Introduction to *BigData*

- BigData is data that has high volume, variety, velocity, veracity, and value.
- Used in - **banking**, healthcare, energy, technology, consumer, manufacturing
    - Banking:
        - Transaction data is stored. Based on this data, bank decides factors like loans, pre-approved loan amounts.
        - Credit card limit may also be set using this data.
        - Credit score
        - Investments, medical insurance
    - Healthcare:
        - Food recommendation
        - Fitness routines
        - ECG, EEG, and other scans like X-Ray, MRI, etc
        - Personalized diagnosis
        - Early disease prediction
    - Energy:
        - EV + Hybrid, forecasting requirement and usage to set up new charging points
        - Energy consumption requirement to scale up production
    - Technology:
        - advancements in tecnnology
    - Consumer:
        - Forecasting product requirements and demand as per consumer purchasing it.
        - Logistics management [eg: grocery at your door]
    - Manufacturing:

    - Where to set up manufacturing plants

Traditional Decision Making

Traditional decison making has various problems -

- takes time to arrive at a decison, losing competitive advantage
- human intervention required at various stages, and human intervention can lead to regular failures
- lacks systematic linkage among strategy, planning, execution and reporting
- due to limitation of storage, provides limited scope of data analysis, ie, provides only bird's eye view
- obstructs company's ability to make fully formed decision

The Solution: Big Data Analytics

- Decision making based on what you know, which is based on data analytics
- Provides comprehensive view of overall picture, which is a result of analyzing data from various sources [sources like SQL and NoSQL databases, URLs, APIs]
- Provides streamlined decision amking from top to bottom
- Big data analytics helps analyze unstructured data
- Helps make faster decisions, giving a competitive advantage and reducing the time takes

Case Study: Google's Self Driving Car

Technical Data [Sensor Data from vehicles] - Camera sensor, LiDAR, Ultrasonic sensor, IR sensor [proximity], GPS, accelerometer|gyroscope, temperature+pressure+humidity sensor

Community Data - Data collected from other vehicles to prepare route maps by analyzing traffic data and others

Personal Data - For personalization, like frequent routes, settings, etc

Big Data

It refers to extremly large data sets thet may be analyzed computationally to reveal patterns [clusters], trends, and associations [data mining, for recommendation systems], especially relating to human behavior and interactions.

Forecasting [or Time Series Forecasting] is usually done using LSTM and ARIMA models

4 V's of Big Data. as per IBM

- Volume - Entire amount of generated data
- Variety - Various types of data from various different sources
- Veracity - Truthfulness of data
- Velocity - Rate of generation being generated
- [There is a 5th V, Value - Data must provide information that add value to organization/company]

Different Types of Data

- Structured - Data in tablualar format, having rows and columns. Collected from databases, excel sheets, spreadsheets, etc.
- Semi-Structured - textual data files with an apparent pattern. Collected from XML, JSON, CSV, TSV files

  ```
  <!-- XML format example -->
  <row_id>
  <column_name> value </column_name>
  ```

- Quasi-Structured - Textual data, with erratic formats that can be formatted with effort and software tools. Example is clickstream data.
- Unstructured - No structure in data, stored as different type of file usually. Example is PDFs, images, videos, text documents

