**UNIT – 3 BIG DATA**

**What is Hadoop?**

Hadoop is an open source software programming framework for storing a large amount of data and performing the computation. Its framework is based on Java programming with some native code in C and shell scripts.

Hadoop is an open-source software framework that is used for storing and processing large amounts of data in a distributed computing environment. It is designed to handle big data and is based on the MapReduce programming model, which allows for the parallel processing of large datasets.

**Hadoop has two main components:**

* HDFS (Hadoop Distributed File System): This is the storage component of Hadoop, which allows for the storage of large amounts of data across multiple machines. It is designed to work with commodity hardware, which makes it cost-effective.
* YARN (Yet Another Resource Negotiator): This is the resource management component of Hadoop, which manages the allocation of resources (such as CPU and memory) for processing the data stored in HDFS.
* Hadoop also includes several additional modules that provide additional functionality, such as Hive (a SQL-like query language), Pig (a high-level platform for creating MapReduce programs), and HBase (a non-relational, distributed database).
* Hadoop is commonly used in big data scenarios such as data warehousing, business intelligence, and machine learning. It’s also used for data processing, data analysis, and data mining. It enables the distributed processing of large data sets across clusters of computers using a simple programming model.

**History of Hadoop**

**Apache Software Foundation** is the developers of Hadoop, and it’s co-founders are **Doug Cutting** and **Mike Cafarella**. It’s co-founder Doug Cutting named it on his son’s toy elephant. In October 2003 the first paper release was Google File System. In January 2006, MapReduce development started on the Apache Nutch which consisted of around 6000 lines coding for it and around 5000 lines coding for HDFS. In April 2006 Hadoop 0.1.0 was released.

Hadoop is an open-source software framework for storing and processing big data. It was created by Apache Software Foundation in 2006, based on a white paper written by Google in 2003 that described the Google File System (GFS) and the MapReduce programming model. The Hadoop framework allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. It is used by many organizations, including Yahoo, Facebook, and IBM, for a variety of purposes such as data warehousing, log processing, and research. Hadoop has been widely adopted in the industry and has become a key technology for big data processing.

**Features of hadoop:**

1. it is fault tolerance.

2. it is highly available.

3. it’s programming is easy.

4. it have huge flexible storage.

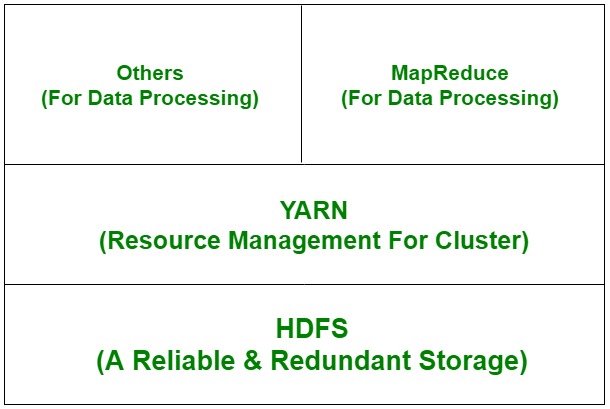
5. it is low cost.

**Hadoop has several key features that make it well-suited for big data processing:**

* Distributed Storage: Hadoop stores large data sets across multiple machines, allowing for the storage and processing of extremely large amounts of data.
* Scalability: Hadoop can scale from a single server to thousands of machines, making it easy to add more capacity as needed.
* Fault-Tolerance: Hadoop is designed to be highly fault-tolerant, meaning it can continue to operate even in the presence of hardware failures.
* Data locality: Hadoop provides data locality feature, where the data is stored on the same node where it will be processed, this feature helps to reduce the network traffic and improve the performance
* High Availability: Hadoop provides High Availability feature, which helps to make sure that the data is always available and is not lost.
* Flexible Data Processing: Hadoop’s MapReduce programming model allows for the processing of data in a distributed fashion, making it easy to implement a wide variety of data processing tasks.
* Data Integrity: Hadoop provides built-in checksum feature, which helps to ensure that the data stored is consistent and correct.
* Data Replication: Hadoop provides data replication feature, which helps to replicate the data across the cluster for fault tolerance.
* Data Compression: Hadoop provides built-in data compression feature, which helps to reduce the storage space and improve the performance.
* YARN: A resource management platform that allows multiple data processing engines like real-time streaming, batch processing, and interactive SQL, to run and process data stored in HDFS.

**Hadoop Distributed File System**

It has distributed file system known as HDFS and this HDFS splits files into blocks and sends them across various nodes in form of large clusters. Also in case of a node failure, the system operates and data transfer takes place between the nodes which are facilitated by HDFS.

