

* Big Data Analytics !→

Big Data Analytics is the often complex process of examining big data to uncover information -- such as hidden patterns, correlations, market trends and customer preferences -- that can help organizations make informed business decisions.

on a broad scale, data analytics technologies and techniques give organizations a way to analyze data sets and gather new information. Business Intelligence (BI) queries answer basic questions about business operations and performance.

Big Data Analytics is a form of advanced analytics, which involve complex applications with elements such as predictive models, statistical algorithms and what-if analysis powered by analytics systems.

* Objective of Big Data Analytics !→

Big Data analytics helps organizations take advantage of their data and use it to identify new opportunities.

1. To provide an overview of an exciting growing field of big data analytics.
2. To Introduce the tools required to manage and analyze big data like Hadoop, NoSQL, MapReduce.

3. To teach the fundamental techniques and principles in achieving big data analytics with scalability and streaming capability.

* Scope & outcome of Big Data Analytics :->

Big Data is defined as massive amount of data which is too large and complex to be stored in traditional databases. Data has evolved over the last 5 years. Lots of data is being generated each day in every business sector.

Below are some facts about Big Data for some of the companies ->

1. 40,000 search queries are performed on Google per second.
i.e. 3.46 million searches a day.
2. Every minute, users send 31.25 million messages and watch 9.77 million videos on Facebook.
3. 55 billion messages and 4.5 billion photos are sent each day on WhatsApp.
4. By 2025, the volume of digital data will increase to 163 Zettabytes.
5. Walmart handles more than 1 million customer transactions every hour.

* Big Data →

According to "Gartner" Big Data is high-volume, Velocity and Variety information assets that demand Cost-effective, ~~inn~~ innovative forms of information processing for enhanced insight and decision making.

Big Data is a term that describes the large volume of data - both structured and unstructured - that inundates a business on a day-to-day basis. Big data can be analyzed for insights that lead to better decisions and strategic business moves.

* Challenges of Big Data →

Many Companies get stuck at the initial stage of their Big Data projects. This is because they are ~~neig~~ neither aware of the challenges of Big Data nor are equipped to tackle those challenges. Let us understand them one by one -

(1) Lack of proper understanding of Big Data →

Companies fail in their Big Data initiatives due to insufficient understanding. Employees may not know what data is, its storage, processing, importance, and sources.

(2) Data growth issues →

one of the most pressing challenges of Big Data is storing all these huge sets of data properly. The amount of data being stored in data centers and

databases of Companies is increasing rapidly. As these data sets grow exponentially with time, it gets extremely difficult to handle.

Most of the data is unstructured and comes from documents, Videos, audios, text files and other sources. This means that you cannot find them in databases.

(3) Confusion while Big Data tool selection →

Companies often get confused while selecting the best tool for Big Data analysis and storage. Is HBase or Cassandra the best technology for data storage.

(4) Lack of data professionals →

To run these modern technologies and Big Data tools, Companies need skilled data professionals. These professionals will include data scientists, data analysts and data engineers who are experienced in working with the tools and making sense out of huge data sets.

Companies face a problem of lack of Big Data professionals. This is because data handling tools have evolved rapidly, but in most cases, the professionals have not.

(5) Integrating data from a variety of sources →

Data in an organization comes from a variety of sources, such as social media pages, ERP applications, customer logs, financial reports, e-mails, presentations and reports created by employees. Combining all this data to prepare reports is a challenging task.

(6) Sharing

(7) Searching

(8) Presentation

* * Problems with Traditional Large-Scale System :→

Because data is so expensive to store in traditional systems, data is filtered and aggregated, and large volumes are thrown out because of the cost of storage. Minimizing the data to be analyzed reduces the accuracy and confidence of the results. Not only are accuracy and confidence to the resulting data affected, but it also limits an organization's ability to identify business opportunities. Atomic data can yield more insights into the data than aggregated data.

* * Sources of Big Data :→

The bulk of big data generated comes from Primary Sources. In addition, Companies need to make the distinction between data which is generated internally, that is to say it resides behind a Company's firewall and externally data generated which needs to be imported into a system.