Case study in BigData: Netflix

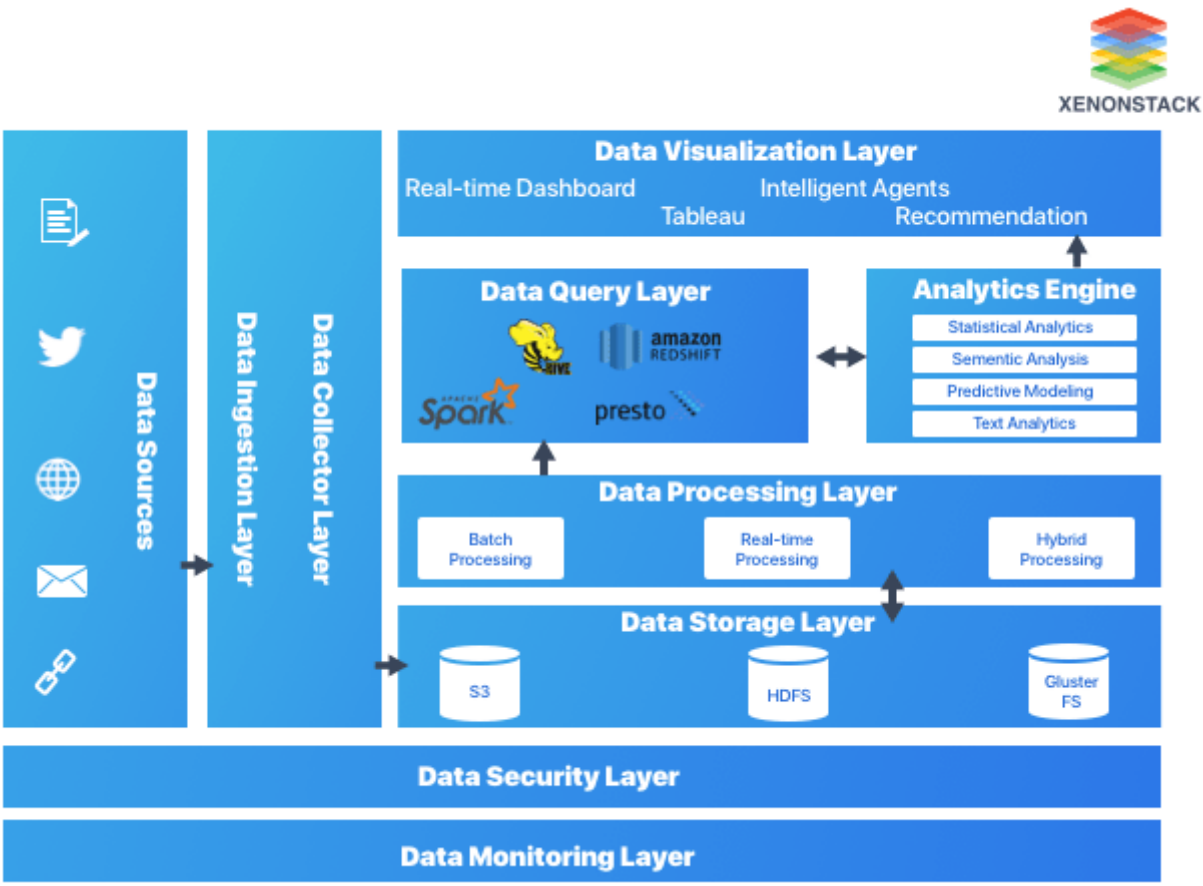
1. When do users watch a show?
2. Where do they watch?
3. On which device do they watch?

4. How often do they pause the program?
5. How often do they re-watch?
6. Do they skip credits?
7. What are the keywords searched?

Big Data Analytics Pipeline

Big data is a problem, because how should be handle this problem of humungous size of data?

Data is collected from various sources - Emails, URLs, weblogs(clickstream data), text, etc



Tools in data ingestion layer

- SQOOP tool helps import and export data from structured source to a destination [dest. is Hadoop Distributed File System]. It is bidirectional. DATABASE<===>SQOOP<===>HDFS
- Flume is unidirectional, used for migrating data from URL (live streaming source) and collect it to store in HDFS. Collect data at regular interval from source and send it to HDFS.
- Kafka is used for collecting data in streaming format.

Batch Processing - Result obtained by performing some process on data collected over a large period of time. Data in sql server is imported to HDFS using SQOOP. On this huge volume of data, analysis is performed to get some result. Analysis is on stored data.

Stream Processing/Realtime Processing - Data collected in small small intervals and processed collectively. Analysis is on live data, not stored data.

Big Data Case Study

Traditional vs Distributed System -

Distributed machines share storage and resources. So, a machine taking 1hr to do some task can be done in few seconds by using distributed machines with same specs.

So, as data grows we can add more devices to the distributed network - instead of buying a completely new machine setup.

Traditional machines support only vertical scaling [adding more storage, new cpu, more ram, etc which are already present in the machine]. But in distributed, systems, we can just add a new machine to the distributed network [horizontal scaling - add a new machine, vertical scaling - update existing machine].

Distributed machines are commodity hardware

Features of Big Data

Scalability in Big Data

Scalable platform accomodates rapid changes in growth of data, either traffic or volume. We add hardware and/or software in order to increase output and storage of data. Having a scalable platform means that the company will be ready for potential growth in data needs.

