**Data: -**

Data is a useful information which have life-cycle phase of generation, capturing, storage, processing, down streaming & visualization.

**Big Data: -**

Big data is an information which have a challenge in capturing, storage & processing.

Big data is the term for a collection of datasets so large and complex that it become difficult to process using on-hand database management tools or traditional data processing applications.

Big data size can be Terabyte, petabyte, zettabyte, yottabyte etc.

**Unit of data: -**

|  |  |
| --- | --- |
| Data Measurement | Size |
| Bit | Single Binary Digit (1 or 0) |
| Byte | 8 bits |
| Kilobyte (KB) | 1,024 Bytes |
| Megabyte (MB) | 1,024 Kilobytes |
| Gigabyte (GB) | 1,024 Megabytes |
| Terabyte (TB) | 1,024 Gigabytes |
| Petabyte (PB) | 1,024 Terabytes |
| Exabyte (EB) | 1,024 Petabytes |
| Zettabyte (ZB) | 1,024 Exabytes |
| Yottabyte (YB) | 1,024 Zettabytes |
| Brontobytes (BB) | 1,024 Yottabytes |
| Geopbyte (GB) | 1,024 Brontobytes |

**What comes Under big data: -**

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

* **Black Box Data** − It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.
* **Social Media Data** − social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.
* **Stock Exchange Data** − The stock exchange data holds information about the ‘buy’ and ‘sell’ decisions made on a share of different companies made by the customers.
* **Power Grid Data** − The power grid data holds information consumed by a particular node with respect to a base station.
* **Transport Data** − Transport data includes model, capacity, distance and availability of a vehicle.
* **Search Engine Data** − Search engines retrieve lots of data from different databases.

**Characteristics of Big Data: -**

There are 16 characteristics exit for data

1. value
2. volume
3. velocity
4. variance
5. visualization
6. variety
7. veracity
8. validity
9. volatility
10. variability
11. viscosity
12. valence
13. vulnerability… etc.

Most of the companies only work on 3V’S Volume, Velocity and Variety.

But IBM introduce 5V’S according to their data requirement Volume, velocity, variety, Value and Variance.

These characteristics are different for different type of data. But widely used only 3 v’s According to "Doug Cutting" (father of Hadoop) if 3 v's are present (volume, velocity, variety) then data can be considering as big data.

1. **Volume: -**

It refers to the size or amount of data.

**Example: -** for social media volume refer to the amount of data generated by websites, portals and online applications.

like for Facebook data is generated when we update status, posts pics, video, commenting. Liking the status etc.

1. **Velocity: -**

It refers to the Speed with which data are being generated and reported to the landing zone.

Landing zone is edge node where data is store for user to fetch.

Big data help the company to accept the large incoming flow of data and at the same time process it fast so that it does not create bottleneck (restriction).

1. **Variety: -**

It refers to the type of data. Data can be stored in multiple formats. It refers

All Structured and Un-Structured data that is generated by either by humans or machines.

**Example: -** text file, ECG recording, audio, video, script, emails, tabular data etc.

# The data that Contains velocity is called Scalable data and scalability is the very first pillar of Hadoop.

**Types of data in Big Data: -**

The data in Big Data will be of 3 types.

1. **Structured data: -**

Any data can be stored, accessed and processed in the form of fixed format is termed as a structured data.

**Example: -** RDBMS (tabular data) etc.

1. **Unstructured data: -**

Any data with unknown form/Structure is classified as un-structured data.