The Primary Sources of Big Data :→

(1) Social Data :→

Social data comes from the likes, Tweets & Retweets, Comments, Video uploads, and general media that are uploaded and shared via the world's favorite social media platforms.

(2) Machine Data :→

Machine data is defined as information which is generated by industrial equipment, sensors that are installed in machinery, and even web logs which track user behavior. This type of data is expected to grow exponentially as the Internet of things grows ever more pervasive and expands around the world.

(3) Transactional Data :→

Transactional Data is generated from all the daily transactions that take place both online and offline. Invoices, Payment orders, Storage records, delivery receipts - all these are characterized as transactional data yet data alone is almost meaningless, and most organizations struggle to make sense of the data that they are generating and how it can be put to good use.

(4) Share Market :→

Stock exchange across the world generates huge amount of data through its daily transaction.

(5) Telecom Company :→

Telecom giants like Airtel, Vodafone study the user trends and accordingly publish their plans and for this they store the data of its million users.

(6) E-Commerce Site :→

Sites like Amazon, Flipkart, Alibaba generate huge amount of logs from which users buying trends can be traced.

(7) Weather Station :→

All the weather station and satellite gives very huge data which are stored and manipulated to forecast weather.

3 Vs of Big Data :→ (3 properties)

This Conception theory gained thrust in the early 2000s when trade and business analyst Mr. Doug Laney expressed the mainstream explanation of the keyword big data over the 3 pillars of 3V's.

(1) Volume :→

Organizations and firms gather as well as pull together different data from different sources, which includes business transactions and data, data from social media, login data, as well as information from the sensor as well as machine-to-machine data. Earlier, this data storage would have been an issue - but because of the advent of new technologies for handling extensive data with tools like Apache Spark, Hadoop, the burden of enormous data got decreased.

(2) Velocity :→

Data is now streaming at an exceptional speed, which has to be dealt with suitably. Sensors, smart metering, user data as well as RFID tags are lashing the need for dealing with an inundation of data in near real-time.

(3) Variety !→

The ~~related~~ releases of data from various systems have diverse types and formats. They range from structured to unstructured, numeric data of traditional databases to non-numeric or text documents, emails, audios and videos, stock ticker data, login data, Blockchains encrypted data, or even financial transactions.

* Importance of Big Data !→

Big Data does not take care of how much data is there, but how it can be used. Data can be taken from various sources for analyzing it and finding answers which enable:

- Reduction in Cost
- Time reductions
- New product development with optimized offers.
- Well-informed decision making.

When we merge big data with high-powered data analytics, it is possible to achieve business-related tasks like -

- Real time determination of core causes of failures, problems or faults.
- Product token and Coupons as per the Customer's buying behavior.
- Risk-management can be done in minutes by calculating stock portfolios.
- Detection of deceptive behaviour before its influence.

* Types of Big Data :->

1. Structured :->

Structured is one of the types of big data and By Structured data, we mean data that can be processed, stored, and retrieved in a fixed format. It refers to highly organized information that can be readily and seamlessly stored and accessed from a database by simple search engine algorithms. for example -> Relational Data.

For instance, the Employee table in a Company database will be structured as the Employee details, their job positions, their Salaries, etc. will be present in an organized manner.

2. Unstructured :->

Unstructured data refers to the data that lacks any specific form or structure whatsoever. This makes it very difficult and time-consuming to process and analyze Unstructured data. for example:- word, pdf, Text, media Logs. Email is an example of Unstructured data.

3. Semi-Structured :->

Semi-Structured is the third type of big data. Semi-Structured data pertains to the data containing both the formats mentioned above, that is, Structured and Unstructured data. for example - XML Data.

Working with Big Data

* Google File System (GFS) →

Google File System (GFS) is a Scalable Distributed File System (DFS) Created by Google Inc. and developed to accommodate Google's expanding data processing requirements. GFS provides fault tolerance, reliability, scalability, availability and performance to large networks and connected nodes. GFS is made up of several storage systems built for low-cost commodity hardware components. It is optimized to accommodate Google's different data use and storage needs, such as its search engine, which generates huge amounts of data that must be stored.

The Google file system capitalized on the strength of off-the-shelf servers while minimizing hardware weaknesses.

[GFS is also known as GoogleFS.]