Fault Tolerance in Big Data

It refers to working strength of a system in unfavourable conditions and how that system can handle such a condition. HDFS also maintains replication factor by creating a replica of data on other machines

Data Inconsistency

Conflicting phenomenon may occur at various granularities. It occurs from knowledge content, data, meta-information, and can adversely affect quality of outcomes in BigData analysis process.

Distributed System

It is a model in which componentslocated on networked computers communicate and coordinate their actions by passing messages.

Challenges of Distributed Systems

- System failures
- Limited bandwidth
- High programming complexity

  The solution to these probles is : ***Hadoop***

Solution to Big Data Storage - Hadoop

It is a framework that allows distributed processing of large datasets across clusters of commodity computers using simple programming models.

Created by Doug Cutting.

Storage Management - Hadoop Distributed File System,

Programming Methodology - MapReduce,

Resource Management - YARN [Yet-Another-Resource-Negotiator]

Characteristics of Hadoop

- Scalable - Can follow both horizontal and vertical scaling
- Highly Available [Reliable] - Stores copies of data on different machines, thus resistant to mechanical failure
- Economical - CAn use ordinary computers for data processing
- Flexible - Can store huge data and decide to use it later

  In traditional system, data is sent to the program. In hadoop, program is sent to the data.

  Hadoop follows disc-only processing

Traditional Database Systems vs Hadoop

RDBMS	------	HADOOP
Structured	Data Types	Multi and Unstructured
Limited, no Data Processing	Processing	Processing coupled with Data
Standards and Structured	Governance	Loosely Structured
Required on Write	Schema	Required on Read
Reads are fast [it is read many/write many]	Speed	Writes are fast [it is read many/write once]
Software License	Cost	Support Only
Known Entity	Resources	Growing, Complexities, Wide
OLTP, Complex ACID Transactions, Operational Data Store	Best Fit USe	Data Discovery, Processing Unstructured Data, Massive Storage/Processing --> Batch Processing

Hadoop

It is an open-source software framework developed on Distributed Files System for storing data and runnig applications on clusters of commodity hardware. HDFS is highly fault tolerant.

Hadoop is programmed using JAVA.

Modes of Hadoop conifigurations -

- Standalone Mode - All Hadoopp services runs in a single JVM on a single machine
- Pseudo-Distributed Mode - Each hadoop runs on its own JVM, but on a single machine
- Fully Distributed Mode - Hadoop services runs on individual JVM, but reside in separate commodity machine in single cluster

HDFS Architecture

Hadoop has 3 components -

- hadoop distributed file system (HDFS) -> for distributed storage
- Yet-Another-Resource-Negotiator(YARN)-> for efficient reusource managementwhile using processing of data

