Big Data Analytics

UNIT- I (Introduction to Big Data)

What is Big Data:

Definition: Big data refers to the technologies and initiatives that involve data that is too diverse, fast changing or massive for conventional technologies, skills and infra structure to address efficiently.

Big Data is "data of a very large size, typically to the extent that its manipulation and management presents significant logistical challenges for an enterprise.

The emerging technologies and practices that enable the collection, processing, discovery, analysis and storage of large volumes and disparate types of data quickly and cost effectively.

Characteristics of Big Data: Following are the big data core characteristics. Understanding the characteristics of big data is vital to know how it works and how you can use it. There are primarily seven characteristics of big data analytics:

1. Volume:

Volume refers to the amount of data that you have. We measure the volume of our data in Gigabytes, Zettabytes (ZB), and Yottabytes (YB). According to the industry trends, the volume of data will rise substantially in the coming years.

2. Velocity: Velocity refers to the speed of data processing. High velocity is crucial for the performance of any big data process. It consists of the rate of change, activity bursts, and the linking of incoming data sets.

3. Value:

Value refers to the benefits that your organization derives from the data. Does it match your organization's goals? Does it help your organization enhance itself? It's among the most important big data core characteristics.

4. Variety:

Variety refers to the different types of big data. It is among the biggest issues faced by the big data industry as it affects performance. It's vital to manage the variety of your data properly by organizing it. Variety is the various types of data that you gather from different kinds of sources.

5. Veracity:

Veracity refers to the accuracy of your data. It is among the most important Big Data characteristics as low veracity can greatly damage the accuracy of your results.

6. Validity:

How valid and relevant is the data to be used for the intended purpose.

7. Volatility: Big data is constantly changing. The data you gathered from a source a day ago might be different from what you found today. This is called variability of data, and it affects your data homogenization.

8. Visualization

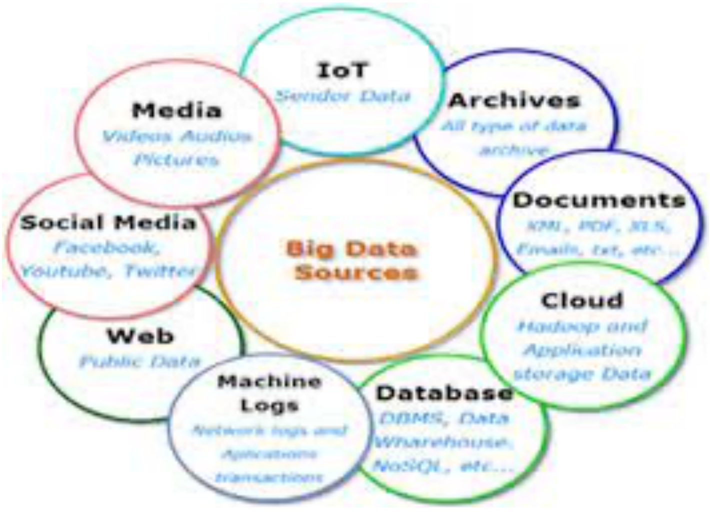
Visualization refers to showing your big data-generated insights through visual representations such as charts and graphs. It has become prevalent recently as big data professionals regularly share their insights with non-technical audiences.

Sources of Big Data: A significant part of big data is generated from three primary resources:

Machine data

Social data, and

Transactional data.


IoT
Sensor Data
Archives
All type of data archive
Documents
XML, PDF, XLS, Emails, txt, etc...
Cloud
Hadoop and Application storage Data
Database
DBMS, Data Wharehouse, NoSQL, etc...
Machine Logs
Network logs and Aplications transactions
Web
Public Data
Social Media
Facebook, Youtube, Twitter
Media
Videos Audios Pictures
Big Data Sources

1. Machine Data

Machine data is automatically generated, either as a response to a specific event or a fixed schedule. It means all the information is developed from multiple sources such as smart sensors, SIEM logs, medical devices and wearables, road cameras, IoT devices, satellites, desktops, mobile phones, industrial machinery, etc. These sources enable companies to track consumer behaviour. Data extracted from machine sources grow exponentially along with the changing external environment of the market.

In a broader context, machine data also encompasses information churned by servers, user applications, websites, cloud programs, and so on.

2. Social Data

It is derived from social media platforms through tweets, retweets, likes, video uploads, and comments shared on Facebook, Instagram, Twitter, YouTube, Linked In etc. The extensive data generated through social media platforms and online channels offer qualitative and quantitative insights on each crucial facet of brand-customer interaction.