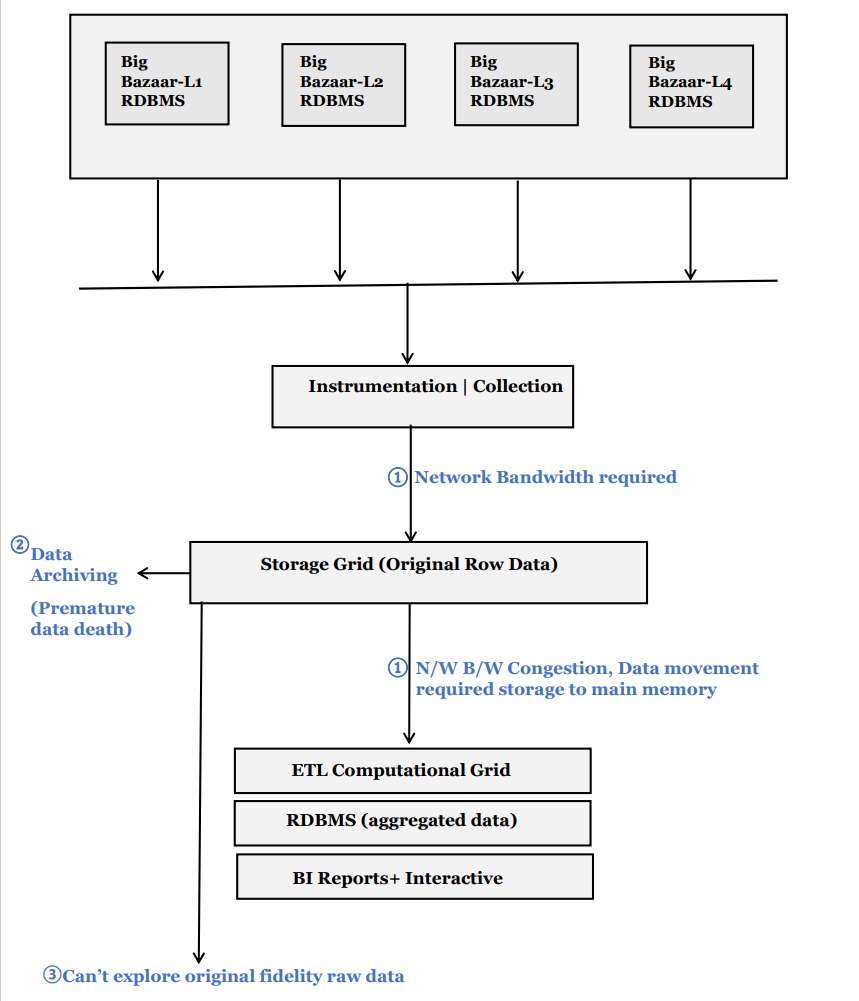
**Example: -** Word file, PDF, Text, Media log etc.

1. **Semi-structure data: -**

It is a structured data but it is not organized into a relational model, like tabular format.

**Example: -** XML files, JSON files, script etc.

**Limitations of Existing data Analytics Architecture: -**



1. **Diversified programming module: -**

Existing data analytical architecture have limitation to run different

Programming language.

Different data tools are required for different type of data

**Example: -** MMP (massive parallel process) graph etc.

1. **Storage area network issue: -**

In existing data analytical architecture data movement is need from

Storage grid area to computational grid area. There is utilization of network

Band width exist which cause network congestion and network congestion

will cause missing SLA`s (service level agreement).

1. **No direct access: -**

BI (Business Intelligence) person does not have direct access to actual data

So, they cannot process different type of analysis.

1. **Premature data death: -**

Before processing on top of data if data is not available then have to access

data from archival zone as 90% of data is available on archival zone.

And again, this causes missing SLA`s.

**SOLUTIONS: A Combined storage computer layer:** -

All the above problem are over come in Hadoop.

1. **Diversified programming Model support: -**

Hadoop support different types of data using “**YARN”** (yet another resource Negotiator) and spark introduction for “**Polyglot”.**

1. There is no separate computational grid, the storage grid itself do computational. So, no data movement needed.
2. The storage grid exposes it`s services to the end users using “Thrift” services.
3. Replication of data is available so data is highly available all the time.

**What is Hadoop: -**

Hadoop is an open-source software frame work for distribution processing & starting large set of data and running applications on clusters of commodity hardware.

**Custer: -**

Cluster is a collection of many computers which worked together as a one system.

**Commodity hardware: -**

Computer hardware that is affordable and easy to obtain. Typically, it is a low

Performance system and is capable of running windows, Linux, or MS-DOS

Without requiring any special device or equipment.

**Distributed File system: -**

Traditional DFS is a file system that allows us to store data over multiple

Nodes or machines in a cluster and allow multiple users to access data. in case

Of DFS data is stored in multiple machines rather than single machine.

**EXAMPLE: -**

A man is filling 10 forms in 100 minutes if he distributes his work in 10 peoples than each person will take 10 min for this task and as they all start simultaneously so parallelly all people complete their total 10min.