- GFS has Snapshot and Record append operations. Snapshot creates a copy of a file or a directory tree at low cost.
- Record append allows multiple clients to append data to the same file concurrently while guaranteeing the atomicity of each individual client's append.

Architecture of GFS →

A GFS cluster consists of a single master and multiple chunk servers and is accessed by multiple clients, as shown in the following figure :-

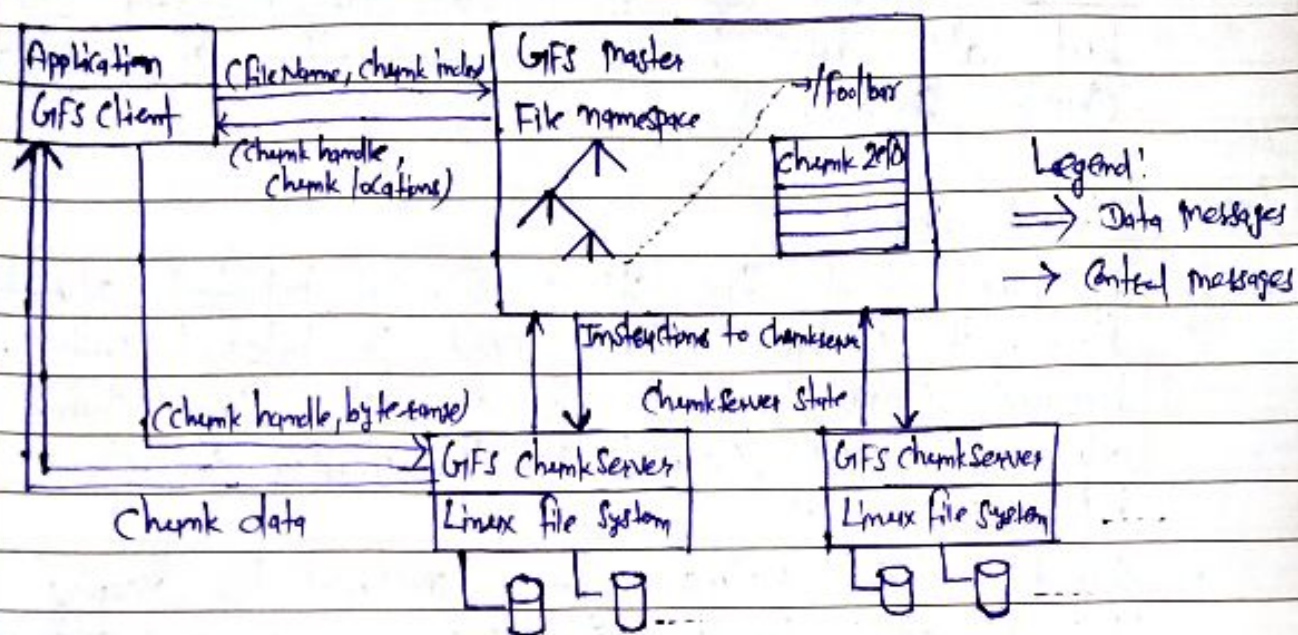


Figure -1 : GFS Architecture

- Each of these is typically a Commodity Linux machine running a user-level Server Process.
- Files are divided into fixed-size chunks. Each chunk is identified by a fixed and globally unique 64-bit chunk handle assigned by the master at the time of chunk creation.
- Chunk Servers store chunks on local disks as Linux files.
- For scalability, each chunk is replicated on multiple chunk servers.
- The master maintains all file system metadata. This includes the namespace, access control information, the mapping from files to chunks and the current locations of chunks.
- It also controls system-wide activities such as chunk lease management, garbage collection of orphaned chunks, and chunk migration between chunk servers.

- The master periodically Communication with each chunk Server in Heart Beat messages to give it instructions and collect its State.
- Clients interact with the master for metadata operations, but all data-bearing Communication goes directly to the chunk Servers.

Chunk Size's

- A large chunk size offers several important advantages.
- First, it reduces clients need to interact with the master because reads and writes on the same chunk require only one initial request to the master for chunk location information.
- Second, it can reduce network overhead by keeping a persistent TCP Connection to the chunkserver over an extended period of time.
- Third, it reduces the size of the metadata stored on the master.