3. Transactional Data

As the name suggests, transactional data is information gathered via online and offline transactions during different points of sale. The data includes vital details like transaction time, location, products purchased, product prices, payment methods, discounts/coupons used, and other relevant quantifiable information related to transactions.

The sources of transactional data include:

Payment orders

Invoices

Storage records and

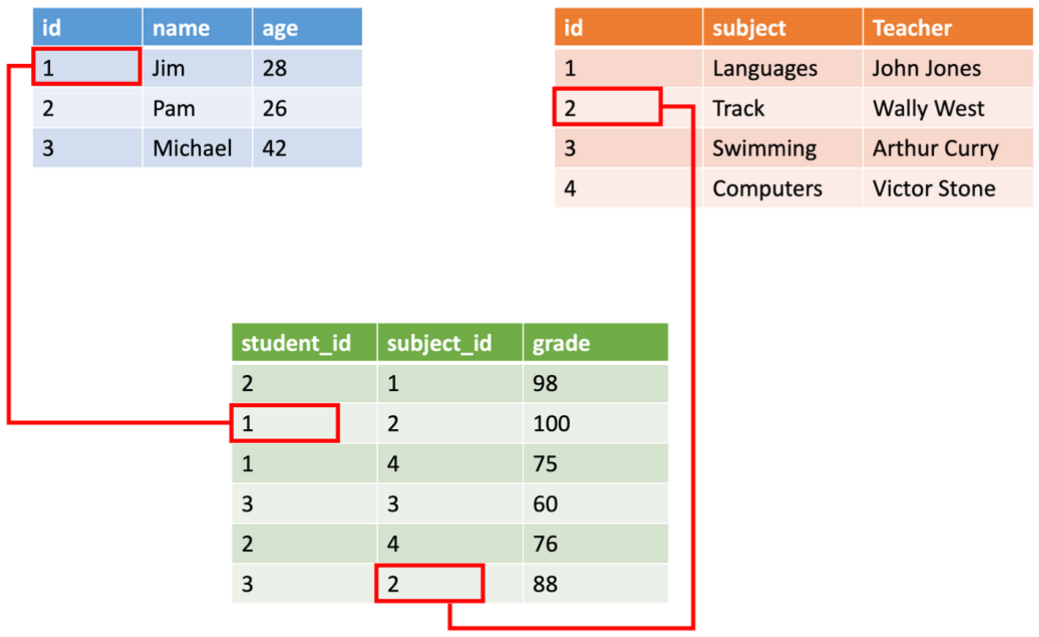
E-receipts

Types of Big Data

There are primarily three types of data in big data:

1. Structured

Structured data refers to the data that you can process, store, and retrieve in a fixed format. It is highly organized information that you can readily and seamlessly store and access from a database by using simple algorithms. This is the easiest type of data to manage as you know what data format you are working with in advance. For example, the data that a company stores in its databases in the form of tables and spreadsheets is structured data.



id	name	age
1	Jim	28
2	Pam	26
3	Michael	42

id	subject	Teacher
1	Languages	John Jones
2	Track	Wally West
3	Swimming	Arthur Curry
4	Computers	Victor Stone

student_id	subject_id	grade
2	1	98
1	2	100
1	4	75
3	3	60
2	4	76
3	2	88

2. Unstructured

Data with an unknown structure is termed unstructured data. Its size is substantially bigger than structured data and is heterogeneous in nature. A great example of unstructured data includes the results you get when you perform a Google search. You get webpages, videos, images, text, and other data formats of varying sizes.

Examples of unstructured data include:

Media files, like photos, videos, and audio files

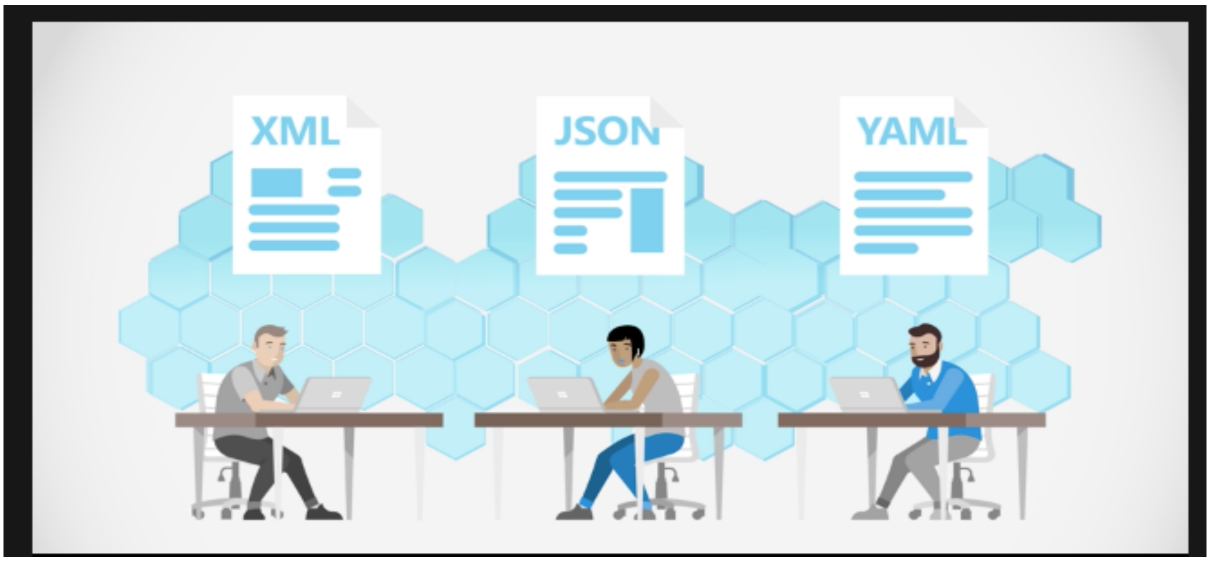
Microsoft 365 files, like Word documents

