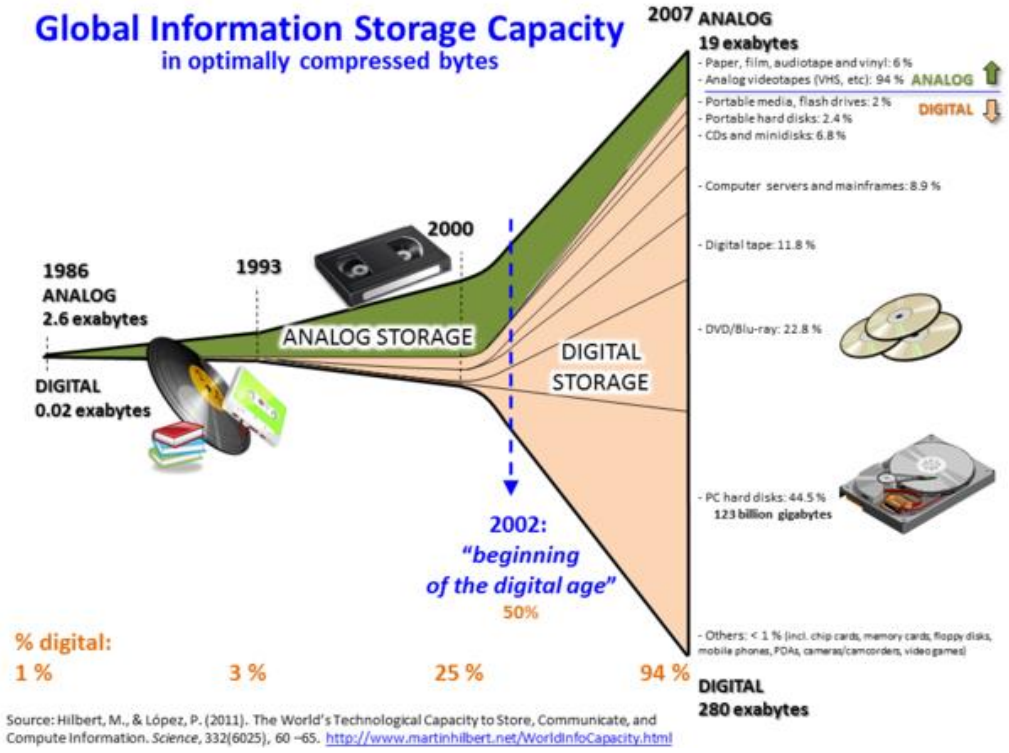
Big data

Big data primarily refers to data sets that are too large or complex to be dealt with by traditional data-processing application software. Data with many entries (rows) offer greater statistical power, while data with higher complexity (more attributes or columns) may lead to a higher false discovery rate. Though used sometimes loosely partly due to a lack of formal definition, the best interpretation is that it is a large body of information that cannot be comprehended when used in small amounts only.



Non-linear growth of digital global information-storage capacity and the waning of analog storage

Big data analysis challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating, information privacy, and data source. Big data was originally associated with three key concepts: *volume*, *variety*, and *velocity*.

The term *big data* has been in use since the 1990s, with some giving credit to John Mashey for popularizing the term.[22][23] Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time.[24] Big data philosophy encompasses unstructured, semi-structured and structured data; however, the main focus is on unstructured data.[25] Big data "size" is a constantly moving target; as of

2012 ranging from a few dozen terabytes to many zettabytes of data.[26] Big data requires a set of techniques and technologies with new forms of integration to reveal insights from data-sets that are diverse, complex, and of a massive scale.

"Variety", "veracity", and various other "Vs" are added by some organizations to describe it, a revision challenged by some industry authorities.[28] The Vs of big data were often referred to as the "three Vs", "four Vs", and "five Vs". They represented the qualities of big data in volume, variety, velocity, veracity, and value. Variability is often included as an additional quality of big data.

A 2018 definition states "Big data is where parallel computing tools are needed to handle data", and notes, "This represents a distinct and clearly defined change in the computer science used, via parallel programming theories, and losses of some of the guarantees and capabilities made by Codd's relational model."

In a comparative study of big datasets, Kitchin and McArdle found that none of the commonly considered characteristics of big data appear consistently across all of the analyzed cases.[30] For this reason, other studies identified the redefinition of power dynamics in knowledge discovery as the defining trait.[31] Instead of focusing on the intrinsic characteristics of big data, this alternative perspective pushes forward a relational understanding of the object claiming that what matters is the way in which data is collected, stored, made available and analyzed.

