional relational database management system.

Advantages of Hadoop

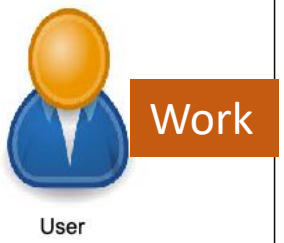
- **Resilient to failure**
  - HDFS has the property with which it can replicate data over the network, so if one node is down or some other network failure happens, then Hadoop takes the other copy of data and use it.
  - Normally, data are replicated thrice but the replication factor is configurable.

BIG DATA ANALYTICS

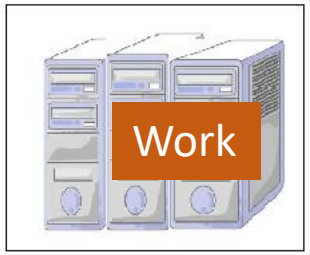
Advantages of Hadoop

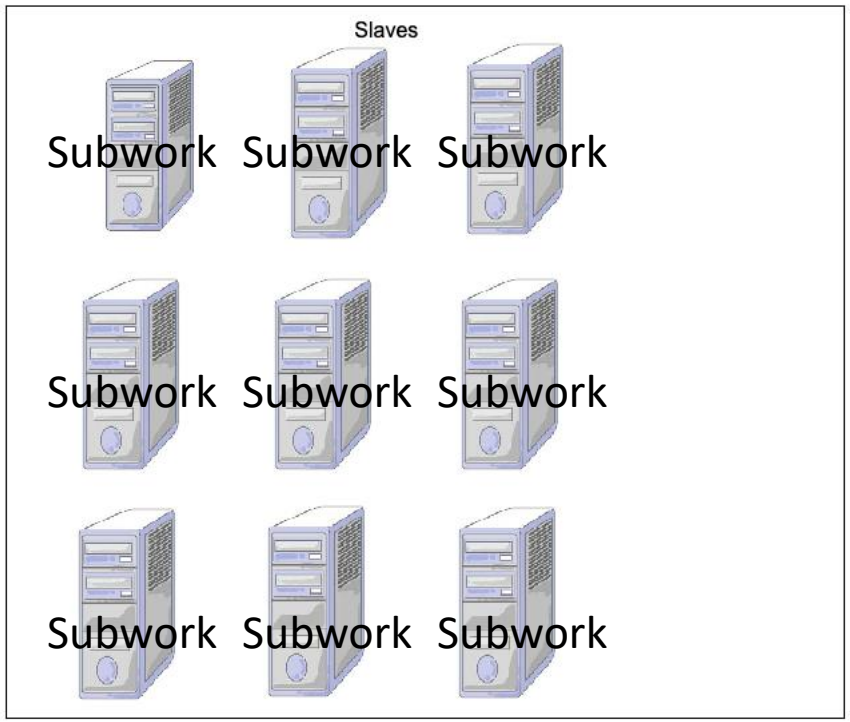
- Master assigns Subwork to slaves

User submits the work on the master



Work





Develops work

Advantages of Hadoop

- **Resilient to failure**
  - HDFS has the property with which it can replicate data over the network, so if one node is down or some other network failure happens, then Hadoop takes the other copy of data and use it.
  - Normally, data are replicated thrice but the replication factor is configurable.

BIG DATA ANALYTICS

Recap

- What is big data?
- How large is big data?
- What is Hadoop?
- History of Hadoop
- Components
- Eco System
- Architecture of Hadoop
  - DataNode
  - NameNode
  - Job Tracker
  - Task Tracker
  - Master Node
  - Slave Node
  - MapReduce Layer
  - HDFS Layer
- Advantages of Hadoop

THANK YOU

Lekha A
Computer Applications

BIG DATA ANALYTICS

Lekha A
Computer Applications

PES UNIVERSITY
CELEBRATING 50 YEARS

BIG DATA ANALYTICS

Introduction to Hadoop Eco System

Hadoop Distributed File System

Lekha A
Computer Applications

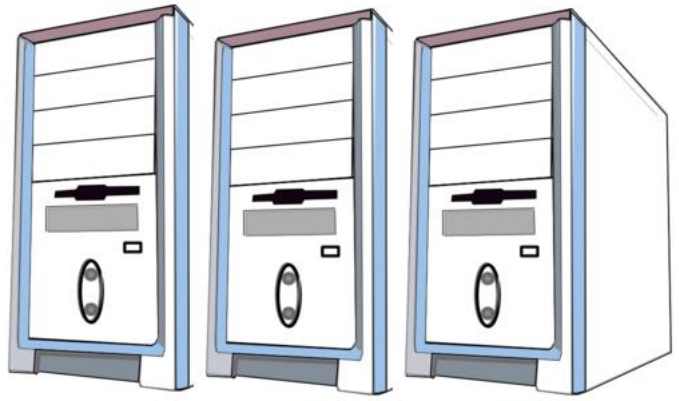
BIG DATA ANALYTICS

Recap

- What is big data?
- How large is big data?
- What is Hadoop?
- History of Hadoop
- Components
- Eco System
- Architecture of Hadoop
  - DataNode
  - NameNode
  - Job Tracker
  - Task Tracker
  - Master Node
  - Slave Node
  - MapReduce Layer
  - HDFS Layer
- Advantages of Hadoop

Hadoop Distributed File System

- HDFS is a file system designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware.
- Hadoop Distributed File System
- It is a file system that is spread across multiple machines.



Hadoop Distributed File System

- Built on commodity hardware
- Highly fault tolerant, hardware failure is the norm
- Suited to batch processing - data access has high throughput rather than low latency
- Supports very large data sets

Hadoop Distributed File System

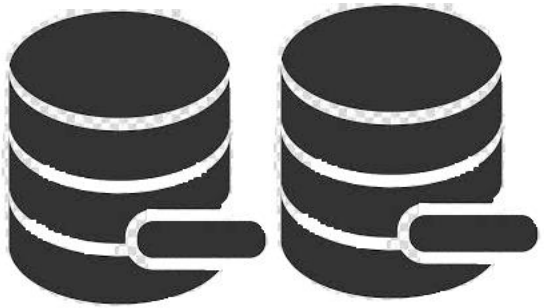
- Any data that is stored in HDFS is split across multiple storage disks.
- Each disk is present on a different machine in a cluster.
- The file system's responsibility is to manage all the machines and all the storage space.
  - This is done by master-slave architecture.

- It randomly sets up one machine as master node.
  - The master node is responsible for coordinating storage across all other nodes on the machine which are slave nodes.

- On the master node HDFS runs a Java Process that receives all requests that are made to the cluster and then forwards these requests to the slave nodes in the cluster.
- This process is called the name node.
  - The node itself is designated as name node.
- On all the other nodes HDFS runs the data node processes.
  - All other machines are designated as data node.

- One name node per cluster and many data nodes depending on the number of machines in the cluster.

Name Node



Data Node

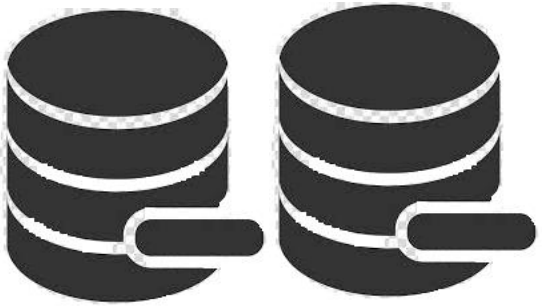
Hadoop Distributed File System

- If the data in the distributed file system is a book

- The name node is the table of contents

Name Node

- The data nodes hold the actual text in each page



Data Node

Name Node

- Manages the overall file system
  - No data is stored in the name node
- Stores
  - The directory structure
  - Metadata of the files

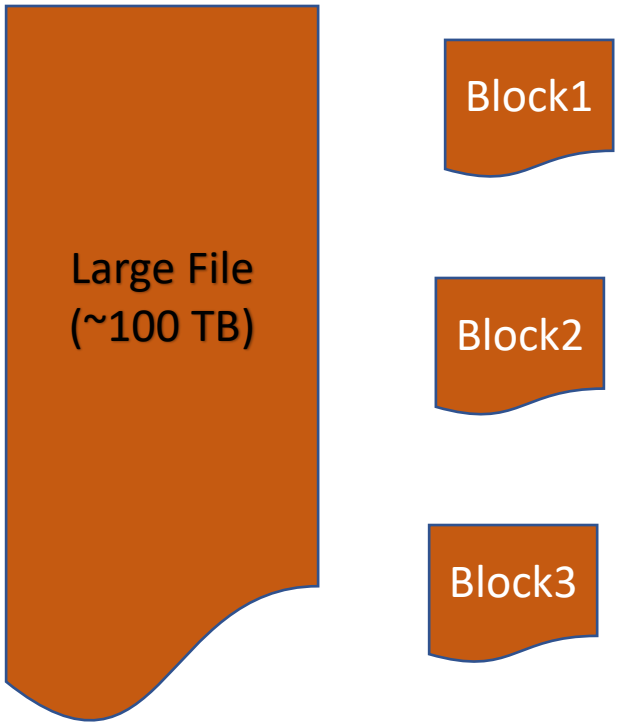
Data Node

- Physically stores the data in the files.
  - This is usually unformatted and unstructured data

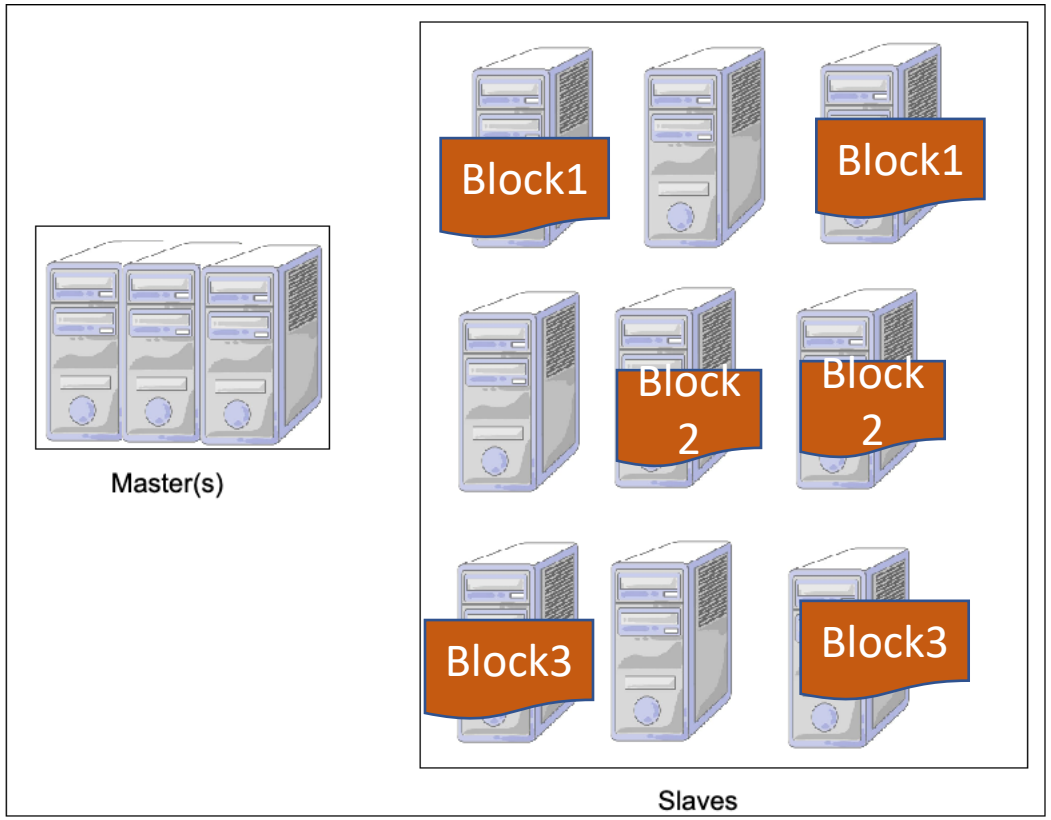
Hadoop Distributed File System

File is divided into smaller

- blocks of size 128 MB



Blocks are stored distributed across cluster



Hadoop Cluster

Blocks are replicated for fault tolerance

BIG DATA ANALYTICS

Storing Files in HDFS

- Data stored in HDFS is normally a large text file of data that is in peta bytes.