[](https://media.geeksforgeeks.org/wp-content/uploads/had.jpg)

*HDFS*

**Advantages of HDFS:** It is inexpensive, immutable in nature, stores data reliably, ability to tolerate faults, scalable, block structured, can process a large amount of data simultaneously and many more. **Disadvantages of HDFS:** It’s the biggest disadvantage is that it is not fit for small quantities of data. Also, it has issues related to potential stability, restrictive and rough in nature. Hadoop also supports a wide range of software packages such as Apache Flumes, Apache Oozie, Apache HBase, Apache Sqoop, Apache Spark, Apache Storm, Apache Pig, Apache Hive, Apache Phoenix, Cloudera Impala.

**Some common frameworks of Hadoop**

1. Hive- It uses HiveQl for data structuring and for writing complicated MapReduce in HDFS.
2. Drill- It consists of user-defined functions and is used for data exploration.
3. Storm- It allows real-time processing and streaming of data.
4. Spark- It contains a Machine Learning Library(MLlib) for providing enhanced machine learning and is widely used for data processing. It also supports Java, Python, and Scala.
5. Pig- It has Pig Latin, a SQL-Like language and performs data transformation of unstructured data.
6. Tez- It reduces the complexities of Hive and Pig and helps in the running of their codes faster.

Hadoop framework is made up of the following modules:

1. Hadoop MapReduce- a MapReduce programming model for handling and processing large data.
2. Hadoop Distributed File System- distributed files in clusters among nodes.
3. Hadoop YARN- a platform which manages computing resources.
4. Hadoop Common- it contains packages and libraries which are used for other modules.

**Advantages and Disadvantages of Hadoop**

**Advantages:**

* Ability to store a large amount of data.
* High flexibility.
* Cost effective.
* High computational power.
* Tasks are independent.
* Linear scaling.

**Hadoop has several advantages that make it a popular choice for big data processing:**

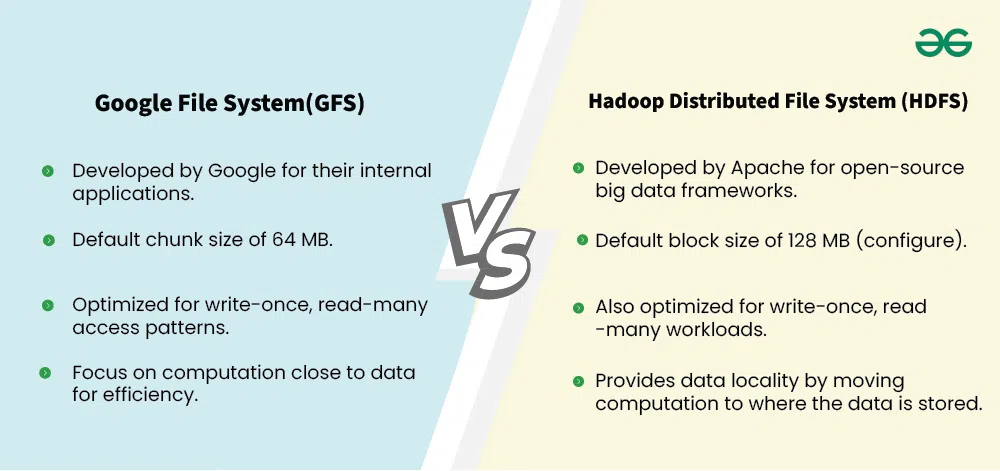
* Scalability: Hadoop can easily scale to handle large amounts of data by adding more nodes to the cluster.
* Cost-effective: Hadoop is designed to work with commodity hardware, which makes it a cost-effective option for storing and processing large amounts of data.
* Fault-tolerance: Hadoop’s distributed architecture provides built-in fault-tolerance, which means that if one node in the cluster goes down, the data can still be processed by the other nodes.
* Flexibility: Hadoop can process structured, semi-structured, and unstructured data, which makes it a versatile option for a wide range of big data scenarios.
* Open-source: Hadoop is open-source software, which means that it is free to use and modify. This also allows developers to access the source code and make improvements or add new features.
* Large community: Hadoop has a large and active community of developers and users who contribute to the development of the software, provide support, and share best practices.
* Integration: Hadoop is designed to work with other big data technologies such as Spark, Storm, and Flink, which allows for integration with a wide range of data processing and analysis tools.