* Hadoop :->

Hadoop is an Apache open source framework written in Java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single servers to thousands of machines, each offering local computation and storage. It is currently used by Google, Facebook, Yahoo, Twitter etc. It is used for batch/offline processing.

* Advantages of Hadoop :->

1. 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 petabyte in hours.

2. Scalable :->

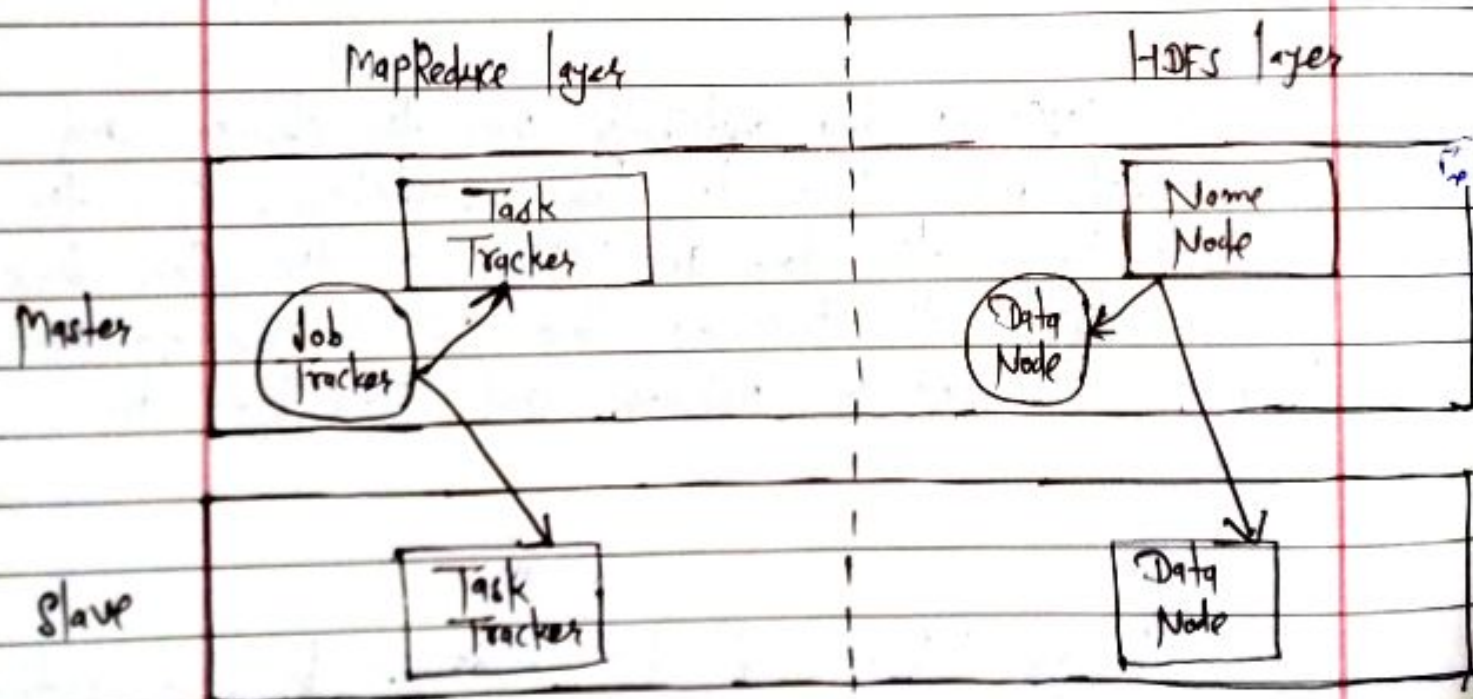
Hadoop Cluster can be extended by just adding nodes in the cluster.

3. 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.

* 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.



1. MapReduce Layer! →

The MapReduce comes into existence when the client application submits the MapReduce job to Job Tracker.

* Hadoop Distributed File System (HDFS) →

The Hadoop Distributed File System (HDFS) is the primary data storage system used by Hadoop Application. HDFS employs a NameNode and DataNode architecture to implement a distributed file system that provides high-performance access to data across highly scalable Hadoop clusters.