- MapReduce (MapR) -> Programming methodology

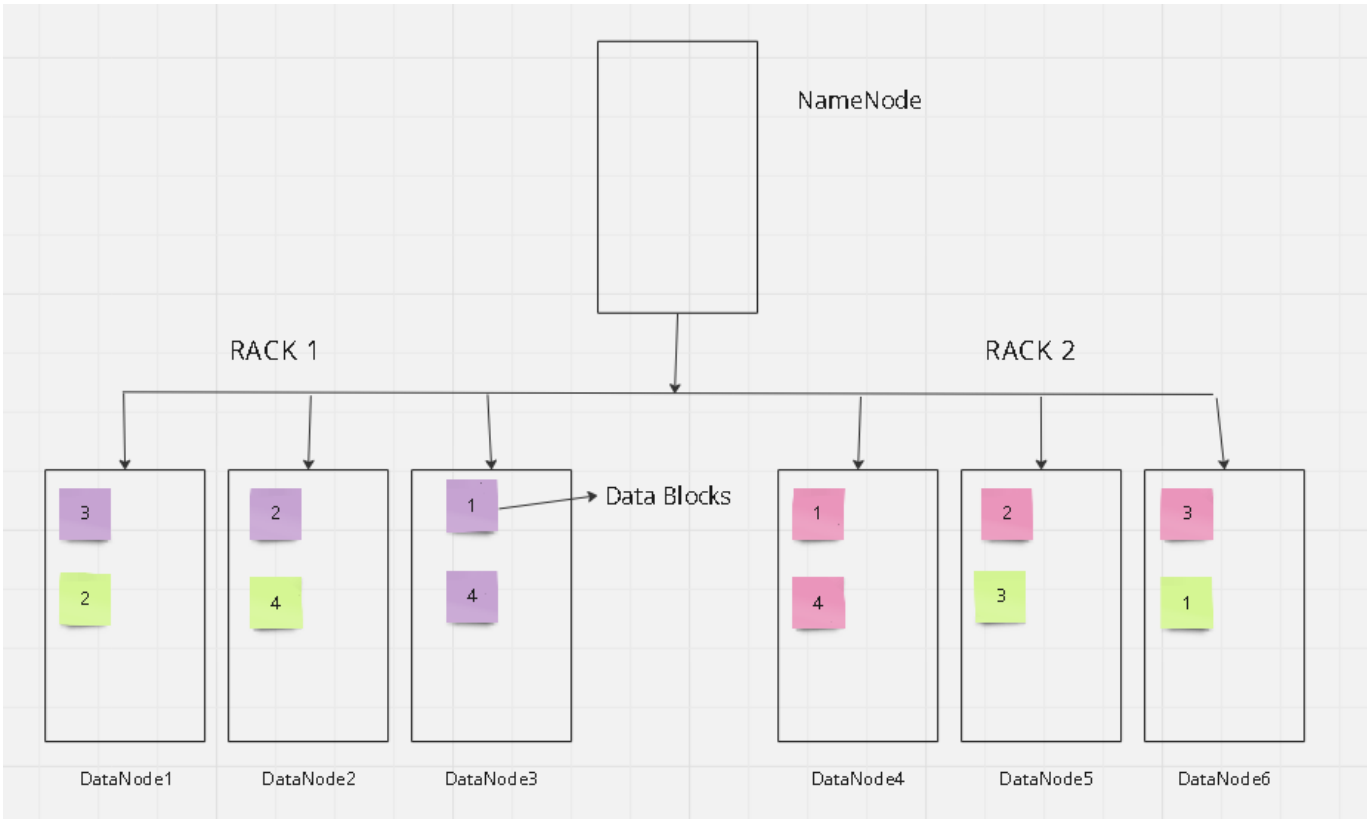
Hadoop Distributed File System is one of the key components of Hadoop Ecosystem, designed to store and manage large volumes of data across a distributed network machine. Its architecture is designed to handle high throughout (output) access to data and providefault-tolerance and high availability.

HDFS Architecture Components [3 services of HDFS]

- NameNode - It acts as master server that manages metadata of file system. Its role is to keep the file system info [path to data blocks]. There is only one NameNode, called as active NameNode. Also keeps track of replicas and file permissions. As user, you will only interact with NameNode
- DataNode - Acts as worker node and keeps actual block of data.
    - HDFS has default block size of 128MB. File will be separated into different blocks , so for larger files, they will be separated into different blocks based on block size.
- Secondary NameNode - Backup for NameNode.

Rules:

- NameNode tries to access nearest DataNode while performing Read and Write operations
- NamedNode distributes data block in such a way all resources are efficiently utilized
- Same data block will never be replicated in same DataNode



-- If any data node goes down, data blocks in that node will be replicated in all the other available data nodes.