**Disadvantages:**

* Not very effective for small data.
* Hard cluster management.
* Has stability issues.
* Security concerns.
* Complexity: Hadoop can be complex to set up and maintain, especially for organizations without a dedicated team of experts.
* Latency: Hadoop is not well-suited for low-latency workloads and may not be the best choice for real-time data processing.
* Limited Support for Real-time Processing: Hadoop’s batch-oriented nature makes it less suited for real-time streaming or interactive data processing use cases.
* Limited Support for Structured Data: Hadoop is designed to work with unstructured and semi-structured data, it is not well-suited for structured data processing
* Data Security: Hadoop does not provide built-in security features such as data encryption or user authentication, which can make it difficult to secure sensitive data.
* Limited Support for Ad-hoc Queries: Hadoop’s MapReduce programming model is not well-suited for ad-hoc queries, making it difficult to perform exploratory data analysis.
* Limited Support for Graph and Machine Learning: Hadoop’s core component HDFS and MapReduce are not well-suited for graph and machine learning workloads, specialized components like Apache Graph and Mahout are available but have some limitations.
* Cost: Hadoop can be expensive to set up and maintain, especially for organizations with large amounts of data.
* Data Loss: In the event of a hardware failure, the data stored in a single node may be lost permanently.
* Data Governance: Data Governance is a critical aspect of data management, Hadoop does not provide a built-in feature to manage data lineage, data quality, data cataloging, data lineage, and data audit.

**Google File System(GFS) vs. Hadoop Distributed File System (HDFS)**

In distributed file systems, Google File System (GFS) and[Hadoop Distributed File System (HDFS)](https://www.geeksforgeeks.org/hadoop-hdfs-hadoop-distributed-file-system/) stand out as crucial technologies. Both are designed to handle large-scale data, but they cater to different needs and environments. In this article, we will understand the differences between them.

  
 Google File System(GFS) vs. Hadoop Distributed File System (HDFS)

**What is Google File System (GFS)?**

**Google File System (GFS)** is a distributed file system designed by Google to handle large-scale data storage across multiple machines while providing high [reliability](https://www.geeksforgeeks.org/reliability-in-system-design/) and performance.

* It was developed to meet the needs of Google's massive data processing and storage requirements, particularly for its search engine and other large-scale applications.
* GFS is optimized for storing and processing very large files (in the range of gigabytes or terabytes) and supports high-throughput data operations rather than low-latency access.

**Key Features of Google File System(GFS)**

Below are the key features of Google File System(GFS):

* [**Scalability**](https://www.geeksforgeeks.org/what-is-scalability-and-how-to-achieve-it-learn-system-design/): GFS can scale to thousands of storage nodes and manage petabytes of data.
* [**Fault Tolerance**](https://www.geeksforgeeks.org/fault-tolerance-in-distributed-system/): Data is replicated across multiple machines, ensuring reliability even in case of hardware failures.
* **High**[**Throughput**](https://www.geeksforgeeks.org/latency-in-system-design/): It’s optimized for large data sets and supports concurrent read and write operations.
* **Chunk-based Storage**: Files are divided into fixed-size chunks (usually 64 MB) and distributed across many machines.
* **Master and Chunkserver Architecture**: GFS employs a master server that manages metadata and multiple chunkservers that store the actual data.

**What is Hadoop Distributed File System (HDFS)?**

[**Hadoop Distributed File System (HDFS)**](https://www.geeksforgeeks.org/hadoop-hdfs-hadoop-distributed-file-system/) is a open source distributed file system inspired by GFS and is designed to store large amounts of data across a cluster of machines, ensuring [fault tolerance](https://www.geeksforgeeks.org/fault-tolerance-in-distributed-system/) and [scalability](https://www.geeksforgeeks.org/what-is-scalability-and-how-to-achieve-it-learn-system-design/). It is a core component of the Apache Hadoop ecosystem and is designed to handle large-scale data processing jobs such as those found in big data environments.

**Key Features of Hadoop Distributed File System (HDFS)**

Below are the key features of Hadoop Distributed File System:

* [**Distributed Architecture**](https://www.geeksforgeeks.org/what-is-a-distributed-system/): HDFS stores files across a distributed cluster of machines.
* [**Fault Tolerance**](https://www.geeksforgeeks.org/fault-tolerance-in-distributed-system/): Data is replicated across multiple nodes, ensuring that the system can recover from failures.
* [**Master-Slave Architecture**](https://www.geeksforgeeks.org/master-slave-architecture/): HDFS consists of a single master node (NameNode) that manages metadata and multiple slave nodes (DataNodes) that store actual data.
* **Large Block Size**: HDFS breaks files into large blocks (default 128 MB or 64 MB) to optimize read/write operations for large datasets.
* **Write Once, Read Many**: HDFS is optimized for workloads that involve writing files once and reading them multiple times

**Google File System(GFS) vs. Hadoop Distributed File System (HDFS)**

Below are the key differences between Google File System and Hadoop Distributed File System:

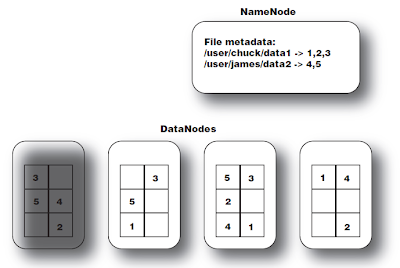
| **Aspect** | **Google File System (GFS)** | **Hadoop Distributed File System (HDFS)** |
| --- | --- | --- |
| **Origin** | Developed by Google for their internal applications. | Developed by Apache for open-source big data frameworks. |
| **Architecture** | Master-slave architecture with a single master (GFS master) and chunkservers. | Master-slave architecture with a NameNode and DataNodes. |
| **Block/Chunk Size** | Default chunk size of 64 MB. | Default block size of 128 MB (configurable). |
| **Replication Factor** | Default replication is 3 copies. | Default replication is 3 copies (configurable) |
| **File Access Pattern** | Optimized for write-once, read-many access patterns. | Also optimized for write-once, read-many workloads. |
| **Fault Tolerance** | Achieves fault tolerance via data replication across multiple chunkservers. | Achieves fault tolerance via data replication across multiple DataNodes. |
| **Data Integrity** | Uses checksums to ensure data integrity. | Uses checksums to ensure data integrity. |
| **Data Locality** | Focus on computation close to data for efficiency. | Provides data locality by moving computation to where the data is stored. |
| **Cost Efficiency** | Designed to run on commodity hardware. | Also designed to run on commodity hardware. |

**The building blocks of Hadoop**

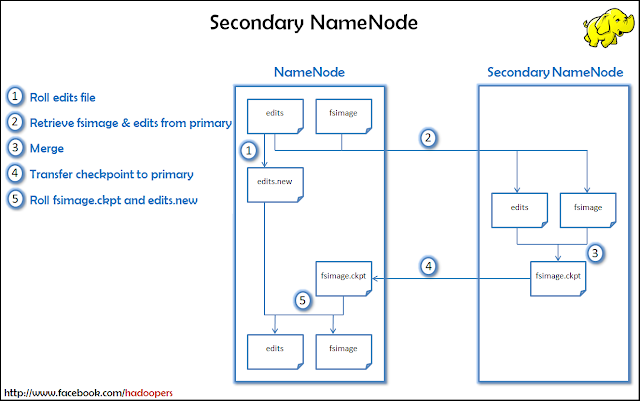
Hadoop employs a master/slave architecture for both distributed storage and distributed computation. The distributed storage system is called the Hadoop Distributed File System (HDFS).  
  
On a fully configured cluster, "running Hadoop" means running a set of daemons, or resident programs, on the different servers in you network. These daemons have specific roles; some exists only on one server, some exist across multiple servers. The daemons include:

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

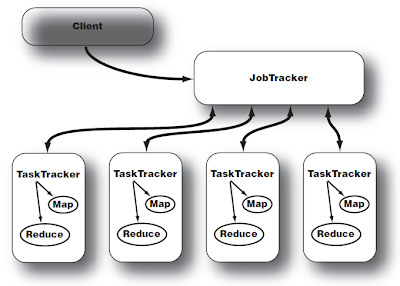
**NameNode:**The NameNode is the *master*of HDFS that directs the *slave*DataNode daemons to perform the low-level I/O tasks. It is the bookkeeper of HDFS; it keeps track of how your files are broken down into file blocks, which nodes store those blocks and the overall health of the distributed filesystem.  
  
The server hosting the NameNode typically doesn't store any user data or perform any computations for a MapReduce program to lower the workload on the machine, hence memory & I/O intensive.  
  
There is unfortunately a negative aspect to the importance of the NameNode - it's a single point of failure of your Hadoop cluster. For any of the other daemons, if their host fail for software or hardware reasons, the Hadoop cluster will likely continue to function smoothly or you can quickly restart it. Not so for the NameNode.  
  
**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 file system  
  
When you 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. Your client communicates directly with the DataNode daemons to process the local files corresponding to the blocks.  
  
Furthermore, a DataNode may communicate with other DataNodes to *replicate*its data blocks for redundancy.This ensures that if any one DataNode crashes or becomes inaccessible over the network, you'll still able to read the files.  
  
DataNodes are constantly reporting to the NameNode. Upon initialization, each of the DataNodes informs the NameNode of the blocks it's currently storing. After this mapping is complete, the DataNodes continually poll the NameNode to provide information regarding local changes as well as receive instructions to create, move, or delete from the local disk.

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEi2teoP3c1Q2Xb6sKCOi49B7E1kogG0nPdRlwc1KpNEtWM6K4E8D87g0NfONbqFSjWhIUq8U58TXakLoaT2i65uE2eM_hrvC9cl7DoaVav4UPDRsymERw9f47DDDZzPIVl9NQ-4-VAxeG0/s1600/NameData_Node+Image.png)

**Secondary NameNode(SNN):**The SNN is an assistant 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 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. Instead, it communicates with the NameNode to take snapshots of the HDFS metadata at intervals defined by the cluster configuration.  
  