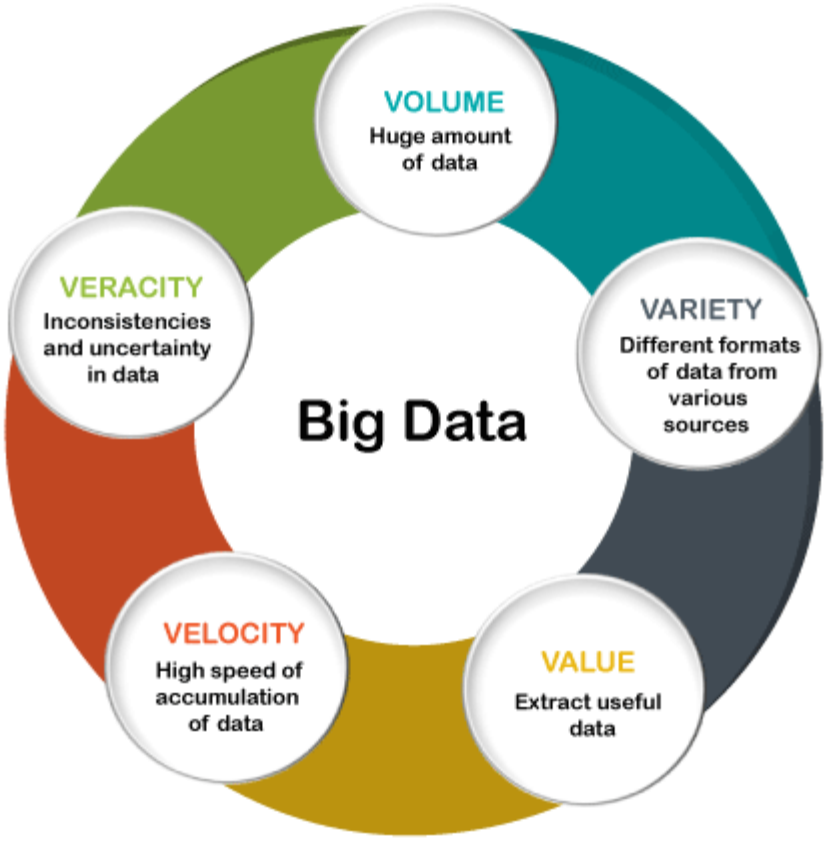
Big Data Characteristics

Big Data contains a large amount of data that is not being processed by traditional data storage or the processing unit. It is used by many **multinational companies** to **process** the data and business of many **organizations**. The data flow would exceed **150 exabytes** per day before replication.

There are five v's of Big Data that explains the characteristics.

5 V's of Big Data

- **Volume**
- **Veracity**
- **Variety**
- **Value**
- **Velocity**

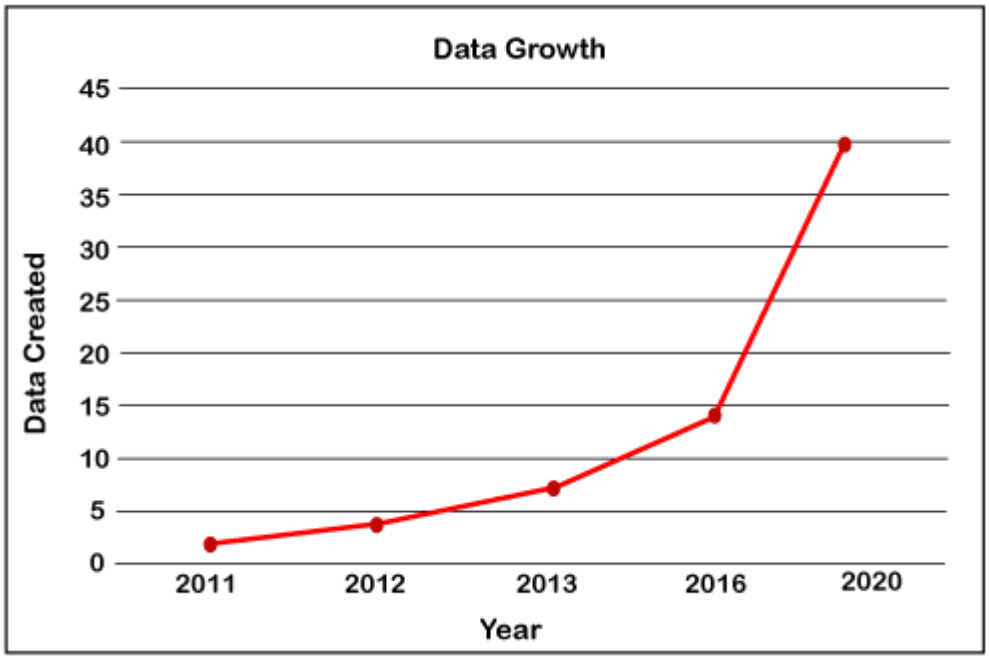


Volume

The name Big Data itself is related to an enormous size. Big Data is a vast 'volumes' of data generated from many sources daily, such as **business processes, machines, social media platforms, networks, human interactions,** and many more.