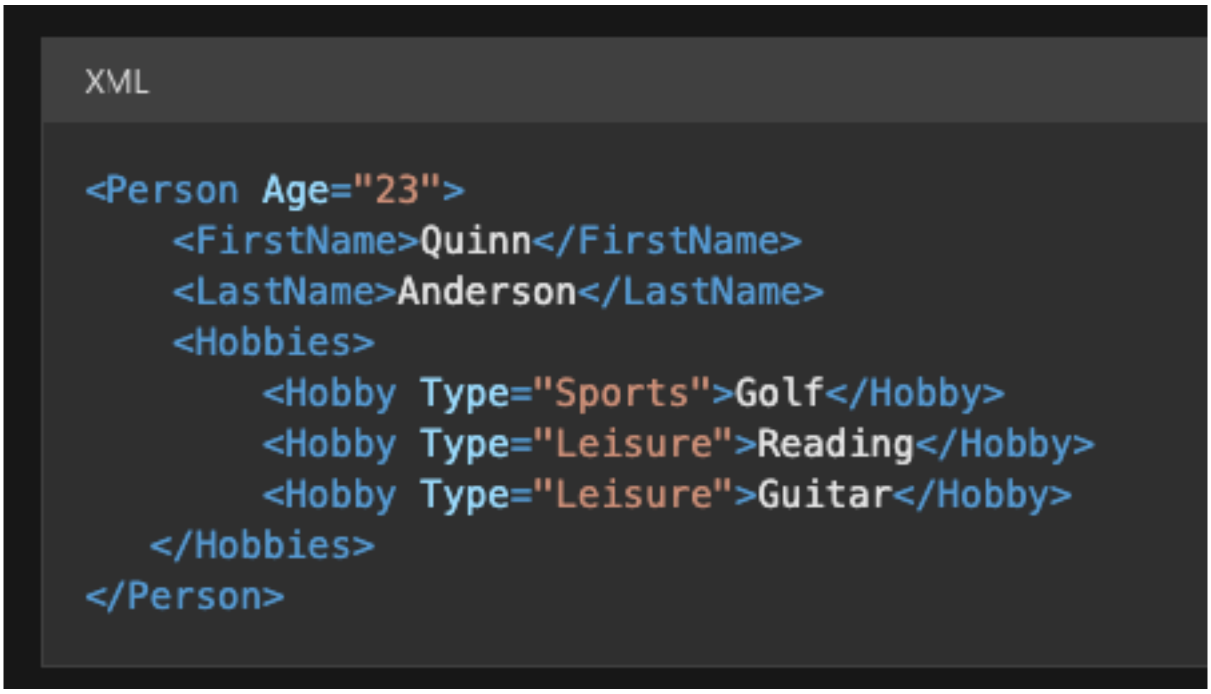
Text files

Log files

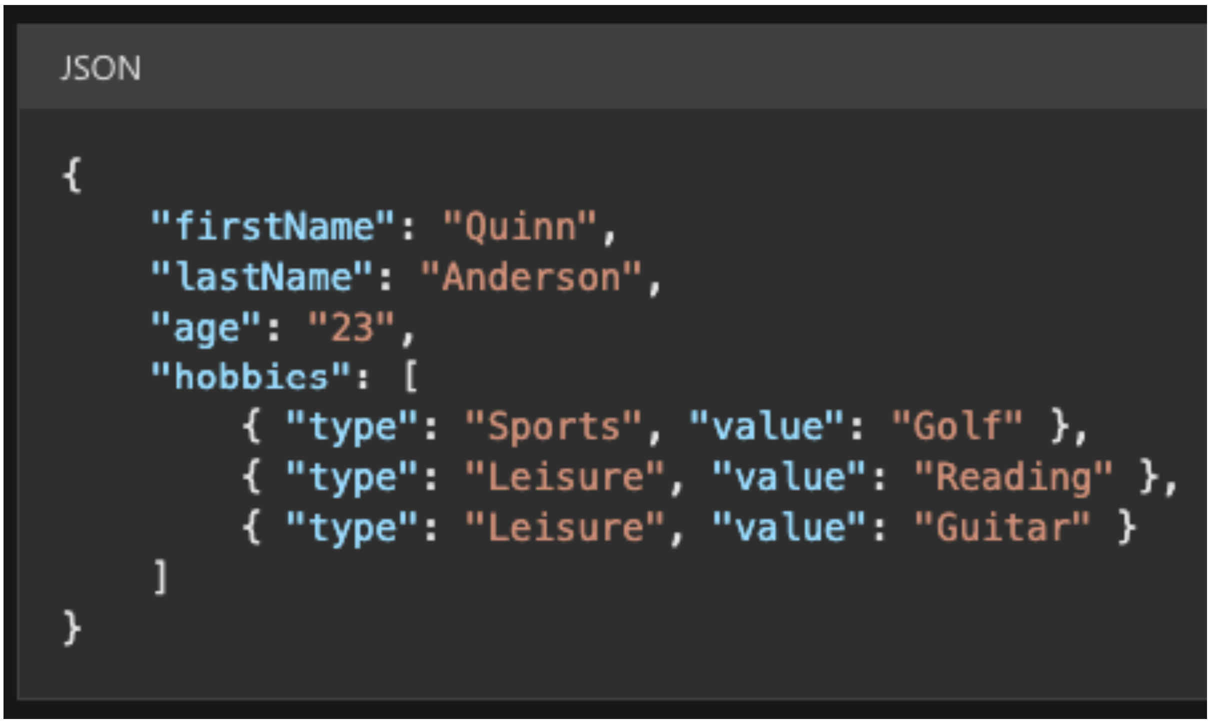
3. Semi-structured

As the name suggests, semi-structured data contains a combination of structured and unstructured data. Semi-structured data is less organized than structured data. Semi-structured data isn't stored in a relational format because the fields don't fit neatly into tables, rows, and columns. Semi-structured data contains tags that make the organization and hierarchy of the data apparent. One example is key/value pairs. Semi-structured data is also referred to as non-relational or not only SQL (NoSQL) data.




XML

```
<Person Age="23">
    <FirstName>Quinn</FirstName>
    <LastName>Anderson</LastName>
    <Hobbies>
        <Hobby Type="Sports">Golf</Hobby>
        <Hobby Type="Leisure">Reading</Hobby>
        <Hobby Type="Leisure">Guitar</Hobby>
   </Hobbies>
</Person>
```



```
{
    "firstName": "Quinn",
    "lastName": "Anderson",
    "age": "23",
    "hobbies": [
        { "type": "Sports", "value": "Golf" },
        { "type": "Leisure", "value": "Reading" },
        { "type": "Leisure", "value": "Guitar" }
    ]
}
```

Challenges in Big Data: These are the following challenges that come into the way while dealing with big data.

1. **Lack of Knowledge professional:** To run the modern technologies and large data tools, companies need skill data professionals. (data scientists, data analysts, data engineers).

2. **Lack of proper understanding of massive data:** Employees might not know what data is , its storage, processing, importance and sources.

3. **Data Growth Issues:** In RDBMS the data is kept in different-different tables to ensure normalization and eliminate redundancy. For Selection of information, the join queries will be fired on the respective tables in the dataset. So these join queries will take more time if the data is massive or keep

on increasing everyday. As these data grow exponentially with time it get more challenging to handle the data.