As mentioned earlier, the NameNode is a single point of failure for a Hadoop cluster, and the SNN snapshots help minimize the downtime and loss of data.  
  
There is another topic which can be covered under SNN, i.e., fsimage(filesystem image) file and edits file:  
  
The HDFS namespace is stored by the NameNode. The NameNode uses a transaction log called the **EditLog**to persistently record every change that occurs to file system metadata. For example, creating a new file in HDFS causes the NameNode to insert a record into the EditLog indicating this. Similarly, changing the replication factor of a file causes a new record to be inserted into the EditLog. The NameNode uses a file in its local host OS file system to store the EditLog. The entire file system namespace, including the mapping of blocks to files and file system properties, is stored in a file called the **FsImage**. The FsImage is stored as a file in the NameNode’s local file system too.

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEiznRkuDmSiMJn1ME36RGtaZzw_UPVdcplTlwdcqpOHR3oSIJtv6qEkaRphpv-OAB0NVwHqLr1_mvQuLJ4cWckth4bZjUWh9N140kDtQMLyAETgblfsCzj7eRflZJa9VS-HGGRk3tlZvQ0/s1600/Lesson+10+-+Secondary+NameNode+Detailed.png)

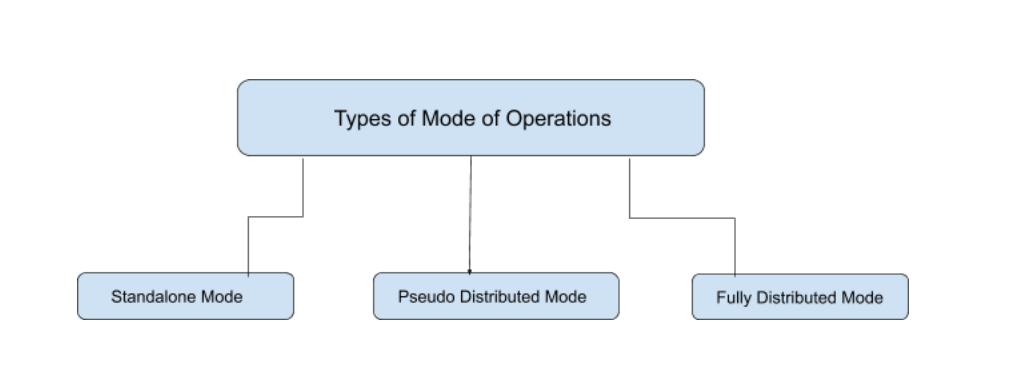
**JobTracker:**Once you submit your code to your cluster, the JobTracker determines the execution plan by determining which files to process, assigns nodes to different tasks, and monitors all tasks as they're running. should a task fail, the JobTracker will automatically relaunch the task, possibly 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.  
  
**TaskTracker:**As with the storage daemons, the computing daemons also follow a master/slave architecture: the JobTracker is the master overseeing the overall execution of a MapReduce job and the TaskTracker manage the execution of individual tasks on each slave node.  
  
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.

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjdtKmfTEOivsYgjEwz7cKesCCK9AN7Ju5tGjjZgLC8Aden28kXoKETohS9WcDogbDl2rdAqX7B2sp6B1SmB6MjT9l8TZ2NVOml9QObzIeWMqni7eh3P5lS31Il_pwSgdhIvHVx74xgF1I/s1600/JobTask_Tracker+Image.png)

**Hadoop – Different Modes of Operation**

Last Updated : 22 Jun, 2020

As we all know Hadoop is an open-source framework which is mainly used for storage purpose and maintaining and analyzing a large amount of data or datasets on the clusters of commodity hardware, which means it is actually a data management tool. Hadoop also posses a scale-out storage property, which means that we can scale up or scale down the number of nodes as per are a requirement in the future which is really a cool feature.



**Hadoop Mainly works on 3 different Modes:**

1. Standalone Mode
2. Pseudo-distributed Mode
3. Fully-Distributed Mode

**1. Standalone Mode (Local)**

In *Standalone Mode* none of the Daemon will run i.e. Namenode, Datanode, Secondary Name node, Job Tracker, and Task Tracker. We use job-tracker and task-tracker for processing purposes in Hadoop1. For Hadoop2 we use Resource Manager and Node Manager. Standalone Mode also means that we are installing Hadoop only in a single system. By default, Hadoop is made to run in this Standalone Mode or we can also call it as the *Local mode*. We mainly use Hadoop in this Mode for the Purpose of Learning, testing, and debugging.