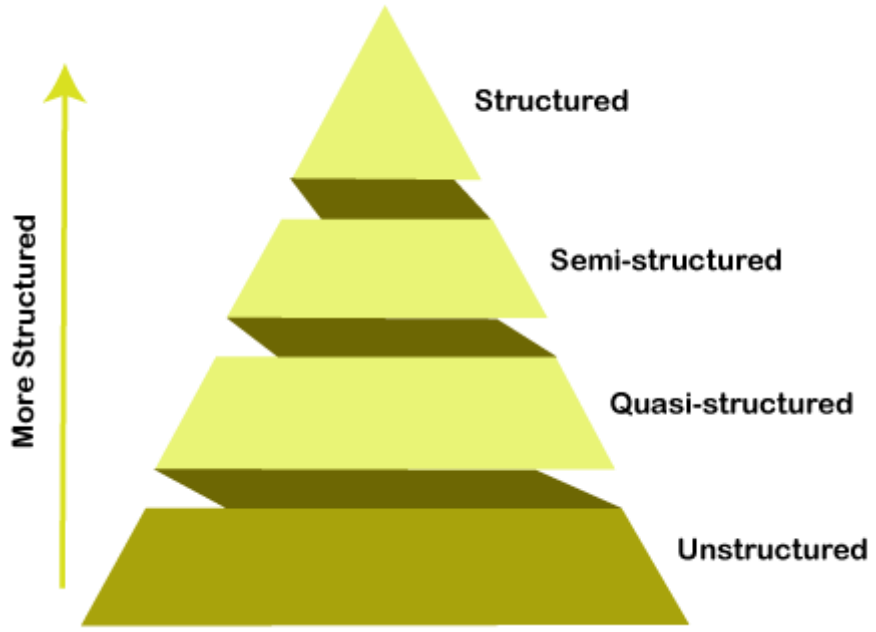
Facebook can generate approximately a **billion** messages, **4.5 billion** times that the "**Like**" button is recorded, and more than **350 million** new posts are uploaded each day. Big data technologies can handle large amounts of data.

Backward Skip 10sPlay VideoForward Skip 10s



Variety

Big Data can be **structured, unstructured, and semi-structured** that are being collected from different sources. Data will only be collected from **databases** and **sheets** in the past, But these days the data will comes in array forms, that are **PDFs, Emails, audios, SM posts, photos, videos,** etc.



The data is categorized as below:

a. **Structured data:** In Structured schema, along with all the required columns. It is in a tabular form. Structured Data is stored in the relational database management system.

b. **Semi-structured:** In Semi-structured, the schema is not appropriately defined, e.g., **JSON, XML, CSV, TSV**, and **email**. OLTP (**Online Transaction Processing**) systems are built to work with semi-structured data. It is stored in relations, i.e., **tables**.

c. **Unstructured Data**: All the **unstructured files, log files, audio files**, and **image** files are included in the unstructured data. Some organizations have much data available, but they did not know how to **derive** the value of data since the data is raw.

d. **Quasi-structured Data:**The data format contains textual data with inconsistent data formats that are formatted with effort and time with some tools.

Example: Web server logs, i.e., the log file is created and maintained by some server that contains a list of **activities**.

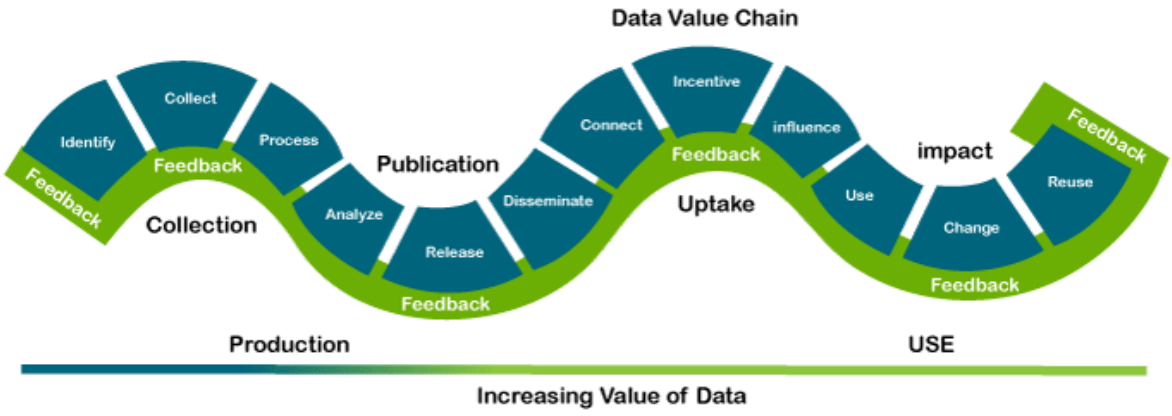
Veracity

Veracity means how much the data is reliable. It has many ways to filter or translate the data. Veracity is the process of being able to handle and manage data efficiently. Big Data is also essential in business development.