So, in disturbed manner instead of 100 minutes he completes his work in 10 minutes.

**Problem with DFS: -**

1. **Storage: -**

In DFS we can’t storage large size of files but we can store small size of file in large number. So, it creates bottleneck at the time of storage & retrieval of data.

1. **Processing: -**

As large number of small files are present so at the time of processing congestion occur at network band width due to heavy traffic. Synchronization problem may also occur.

1. **Data loss: -**

If any of the nodes get down then complete data goes loss as there is no replication of data is available.

Due to node failure other nodes get impacted as they got more data load then previously, they are getting.

1. **Meta data: -**

No meta data is available about the cluster data, so accessing data take more time.

All the about problems are over come in Hadoop.

**History of Hadoop: -**

In 2003 “Drug Cutting” was working for project “Apache Nautch” and doing work for a use case like “web crawling” and “search engine”.

He needs a scalable solution for the above use cases as data keeps on growing.

During the same time google has published a white paper called GFS and

MapReduce Doug cutting implemented google file system (GFS)&MapReduce and name the content as GFS ->Hadoop Distributed file system (HDFS) and same MapReduce.

Later he separated out the HDFS and MapReduce from Apache Notch and created a project called Hadoop. His son was planning with a yellow elephant shapes toy named Hadoop so he took his project name Hadoop.

At yahoo he performed sort of a terabyte data on 910m\c and response time was 3.5 min in April 2008 next in May 2009 yahoo does fastest sort of a TB. 62 sec over 1460 needs and PB in 16.25 hrs over 3658 nodes.

Later he resigned from the yahoo and started a company “Coludera” and make Hadoop to open-source community from this onwards it is Apache Hadoop.

**Apache Hadoop Components: -**



**Generation-1: -**

Hadoop is a frame work that store & process large dataset on DFS which are made up of commodity H|W and uses simple programming model called MapReduce application.

**Generation-2: -**

Apache Hadoop is a framework,

Which store & process large data sets

In Distributed file System (DFS),

Using commodity H|W,

Use simple (MapReduce) and diversified programming model.

**Commodity H/W: -**

It is a relatively low-cost h/w which is highly available in the market.

**HDFS Components: -**

1. Nome Node.

2 Data Node

3. Secondary Node

4. Resource Manager

5. Node Manager

**HDFS: - [Hadoop Distributed file System] Architecture: -**

It follows Master - Slave architecture.

In Master-solve architecture slave Computers are attached to the master Computer and perform data. processing function as directed by their master. It have5 Component's in Hadoop given above.

**Name Node:**

#It is master of the system.

# It stores meta data in RAM for fast read/ high availability performance, as

huge data have to access in less time.

#It works on Java object.

# It is not a Commodity H/W

# No demand for file system paging. All meta-data is stored only in RAM So no data is present in ROM and if all date is present in RAM (main memory) itself then no need to load required from ROM to RAM. hence in "Name Node, No demand paging of FS meta data.

**Configuration: -**

RAM - 1 128GB

Processor – XENON

Core - 128 Core

Hard Disk - 6TB (6XITB)

Frequency – 12.6

Ethernet -3 x 10Gb

0s – Centos

Version - 6 or 7

estimated cost - 23 lack

# Redundant power supply

# RAM Consist JVM..

#Rom Consist FS image & edit logs



Public Class Name Node

{

Store Metadata();

listen Heart Beat();

RollEdits();

RollFsimages();

}

**Data Node:**

#It is a slave node.

#It Store actual data in terms of Block in Rom

#It works on Java object.

# It is a Commodity H/W.

# Client directly read/write data to/from Data node.

# Data node will serve read/ write request from client. Basically, Name node provide the meta data the client and client extract data from the data node directly.

This is because of n/w congestion its Nome node extract the data then definitely there will be huge truffle at Nome node side.

# Data node writes the replication of data by default 3 times. Means it client store data, it will only be store in 3 diff locations.

#Replication factor can be modified according to the requirement

# Send heart beat to name node in 5 minutes showing it is alive.

**# Configuration: -**

RAM - 64 GB

Processor -Xenon

Core - 64 Core

HDD - 6X12 TB

OS - Centos

Frequency - 8.5 to 12

Ethernet - 3 X 10G

**Why Replication factor is 3:**

It is Basically the number of times Hadoop framework replicate each and every Data Block. Block is replicated to provide faut Tolerance. By default, it is 3 it can be changed according the requirement.