Hadoop itself is an open source distributed processing framework that manages data processing and storage for big data applications. HDFS is a key part of the many Hadoop ecosystem technologies. It provides a reliable means for managing parts of big data and supporting related big data analytics applications.

⇒ Advantages of HDFS →

1. Hadoop framework allows the user to quickly write and test distributed systems. It is efficient and it automatically distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
2. Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
3. Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

* Features of HDFS :->

1. It is suitable for the distributed storage and processing.
2. Hadoop provides a Command interface to interact with HDFS.
3. The built-in servers of NameNode and DataNode help users to easily check the status of cluster.
4. Streaming access to file system data.
5. HDFS provides file permissions and authentication.

* Goals of HDFS :->

1. Fault detection and recovery :->

Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.

2. Huge dataset :->

HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.

3. Hardware at data :->

A requested task can be done efficiently when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.

* HDFS Architecture / Building blocks of Hadoop / Hadoop Daemons →

HDFS is responsible for storing data on the cluster in Hadoop. Files in HDFS are split into blocks before they are stored on a cluster of size 64 MB or 128 MB. On a fully configured cluster, "running Hadoop" means running a set of ~~daemons~~ resident programs, on the different servers in your network. These daemons have specific roles; some exist only on one server, some exist across multiple servers. The daemons include

1. NameNode
2. DataNode
3. Secondary NameNode
4. JobTracker
5. TaskTracker

1. NameNode →

Hadoop employs a master/slave architecture for both distributed storage and distributed computation. The distributed storage system is called the Hadoop File System, or HDFS.

The NameNode is the master of HDFS that directs the slave DataNode daemons to perform the low-level I/O tasks.

The NameNode is the bookkeeper of HDFS; it keeps track of how your files are broken down into file blocks, which nodes store those blocks, and the overall health of the distributed file system.

The function of the NameNode is memory and I/O intensive. As such, the server hosting the NameNode typically doesn't store

only store data or perform any Computations for a MapReduce program to lower the workload on the machine.

9. DataNode →

Each slave machine in your cluster will host a DataNode daemon to perform the grunt work of the distributed filesystem reading and writing HDFS blocks to actual files on the local filesystem. When we want to read or write a HDFS file, the file is broken into blocks and the NameNode will tell your client which DataNode each block resides in.

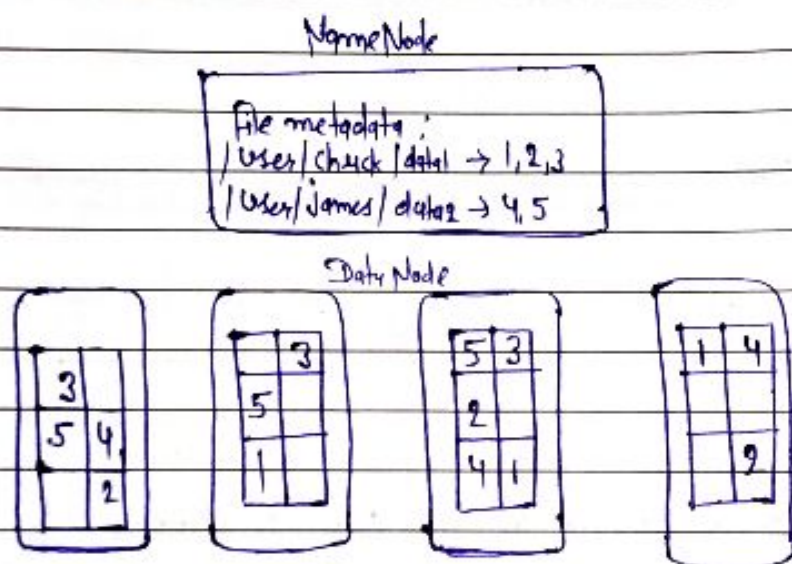


Figure :- NameNode / DataNode Interaction in HDFS

The above figure illustrates the role of the NameNode and DataNodes. In the figure, we show two data files, one at /user/chuck/data1 and another at /user/james/data2. The data1 file takes up three blocks, which we denote 1, 2 and 3 and the data2 file consists of block 4 and 5. The content of the files are distributed among the DataNodes. In the illustration, each block has three replicas.

for example, block 1 (used for data) is replicated over the three rightmost DataNodes. This ensures that if any one DataNode crashes or ~~becs~~ becomes inaccessible over the network, you'll still be able to read the files.