For example, **Facebook posts** with hashtags.

Value

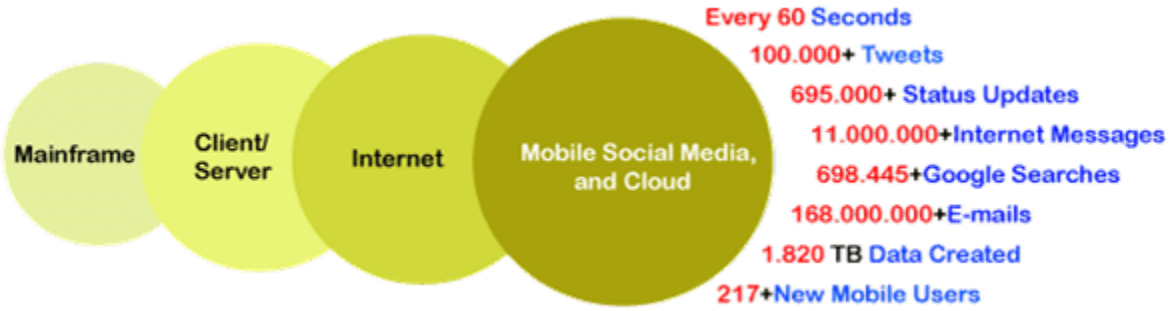
Value is an essential characteristic of big data. It is not the data that we process or store. It is **valuable** and **reliable** data that we **store, process**, and also **analyze**.



Velocity

Velocity plays an important role compared to others. Velocity creates the speed by which the data is created in **real-time**. It contains the linking of incoming **data sets speeds, rate of change**, and **activity bursts**. The primary aspect of Big Data is to provide demanding data rapidly.

Big data velocity deals with the speed at the data flows from sources like **application logs, business processes, networks, and social media sites, sensors, mobile devices,** etc.



Different Types of Big Data

Big data types in Big Data are used to categorize the numerous kinds of data generated daily. Primarily there are 3 types of data in analytics. The following types of Big Data with examples are explained below:-

1. Structured Data: Any data that can be processed, is easily accessible, and can be stored in a fixed format is called structured data. In Big Data, structured data is the easiest to work with because it has highly coordinated measurements that are defined by setting parameters. Structured types of Big Data are:-

- Address
- Age
- Credit/debit card numbers
- Contact
- Expenses
- Billing

2. Unstructured Data: Unstructured data in Big Data is where the data format constitutes multitudes of unstructured files (images, audio, log, and video). This form of data is classified as intricate data because of its unfamiliar structure and relatively huge size. A stark example of unstructured data is an output returned by 'Google Search' or 'Yahoo Search.'

3. Semi-structured Data: In Big Data, semi-structured data is a combination of both unstructured and structured types of data. This form of data constitutes the features of structured data but has unstructured information that does not adhere to any formal structure of data models or any relational database. Some semi-structured data examples include XML and JSON.

Subtypes of Data

This form of data does not typically fall under the umbrella of Big Data. However, certain subtypes of data are relevant in the field of analytics. This type of data often refers to the origin of data. They are classified as follows:-

- Machine (operational logging)

- Social media or event-triggered
- Geospatial (locational)