Two years ago, on the first of July, residents of Minneapolis saw Midwest summer weather at its ugliest. According to the Minneapolis Star-Tribune, the skies turned a "surreal" shade of black and green as a squall line moved toward the city. In just half an hour, three inches of rain doused parts of the city. A foot of water rushed into the Mall, and a power outage forced doctors in one hospital to deliver babies by flashlight.

To the northwest, things were a whole lot worse. Twelve tornadoes rampaged through the surrounding counties, knocking down houses by the hundreds and trees by the thousands. The town of Monticello, the Star-Tribune reported, looked like a logging camp. "There were no fatalities - amazingly," says Gregory Tipton, who at the time was a journeyman forecaster in charge of reading the Doppler radar at the National Center.

There was little warning for these tornadoes because, according to Tipton, the July 1 storm fooled meteorologists. Most of its tornadoes formed from the ground up - the opposite of a textbook tornado, which is supposed to descend from the clouds. The standard model of tornado development invokes the "dynamic pipe effect." In this model, a tornado gets started three miles or so above the ground with the formation of a broad vortex up to several miles wide, called a mesocyclone. Within the vortex air converges into smaller eddies, and as it converges it starts spinning faster, in the same way that a figure skater spins faster when she pulls in her arms. Eventually, the rotating air column becomes condensed enough to form a barrier that keeps any more air at its own level from getting in. Then the vortex has to suck air in from below. This, in turn, causes the air at the lower levels to circulate faster. Eventually the barrier moves down to the lower levels, and the column of spinning air (the "dynamic pipe") grows longer. It's a fairly slow process, so forecasters usually have about 20 minutes from the appearance of the first eddy on their radar screens until the touchdown of the tornado and the extra point.

But in a forthcoming study of tornado patterns all over the country, nearly half the tornadoes failed to conform to the dynamic-pipe model. These "nondescending" tornadoes, which form either from the bottom up or at all heights simultaneously, are more likely to be associated with squall lines, as the Minnesota tornadoes were, and they give much less advance warning of their presence than the typical top-down tornado.

Three years ago, Jeffrey Trapp and Robert Davies-Jones of the National Severe Storms Laboratory in Norman, Oklahoma derived a mathematical model that explains how a tornado can grow from the ground up, describing their work in the Journal of Atmospheric Sciences. If air within the mesocyclone starts to converge first at ground level (Trapp says the formation of the mesocyclone at ground level is "still hotly debated"), then updrafts resulting from the buoyancy of hot air can stretch the vortex upward. As it stretches, it becomes narrower, and (think of the figure skater again) spins faster and faster. In as little as five minutes, according to computer simulations, it can spin up to a full-fledged tornado and wreak havoc on animals, barns, and children.

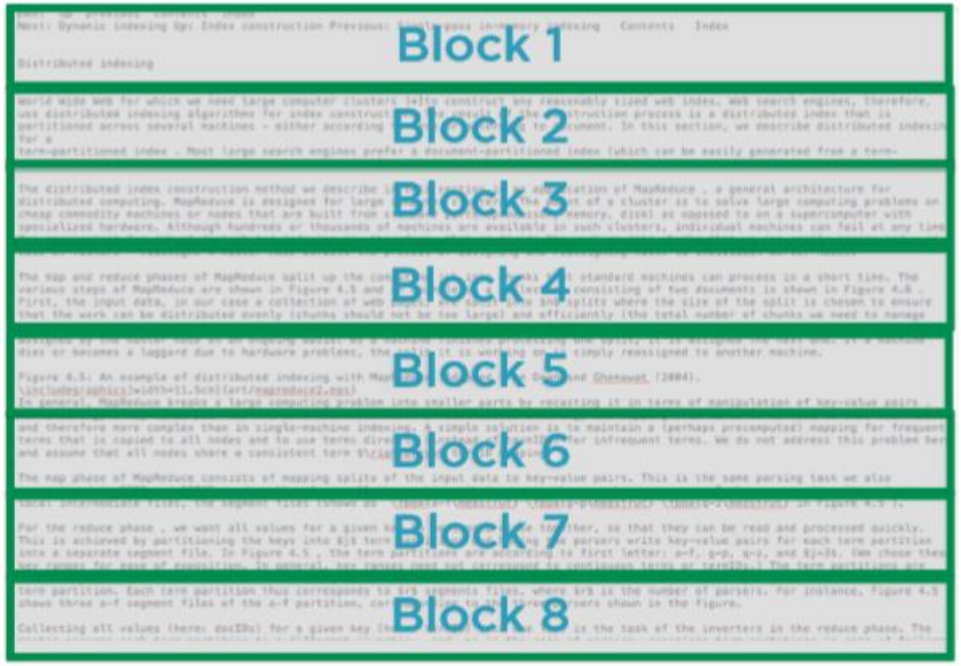
To see how often this behavior is actually observed in the field, Trapp went to DeWayne Mitchell, the developer of the Tornado Detection Algorithm now installed on all Doppler radar stations nationwide. "Someone on the ground would not be able to distinguish a tornado that formed on the ground from one that formed aloft," explains Mitchell. The visible shape of a funnel cloud depends on the debris it collects and the condensation point of water, and not directly on the winds that give it birth, which might be quite invisible. They are not invisible, however, to Doppler radar. The telltale sign is

Storing Files in HDFS

- These files are normally broken in to smaller chunks of information called blocks.
- Each of these blocks are stored on different nodes in a cluster.
- The entire file is not stored in one node.

BIG DATA ANALYTICS

Storing Files in HDFS

- 

Storing Files in HDFS

- Each of these blocks are of equal size.
  - Different length files are treated in the same way.
  - Equal block size ensures equal amount of processing time for all queries.
- Storage is simplified

Storing Files in HDFS

- A block is the unit for replication and fault tolerance
  - Multiple copies of blocks of data are kept and not multiple copies of the entire data.
- The blocks are of an optimal size of 128 MB

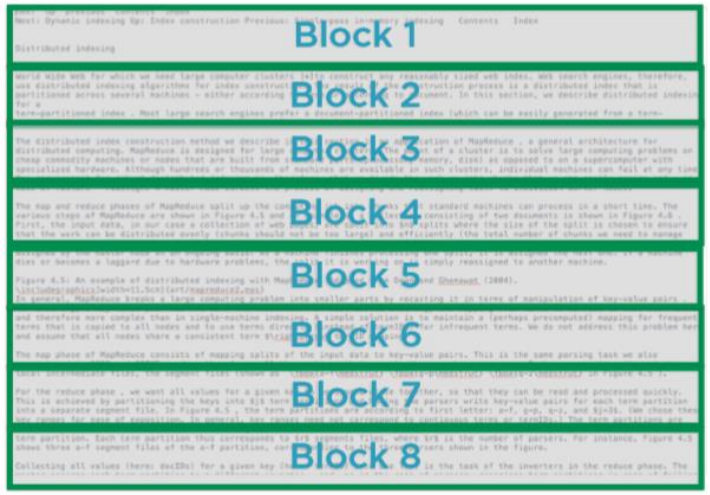
Storing Files in HDFS

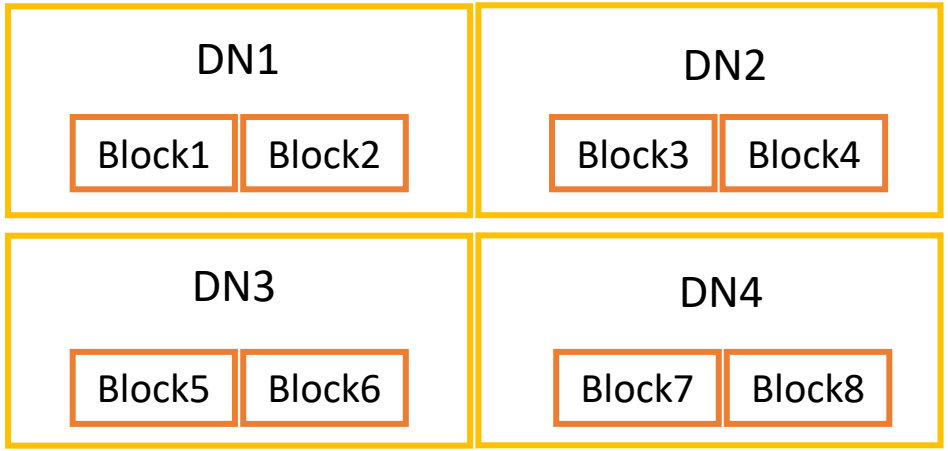
- Block size is a tradeoff .
  - If block size is increase, parallelism is reduced.
    - Fewer chunks of data.
    - Fewer processes.
  - If block size is too small, increases overheads.
    - One file will have many hundreds of splits
    - Requires hundreds of splits

- Time taken to read a block of data from disk is divided into
  - Seek time
    - Time taken to seek that position
    - Is roughly 1% of transfer time
  - Read time / Transfer time
    - Time taken to read the block

Storing Files in HDFS

- Store the blocks across the data nodes





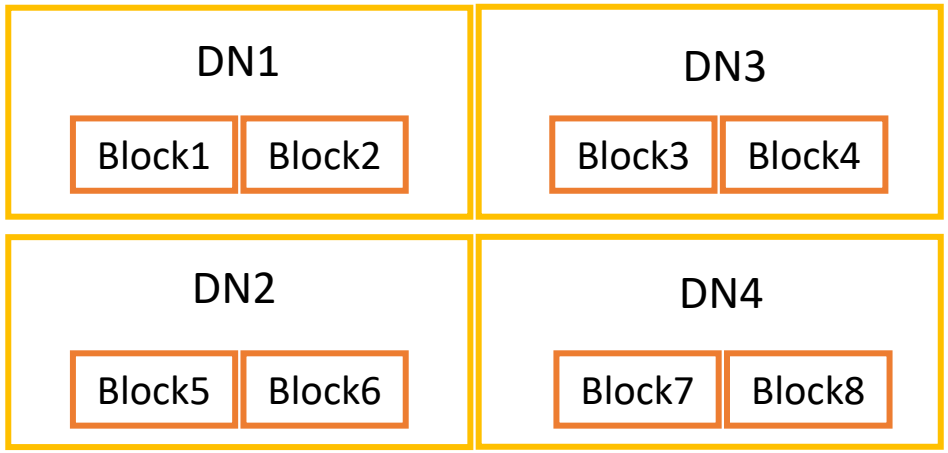
- Each node contains a partition or a split of data

Reading a File in HDFS

- Use metadata in the name node to look up block locations
- Read the blocks from respective locations