The NameNode keeps track of the file metadata - which files are in the system and how each file is broken down into blocks. The DataNodes provide backup store of the blocks and constantly report to the NameNode to keep the metadata current.

3. Secondary NameNode →

The Secondary NameNode (SNN) is an assistant dae daemon for monitoring the state of the cluster HDFS. Like the NameNode, each cluster has one SNN, and it typically resides on its own machine as well. No other DataNode or TaskTracker dae daemons run on the same server. The SNN differs from the NameNode in that this process doesn't receive or record any real-time changes to HDFS.

The NameNode is a single point of failure for a Hadoop cluster, and the SNN snapshots help minimize the downtime and loss of data.

4. Job Tracker →

The JobTracker daemon is the liaison between your application and Hadoop. once you submit your code to our cluster, the JobTracker determines the execution plan by determining which files to process, assigns nodes to different tasks, and monitors all tasks as they're running.

Should a task fail, the JobTracker will automatically resubmit the task, possibly on a different node, up to a predefined limit of retries. There is only one JobTracker daemon per Hadoop cluster. It's typically run on a server as a master node of the cluster.

5. TaskTracker →

The JobTracker is the master for the overall execution of a MapReduce job and the TaskTrackers manage the execution of individual tasks on each slave node.

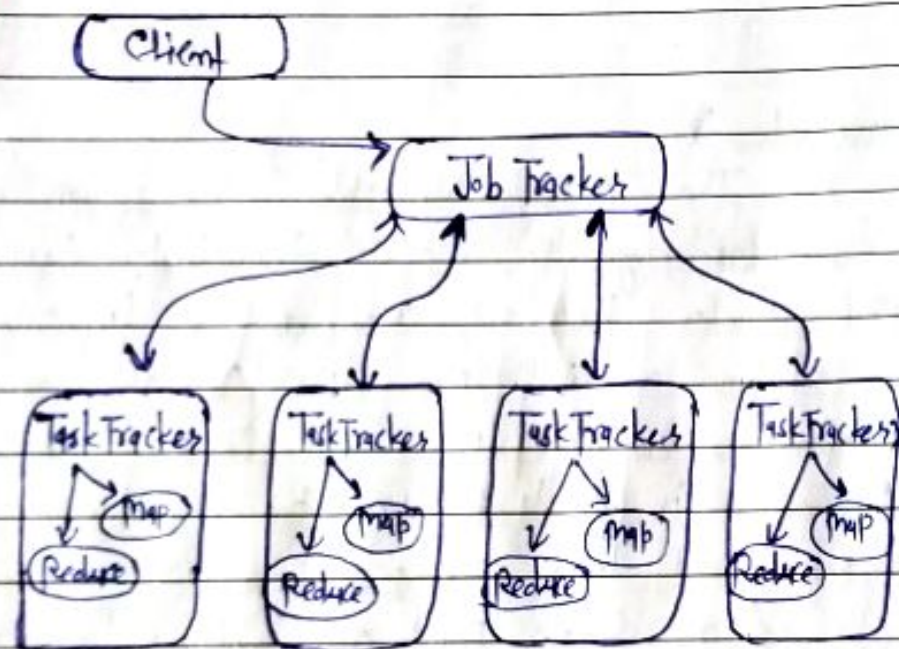


Figure: JobTracker and TaskTracker interaction

The above figure shows this interaction. Each TaskTracker is responsible for executing the individual tasks that the JobTracker assigns. Although there is a single TaskTracker per slave node, each TaskTracker can spawn multiple JVMs to handle many map or reduce tasks in parallel. One responsibility of the TaskTracker is to constantly communicate with the JobTracker.

If the JobTracker fails to receive a heartbeat from a TaskTracker within a specified amount of time, it will assume the TaskTracker has crashed and will resubmit the corresponding tasks to other nodes in the cluster.