4. Confusion while Big data tool Selection: Companies often get confused while selecting the simplest tool for giant data analysis and storage.
   - Is HBase or Cassandra for data Storage.
   - Hadoop Map Reduce or Spark for Data Analysis.

5. Integrating data from a variety of Sources: Data comes from variety of resources such as social media pages, ERP applications, Customer logs, financial reports, email, presentations and various reports created by employees.

   So combining all this data into an organize reports might be challenging tasks for clients.

6. Securing Data: Securing these huge set of knowledge is one of the daunting challenges of massive data. Often companies are busy in understanding , storing and analyzing their datasets, that they push data security for later stages.
7. Storage and Transport Issues: Means Data transfer takes larger time then data processing.
8. Data Management Issues:
   - Sources of data is varied by size, format, methods of collections
   - What, Where, why, Who, When and How the data is collected and its provenance.

- Rigorous protocols are followed to ensure its accuracy and validity.

9. Processing Issues:

-Extensive processing and algorithms are required to provide timely and actionable information.

10. Characteristics Issues:

-Data Volume- Terabyte of data being produced everyday which is might be unmanageable using existing traditional systems.

- Data Velocity: Traditional System are not capable of performing analytics of data which is in motion. Ecommerce has increased the speed and richness of data used for business transactions.

- Data Variety- Since the data comes in various forms from various sources.

-Incompatible data formats, non-aligned data structure represents significant challenges that can lead to analytic spread.

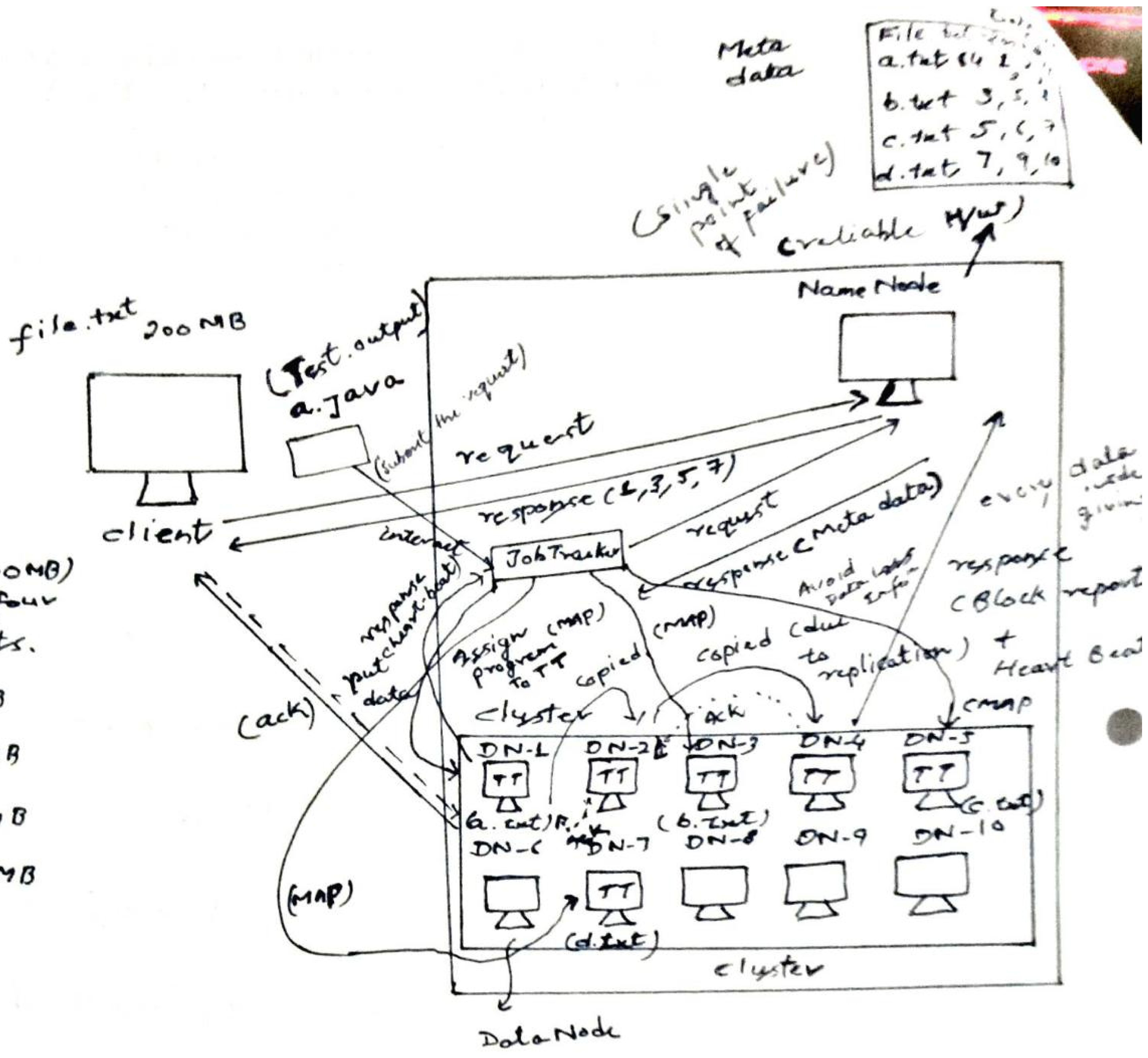
file.txt-(200MB) spilts into four input spilts.