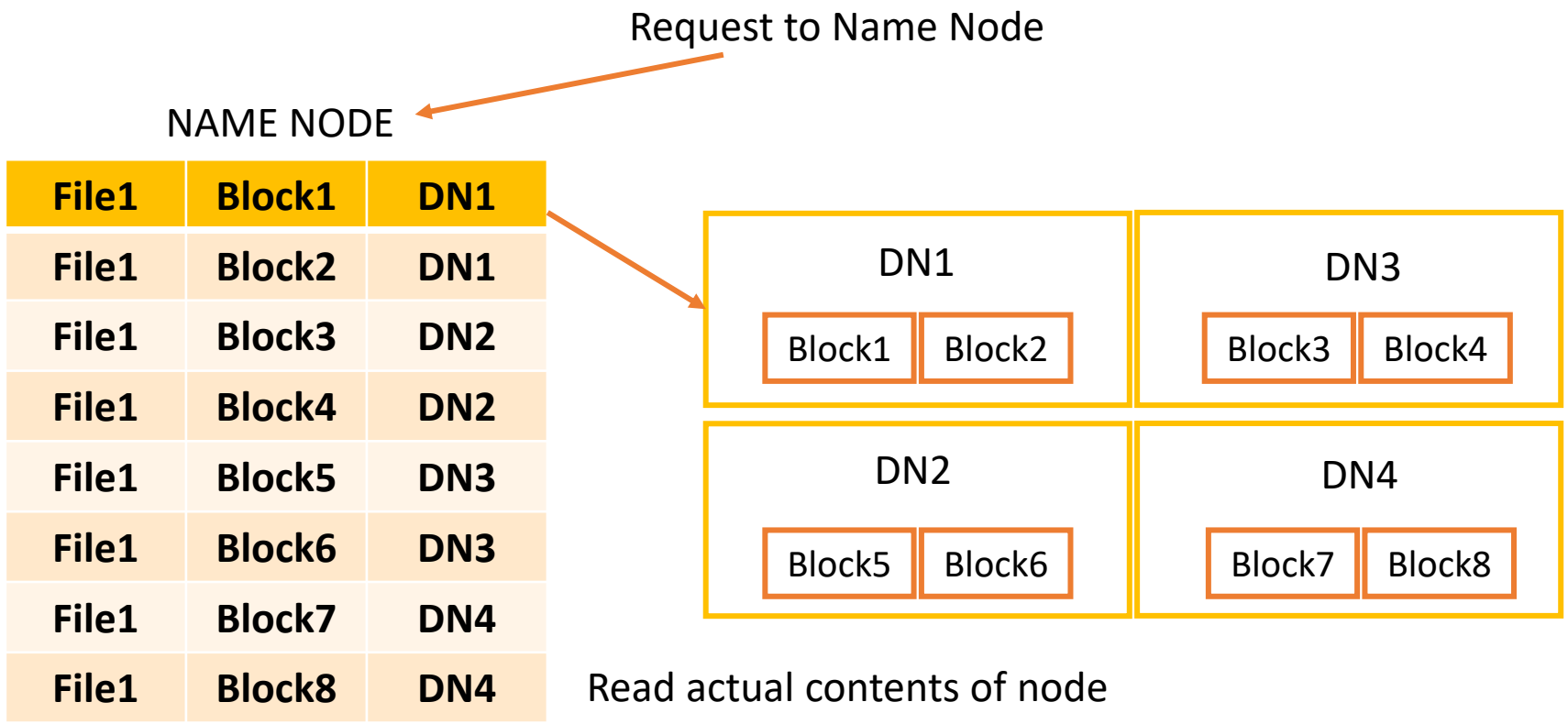
NAME NODE

File1	Block1	DN1
File1	Block2	DN1
File1	Block3	DN2
File1	Block4	DN2
File1	Block5	DN3
File1	Block6	DN3
File1	Block7	DN4
File1	Block8	DN4



BIG DATA ANALYTICS

Reading a File in HDFS



Hadoop Installation Modes

- Hadoop has three installation modes
  - Standalone
  - Pseudo-distributed
  - Fully Distributed

Standalone Installation Mode

- The default mode in which Hadoop runs.
- Runs on a single node
- A single JVM process
- Local File System for Storage
- HDFS and YARN do not run
- Used to test MapReduce programs before running them on a cluster

BIG DATA ANALYTICS
Pseudo-distributed Mode

- Runs on a single node
- 2 JVM processes to simulate 2 nodes
- HDFS for storage
- YARN for managing tasks
- Used as a fully-fledged test environment

BIG DATA ANALYTICS
Fully-distributed Mode

- Runs on a cluster of machines
  - Linux servers in a data center
  - VMs requisitioned on a cloud service
- Manual configuration of a cluster is complicated
- Usually use enterprise editions
  - Cloudera, MapR, Hortonworks

BIG DATA ANALYTICS
Fully-distributed Mode

- Runs on a cluster of machines
  - Linux servers in a data center
  - VMs requisitioned on a cloud service
- Manual configuration of a cluster is complicated
- Usually use enterprise editions
  - Cloudera, MapR, Hortonworks

BIG DATA ANALYTICS

Recap

- Architecture of Hadoop
- Advantages of Hadoop
- Hadoop Distributed File System
- Storing files
- Reading files
- Hadoop Installation Modes

THANK YOU

Lekha A
Computer Applications

BIG DATA ANALYTICS

Lekha A
Computer Applications

PES UNIVERSITY
CELEBRATING 50 YEARS

BIG DATA ANALYTICS

Introduction to Hadoop Eco System

HDFS concepts in detail

Lekha A
Computer Applications

BIG DATA ANALYTICS

Recap

- Architecture of Hadoop
- Advantages of Hadoop
- Hadoop Distributed File System
- Storing files
- Reading files
- Hadoop Installation Modes

Design of HDFS

- HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware.

Design of HDFS

- Very large files
  - "Very large" in this context means files that are hundreds of megabytes, gigabytes, or terabytes in size.

Design of HDFS

- Streaming data access
  - HDFS is built around the idea that the most efficient data processing pattern is a write-once, read-many-times pattern.
  - A dataset is typically generated or copied from source, then various analyses are performed on that dataset over time.
  - Each analysis will involve a large proportion, if not all, of the dataset, so the time to read the whole dataset is more important than the latency in reading the first record.

Design of HDFS

- Commodity hardware
  - Hadoop doesn't require expensive, highly reliable hardware to run on. It's designed to run on clusters of commodity hardware (commonly available hardware available from multiple vendors3) for which the chance of node failure across the cluster is high, at least for large clusters.
  - HDFS is designed to carry on working without a noticeable interruption to the user in the face of such failure.

Blocks

- A disk has a block size, which is the minimum amount of data that it can read or write.
- HDFS, too, has the concept of a block, but it is a much larger unit—64 MB by default.
- Like in a filesystem for a single disk, files in HDFS are broken into block-sized chunks, which are stored as independent units.
- Unlike a filesystem for a single disk, a file in HDFS that is smaller than a single block does not occupy a full block's worth of underlying storage.

Blocks

- HDFS blocks are large compared to disk blocks, and the reason is to minimize the cost of seeks.
- By making a block large enough, the time to transfer the data from the disk can be made to be significantly larger than the time to seek to the start of the block.
- Thus the time to transfer a large file made of multiple blocks operates at the disk transfer rate.

Blocks

- A quick calculation shows that if the seek time is around 10 ms, and the transfer rate is 100 MB/s, then to make the seek time 1% of the transfer time, there is a need to make he block size around 100 MB.
- The default is actually 64 MB, although many HDFS installations use 128 MB blocks

Advantages of Blocks

- A file can be larger than any single disk in the network.
  - There's nothing that requires the blocks from a file to be stored on the same disk, so they can take advantage of any of the disks in the cluster.
  - Making the unit of abstraction a block rather than a file simplifies the storage subsystem.
  - Simplicity is something to strive for all in all systems, but is especially important for a distributed system in which the failure modes are so varied.

Advantages of Blocks

- The storage subsystem deals with blocks
  - simplifying storage management (since blocks are a fixed size, it is easy to calculate how many can be stored on a given disk)
  - eliminating metadata concerns (blocks are just a chunk of data to be stored—file metadata such as permissions information does not need to be stored with the blocks, so another system can handle metadata separately).

Advantages of Blocks

- Blocks fit well with replication for providing fault tolerance and availability.
  - To insure against corrupted blocks and disk and machine failure, each block is replicated to a small number of physically separate machines (typically three).
  - If a block becomes unavailable, a copy can be read from another location in a way that is transparent to the client.
  - A block that is no longer available due to corruption or machine failure can be replicated from its alternative locations to other live machines to bring the replication factor back to the normal level.

Advantages of Blocks

- Blocks fit well with replication for providing fault tolerance and availability.
  - To insure against corrupted blocks and disk and machine failure, each block is replicated to a small number of physically separate machines (typically three).
  - If a block becomes unavailable, a copy can be read from another location in a way that is transparent to the client.
  - A block that is no longer available due to corruption or machine failure can be replicated from its alternative locations to other live machines to bring the replication factor back to the normal level.

Namenode and Datanodes

- An HDFS cluster has two types of node operating in a master-worker pattern
  - a namenode (the master)
  - a number of datanodes (workers).

Namenode and Datanodes

- 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
  - the edit log.

Namenode and Datanodes

- 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.

Namenode and Datanodes

- Datanodes are the workhorses 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.

Namenode and Datanodes

- Without the namenode, the filesystem cannot be used.
- If in the machine running the namenode were destroyed, 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.

Namenode and Datanodes

- Hadoop provides two mechanisms for making namenode resilient to failure.
  - 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.

Namenode and Datanodes

- Hadoop provides two mechanisms for making namenode resilient to failure.
  - Run a secondary namenode, which 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.

Namenode and Datanodes

- Hadoop provides two mechanisms for making namenode resilient to failure.
  - It 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.

Namenode and Datanodes

- Each namenode manages a namespace volume which is made up of
  - the metadata for the namespace
  - a block pool containing all the blocks for the files in the namespace.
- Namespace volumes are independent of each other.
  - Namenodes do not communicate with one another
  - Thee failure of one namenode does not affect the availability of the namespaces managed by other namenodes.

Namenode and Datanodes

- Block pool storage is not partitioned.
  - Datanodes register with each namenode in the cluster and store blocks from multiple block pools.

Namenode and Datanodes - Recovery

- The namenode is still a single point of failure (SPOF)
  - If it fails, all clients—including MapReduce jobs—would be unable to read, write, or list files, because the namenode is the sole repository of the metadata and the file-to-block mapping.

Namenode and Datanodes - Recovery

- To recover from a failed namenode in this situation, an administrator starts a new primary namenode with one of the filesystem metadata replicas, and configures Datanodes and clients to use this new namenode.

Namenode and Datanodes - Recovery

- The new namenode is not able to serve requests until it has
  - Loaded its namespace image into memory,
  - Replayed its edit log
  - Received enough block reports from the datanodes to leave safe mode.
- On large clusters with many files and blocks, the time it takes for a namenode to start from cold can be 30 minutes or more.

Namenode and Datanodes - Recovery

- There is a pair of namenodes in an activestandby configuration.
- In the event of the failure of the active namenode, the standby takes over its duties to continue servicing client requests without a significant interruption.