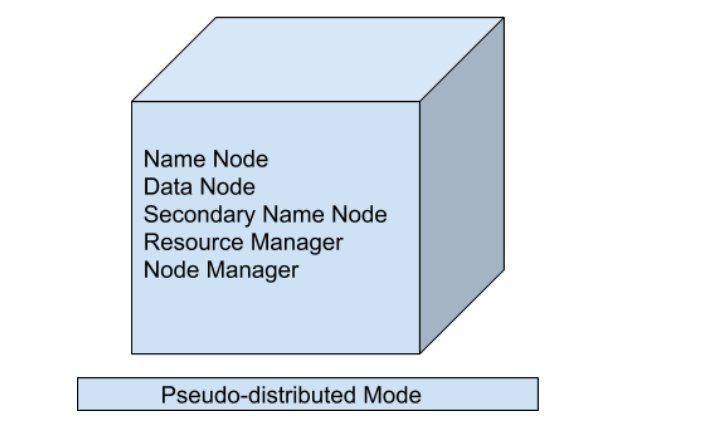
Hadoop works very much Fastest in this mode among all of these 3 modes. As we all know HDFS (Hadoop distributed file system) is one of the major components for Hadoop which utilized for storage Permission is not utilized in this mode. You can think of HDFS as similar to the file system’s available for windows i.e. NTFS (New Technology File System) and FAT32(File Allocation Table which stores the data in the blocks of 32 bits ). when your Hadoop works in this mode there is no need to configure the files – *hdfs-site.xml*, *mapred-site.xml*, *core-site.xml* for Hadoop environment. In this Mode, all of your Processes will run on a single [JVM(Java Virtual Machine)](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/) and this mode can only be used for small development purposes.

**2. Pseudo Distributed Mode (Single Node Cluster)**

In Pseudo-distributed Mode we also use only a single node, but the main thing is that the cluster is simulated, which means that all the processes inside the cluster will run independently to each other. All the daemons that are Namenode, Datanode, Secondary Name node, Resource Manager, Node Manager, etc. will be running as a separate process on separate JVM(Java Virtual Machine) or we can say run on different java processes that is why it is called a Pseudo-distributed.

One thing we should remember that as we are using only the single node set up so all the Master and Slave processes are handled by the single system. Namenode and Resource Manager are used as Master and Datanode and Node Manager is used as a slave. A secondary name node is also used as a Master. The purpose of the Secondary Name node is to just keep the hourly based backup of the Name node. In this Mode,

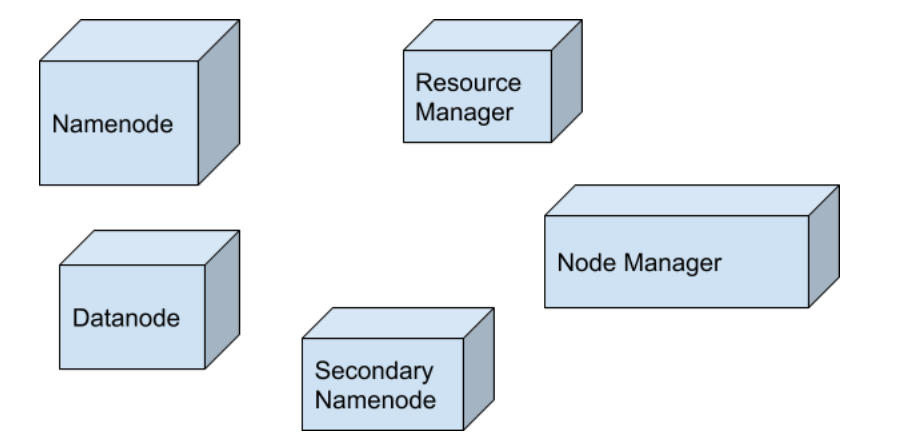
* Hadoop is used for development and for debugging purposes both.
* Our HDFS(Hadoop Distributed File System ) is utilized for managing the Input and Output processes.
* We need to change the configuration files *mapred-site.xml*, *core-site.xml*, *hdfs-site.xml* for setting up the environment.