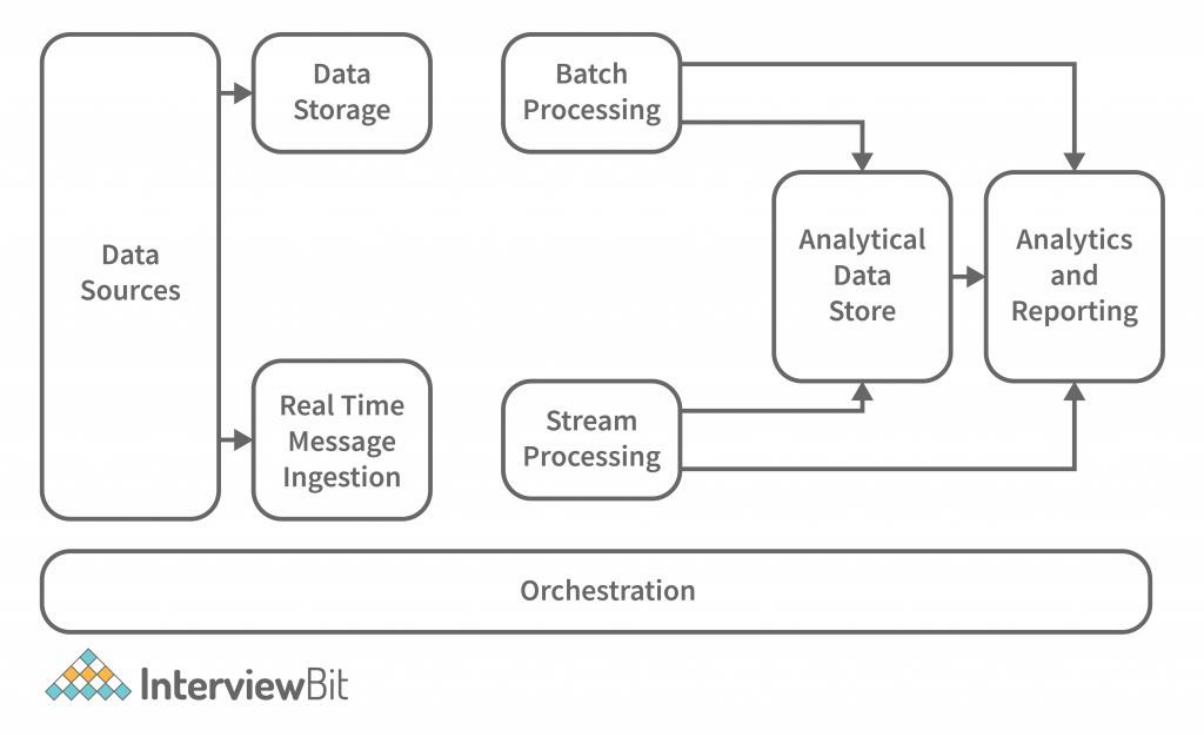
Data subtypes can fall under other access levels, such as:-

- Open (i.e. open source)
- Dark/lost (siloed systems such as CCTV)
- Linked (transmitted web data through APIs and other methods)

Big Data Architecture

There is more than one workload type involved in big data systems, and they are broadly classified as follows:

1. Merely batching data where big data-based sources are at rest is a data processing situation.
2. Real-time processing of big data is achievable with motion-based processing.
3. The exploration of new interactive big data technologies and tools.
4. The use of machine learning and predictive analysis.



- **Data Sources:** All of the sources that feed into the data extraction pipeline are subject to this definition, so this is where the starting point for the big data pipeline is located. Data sources, open and third-party, play a significant role in architecture. Relational databases, data warehouses, cloud-based data warehouses, SaaS applications, real-time data from company servers and sensors such as IoT devices, third-party data providers, and also static files such as Windows logs, comprise several data sources. Both batch processing and real-time processing are possible. The data managed can be both batch processing and real-time processing.

- **Data Storage:** There is data stored in file stores that are distributed in nature and that can hold a variety of format-based big files. It is also possible to store large numbers of different format-based big files in the data lake. This consists of the data that is managed for batch built operations and is saved in the file stores. We provide HDFS, Microsoft Azure, AWS, and GCP storage, among other blob containers.

- **Batch Processing:** Each chunk of data is split into different categories using long-running jobs, which filter and aggregate and also prepare data for analysis. These jobs typically require sources, process them, and deliver the processed files to new files. Multiple approaches to batch processing are employed, including Hive jobs, U-SQL jobs, Sqoop or Pig and custom map reducer jobs written in any one of the Java or Scala or other languages such as Python.

- **Real Time-Based Message Ingestion:** A real-time streaming system that caters to the data being generated in a sequential and uniform fashion is a batch processing system. When compared to batch processing, this includes all real-time streaming systems that cater to the data being generated at the time it is received. This data mart or store, which receives all incoming messages and discards them into a folder for data processing, is usually the only one that needs to be contacted. Message-based ingestion stores such as Apache Kafka, Apache Flume, Event hubs from Azure, and others, on the other hand, must be used if message-based processing is required. The delivery process, along with other message queuing semantics, is generally more reliable.

- **Stream Processing:** Real-time message ingest and stream processing are different. The latter uses the ingested data as a publish-subscribe tool, whereas the former takes into account all of the ingested data in the first place and then utilises it as a publish-subscribe tool. Stream processing, on the other hand, handles all of that streaming data in the form of windows or streams and writes it to the sink. This includes Apache Spark, Flink, Storm, etc.

- **Analytics-Based Datastore:** In order to analyze and process already processed data, analytical tools use the data store that is based on HBase or any other NoSQL data warehouse technology. The data can be presented with the help of a hive database, which can provide metadata abstraction, or interactive use of a hive database, which can provide metadata abstraction in the data store. NoSQL databases like HBase or Spark SQL are also available.

- **Reporting and Analysis:** The generated insights, on the other hand, must be processed and that is effectively accomplished by the reporting and analysis tools that utilize embedded technology and a solution to produce useful graphs, analysis, and insights that are beneficial to the businesses. For example, Cognos, Hyperion, and others.