Among three are located at diff -2 locations. The first Copy is stored on local node. The other Some Re most rack but in node goes down also store in the different node. so, it first local then other rack is alive and it local or other one also dead then lost nodes is still alive. But it both rack fails then data will be lost. failure chances of two locations at the same time is very very impossible. Additional replica is randomly placed.

**Example: -**



In This example if any of the node or Rack fails, replica of the data will be available on other racks.

**Edge Node:**

It is also called app servers or UBUNTU. User\developer can't directly connect to name node or Data node... They Connect with edge node to work. Edge node is a m/c where a developer, QA engg. or Deploy manager connect & executes the Commands.

Edge node can be 6.8.10 in number as per, number of user present. Minimum it can be. 3. for according to requirement.



Putty is used here to connect with other OS. Jump Box Contain all Java program and using Putty Connect to Dev Box and then again putty is used to Connect DEV BOX to edge node.

**Data Set: -**

# It is never created by us. It will be downloaded oh pushed by other process.

#It is not very large in size but further can be accumulated as a single filler.

#Data Set are extracted from the source like oracle... Teradata, facebook etc...

#Data set contains metadata.

#Data set States in Block.

**Meta Data:**

Meta data is a data about the actual data. for example, meta data contains properties of the file. like name of the file, size, owner, created by time, update time, permissions etc. It also contains Block info that which data part is placed at which block. Like Block1 – DN1, DN2, DN7, Block2 DN3, DN5 DN4

**Block: -**

A block is the smallest unit of data In Generation-1 Storage in HDFS by default the block size is 64MB. And in Generation - 2 by default it is 128MB.

In our traditional system smallest black size is 4k.

**Why Block: -**

Any file that to be stored in HDFS should be split into blocks to achieve load balance.

To achieve parallelism.

To achieve fault tolerance.

To reduce read preference.

**Example: -**

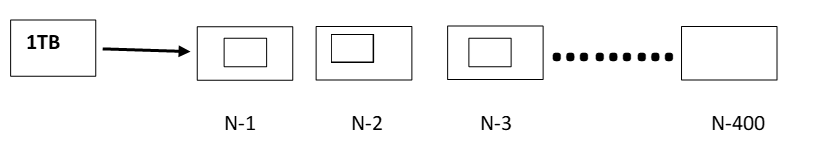
Let a ITB file.

#Without blocking, let divide file into 2 parts and run on 2 different m\c having 64 GB of RAM.

****

It will take around **1** hr. of time to read complete file.

**#**With blocking let there are 400m\c in a cluster.

****

each node will take 40 sec and all nodes will process at the same time so that time will be 40 sec only.

**#**And if any of node get fail then lose time will be less

**Why 64MB: -**

To achieve **constant performance** in the file system Doug cutting found rang of file from 4KB to 64MB constant means read al the block at the same time and end at the same time. He choosen 64MB because to reduce the name node meta data.

The main reason for having large blocks size is to reduce the cost of seek time. If a large will broke into 4K blocks size then meta-DATA will be huge for name node and At the time of reading data will take more time to seek all the blocks.

**Seeking time** – It is a time taken by the paddle to move from one location to another.

So up to 64MB and new 128MB our h\w are not capable to perform constant so due to this, task can take more time to complete.

If our h\w get more mature then block size can be increase more but partially it is very difficult.

**HDFS Architecture: -**

It defines how the components are interacting with each other. To understand architecture let us create a file “wordcount.txt” on “users\ eureka” using the comman 

1. **Hadoop is a utility.**
2. **Fs is a command.**
3. **Put is sub command.**
4. **Wordcount.txt is a file on edge node.**
5. **\user\eureka is a destination path on HDFS directory.**
6. **Hadoop utility perform following 4steps: -**
7. Loading configuration.
8. Identifying the command vale =fs
9. Setting the class path.
10. Identifying the java class: -

org.apache.hadoop.fs.fsshell

and executes main() method-> ToolRunner.run()method-put()

and put() will executes **copyFromLocal()** method.