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

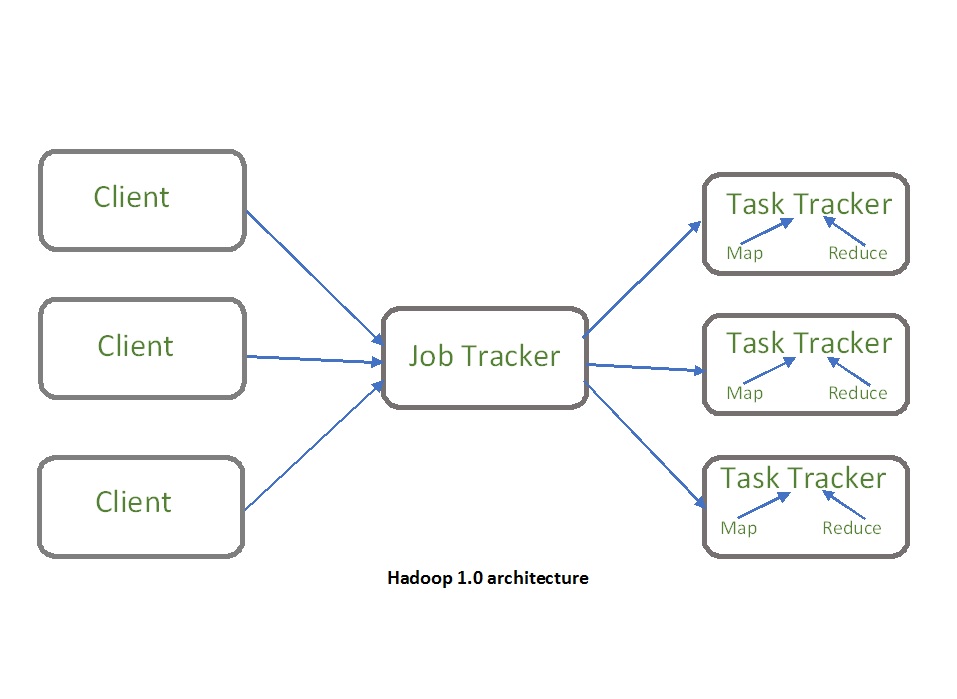
This is the most important one in which multiple nodes are used few of them run the Master Daemon’s that are Namenode and Resource Manager and the rest of them run the Slave Daemon’s that are DataNode and Node Manager. Here Hadoop will run on the clusters of Machine or nodes. Here the data that is used is distributed across different nodes. This is actually the *Production Mode*of Hadoop let’s clarify or understand this Mode in a better way in Physical Terminology.

Once you download the Hadoop in a tar file format or zip file format then you install it in your system and you run all the processes in a single system but here in the fully distributed mode we are extracting this tar or zip file to each of the nodes in the Hadoop cluster and then we are using a particular node for a particular process. Once you distribute the process among the nodes then you’ll define which nodes are working as a master or which one of them is working as a slave.

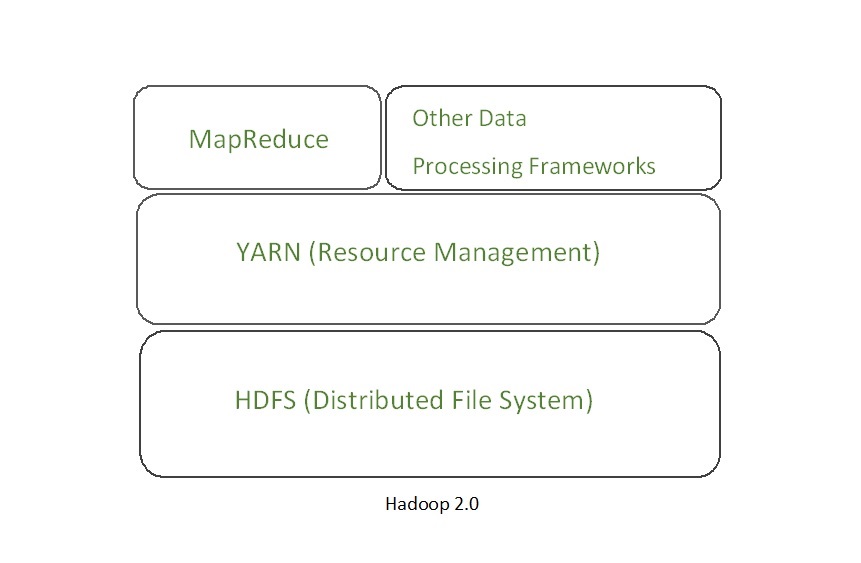


**Hadoop YARN Architecture**

YARN stands for “***Yet Another Resource Negotiator***“. It was introduced in Hadoop 2.0 to remove the bottleneck on Job Tracker which was present in Hadoop 1.0. YARN was described as a “*Redesigned Resource Manager*” at the time of its launching, but it has now evolved to be known as large-scale distributed operating system used for Big Data processing.



YARN architecture basically separates resource management layer from the processing layer. In Hadoop 1.0 version, the responsibility of Job tracker is split between the resource manager and application manager.