- **Orchestration:** Data-based solutions that utilise big data are data-related tasks that are repetitive in nature, and which are also contained in workflow chains that can transform the source data and also move data across sources as well as sinks and loads in stores. Sqoop, oozie, data factory, and others are just a few examples.

Types of Big Data Architecture

- Lambda Architecture
- Kappa Architecture

Map-Reduce and Hadoop

Hadoop

Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume. Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing.It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Moreover it can be scaled up just by adding nodes in the cluster.

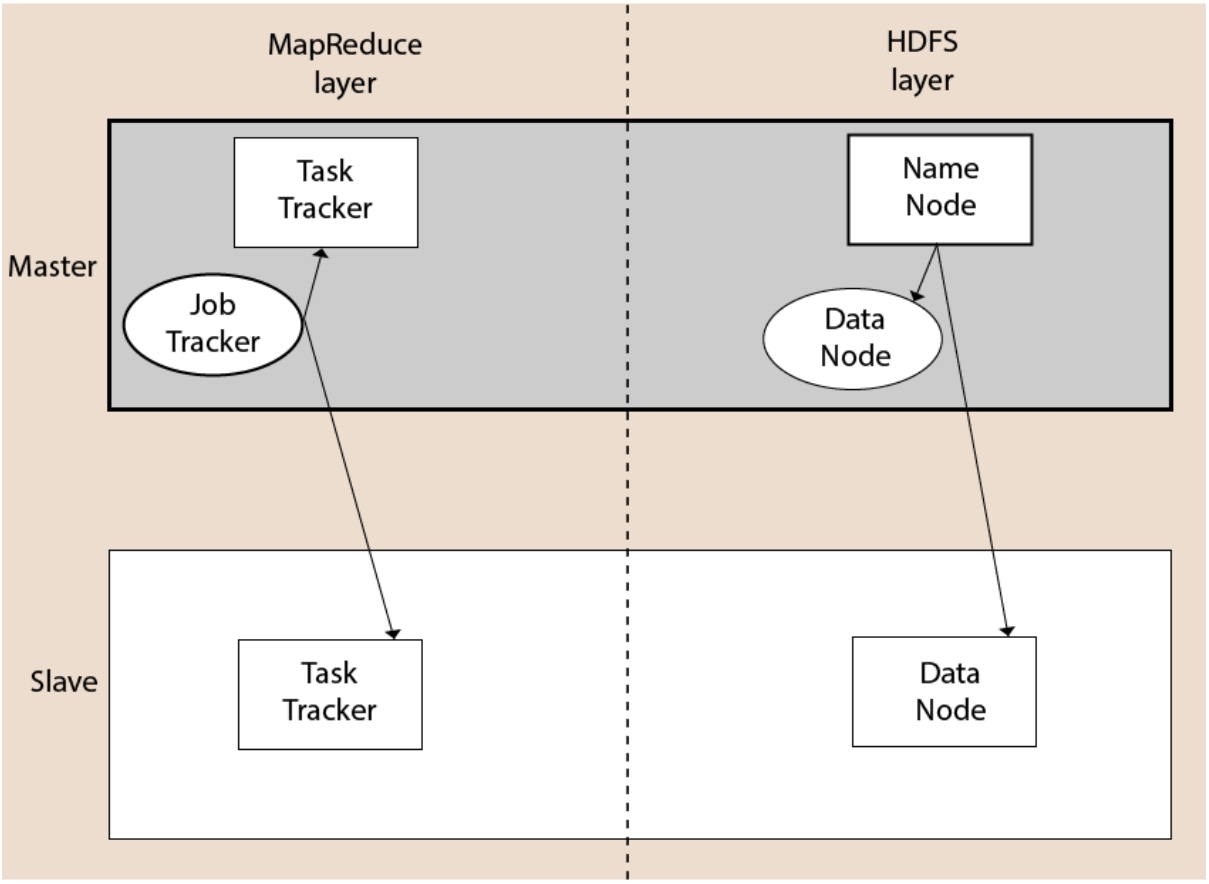
Modules of Hadoop

1. **HDFS:** Hadoop Distributed File System. Google published its paper GFS and on the basis of that HDFS was developed. It states that the files will be broken into blocks and stored in nodes over the distributed architecture.
2. **Yarn:** Yet another Resource Negotiator is used for job scheduling and manage the cluster.
3. **Map Reduce:** This is a framework which helps Java programs to do the parallel computation on data using key value pair. The Map task takes input data and converts it into a data set which can be computed in Key value pair. The output of Map task is consumed by reduce task and then the out of reducer gives the desired result.
4. **Hadoop Common:** These Java libraries are used to start Hadoop and are used by other Hadoop modules.

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. The master node includes Job Tracker, Task Tracker, NameNode, and DataNode whereas the slave node includes DataNode and TaskTracker.



Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. It contains a master/slave architecture. This architecture consist of a single NameNode performs the role of master, and multiple DataNodes performs the role of a slave.

Both NameNode and DataNode are capable enough to run on commodity machines. The Java language is used to develop HDFS. So any machine that supports Java language can easily run the NameNode and DataNode software.

NameNode

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

DataNode

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

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

Job Tracker

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

Task Tracker

- It works as a slave node for Job Tracker.
- It receives task and code from Job Tracker and applies that code on the file. This process can also be called as a Mapper.

MapReduce Layer

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

Advantages of Hadoop

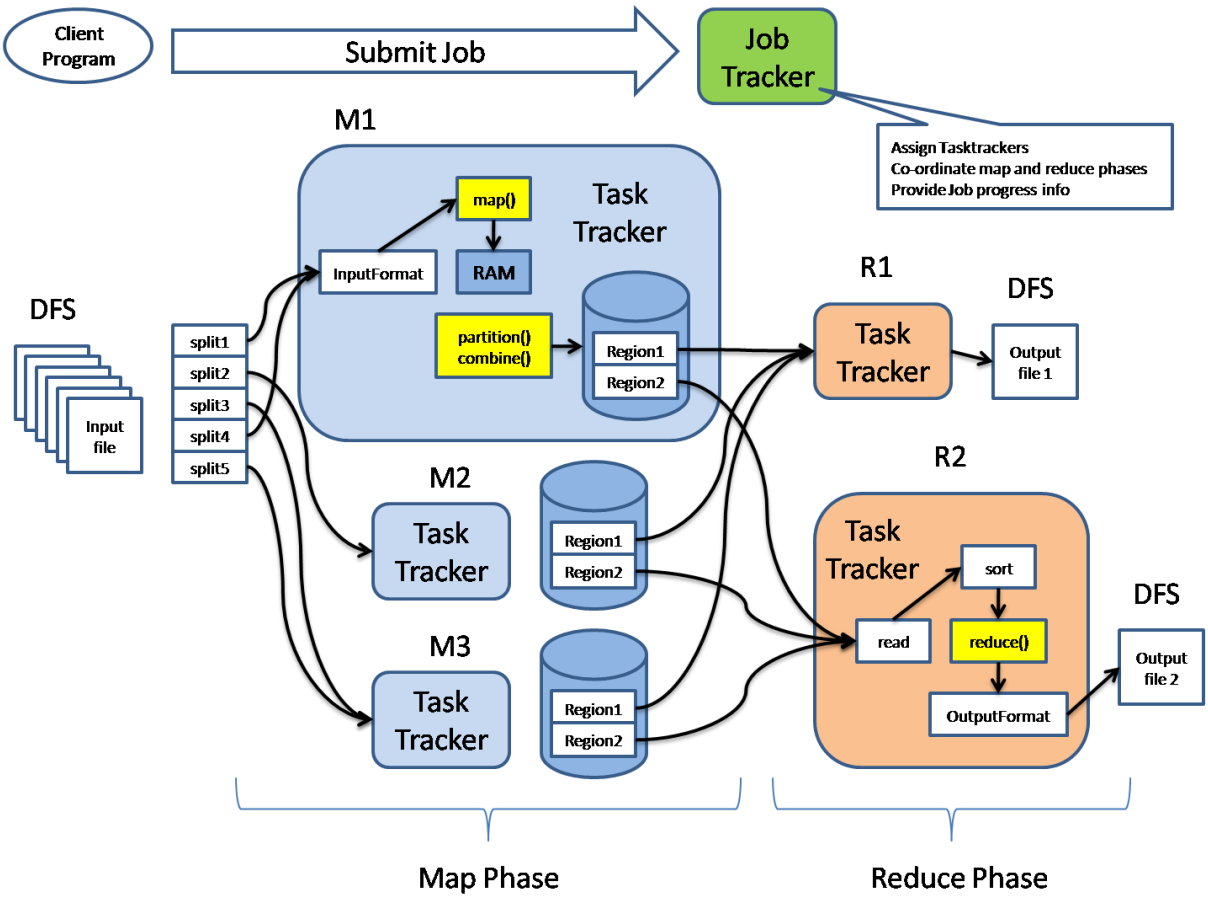
- **Fast:** In HDFS the data distributed over the cluster and are mapped which helps in faster retrieval. Even the tools to process the data are often on the same servers, thus reducing the processing time. It is able to process terabytes of data in minutes and Peta bytes in hours.
- **Scalable:** Hadoop cluster can be extended by just adding nodes in the cluster.
- **Cost Effective:** Hadoop is open source and uses commodity hardware to store data so it really cost effective as compared to traditional relational database management system.
- **Resilient to failure:** HDFS has the property with which it can replicate data over the network, so if one node is down or some other network failure happens, then Hadoop takes the other copy of data and use it. Normally, data are replicated thrice but the replication factor is configurable.