a.txt - 64MB

b.txt - 64MB

c.txt - 64MB

d.txt - 64MB



5 m 6R-7500
250 GB-3750

(HDFS Architecture)

HDFS Architecture:

-- HDFS stands for Hadoop Distributed File Systems.

-- HDFS is used to store (HDFS) and processing (Map-Reduce) a large data sets.

-- HDFS is a specially designed file systems for storing a huge datasets with a cluster of commodity hardware with streaming access patterns. (Means write once and read any number of times but don't try to change the contents of a file on HDFS.)

--Block size of HDFS is 64 MB in Hadoop 1.0 . (By Default) You can also set the block size up to 128 MB. (Hadoop 2.0) (Job of Hadoop Administrator Profile).

--Hadoop Distributed File System is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines. Apache Hadoop HDFS Architecture follows a Master/Slave Architecture, where a cluster comprises of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes).

-- Some of the services that are running on the Hadoop Distributed File Systems are on the top of current operating files systems are as-

-- NameNode
-- Secondary NameNode
-- Job Tracker

} These are called as Master Services

-- DataNode
-- Task Tracker

} These services are called as slave Services or Daemons.

-- Master Services and Slave Services can talk to each other to coordinate their works.(Means **NameNode** can interact to **DataNode** and **JobTracker** can interact to **Tasktracker** to coordinate their work on HDFS).

1. NameNode : NameNode is the master node in the Apache Hadoop HDFS Architecture that maintains and manages the blocks present on the DataNodes (slave nodes).

--NameNode is a very highly available server that manages the File System Namespace and controls access to files by clients. The HDFS architecture is built in such a way that the user data never resides on the NameNode. The data resides on DataNodes only.

--NameNode contains metadata about blocks allocated to data, block replica information of the data over to the clusters of data node, Number of splits of datasets. For example if a dataset file.txt is of 200 MB. If it is deployed over to HDFS then that dataset is divided in four data block of size 64MB, 64 MB, 64 MB, 8 MB block respectively over to HDFS.

	SIZE.	Data Node and Replica information about block
a.txt.	64 MB	1, 2, 4
b.txt.	64 MB.	3, 5, 8
c.txt.	64 MB.	5, 6, 7
d.txt.	64 MB.	7, 9, 10

- HDFS maintains the 3 replica of the data on the Data Node of the cluster.

Functions of NameNode:

--It is the master daemon that maintains and manages the DataNodes (slave nodes)

--It records the metadata of all the files stored in the cluster, e.g. The location of blocks stored, the size of the files, permissions, hierarchy, etc. There are two files associated with the metadata:

-- **FsImage:** It contains the complete state of the file system namespace since the start of the NameNode.

-- **EditLogs:** It contains all the recent modifications made to the file system with respect to the most recent FsImage.

- It records each change that takes place to the file system metadata. For example, if a file is deleted in HDFS, the NameNode will immediately record this in the EditLog.
- It regularly receives a **Heartbeat** and **a block report** from all the DataNodes in the cluster to ensure that the DataNodes are live.
- It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.
- The NameNode is also responsible to take care of the **replication factor** of all the blocks.

- In **case of the DataNode failure**, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

Important Point: If a NameNode is fails then entire concept of HDFS is fail so NameNode is a single point of failure since it contains all the information about the metadata of the blocks and its replicas.

2. Data Node: DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is a commodity hardware, that is, a non-expensive system which is not of high quality or high-availability. The DataNode is a block server that stores the data in the local file ext3 or ext4.

Functions of DataNode:

--These are slave daemons or process which runs on each slave machine.

--The actual data is stored on DataNodes.