After a client calls the JobTracker to begin a data processing job, the JobTracker partitions the work and assigns different map and reduce tasks to each TaskTracker in the cluster.

The following figure shows the topology of a typical Hadoop cluster. This topology features a master node running the NameNode and JobTracker daemons and a standalone node with the SNN in case the master node fails. For small clusters, the SNN can reside on one of the slave nodes.

On the other hand, for large clusters, separate the NameNode and JobTracker on two machines. The slave machines each host a DataNode and TaskTracker, for running tasks on the same node where their data is stored.

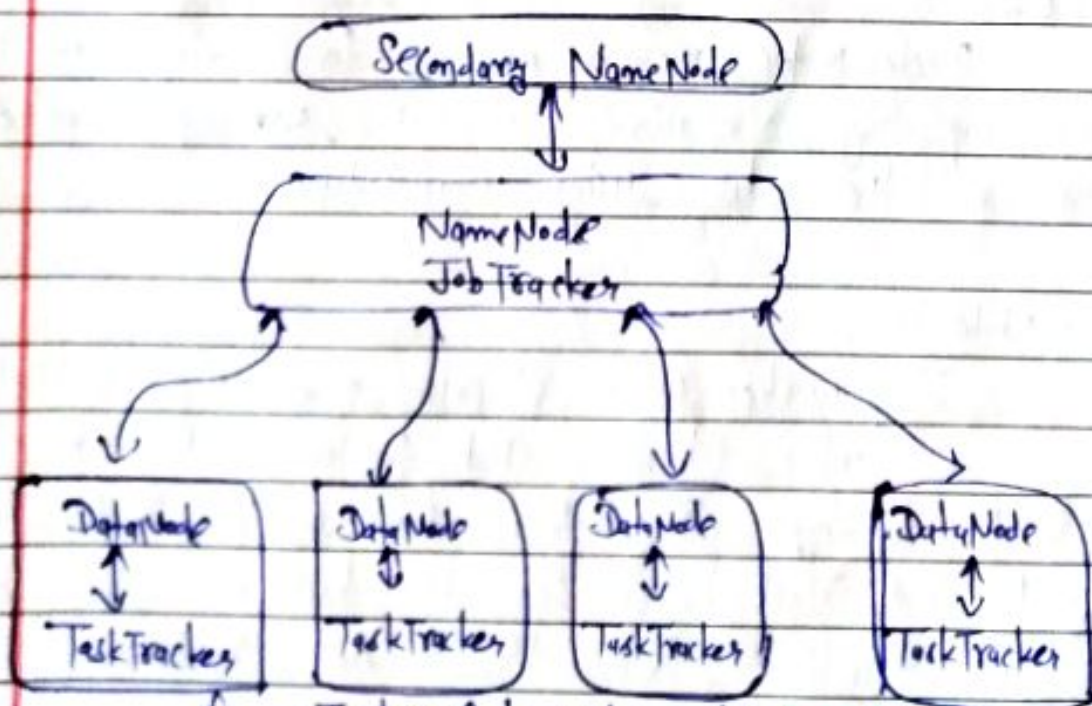


Figure 1: Topology of typical Hadoop cluster

Name of Lecturer

* Hadoop Cluster :->

- Apache Hadoop is an open sources, Java-based, software framework and parallel data processing engine.
- It enables big data analytics processing tasks to be broken down into smaller tasks that can be performed in parallel by using an algorithm (like the MapReduce algorithm) and distributing them across a Hadoop cluster.
- A Hadoop cluster is a collection of computers, known as nodes, that are networked together to perform these kinds of parallel computations on big data sets. Unlike other computer cluster, Hadoop clusters are designed specifically to store and analyze mass amounts of structured and unstructured data in distributed computing environment.

* Hadoop Cluster Architecture :->

Hadoop clusters are composed of a network of master and worker nodes that orchestrate and execute the various jobs across the Hadoop distributed file system.

(i) Master Nodes :->

Master nodes are responsible for storing data in HDFS and overseeing key operations, such as running parallel computations on the data using MapReduce.

(ii) The worker nodes :->

The worker nodes comprise most of the virtual machines in a Hadoop cluster, and perform the job of storing the data and running computations. Each worker node runs the DataNode and TaskTracker services, which are used to receive the instructions from the master nodes.