In **copyFormLocal()** it will identify the **FileSystem.**

we have different types of file system for different types of data for example:

**Local fs (file: /// )**

**HDFS fs (hdfs://IP address Iport of NN)**

**Kosmos FS (KES) (kfs://IP address Master).**

**Amazon FS (53) ($3: // bucket-name)**

Copyfromlocal calls "fileutil" and in fileutil Copy() method. will be called to copy from the edge node (source) to HDFS (destination).

**Anatomy of "file write": -**

❷

❶

File Write = Creating Metadata + Add data

Anatomy is a complete detail of a file.

**example: -** how creating of metadata, writing in a file, creating Content of metadata etc.

**1.) Creating Metadata**: -

Create the empty file and place in the Name Node...

**Step1: -**

The client creates the file by calling **create ( )** on Distributed File System.

**Step2: -**

DFS makes a Remote Procedure call (RPC) call to the name node to Create a new file in the file system’s namespace, with no blocks associated with it.

DFS invoke NN’s java Object **create( )** by summing the metadata .

**Name Node Process Perform 2 Checks: -**

a) NN check, file is already existed of not on NN. If file is already present in NN then it will throw the exception **"file. already exist".**

b) Second NN checks, user has permission to write on the Data Node. If permission not exist then it will throw the exception **"permission denied".**

If all good and no exception occurred than it will create on empty file and put the metadata entry in the metadata file on NN.

**#** It we are creating our own methods then client will read file Name & Destination information from Command line. Argument and convert it into Path because HDFS doesn’t Understand string it only understand URI then it will create method.

**Example: -**

**Java com.laboros.hdfs.HDFSSERVICES wordCount.txt/user/etc**

**String filename= args[0]**

**String dest =args[1]**

**On hdfs =user/ edureka / wordcount.txt**

**String newstr = dest + " / " + filename;**

**Path newpath = new (newstr);**

**Or**

**Path newpath = new (dest, fileName);**



**2.) Adding Data: -**

**Step1:** split data into blocks.

**Step 2:** Identification of Data nodes per data block through BPP.

**Step 3:** writing Data Block to Data Node.

**step 4:** Meet Replication

**Step 5:** Sync to Data Block + DataNode to Nome Node

**Step 6:** Handling failure.

**Step1: -** Split data into Block.

After Creating an empty file on Name Node RAM, (Distributed fs) it will return an object **“FSDataOutputStream”** to the client to start writing data.

Then client will call open() on local file system edge node for file wordCount.txt and gets the inputStream. using inputStream client will read the data in term of byte! (byte [ ] because hdfs support variety of format and byte a format for common for all).

This byte[] will feed to the FSDataOutputStream · FSDOS will create data packet (block) and put it in a data queue.

**How many packets in a queue?**

It will be according to capacity of edge node’s RAM. Suppose RAM = 64GB then for other process let u take 4GB then left 60 GB will be used for data packets.

60 GB RAM will contain **480** packets (Block)

It follows combination of parallel and sequential process. I queue whole at a time-parallel and then next queue-sequentially come and take parallel for it.



**Step-2:** - Identification of DataNode for given data block. The data queue is consumed by the Data Streamer. Data Streamer pick the packet from the data queue and connect to the NameNode.

NameNode Use Block Placement Policy (BPP) and allocate no. of

data node = Replication factor and form a linklist /Pipline.

Let replication factor=3.

B1 placed at DN1, DN3, DN7

|  |  |
| --- | --- |
| **B1** | **1010** |

|  |  |
| --- | --- |
| **B1** | **1070** |

|  |  |
| --- | --- |
| **B1** |  |

**Linklist**

DN1 DN3 DN7

**1010**  **1070**

**step 3: -** writing Data Block to Data Node.

FileSystem DataOutputStream also maintain an internal queue of Packets. when Data Streamer Consume few packets from the data queue. A copy of the consumed data is stored in “ack queue”.

**Step4: -**

Data Streamer stream the packet to the first mode in pipeline. Once the data is written in DNI, the data is forward to Next DN. This repeats till last DN. Once the pkt is written to the last DN, on ack is sent from each ON to DESOS. The packet pi is removed from Ack Queue. The whole. process Continues till a block is filled.