--The DataNodes perform the low-level read and write requests from the file system's clients.

--They send heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds.

Job Tracker and Task Tracker in Hadoop:

- JobTracker and TaskTracker are 2 essential process involved in MapReduce execution in MRv1 (or Hadoop version 1). Both processes are now deprecated in MRv2 (or Hadoop version 2) and replaced by Resource Manager, Application Master and Node Manager Daemons.

Job Tracker –

- JobTracker process runs on a separate node and not usually on a DataNode.
- JobTracker is an essential Daemon for MapReduce execution in MRv1. It is replaced by ResourceManager/ApplicationMaster in MRv2.

- JobTracker receives the requests for MapReduce execution from the client.
- JobTracker talks to the NameNode to determine the location of the data.
- JobTracker finds the best TaskTracker nodes to execute tasks based on the data locality (proximity of the data) and the available slots to execute a task on a given node.
- JobTracker monitors the individual TaskTrackers and the submits back the overall status of the job back to the client.
- JobTracker process is critical to the Hadoop cluster in terms of MapReduce execution.

When the JobTracker is down, HDFS will still be functional but the MapReduce execution can not be started and the existing MapReduce jobs will be halted.

TaskTracker –

- TaskTracker runs on DataNode. Mostly on all DataNodes.

TaskTracker is replaced by Node Manager in MRv2.

- Mapper and Reducer tasks are executed on DataNodes administered by TaskTrackers.
- TaskTrackers will be assigned Mapper and Reducer tasks to execute by JobTracker.

- TaskTracker will be in constant communication with the JobTracker signalling the progress of the task in execution.
- TaskTracker failure is not considered fatal. When a TaskTracker becomes unresponsive, JobTracker will assign the task executed by the TaskTracker to another node.

Secondary NameNode: The **Secondary namenode** is a helper node in hadoop, To understand the functionality of the secondary namenodelet's understand how the namenode works.

Name node stores metadata like file system namespace information, blockinformation etc in the memory.It also stores the persistent copy of the same on the disk. Name node stores information in two files.

fsimage: It's a snapshot of the file system, stores information like modification time access time, permission, replication.

Edit logs: It stores details of all the activities/transactions being performed on the HDFS..

When the namenode is in the active state the edit logs size grows continuously as the edit logs can only be applied to the fsimage at the time of namenode restart, to get the latest state of the HDFS. If edit logs grows significantly and namenode tries to apply it on fsimage at the time of namenode restart, the process can take very long, here secondary node come into the play.

Secondary namenode keeps the checkpoint on the namenode, It reads the edit logs from the namenode continuously after a specific interval and applies it to the fsimage copy of secondary namenode. In this way the fsimage file will have the most recent state of HDFS.

Introducing and Configuring Hadoop cluster (Local Pseudo distributed mode, Fully Distributed mode):

Local Mode:

Standalone mode is the default mode in which Hadoop run. Standalone mode is mainly used for debugging where you don't really use HDFS.

You can use input and output both as a local file system in standalone mode.

You also don't need to do any custom configuration in the files- mapred-site.xml, core-site.xml, hdfs-site.xml.

Standalone mode is usually the fastest Hadoop modes as it uses the local file system for all the input and output. Here is the summarized view of the standalone mode-

- Used for debugging purpose
- HDFS is not being used
- Uses local file system for input and output

- No need to change any configuration files
- Default Hadoop Modes

2) Pseudo-distributed Mode

--The pseudo-distribute mode is also known as a single-node cluster where both NameNode and DataNode will reside on the same machine.

--In pseudo-distributed mode, all the Hadoop daemons will be running on a single node. Such configuration is mainly used while testing when we don't need to think about the resources and other users sharing the resource.

--In this architecture, a separate JVM is spawned for every Hadoop component as they could communicate across network sockets, effectively producing a fully functioning and optimized mini-cluster on a single host.

Here is the summarized view of pseudo distributed Mode-

- Single Node Hadoop deployment running on Hadoop is considered as pseudo distributed mode

- All the master & slave daemons will be running on the same node
- Mainly used for testing purpose
- Replication Factor will be ONE for Block
- Changes in configuration files will be required for all the three files- mapred-site.xml, core-site.xml, hdfs-site.xml.

3) Fully-Distributed Mode (Multi-Node Cluster)

This is the production mode of Hadoop where multiple nodes will be running. Here data will be distributed across several nodes and processing will be done on each node.
Master and Slave services will be running on the separate nodes in fully-distributed Hadoop Mode.