(3) Client Nodes' →

Client Nodes are in charge of loading the data into the cluster. Client nodes first submit MapReduce jobs describing how data needs to be processed, and then fetch the results once the processing is finished.

* Configuring Hadoop Cluster →

- The Majority of Hadoop Settings are Contained in XML Configuration files.
- In order to create a Hadoop cluster we need to configure several XML files.
- All the Configuration files are stored in conf directory of Hadoop-Home.
- In `hadoop - env.sh` define the `JAVA_HOME` Environment Variable to point to the Java installation directory.
- Before Version 0.20, these XML files are `hadoop - default.xml` and `hadoop - site.xml`.
- The `hadoop - default.xml` contains the default Hadoop settings to be used unless they are explicitly overridden in `hadoop - site.xml`.
- In Version 0.20 the `hadoop - site.xml` file has been separated out into three XML files:- `core-site.xml`, `hdfs-site.xml` and `mapred-site.xml`.
- A Hadoop cluster can be configured in one of the following 3 modes by modifying the above XML files.

- (1) Local (Standalone) Mode.
- (2) Pseudo-distributed mode.
- (3) Fully-distributed mode.

} optional operational modes of Hadoop

(b) Local (standalone) mode :->

- The Standalone mode is the default mode for Hadoop.
- When we first unzip the Hadoop source package, it does not consider our hardware setup. Hadoop chooses to be conservative and assumes a minimal configuration.
- All three XML files (or `hadoop-site.xml` before version 0.2.0) are empty under this default mode:

```
<?xml version="1.0"?>
```

```
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
```

```
<!-- Put site-specific property overrides in this file. -->
```

```
<configuration>
```

```
</configuration>
```

- Its primary use is for developing and debugging the application logic of a MapReduce program without the additional complexity of interacting with the daemons.

(c) Pseudo-distributed mode :->

- The pseudo-distributed mode is running Hadoop in a "cluster of one" with all daemons running on a single machine.
- This mode allows us to examine memory usage, HDFS I/O issues and other daemon interactions.
- Listing 2.1 provides sample XML files to configure a single server in the mode.

Listing 2.1 Example of the three configuration files for pseudo-distributed mode.

Core-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
    <description>The name of the default file system. A URI whose
      scheme and authority determine the filesystem implementation.
    </description>
  </property>
</configuration>
```

mapred-site.xml →

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:9001</value>
    <description>The host and port that the MapReduce job tracker runs
      at. </description>
  </property>
</configuration>
```

Name of Lecturer : _____

hdfs-site.xml →

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
    <description>The actual number of replications can be specified
      when the file is created. </description>
  </property>
</configuration>
```

- In Core-site.xml and mapred-site.xml we specify the hostname and port of the NameNode and the JobTracker, respectively.

Properties:

- (i) Configuration is required to given three files for this mode.
- (ii) This is used for local code to test in HDFS.
- (iii) This is a cluster, where all daemons are running on one node itself.

(3) Fully-Distributed Mode:

After continuously emphasizing the benefits of distributed storage and distributed computation, it's time for us to set up a full cluster. In the discussion below we'll use the following server names:-

mapred-site.xml :->

```
<?xml Version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>master:job</value>
    <description>The host and port that the MapReduce job tracker
      runs at. </description>
  </property>
</configuration>
```

hdfs-site.xml :->

```
<?xml Version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
    <description>The actual number of replications can be specified
      when the file is created. </description>
  </property>
</configuration>
```

Properties :->

- (1) This is a production phase.
- (2) Data are used and distributed across many nodes.
- (3) Different nodes will be used as master node/Data node etc.

* Data Measurement Chart :->

Data measurement	Data Size
Bit	1 or 0 (Single Binary Digit)
Byte	8 Bits
Kilobyte (KB)	1024 Bytes
Megabyte (MB)	1024 KB
Gigabyte (GB)	1024 MB
Exa Terabyte (TB)	1024 GB
Petabyte (PB)	1024 TB
Ext Exabyte (EB)	1024 PB
Zettabyte (ZB)	1024 EB
Yottabyte (YB)	1024 ZB