Namenode and Datanodes – Recovery Architecture

- The namenodes must use highly-available shared storage to share the edit log.
  - When a standby namenode comes up it reads up to the end of the shared edit log to synchronize its state with the active namenode, and then continues to read new entries as they are written by the active namenode.
- Datanodes must send block reports to both namenodes since the block mappings are stored in a namenode's memory, and not on disk.
- Clients must be configured to handle namenode failover, which uses a mechanism that is transparent to users.

Hadoop Commands

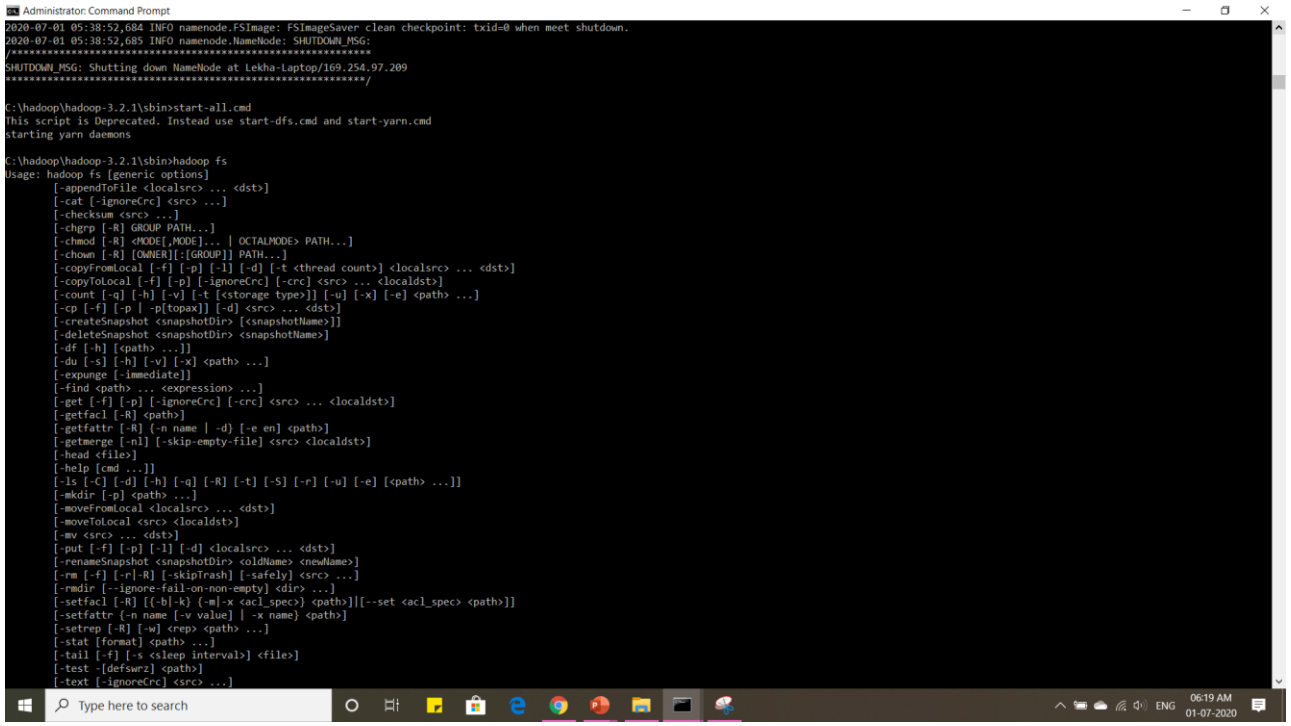
- hadoop.cmd and hdfs.cmd are the commands for interaction with file system.

BIG DATA ANALYTICS

Hadoop Commands

- In the terminal or command prompt type Hadoop fs.

- Allows to manipulate file system.



BIG DATA ANALYTICS

Hadoop Commands

- Allows to manipulate the Hadoop Distributed file system
-

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -help
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
```

Hadoop Commands

- Allows to manipulate the Hadoop Distributed file system

```
C:\hadoop\hadoop-3.2.1\sbin>hdfs dfs
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] [-l] [-d] [-t <thread count>] <localsrc> ... <dst>]
        [-copyToLocal [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-count [-q] [-h] [-v] [-t [<storage type>]] [-u] [-x] [-e] <path> ...]
```

Hadoop Commands

- Create a directory at the root level.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -mkdir /test
```

- To check the contents of the directory

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls /
Found 1 items
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:00 /test
```

Hadoop Commands

- Create a directory at the level.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -mkdir /test/subtest
```

- To check the contents of the subdirectory

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -R /
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:00 /test
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:00 /test/subtest
```

BIG DATA ANALYTICS

Hadoop Commands

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -R /
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:00 /test
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:00 /test/subtest
```

File Permissions

Number of file replicas: Set to 1 since that is the replication factor

Owner and group of files

File Size

Last Modified Time

File name and path

Hadoop Commands

- To copy from local machine to Hadoop,
  - use hadoop fs –copyFromLocal
- The first argument is source files.
- The second argument is the destination folder.
  - The destination folder should exist.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -copyFromLocal C:\hadoop\hadoop-3.2.1\
LICENSE.txt /test
2020-07-01 10:12:17,794 INFO sasl.SaslDataTransferClient: SASL encryption tr
ust check: localHostTrusted = false, remoteHostTrusted = false
```

Hadoop Commands

- To check if it is copied

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -R /
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:12 /test
-rw-r--r--   1 raoal supergroup     150569 2020-07-01 10:12 /test/LICENSE.tx
t
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:00 /test/subtest
```

Hadoop Commands

- To copy from local machine to Hadoop use –put
- The first argument is source files.
- The second argument is the destination folder.
  - The destination folder should exist.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -put C:\hadoop\hadoop-3.2.1\NOTICE.txt
 /test/subtest
2020-07-01 10:18:16,207 INFO sasl.SaslDataTransferClient: SASL encryption tr
ust check: localHostTrusted = false, remoteHostTrusted = false
```

Hadoop Commands

- To check if it is copied

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -R /
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:12 /test
-rw-r--r--   1 raoal supergroup     150569 2020-07-01 10:12 /test/LICENSE.tx
t
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:18 /test/subtest
-rw-r--r--   1 raoal supergroup      22125 2020-07-01 10:18 /test/subtest/NO
TICE.txt
```

Hadoop Commands

- Use cp command to copy between directories on HDFS.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -cp /test/* /test1
2020-07-01 10:22:09,072 INFO sasl.SaslDataTransferClient: SASL encryption tr
ust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-01 10:22:09,390 INFO sasl.SaslDataTransferClient: SASL encryption tr
ust check: localHostTrusted = false, remoteHostTrusted = false
2020-07-01 10:22:09,729 INFO sasl.SaslDataTransferClient: SASL encryption tr
ust check: localHostTrusted = false, remoteHostTrusted = false
```

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls /
Found 2 items
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:12 /test
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:20 /test1
```

Hadoop Commands

- To check if it is copied.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -R /
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:12 /test
-rw-r--r--   1 raoal supergroup     150569 2020-07-01 10:12 /test/LICENSE.tx
t
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:18 /test/subtest
-rw-r--r--   1 raoal supergroup      22125 2020-07-01 10:18 /test/subtest/NO
TICE.txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:22 /test1
-rw-r--r--   1 raoal supergroup     150569 2020-07-01 10:22 /test1/LICENSE.t
xt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:22 /test1/subtest
-rw-r--r--   1 raoal supergroup      22125 2020-07-01 10:22 /test1/subtest/N
OTICE.txt
```

Hadoop Commands

- Create a directory on your local machine.

```
C:\hadoop\hadoop-3.2.1\sbin>mkdir fromhdfs
```

- To copy from HDFS to local directory use copyToLocal.
- The first argument is source files on HDFS.
- The second argument is the destination folder on local machine.
  - The destination folder should exist.

BIG DATA ANALYTICS

Hadoop Commands

-
```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -copyToLocal /test/* fromhdfs
2020-07-01 10:25:17,980 INFO sasl.SaslDataTransferClient: SASL encryption tr
ust check: localHostTrusted = false, remoteHostTrusted = false
```

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>dir
 Volume in drive C is OS
 Volume Serial Number is C44E-6526

 Directory of C:\hadoop\hadoop-3.2.1\sbin\fromhdfs

01-07-2020  10.25 AM    <DIR>          .
01-07-2020  10.25 AM    <DIR>          ..
01-07-2020  10.25 AM           150,569 LICENSE.txt
01-07-2020  10.25 AM    <DIR>          subtest
               1 File(s)        150,569 bytes
               3 Dir(s)  266,465,554,432 bytes free
```

BIG DATA ANALYTICS

Hadoop Commands

-
```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs\subtest>dir
 Volume in drive C is OS
 Volume Serial Number is C44E-6526

 Directory of C:\hadoop\hadoop-3.2.1\sbin\fromhdfs\subtest

01-07-2020  10.25 AM    <DIR>          .
01-07-2020  10.25 AM    <DIR>          ..
01-07-2020  10.25 AM            22,125 NOTICE.txt
               1 File(s)         22,125 bytes
               2 Dir(s)  266,467,926,016 bytes free
```

Hadoop Commands

- Delete the files from fromhdfs using del and rmdir.
-

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>dir
 Volume in drive C is OS
 Volume Serial Number is C44E-6526

 Directory of C:\hadoop\hadoop-3.2.1\sbin\fromhdfs

01-07-2020  10.28 AM    <DIR>          .
01-07-2020  10.28 AM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  266,475,380,736 bytes free
```

BIG DATA ANALYTICS

Hadoop Commands

- Use the hadoop get command to copy from HDFS to local directory

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -get /test/* fromhdfs
2020-07-01 10:29:27,744 INFO sasl.SaslDataTransferClient: SASL encryption tr
ust check: localHostTrusted = false, remoteHostTrusted = false
```

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>dir
 Volume in drive C is OS
 Volume Serial Number is C44E-6526

 Directory of C:\hadoop\hadoop-3.2.1\sbin\fromhdfs

01-07-2020  10.29 AM    <DIR>          .
01-07-2020  10.29 AM    <DIR>          ..
01-07-2020  10.29 AM           150,569 LICENSE.txt
01-07-2020  10.29 AM    <DIR>          subtest
               1 File(s)        150,569 bytes
               3 Dir(s)  266,477,834,240 bytes free
```

Hadoop Commands

- To see the contents of a file in HDFS.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -cat /test/LICENSE.txt
```

- To delete a file on HDFS.

```
C:\hadoop\hadoop-3.2.1\sbin>
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -rm -r /test/LICENSE.txt
Deleted /test/LICENSE.txt
```

Hadoop Commands

- To delete all files that match the specified file pattern use –rm command.
- -rm [-f] [-r|-R] [-skipTrash] [-safely] <src> ...
  - -f If the file does not exist, do not display a diagnostic message or modify the exit status to reflect an error.

Hadoop Commands

- -rm [-f] [-r|-R] [-skipTrash] [-safely] <src> ...
  - -[rR] Recursively deletes directories.
  - -skipTrash option bypasses trash, if enabled, and immediately deletes <src>.

Hadoop Commands

- -rm [-f] [-r|-R] [-skipTrash] [-safely] <src> ...
  - safely option requires safety confirmation, if enabled, requires confirmation before deleting large directory with more than **<hadoop.shell.delete.limit.num.files>** files.
- Delay is expected when walking over large directory recursively to count the number of files to be deleted before the confirmation.