**Step5: -**

After that, the pipeline, is closed and DataStream asks NN fish fresh set of DNs for next block. And the cycle, Repeats.

HDFS client calls the close () method once the write is finished. This would flush all the remaining packets to the pipeline & waits for ack before informing the NN that the white is complete.

**Step6: -**

At any point the data node gars down following are the two process the NN perform: -

first if there is a write operation still continue the NN will Remove the failed DN and assign new DataNode and start writing the data.

Second if there is no write the name node will identify under Replicated or Over Replicated and initiate the load balances for the write operation.

**Anatomy of File Read: -**

Reading file= Reading Metadata + Read Data

1. **Reading Metadata: -**

Identify which file client/usen need to read from the HDFS (from Command line args).

string hdfsfile= args[0]

Convert string into URI using path as Hadoop does not understand string.

**URI (uniform Resource Identifier): —**

A URI identifies the name and the location of a file. of resource in a uniform format. It is unique all over the n/w. Example: WWIN. Google.com etc.

Create the FS object using fileSystem.get(cont).

Invoke Open() on hdfsFileSystem Object by submming the path.

It will connect to the NN and submit the meta data to read the file.

**Here Name node will perform 2 validations: -**

1. Requested file is present or not. If file not exist then NN will throw an exception “Invalid input path exception “Otherwise.
2. NN will check that client has read permission or not if permission not exit then NN will throw an exception “Permission Denied”.

Otherwise, it will return an object called FSInputstream to the client side.

**(2) Reading Data: -**

Connect to the DataNode

Read the Block

Synchronization

Merging

FSInputStream will start reading the data from the DataNode based on the proximity (Nearest in Distance) of the Data Node, and creates temporary files.

The merging of the file and synchronization will be. happen using "sync Marker". It is a 4 Byte into out of which Byte is useful for reading the next. Data Block

We never ever bring large file to the edge node. for processing MapReduce, because edge node can't. handle big files as these will a lot of traffic congestion. Lots of RAM will be required and it will be a costlier operation so basically, we send our program to all Data Node where our file blocks are exit and DN itself do the calculation and send calculated data to the. Edge Node.

**Why Replication factor is 3 by default /Rack Awareness /BPP**

1. **Flat N/W Topology: -** Hadoop uses. by default, flat topology for a smaller number of machines. In this all nodes of cluster are in one read
2. **T-N/W Topology: -** If there is a greater number of m/c are in Hadoop cluster in order to accommodate all nodes Hadoop. introduce T- Topology, and Racks Awareness.

**RACK Awareness: -**

A rack is a collection of 30-40 nodes that are physically stored close together and are all come to the same n/w switch N/W bandwidth b/w 2 node in rack is greater then b/w two node in rack is greater than “b/w two node on different racks. A Hadoop cluster is a Collection of racks.



In order to improve the n/w tragic in a large cluster of Hadoop while reading/writing HDFS file, NameNode Choose the DataNode which is closer to the same rack or nearby rack to Read/Write request. This Concept of choose closer DataNode based on the rack information is called Rack Awareness in Hadoop.

In Hadoop rack awareness is required to improve data high availability and reliability.

To improve the performance of the cluster.

To improve n/w bandwidth.

To avoid losing data and keep bulk data in the rack.

**Block placement policy: -**

Block replication is multiple racks in HDFS is done using a policy as follows: -

1. When a new block is created, the first Block go to a node which is very nearest to the edge node.

In Order to compute distance, Hadoop follow for formula: -

R1N1-R1N2<R1N1-R2N4

Means distance of 2 node in same rack will always be in comparison of distance of 2 node in diff rack.

1. Replication 1 -

According to BPP the second Block (Replicated Black) be to place on the m/c which handle failure.

There are different types of failure: -

1) Data Centre failure

2) cluster failure.

3) Rack failure

4) Data Node failure

Data Centre and cluster will be handled by keeping replica of data centre.

Rack failure is handled by keeping the Data Block another node in another rack.

Data Node failure is handled by keeping the Data on Same Rock.

So, the first replication of Data Block will place. in different rack.

1. Replication 2: -

The Replication factor 3 was so Choosen to handle. read performance. According to the Read performance. effect the replicated block have to be choosen to place the data block on same rack with in the other Node.

Example: Black-A, Block B, Black-C