MapReduce: MapReduce is a data processing tool which is used to process the data parallelly in a distributed form. It was developed in 2004, on the basis of paper titled as "MapReduce: Simplified Data Processing on Large Clusters," published by Google.

The MapReduce is a paradigm which has two phases, the mapper phase, and the reducer phase. In the Mapper, the input is given in the form of a key-value pair. The output of the Mapper is fed to the reducer as input. The reducer runs only after the Mapper is over. The reducer too takes input in key-value format, and the output of reducer is the final output.

Steps in Map Reduce

- The map takes data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.
- Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.
- An output of sort and shuffle sent to the reducer phase. The reducer performs a defined function on a list of values for unique keys, and Final output <key, value> will be stored/displayed.



Sort and Shuffle

The sort and shuffle occur on the output of Mapper and before the reducer. When the Mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk. Using the input from each Mapper <k2,v2>, we collect all the values for each unique key k2. This output from the shuffle phase in the form of <k2, list(v2)> is sent as input to reducer phase.

Usage of MapReduce

- It can be used in various application like document clustering, distributed sorting, and web link-graph reversal.
- It can be used for distributed pattern-based searching.
- We can also use MapReduce in machine learning.
- It was used by Google to regenerate Google's index of the World Wide Web.
- It can be used in multiple computing environments such as multi-cluster, multi-core, and mobile environment.

Distributed File System, HDFS

Distributed File System

Distributed File System (**DFS**) is a set of client and server services that allow an organization using Microsoft Windows servers to organize many distributed SMB file shares into a distributed file system. DFS has two components to its service: Location transparency (via the namespace component) and Redundancy (via the file replication component). Together, these components enable data availability in the case of failure or heavy load by allowing shares in multiple different locations to be logically grouped under one folder, the "DFS root".

HDFS

Apache Hadoop is a collection of open-source software utilities that facilitates using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model. Hadoop was originally designed for computer clusters built from commodity hardware, which is still the common use. It has since also found use on clusters of higher-end hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part which is a MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality,[7] where nodes manipulate the data they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

The base Apache Hadoop framework is composed of the following modules:

- *Hadoop Common* – contains libraries and utilities needed by other Hadoop modules;
- *Hadoop Distributed File System (HDFS)* – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
- *Hadoop YARN* – (introduced in 2012) is a platform responsible for managing computing resources in clusters and using them for scheduling users' applications;[10][11]
- *Hadoop MapReduce* – an implementation of the MapReduce programming model for large-scale data processing.

- *Hadoop Ozone* – (introduced in 2020) An object store for Hadoop

File systems

Hadoop distributed file system

The *Hadoop distributed file system* (HDFS) is a distributed, scalable, and portable file system written in Java for the Hadoop framework. Some consider it to instead be a data store due to its lack of POSIX compliance,[29] but it does provide shell commands and Java application programming interface (API) methods that are similar to other file systems.[30] A Hadoop instance is divided into HDFS and MapReduce. HDFS is used for storing the data and MapReduce is used for processing data. HDFS has five services as follows:

1. Name Node
2. Secondary Name Node
3. Job tracker
4. Data Node
5. Task Tracker

Top three are Master Services/Daemons/Nodes and bottom two are Slave Services. Master Services can communicate with each other and in the same way Slave services can communicate with each other. Name Node is a master node and Data node is its corresponding Slave node and can talk with each other.

Name Node: HDFS consists of only one Name Node that is called the Master Node. The master node can track files, manage the file system and has the metadata of all of the stored data within it. In particular, the name node contains the details of the number of blocks, locations of the data node that the data is stored in, where the replications are stored, and other details. The name node has direct contact with the client.

Data Node: A Data Node stores data in it as blocks. This is also known as the slave node and it stores the actual data into HDFS which is responsible for the client to read and write. These are slave daemons. Every Data node sends a Heartbeat message to the Name node every 3 seconds and conveys that it is alive. In this way when Name Node does not receive a heartbeat from a data node for 2 minutes, it will take that data node as dead and starts the process of block replications on some other Data node.

Secondary Name Node: This is only to take care of the checkpoints of the file system metadata which is in the Name Node. This is also known as the checkpoint Node. It is the helper Node for the Name Node. The secondary name node instructs the name node to create & send fsimage & editlog file, upon which the compacted fsimage file is created by the secondary name node.[31]

Job Tracker: Job Tracker receives the requests for Map Reduce execution from the client. Job tracker talks to the Name Node to know about the location of the data that will be used in processing. The Name Node responds with the metadata of the required processing data.

Task Tracker: It is the Slave Node for the Job Tracker and it will take the task from the Job Tracker. It also receives code from the Job Tracker. Task Tracker will take the code and apply on the file. The process of applying that code on the file is known as Mapper.

HDFS Architecture