BIG DATA ANALYTICS

Hadoop Commands

- ```
  C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -rm /test/LICENSE.txt
  rm: `/test/LICENSE.txt': No such file or directory
  ```

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -rm -f /test/LICENSE.txt
```
```

Hadoop Commands

- To know the disk usage of all files in a particular folder use –du.
- It gives the result in bytes.
- The output is in the form
  - size disk space consumed name(full path)

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -du /test
22125  22125  /test/subtest

C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -du /test1
150569  150569  /test1/LICENSE.txt
22125   22125   /test1/subtest
```

Hadoop Commands

- -du [-s] [-h] [-v] [-x] <path>
  - -s Rather than showing the size of each individual file that matches the pattern, shows the total (summary) size.
  - -h Formats the sizes of files in a human-readable fashion rather than a number of bytes.
  - -v option displays a header line.
  - -x Excludes snapshots from being counted.

Hadoop Commands

- To get the usage in human readable format use –h option in –du.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -du -s /test
22125  22125  /test

C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -du -s /test1
172694  172694  /test1
```

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -du -h /test
21.6 K  21.6 K  /test/subtest

C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -du -h /test1
147.0 K  147.0 K  /test1/LICENSE.txt
21.6 K   21.6 K   /test1/subtest
```

Hadoop Commands

- To move a file from local directory to HDFS and delete the same from local directory

  use the command –moveFromLocal.

- It requires two parameters.
  - The first argument is source files.
  - The second argument is the destination folder.

BIG DATA ANALYTICS

Hadoop Commands

-

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>dir
 Volume in drive C is OS
 Volume Serial Number is C44E-6526

 Directory of C:\hadoop\hadoop-3.2.1\sbin\fromhdfs

01-07-2020  10.29 AM    <DIR>          .
01-07-2020  10.29 AM    <DIR>          ..
01-07-2020  10.29 AM           150,569 LICENSE.txt
01-07-2020  10.29 AM    <DIR>          subtest
               1 File(s)        150,569 bytes
               3 Dir(s)  266,467,266,560 bytes free
```

BIG DATA ANALYTICS

Hadoop Commands

-
```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -moveFromLocal fromhdfs/LICENSE.txt
/test/
2020-07-01 10:42:21,306 INFO sasl.SaslDataTransferClient: SASL encryption
trust check: localHostTrusted = false, remoteHostTrusted = false
```

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -R /
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:42 /test
-rw-r--r--   1 raoal supergroup     150569 2020-07-01 10:42 /test/LICENSE.
txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:18 /test/subtest
-rw-r--r--   1 raoal supergroup      22125 2020-07-01 10:18 /test/subtest/
NOTICE.txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:22 /test1
-rw-r--r--   1 raoal supergroup     150569 2020-07-01 10:22 /test1/LICENSE
.txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:22 /test1/subtest

-rw-r--r--   1 raoal supergroup      22125 2020-07-01 10:22 /test1/subtest
/NOTICE.txt
```

Hadoop Commands

- To move a file or directory within different HDFS directories

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -mv /test/subtest/NOTICE.txt /test1/
```

Hadoop Commands

- To merge n number of files in the HDFS distributed file system and put it into a single file in local file system use the command getmerge.
  - The first set of parameters are the source files.
  - The last parameter is the destination folder with file name.

Hadoop Commands

- -getmerge [-nl] [-skip-empty-file] <src> <localdst> :
  - Get all the files in the directories that match the source file pattern and merge and sort them to only one file on local fs. <src> is kept.
  - -nl Add a newline character at the end of each file.
  - -skip-empty-file Do not add new line character for empty file.

BIG DATA ANALYTICS

Hadoop Commands

-

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -getmerge /test/LICENSE.txt /test1/
NOTICE.txt fromhdfs/hadoop.txt
2020-07-01 10:51:09,907 INFO sasl.SaslDataTransferClient: SASL encryption
 trust check: localHostTrusted = false, remoteHostTrusted = false
```

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>dir
 Volume in drive C is OS
 Volume Serial Number is C44E-6526

 Directory of C:\hadoop\hadoop-3.2.1\sbin\fromhdfs

01-07-2020  10.51 AM    <DIR>          .
01-07-2020  10.51 AM    <DIR>          ..
01-07-2020  10.51 AM             1,360 .hadoop.txt.crc
01-07-2020  10.51 AM           172,694 hadoop.txt
01-07-2020  10.29 AM    <DIR>          subtest
               2 File(s)        174,054 bytes
               3 Dir(s)  266,496,704,512 bytes free
```

Hadoop Commands

- To print information about path use stat command.
- Format is a string which accepts file size in blocks (%b), filename (%n), block size (%o), replication (%r), and modification date (%y, %Y).

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -stat /test
2020-07-01 05:12:21
```

Hadoop Commands

- To show the last 1KB of a file in HDFS on stdout use -tail command.
  - -f Shows appended data as the file grows.

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -tail /test/LICENSE.txt
2020-07-01 10:55:08,854 INFO sasl.SaslDataTransferClient: SASL encryption
 trust check: localHostTrusted = false, remoteHostTrusted = false
        this trademark restriction does not form part of this License.

        Creative Commons may be contacted at https://creativecommons.org/
.
--------------------------------------------------------------------------
-------
```

Hadoop Commands

- Appends the contents of all the given local files to the given destination file.
- The destination file will be created if it does not exist.
- If <localSrc> is -, then the input is read from stdin.

Hadoop Commands

-
```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -appendToFile -
 /test/Sample.txt
How are you?
I am fine.
What about you?
^Z
2020-07-01 10:57:25,346 INFO sasl.SaslDataTransferClient: SASL
 encryption trust check: localHostTrusted = false, remoteHostT
rusted = false
```

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -cat /test/Samp
le.txt
2020-07-01 10:57:41,277 INFO sasl.SaslDataTransferClient: SASL
 encryption trust check: localHostTrusted = false, remoteHostT
rusted = false
How are you?
I am fine.
What about you?
```

Hadoop Commands

- To Count the number of directories, files and bytes under the paths that match the specified file pattern use –count.
- The output columns are:
  - DIR_COUNT FILE_COUNT CONTENT_SIZE PATHNAME

Hadoop Commands

- With –q option, the output columns are
  - QUOTA REM_QUOTA SPACE_QUOTA REM_SPACE_QUOTA DIR_COUNT FILE_COUNT CONTENT_SIZE PATHNAME
- The -h option shows file sizes in human readable format.

- If a comma-separated list of storage types is given after the -t option, it displays the quota and usage for the specified types.
- Otherwise, it displays the quota and usage for all the storage types that support quota.

- The list of possible storage types(case insensitive):
  - ram_disk, ssd, disk and archive.
- It can also pass the value '', 'all' or 'ALL' to specify all the storage types.

- The -u option shows the quota and the usage against the quota without the detailed content summary.

BIG DATA ANALYTICS

Hadoop Commands

-
```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -count /test
           2            2                150612 /test
```

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -count -q /test
       none             inf            none             inf            2            2           150612 /test
```

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -count -u /test
        none               inf              none               inf /test
```

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -count -v /test
   DIR_COUNT   FILE_COUNT       CONTENT_SIZE PATHNAME
           2            2             150612 /test
```

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -count -h /test
           2            2             147.1 K /test
```

BIG DATA ANALYTICS

Hadoop Commands

-
```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -count -u -v /test
        QUOTA       REM_QUOTA     SPACE_QUOTA REM_SPACE_QUOTA PATHNAME
         none             inf            none             inf /test
```

```
C:\hadoop\hadoop-3.2.1\sbin\fromhdfs>hadoop fs -count -q -v /test
      QUOTA       REM_QUOTA     SPACE_QUOTA REM_SPACE_QUOTA    DIR_COUNT   FILE_COUNT       CONTENT_SIZE PATHNAME
       none             inf            none             inf           2            2             150612 /test
```

Hadoop Commands

- -text
  - Takes a source file and outputs the file in text format on the terminal.
  - The allowed formats are zip and TextRecordInputStream

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -text /test/Sample.txt

2020-07-01 15:06:48,903 INFO sasl.SaslDataTransferClient: SA
SL encryption trust check: localHostTrusted = false, remoteH
ostTrusted = false
How are you?
I am fine.
What about you?
```

Hadoop Commands

- Checksum - a digit representing the sum of the correct digits in a piece of stored or transmitted digital data, against which later comparisons can be made to detect errors in the data.
- To dump checksum information for files that match the file pattern <src> to stdout use –checksum command.
  - This requires a round-trip to a datanode storing each block of the file, and thus is not efficient to run on a large number of files.

Hadoop Commands

- The checksum of a file depends on its content, block size and the checksum algorithm and parameters used for creating the file.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -checksum /test/Sample
.txt
/test/Sample.txt        MD5-of-0MD5-of-512CRC32C        0000
0200000000000000000000aba820455ac1d7e933783bb3da1a40e
```

Hadoop Commands

- The checksum of a file depends on its content, block size and the checksum algorithm and parameters used for creating the file.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -checksum /test/Sample
.txt
/test/Sample.txt        MD5-of-0MD5-of-512CRC32C        0000
0200000000000000000000aba820455ac1d7e933783bb3da1a40e
```

Hadoop Commands

- Display the folder contents using –ls command.
- The options can be
  - -C Display the paths of files and directories only.
  - -d Directories are listed as plain files.
  - -h Formats the sizes of files in a human-readable fashion rather than a number of bytes.
  - -q Print ? instead of non-printable characters.

Hadoop Commands

- -R Recursively list the contents of directories.
- -t Sort files by modification time (most recent first).
- -S Sort files by size.
- -r Reverse the order of the sort.
- -u Use time of last access instead of modification for display and sorting.

Hadoop Commands

- Display the folder contents.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -h -R /test
-rw-r--r--   1 raoal supergroup    147.0 K 2020-07-01 10:42 /test/LICENSE.txt
-rw-r--r--   1 raoal supergroup         43 2020-07-01 10:57 /test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:45 /test/subtest

C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -R /test
-rw-r--r--   1 raoal supergroup     150569 2020-07-01 10:42 /test/LICENSE.txt
-rw-r--r--   1 raoal supergroup         43 2020-07-01 10:57 /test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:45 /test/subtest
```

Hadoop Commands

- Display the folder contents.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -d /test
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:57 /test

C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls /test
Found 3 items
-rw-r--r--   1 raoal supergroup     150569 2020-07-01 10:42 /test/LICENSE.txt
-rw-r--r--   1 raoal supergroup         43 2020-07-01 10:57 /test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:45 /test/subtest
```

Hadoop Commands

- Display the folder contents.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -C /test
/test/LICENSE.txt
/test/Sample.txt
/test/subtest
```

Hadoop Commands

- Display the folder contents.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -S /test
Found 3 items
-rw-r--r--   1 raoal supergroup     150569 2020-07-01 10:42
/test/LICENSE.txt
-rw-r--r--   1 raoal supergroup         43 2020-07-01 10:57
/test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:45
/test/subtest
```

Hadoop Commands

- Display the folder contents.

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls -r /test
Found 3 items
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:45
/test/subtest
-rw-r--r--   1 raoal supergroup         43 2020-07-01 10:57
/test/Sample.txt
-rw-r--r--   1 raoal supergroup     150569 2020-07-01 10:42
/test/LICENSE.txt
```

Hadoop Commands

- To find all files that match the specified expression and apply selected actions to them use –find command.
- If no <path> is specified then defaults to the current working directory.
- If no expression is specified then defaults to -print.

Hadoop Commands

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -find /test/s*
/test/subtest

C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -find /test/S*
/test/Sample.txt
```

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -find  s*
```

Hadoop Commands

- hdfs dfs -getfacl [-R] <path>
  - Displays the Access Control Lists (ACLs) of files and directories. If a directory has a default ACL, then getfacl also displays the default ACL.