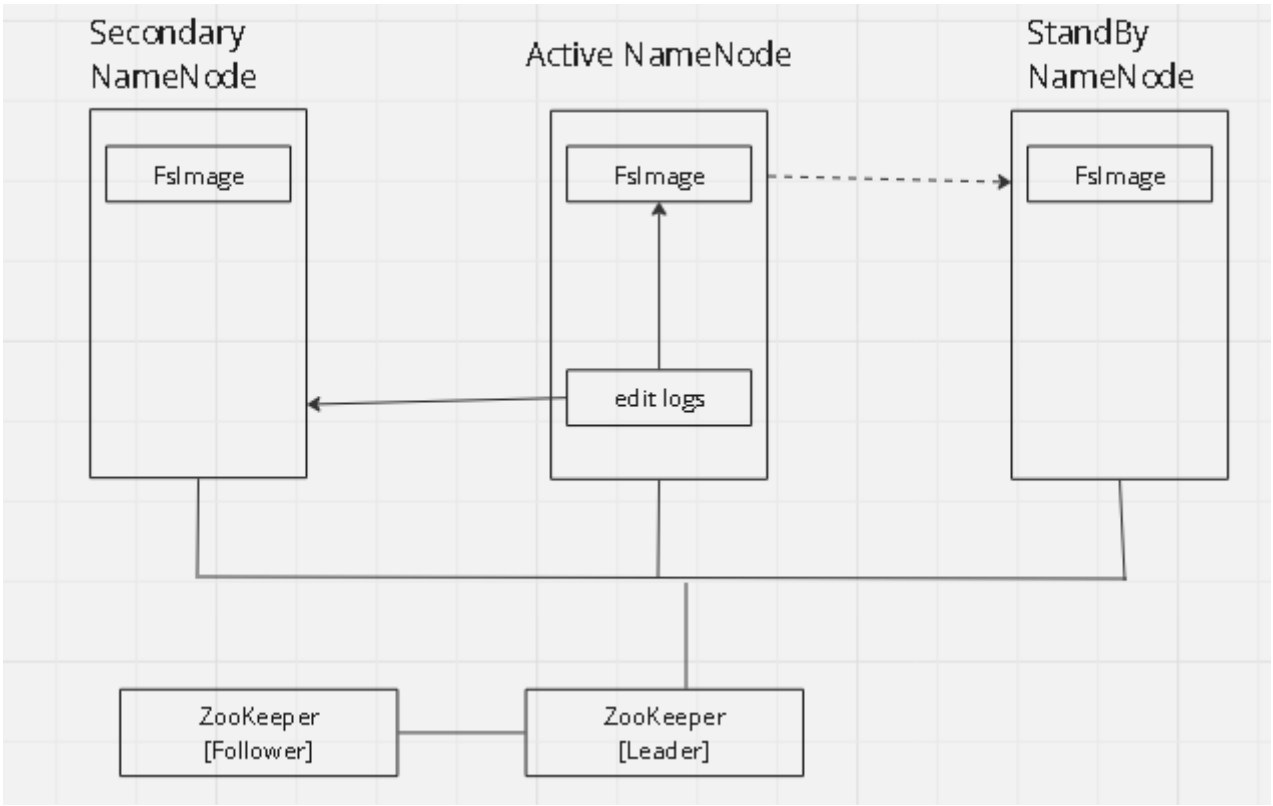
[In image, Violet 1-4 are 4 parts of same data, Pink is its replica, and Yellow is also its replica. Three colors indicate the 3 copies of same data. 1Pink and 1Yellow and 1Violet will never be in same DataNode]

-- if a downed DataNode comes back online, the newly created replicas are automatically deleted.

This is a Master-Slave architecture. NameNode is master, DataNodes are slave nodes

Concept of Multiple NameNodes

We have active NameNode and secondary NameNode.



Above is HDFS High Availability Architecture

FsImage is snapshots of activities, edit logs show just the differences. It contains the complete directory structure (namespace) of the HDFS with details about the location of the data on the Data Blocks and which blocks are stored on which node.

At regular intervals, FsImage will be copied to **Secondary NameNode**, same is copied to **Standby NameNode** also.

Incase active one goes down, secondary or standby becomes new active NameNode.

[2 alternatives are not compulsory, but it is good.]

[StandBy becomes Active if Active goes down. Only if StandBy is not present, will Secondary be taken as new Active node. But in that case,]

Any tasks performed by user is represented in terms of edit logs.

[Edit logs are like temporary, FsImage is permanent]

Notes -|

Primary function of secondary NameNode is to perform checkpointing. Checkpointing involves periodically merging NameNode's edit logs [changes in file system] with current file system [FsImage]. This happens at regular interval in Secondary NameNode.

NameNode keeps edit logs of all modifications made to the file system. Over time, this log can grow large, hence Secondary NameNode helps to convertthese edit logs into file system [FsImage] by creating checkpointing. Over time, if edit logs can grow large, NameNode will take more time to convert it into FsImage by restarting it. During this phase, cluster will be down.

Secondary NameNode helpsin reducing recovery time of the cluster. Due to checkpointing, downtime of cluster minimizes.

-- FsImage is FileSystem NameSpace, which indicates path to filesystem in cluster's DataNodes. It will also keep track of number of replicas, all replica paths, file system, etc.

In older versions of Hadoop Distributed File System [HDFS], cluster used to be down frequently, as logs and FsImage used to take lot of space in Memory [RAM]. Due to shortage of RAM, cluster used to go down. This got resolved in Hadoop 2 after introducing Secondary NameNode concept.

ZooKeeper has follower which will take backup, and leader which will be directly connected to cluster's NameNode. ZooKeeper acts as coordinator that will monitor health of Active NameNode, and it makes desicions of automatic fail-over where it will replace existing NameNode with StandBy NameNode. [StandBy becomes Active only when it receives all logs from FsImage].