- Production phase of Hadoop
- Separate nodes for master and slave daemons
- Data are used and distributed across multiple nodes

In the Hadoop development, each Hadoop Modes have its own benefits and drawbacks. Definitely fully distributed mode is the one for which Hadoop is mainly known for but again there is no point in engaging the resource while in testing or debugging phase.

So standalone and pseudo-distributed Hadoop modes are also having their own significance.

Follow these steps for installing and configuring Hadoop on a single node:

Step-1. Install Java: we will use Java 1.6 therefore describing the installation of Java 1.6 in detail.

Use the below command to begin the installation of Java

```
$ sudo apt-get install openjdk-6-jdk
```

Step-2. Verify Java installation

You can verify java installation using the following command

```
$ java -version
```

On executing this command, you should see output similar to the following:

```
java version "1.6.0_27"
```

Step-3. SSH configuration

Install SSH using the command.

```
sudo apt-get install ssh
```

Generate ssh key

ssh -keygen -t rsa -P "" (press enter when asked for a file name; this will generate a passwordless ssh file)

Now copy the public key (id_rsa.pub) of current machine to authorized_keys. Below command copies the generated public key in the .ssh/authorized_keys file:

```
cat $HOME/.ssh/id_rsa.pub >>
$HOME/.ssh/authorized_keys
```

Verify ssh configuration using the command

ssh localhost

Pressing yes will add localhost to known hosts

Step-4. Download Hadoop

Download the latest stable release of Apache Hadoop from

http://hadoop.apache.org/ releases.html.

Unpack the release tar – zxvf hadoop-1.0.3.tar.gz

Save the extracted folder to an appropriate location,

HADOOP_HOME will be pointing to this directory.

Step-5. Verify Hadoop

Check if the following directories exist under

HADOOP_HOME: bin, conf, lib, bin

Use the following command to create an environment variable that

points to the Hadoop installation directory

(HADOOP_HOME)

```
export HADOOP_HOME=/home/user/hadoop
```

Now place the Hadoop binary directory on your command-line path by executing the command

```
export PATH=$PATH:$HADOOP_HOME/bin
```

Use this command to verify your Hadoop installation:

```
hadoop version
```

The o/p should be similar to below one

```
Hadoop 1.1.2
```

Step-6. Configure JAVA_HOME:

Use the below command to set JAVA_HOME on Ubuntu

```
export JAVA_HOME=/usr/lib/jvm/java-6-sun
```

JAVA_HOME can be verified by command

```
echo $JAVA_HOME
```

Step-7. Create Data Directory for Hadoop

An advantage of using Hadoop is that with just a limited number of directories you can set it up to work correctly. Let us create a directory with the name hdfs and three sub-directories name, data and tmp.

Since a Hadoop user would require to read-write to these directories you would need to change the permissions of above directories to 755 or 777 for Hadoop user.

Step-8. Configure Hadoop XML files

Next, we will configure Hadoop XML file. Hadoop configuration files are in the HADOOP_HOME/conf dir.

conf/core-site.xml

<! -- Putting site-specific property overrides the file. -->

fs.default.name

hdfs://localhost:9000

hadoop.temp.dir

/home/girish/hdfs/temp

conf/hdfs-site.xml

<! -- Putting site specific property overrides in the file. -->

dfs.name.dir

/home/girish/hdfs/name

dfs.data.dir

/home/girish/hdfs/data

dfs.replication

1

conf/mapred-site.xml

<! -- Putting site-specific property overrides this file. -->

mapred.job.tracker

localhost:9001

conf/masters

Not required in single node cluster.

conf/slaves

Not required in single node cluster.

Step-9. Format Hadoop Name Node-

Execute the below command from hadoop home directory

```
$ ~/hadoop/bin/hadoop namenode -format
```

Step-10. Start Hadoop daemons

```
$ ~/hadoop/bin/start-all.sh
```

Step-11. Verify the daemons are running

$ jps (if jps is not in path, try /usr/java/latest/bin/jps)

output will look similar to this

```
9316 SecondaryNameNode
9203 DataNode
9521 TaskTracker
9403 JobTracker
9089 NameNode
```

Now we have all the daemons running

Step-12. Verify UIs by namenode & job tracker

Open a browser window and type the following URLs:

namenode UI: http://machine_host_name:50070

job tracker UI: http://machine_host_name:50030

substitute 'machine host name' with the public IP of your node

e.g: http://localhost:50070

Now you have successfully installed and configured Hadoop on a single node