```
C:\hadoop\hadoop-3.2.1\sbin\fromhadoop>hdfs dfs -getfacl -R /testing1
# file: /testing1
# owner: raoal
# group: supergroup
user::rwx
group::r-x
other::r-x
```

Hadoop Commands

- To truncate all files that match the specified file pattern to the specified length use –truncate.
- The length has to be specified.
  - -w Requests that the command wait for block recovery to complete, if necessary.

BIG DATA ANALYTICS

Hadoop Commands

-
```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -truncate 100 /test
/LICENSE.txt
Truncating /test/LICENSE.txt to length: 100. Wait for blo
ck recovery to complete before further updating this file
.

C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls /test
Found 3 items
-rw-r--r--   1 raoal supergroup        100 2020-07-01 22:
27 /test/LICENSE.txt
-rw-r--r--   1 raoal supergroup         43 2020-07-01 10:
57 /test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:
45 /test/subtest
```

BIG DATA ANALYTICS

Hadoop Commands

```
C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -truncate -w 50 /test
/LICENSE.txt
Waiting for /test/LICENSE.txt ...
Truncated /test/LICENSE.txt to length: 50

C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls /test
Found 3 items
-rw-r--r--   1 raoal supergroup         50 2020-07-01 22:32
 /test/LICENSE.txt
-rw-r--r--   1 raoal supergroup         43 2020-07-01 10:57
 /test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:45
 /test/subtest
```

Hadoop Commands

- To set the replication level of a file use -setrep command.
- If <path> is a directory then the command recursively changes the replication factor of all files under the directory tree rooted at <path>.
  - -w It requests that the command waits for the replication to complete.
    - This can potentially take a very long time.
  - -R It is accepted for backwards compatibility.
    - It has no effect.

BIG DATA ANALYTICS

Hadoop Commands

- ```
  C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -setrep 2 /test/LICEN
  SE.txt
  Replication 2 set: /test/LICENSE.txt

  C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -ls /test
  Found 3 items
  -rw-r--r--   2 raoal supergroup         50 2020-07-01 22:32
   /test/LICENSE.txt
  -rw-r--r--   1 raoal supergroup         43 2020-07-01 10:57
   /test/Sample.txt
  drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:45
   /test/subtest
  ```
  ```

Hadoop Commands

- ```
  C:\hadoop\hadoop-3.2.1\sbin>hadoop fs -setrep -w 2 /test/Sa
  mple.txt
  Replication 2 set: /test/Sample.txt
  Waiting for /test/Sample.txt .....
  ```

```
Waiting for /test/Sample.txt ..............
```

```
Waiting for /test/Sample.txt ........................
```

```
Waiting for /test/Sample.txt ..............................
................
```
```

Hadoop Commands

- To display the usage for given command or all commands if none is specified use –

  usage.

```
C:\WINDOWS\system32>hadoop fs -usage ls
Usage: hadoop fs [generic options] -ls [-C] [-d] [-h] [-q]
[-R] [-t] [-S] [-r] [-u] [-e] [<path> ...]
```

Hadoop Commands

- To create a file of zero length at <path> with current time as the timestamp of that

  <path> use -touchz.

- An error is returned if the file exists with non-zero length

```
C:\WINDOWS\system32>hadoop fs -touchz /test/s.txt

C:\WINDOWS\system32>hadoop fs -ls /test
Found 4 items
-rw-r--r--   2 raoal supergroup         50 2020-07-01 22:32
 /test/LICENSE.txt
-rw-r--r--   2 raoal supergroup         43 2020-07-01 10:57
 /test/Sample.txt
-rw-r--r--   1 raoal supergroup          0 2020-07-01 22:50
 /test/s.txt
drwxr-xr-x   - raoal supergroup          0 2020-07-01 10:45
 /test/subtest
```

Hadoop Commands

- To remove the directory entry specified by each directory argument, provided it is empty use –rmdir command.
  - -rmdir [--ignore-fail-on-non-empty] <dir> ... :

```
C:\WINDOWS\system32>hadoop fs -rmdir /test/subtest

C:\WINDOWS\system32>hadoop fs -ls /test
Found 3 items
-rw-r--r--   2 raoal supergroup         50 2020-07-01 22:32
 /test/LICENSE.txt
-rw-r--r--   2 raoal supergroup         43 2020-07-01 10:57
 /test/Sample.txt
-rw-r--r--   1 raoal supergroup          0 2020-07-01 22:50
 /test/s.txt
```

Hadoop Commands

- To show the capacity, free and used space of the filesystem use –df command.
  - If the filesystem has multiple partitions, and no path to a particular partition is specified, then the status of the root partitions will be shown.
  - -h Formats the sizes of files in a human-readable fashion rather than a number of bytes.

BIG DATA ANALYTICS

Hadoop Commands

-
```
C:\WINDOWS\system32>hadoop fs -df /test
Filesystem                          Size    Used     Available
Use%
hdfs://localhost:9820   502703058944   242807   265699422208
   0%

C:\WINDOWS\system32>hadoop fs -df -h /test
Filesystem                      Size     Used  Available  Use%
hdfs://localhost:9820   468.2 G   237.1 K    247.5 G    0%
```

User Commands

- fsck
  - HDFS supports the fsck command to check for various inconsistencies.
  - It is designed for reporting problems with various files, for example, missing blocks for a file or under-replicated blocks.
-

BIG DATA ANALYTICS
User Commands

- hdfs fsck <path> [-list-corruptfileblocks | [-move | -delete | -openforwrite] [-files [-blocks [-locations | -racks | -replicaDetails | -upgradedomains]]]] [-includeSnapshots] [-showprogress] [-storagepolicies] [-maintenance] [-blockId <blk_Id>]
-

```
C:\hadoop\hadoop-3.2.1\sbin>hdfs fsck /test
Connecting to namenode via http://localhost:9870/fsck?ugi=raoal&path=%2Ftest
FSCK started by raoal (auth:SIMPLE) from /127.0.0.1 for path /test at Wed Jul 15 10:11:22
 IST 2020

/test/LICENSE.txt:  Under replicated BP-1628838345-169.254.97.209-1593562132131:blk_10737
41829_1012. Target Replicas is 2 but found 1 live replica(s), 0 decommissioned replica(s)
, 0 decommissioning replica(s).
```

- hdfs classpath
  - Prints the class path needed to get the Hadoop jar and the required libraries
-

```
C:\hadoop\hadoop-3.2.1\sbin>hdfs classpath
C:\hadoop\hadoop-3.2.1\etc\hadoop;C:\hadoop\hadoop-3.2.1\share\hadoop\common;C:\hadoop\ha
doop-3.2.1\share\hadoop\common\lib\*;C:\hadoop\hadoop-3.2.1\share\hadoop\common\*;C:\hado
op\hadoop-3.2.1\share\hadoop\hdfs;C:\hadoop\hadoop-3.2.1\share\hadoop\hdfs\lib\*;C:\hadoo
p\hadoop-3.2.1\share\hadoop\hdfs\*;C:\hadoop\hadoop-3.2.1\share\hadoop\yarn;C:\hadoop\had
oop-3.2.1\share\hadoop\yarn\lib\*;C:\hadoop\hadoop-3.2.1\share\hadoop\yarn\*;C:\hadoop\ha
doop-3.2.1\share\hadoop\mapreduce\lib\*;C:\hadoop\hadoop-3.2.1\share\hadoop\mapreduce\*
```

- hdfs dfsadmin –report
  - find out how much disk space us used, free, under-replicated, etc.usernames.

```
C:\hadoop\hadoop-3.2.1\sbin\fromhadoop>hdfs dfsadmin -report
Configured Capacity: 502703058944 (468.18 GB)
Present Capacity: 266300265187 (248.01 GB)
DFS Remaining: 266299916288 (248.01 GB)
DFS Used: 348899 (340.72 KB)
DFS Used%: 0.00%
Replicated Blocks:
        Under replicated blocks: 5
        Blocks with corrupt replicas: 0
        Missing blocks: 0
        Missing blocks (with replication factor 1): 0
        Low redundancy blocks with highest priority to recover: 5
        Pending deletion blocks: 0
```

User Commands

- hdfs dfsadmin -safemode get
  - find out if it is in the safemode or not
-
```
C:\hadoop\hadoop-3.2.1\sbin\fromhadoop>hdfs dfsadmin -safemode get
Safe mode is OFF
```

User Commands

- hdfs getconf –namenodes
  - namenodes in the cluster
- hdfs getconf –secondaryNameNodes
  - Secondary namenodes in the cluster
- hdfs getconf –backupNodes
  - Back up nodes in the cluster

BIG DATA ANALYTICS

User Commands