Each DataNode sends it status to NameNode at regular intervals, called "Heartbeat". It can be used to know if a DataNode is down or not.

[User cannot directly interact with the DataNode, only with NameNode]

27-08-2024

Hadoop Hands-On

password: hadoop@123

useful commands

- sudo -> superuser do, avoid issues of permission
- mkdir -> kmake directory
- mv src dest -> move from source to destination
- cd path -> change directory to change into [new path starts with / if its not in continuation from current path]
- ls -> list files and directories

- ls -l -> lists file with many other informations
- sudo tar xvzf 'file' -> to extract .tar.gz file
- history -> all commands that have been run
- cd ~ -> go back to home directory
- sudo apt-get install package -> install some package
- sudo apt-get remove package -> uninstall some package
- sudo touch newfile2.txt -> create a file in specified path
- reboot -> command to reboot the machine
- clear -> clear terminal screen

hadoop installation prerequisites:

- java environment
- all communication between nodes is through secure shell, i.e. SSH protocol
- IPv6 needs to disabled, as secondary namenode might later cause issue with it.

Permissions in linux:

d at front means directory, else it is just file

777 -> -RWX-RWX-RWX [user, group, others]

(7 = 4+2+1)

localhost:50070 --> namenode ui

localhost:8088 --> hadoop cluster

https://medium.com/@amitmishra_393/hadoop-3-0-installation-on-ubuntu-18-04-step-by-step-pseudo-distributed-mode-2808f6b8e71f

commands:

- hadoop fs --> looks for services of Hadoop
- hadoop fs -mkdir /newdir -> make a new direrctory with name newdir
- hadoop fs -mkdir -p /newdir1/newdir2/newdir3 -> makes directory within directory

- hdfs dfs -copyFromLocal newfile.txt /newdir
- hdfs dfs -put source destination --> same function as above command
- hdfs dfs -cat /newdir/newfile.txt --> show content of file in path mentioned
- hdfs dfs touchz /newdir1/tempfile.txt --> create a new empty file in specified path
- hdfs dfs -appendToFile newfile.txt /newdir1/tempfile.txt -->copeid content from newfile.txt to tempfile.txt; there can be multiple source files
- hdfs dfs -cat /newdir1/tempfile.txt > newfile2.txt---> copy content from HDFS to local
- hdfs dfs -put newfile2.txt /hdfs----> copy from local and put it in HDFS directory
- hdfs dfs -cp /hdfs/newfile2.txt /newdir --> copy to newdir folder
- hdfs dfs -copyToLocal /newdir1/tempfile1 --> copy file from Hadoop to local destination not mentioned, so it is home, ie /home/hadoop
- get is alternative to copyToLocal
- hdfs dfs -ls / --> show all files and folders
- hdfs dfs -ls -R / --> show all files and folders, recursively
- hdfs dfs -rm /hdfs/newfile2.txt --> delete file in specified path
- hdfs dfs -rm -r /newdir --> deletes the directory and everything in it
- hdfs dfs -rmdir /hdfs --> deletes the directory if it is empty
- hdfs dfs -expunge --> empty all trash
- hdfs dfs -ls /hdfs/* --> see details of all files in specified folder
- hdfs dfs -chmod 766 /hdfs/newfile.txt --> give specific permission to some file
- hdfs dfs -chown hadoop:hadoop /hdfs --> change ownership of file and folder; owner is hadoop, group is also hadoop
- hdfs dfs -chown -R hadoop:hadoop /hdfs --> change ownership of file and folder recursively
- hdfs dfs -stat /hdfs/newfile.txt --> shows date when file was created
- hdfs dfs -stat %r /hdfs/newfile.txt --> shows status of replicas

- hdfs dfs -stat %b /hdfs/newfile.txt --> shows file size, in bytes
- hdfs dfs -setrep 3 /hdfs/newfile.txt --> set replication factor to 3
- hdfs dfs -du / --> size occupied by each directory, in bytes
- hdfs dfs -du -h / --> size occupied by each directory, in bhuman readable format ie bigger sizes will be shown in kb and mb
- hdfs dfs -df / --> size occupied by cluster [add -h for human readable format]
- hdfs dfs -count / --> number of files, directories, and size
- hdfs dfs -count /hdfs --> number of files, directories, and size in specified directory