```
C:\hadoop\hadoop-3.2.1\sbin>hdfs getconf -namenodes
localhost

C:\hadoop\hadoop-3.2.1\sbin>hdfs getconf -SecondaryNameNodes
0.0.0.0

C:\hadoop\hadoop-3.2.1\sbin>hdfs getconf -backupNodes
0.0.0.0
```

```
C:\hadoop\hadoop-3.2.1\sbin\fromhadoop>hdfs getconf -confkey dfs.replication
1
```

Recap

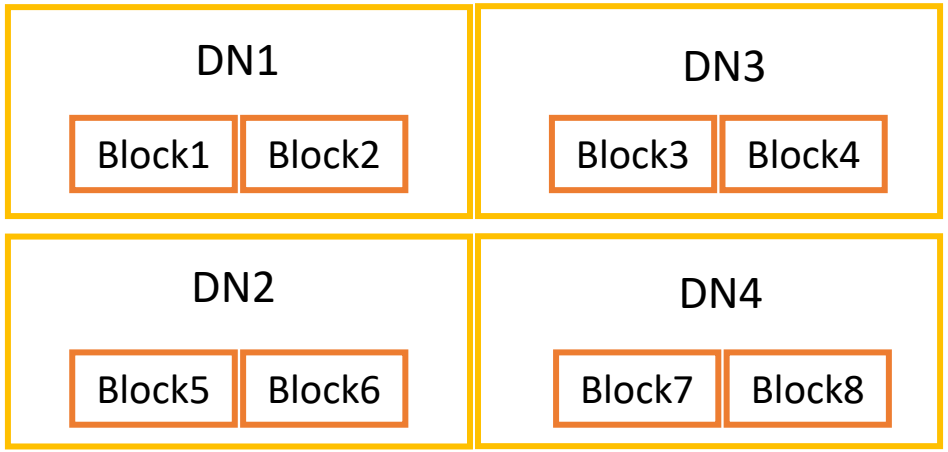
- mkdir
- copyFromLocal
- copyToLocal
- put
- get
- rm

Fault Tolerance

- Challenges in Distributed Mode
- Managing Failures in Namenode
- Managing Failures in Datanode

Fault Tolerance

- A file gets distributed across multiple cluster nodes in a local machine.
- Each of the cluster nodes are commodity hardware.



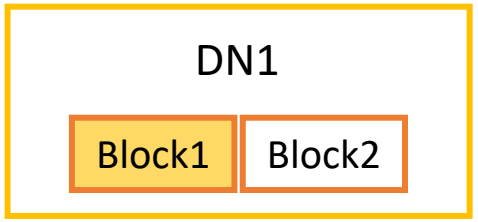
File1	Block1	DN1
File1	Block2	DN1
File1	Block3	DN3
File1	Block4	DN3
File1	Block5	DN2
File1	Block6	DN2
File1	Block7	DN4
File1	Block8	DN4

Challenges in Distributed Storage

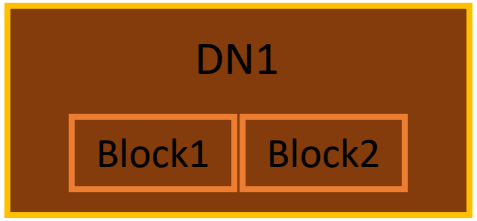
- Failure management in the data nodes
- Failure management for the name node

Challenges in Distributed Storage

- A block can get corrupted.

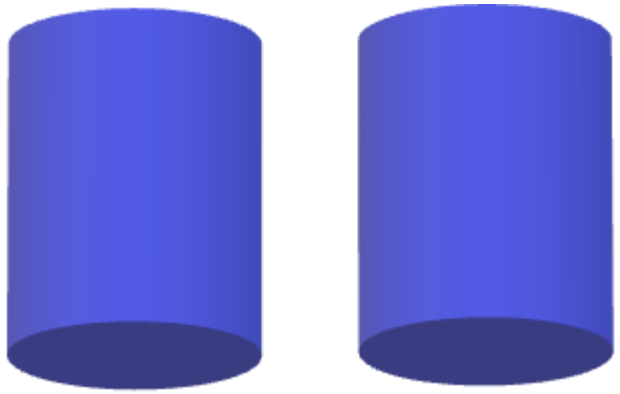


- A data node containing some blocks crashes.



Manage Failure

- Define a replication factor.

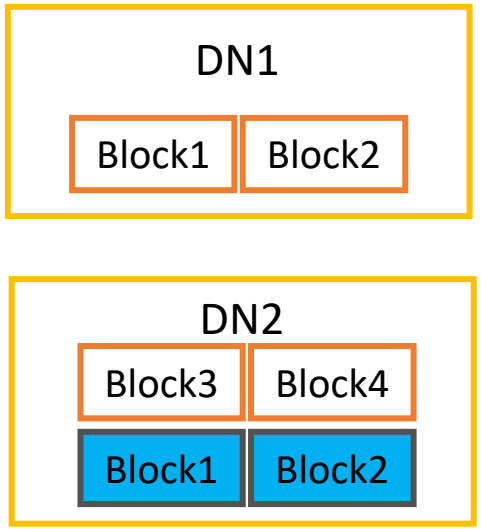


- It is a configuration property that is set on HDFS.

Manage Failure

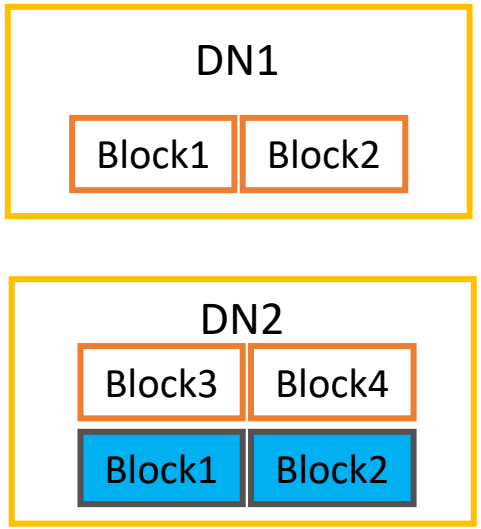
- 1. Replicate blocks based on the replication factor

- 2. Store replicas in different locations



Manage Failure

- The replica locations are also stored in the name node.



File1	Block1	DN1
File1	Block2	DN1
File1	Block3	DN2
File1	Block4	DN2
File1	Block5	DN3
File1	Block6	DN3
File1	Block7	DN4
File1	Block8	DN4
File1	Block1	DN2
File1	Block2	DN2

Manage Failure

- Hadoop has to explicitly determine what nodes in the cluster will hold the replica.
- This is done using the replication strategy.

Manage Failure

- Choosing replica strategy needs to balance two constraints.
  - Maximize redundancy
    - Greater the number of replicas, more the redundancy and more the fault tolerance.

Manage Failure

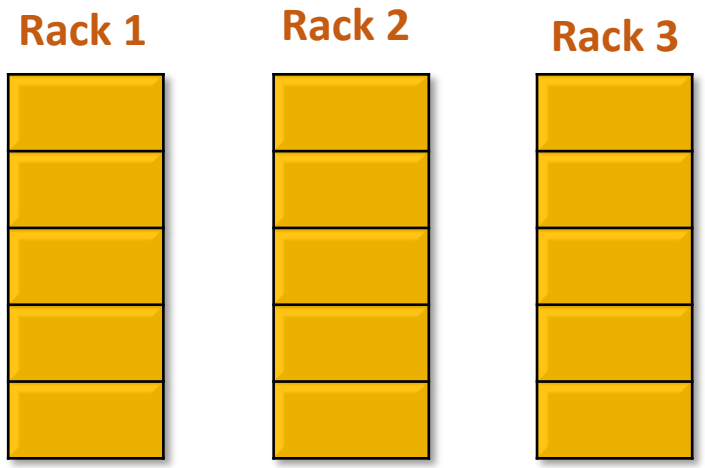
- Choosing replica strategy needs to balance two constraints.
  - Minimize write bandwidth
    - Updating a file should not cause writes to locations which are very far from each other.
    - This will lead to clogging of intra-cluster bandwidth.

Maximize Redundancy

- Any cluster within a data centre has typically machines that are stored in racks.
- Machines on a single rack are close to each other and have very high bandwidth connections between them.

Maximize Redundancy

- Any cluster within a data centre has typically machines that are stored in racks.
- Machines on a single rack are close to each other and have very high bandwidth connections between them.

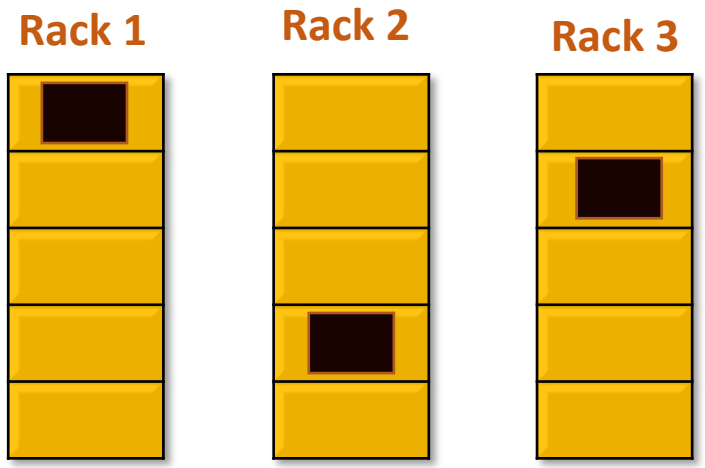


Maximize Redundancy

- A cluster is made up of machines in different racks.
- Nodes on different racks are further away from each other than nodes on the same rack.
  - They are also interconnected but the bandwidth available for write and read operations will be lower than the intra-rack bandwidth.

Maximize Redundancy

- Store replicas “far away” i.e. on different nodes.
  - Replicas should be stored far away from each other on different nodes.

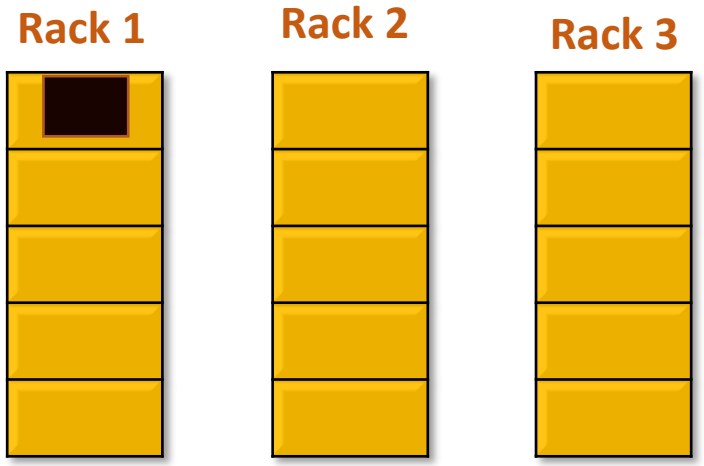


Maximize Redundancy

- This requires that replicas be stored close to each other.
- Write operation
  - There is one node that runs the replication pipeline.
  - This node chooses the location for the first replica.
    - This is chosen by random.
    - Write operations will first write to this replica.
    - The data is then forwarded to the next replica

BIG DATA ANALYTICS

Maximize Redundancy

- 

Maximize Redundancy

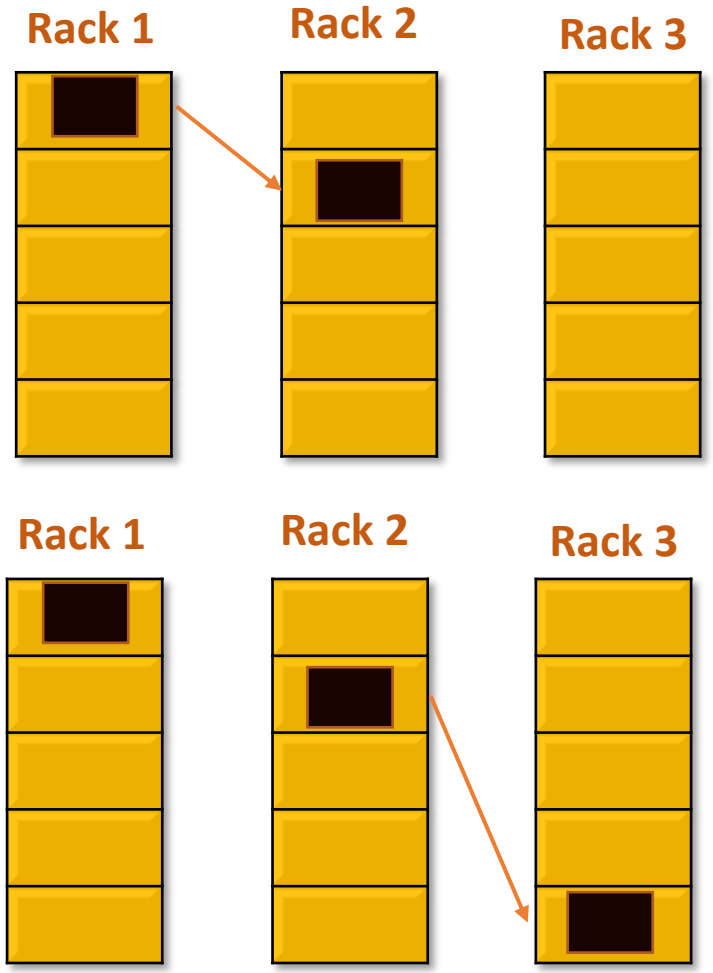
- This requires that replicas be stored close to each other.
- Write operation
  - There is one node that runs the replication pipeline.
  - This node chooses the location for the first replica.
    - This is chosen by random.
    - Write operations will first write to this replica.
    - The data is then forwarded to the next replica

Maximize Redundancy

- If the replica is on a node on a different rack, the write operation has to pass through the intra – rack connections which are not of very high bandwidth.
- They will be then forwarded to the next replica location.

BIG DATA ANALYTICS

Maximize Redundancy

- 

Minimize Write Redundancy

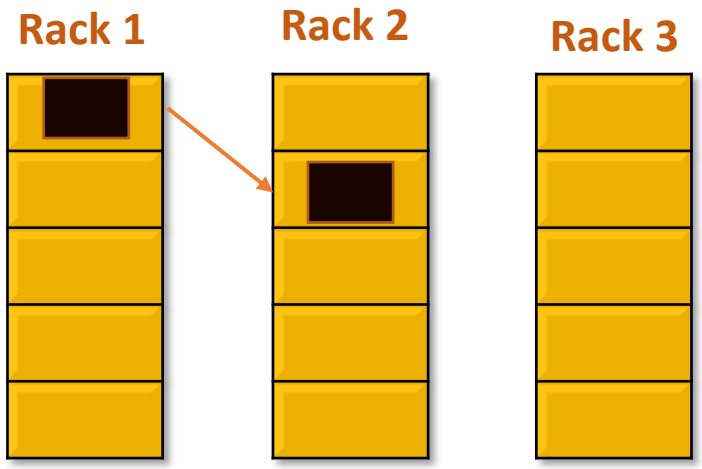
- If the replica is on a node on a different rack, the write operation has to pass through the intra – rack connections which are not of very high bandwidth.
- They will be then forwarded to the next replica location.

Minimize Write Redundancy

- In a fully distributed Hadoop system 3 is the default replication factor.
  - Forwarding requires a large amount of bandwidth.
  - Increases the cost of write operations

Default Hadoop Replication Strategy

- The first node on which the data is placed is chosen at random.
  - Any rack that is closest to the client will be chosen.
- The second location has to be on a different rank if possible.

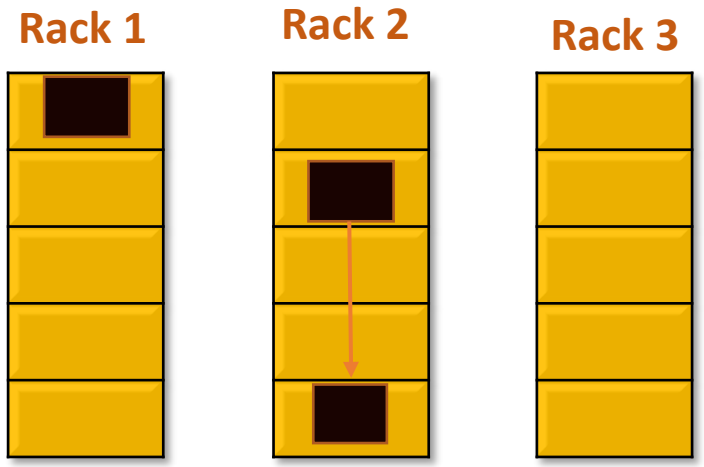


Default Hadoop Replication Strategy

- Third replica is on the same rack as the second but on different nodes.
  - Hadoop goes for a comprise on maximum redundancy and minimum write band width by using 2 racks and 3 nodes rather than 3 racks and 3 nodes.
    - Reduces interrack traffic and improves write performance

BIG DATA ANALYTICS

Default Hadoop Replication Strategy

- 

Default Hadoop Replication Strategy

- Read operations are sent to the rack physically closest to the client.
- Write operation will also happen at the same node.

Setting the Replication factor

- To set the replication factor in a pseudo-distributed file system it will be defined in the configuration file hdfs-site.xml.
- dfs.replication is the name of the attribute.
- In a fully-distributed system the value is default set to 3.
- In a pseudo-distributed system there is just one node so the replication factor cannot be >1

Namenode Failures

- The name node is the most important node of the cluster.
- The name node is the heart of HDFS
- Block locations are not persistent, i.e. they are stored in memory for quick look up.
  - This is called as block caching.

Namenode Failures

- If name node fails block locations will be completely lost.
  - There is no way to reconstruct where a particular file is.
  - There is no way to know which data nodes have the original and which data nodes have the replica.

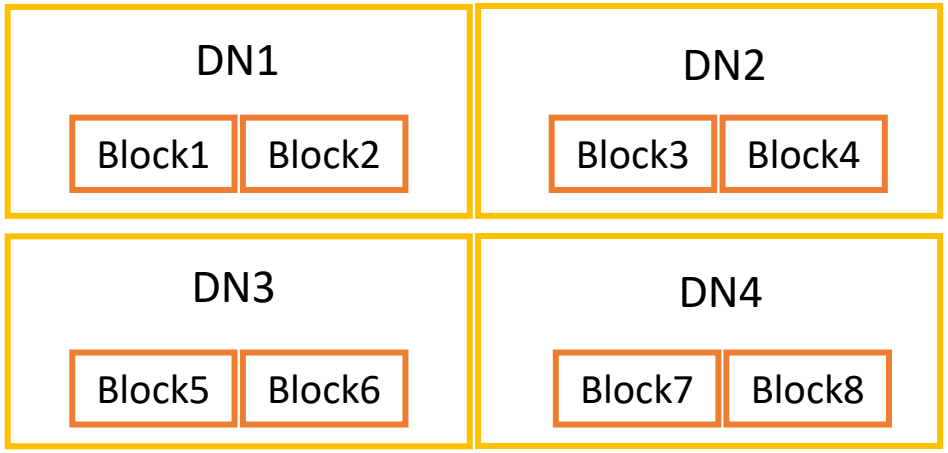
BIG DATA ANALYTICS

Namenode Failures

File1	Block1	DN1
File1	Block2	DN1
File1	Block3	DN2
File1	Block4	DN2
File1	Block5	DN3
File1	Block6	DN3
File1	Block7	DN4
File1	Block8	DN4
File1	Block1	DN2
File1	Block2	DN2

Namenode Failures

- If the name node fails File-Block Location mapping is lost!



- This data is worthless without the name node.

- Backing up the name node information is critical for high availability and reliability

Managing Name node failures

- They are managed in two ways
  - Using Metadata files
  - Using Secondary Name Node

BIG DATA ANALYTICS

Managing Name node failures

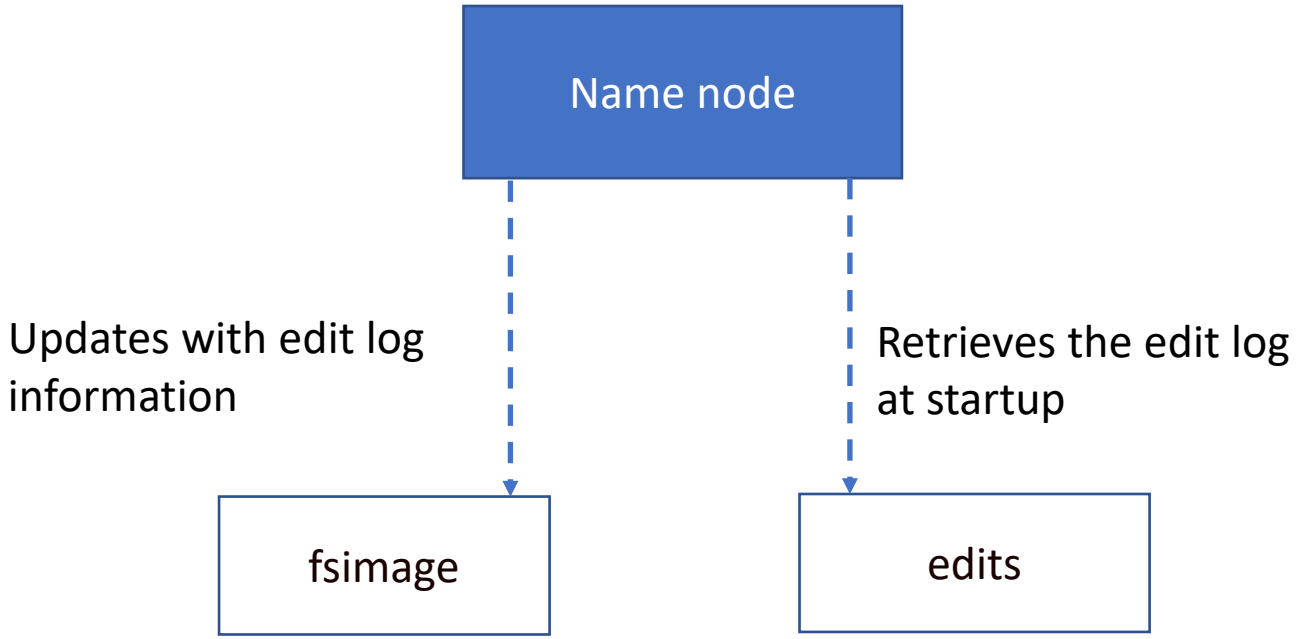
- There are two specific metadata files which allow the reconstruction of information in name node.
  - fsimage
    - File system image file
  - edits
- The above are the two files that store filesystem metadata and provide backup.

fsimage

- A snapshot of the complete file system at start up
  - When the Hadoop cluster is first started up.
- This is the current state of the file system when no operations have been performed on a fresh restart.
- Loaded into memory

edits

- The file contains a log of all in-memory edits made across HDFS to the file system since the Hadoop cluster was restarted.



Managing Name node failures

- Both the files fsimage and edits have a default backup location on the local file system and not on local HDFS.
- It can also be configured to be at a remote disk.
- Can be done in hdfs-site.xml.
  - Set the dfs.namenode.name.dir property to point to the backup location.

```
<property>
   <name>dfs.namenode.name.dir</name>
   <value>C:\BigData\hadoop-2.9.1\data\namenode</value>
</property>
```

Managing Name node failures

- To specify multiple backup locations, this property can take a comma separated list of paths
- Example
  - $PATH_1,$PATH_2,$PATH_3
    - Each path can be on a different host.
    - Metadata files will be backed up to each path.

Managing Name node failures

- Backing up these two metadata files is not straight forward.
- Merging these two files is very compute heavy and non-trivial operation.
- Hadoop clusters tend to be long running.
- They are not meant to be restarted often.
- Bringing a system back online could take a long time

Secondary Name Node

- It is an exact backup of the complete name node which runs on a completely different machine.

Managing Name node failures

- The name node and secondary name node have to sync up.
- Checkpointing is to see whether both of them are up-todate.

Managing Name node failures

- After secondary name node will get these files it will merge both of them to get a updated fsimage file.
  - It will apply every transaction, every file edits in the edits log in fsimage to get an updated fsimage.
  - This updated fsimage is then copied back to the name node.
  - The namenode will replace its fsimage.

Managing Name node failures

- Both the fsimage are in sync and up-to date.
- The edits file on both the machines are reset to empty.
  - This is done because fsimage are up-todate and the logs are not required to change the content.
- Checkpointing is done at a specified frequency.

Managing Name node failures

- In case of failure of Name node then secondary name node is promoted to be the name node.
- A new machine on the cluster is made the secondary new node.

Managing Name node failures

- To configure the checkpoint frequency, set properties in hdfs-site.xml
  - dfs.namenode.checkpoint.period
    - The number of seconds between each checkpoint.

Managing Name node failures

- dfs.namenode.checkpoint.check.period
  - The period when the secondary name node polls the name node for uncheckpointed transactions
  - Whether or not the original check point period has expired, the secondary name node will ask the name node if there is any uncheckpointed transactions it needs to copy over.

Managing Name node failures

- dfs.namenode.checkpoint.txns
  - The number of transactions (edits to the file system) before checkpointing.
  - It is not time based but transaction based.

THANK YOU

Lekha A
Computer Applications