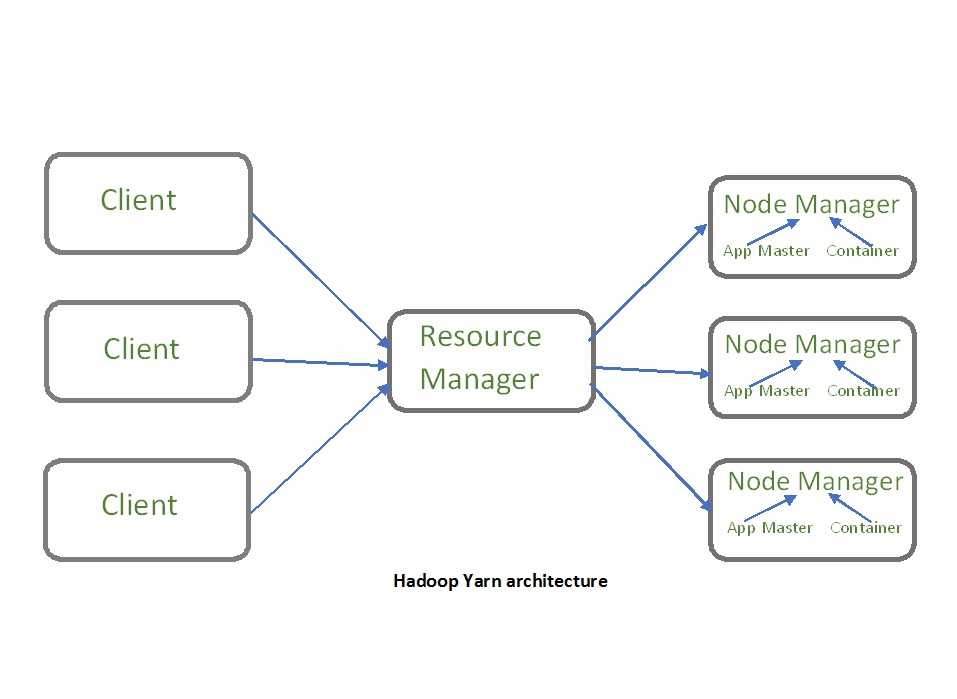


YARN also allows different data processing engines like graph processing, interactive processing, stream processing as well as batch processing to run and process data stored in HDFS (Hadoop Distributed File System) thus making the system much more efficient. Through its various components, it can dynamically allocate various resources and schedule the application processing. For large volume data processing, it is quite necessary to manage the available resources properly so that every application can leverage them.

**YARN Features:** YARN gained popularity because of the following features- 

* **Scalability:** The scheduler in Resource manager of YARN architecture allows Hadoop to extend and manage thousands of nodes and clusters.
* **Compatibility:** YARN supports the existing map-reduce applications without disruptions thus making it compatible with Hadoop 1.0 as well.
* **Cluster Utilization:**Since YARN supports Dynamic utilization of cluster in Hadoop, which enables optimized Cluster Utilization.
* **Multi-tenancy:** It allows multiple engine access thus giving organizations a benefit of multi-tenancy.

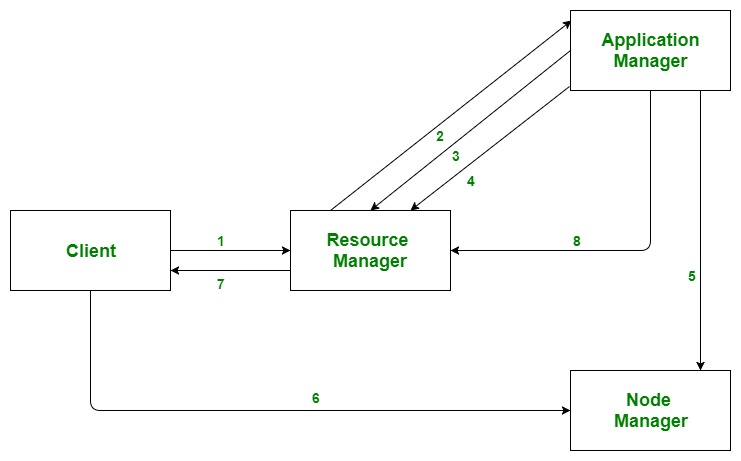
**Hadoop YARN Architecture**



The main components of YARN architecture include:

* **Client:** It submits map-reduce jobs.
* **Resource Manager:** It is the master daemon of YARN and is responsible for resource assignment and management among all the applications. Whenever it receives a processing request, it forwards it to the corresponding node manager and allocates resources for the completion of the request accordingly. It has two major components:
  + **Scheduler:** It performs scheduling based on the allocated application and available resources. It is a pure scheduler, means it does not perform other tasks such as monitoring or tracking and does not guarantee a restart if a task fails. The YARN scheduler supports plugins such as Capacity Scheduler and Fair Scheduler to partition the cluster resources.
  + **Application manager:** It is responsible for accepting the application and negotiating the first container from the resource manager. It also restarts the Application Master container if a task fails.
* **Node Manager:** It take care of individual node on Hadoop cluster and manages application and workflow and that particular node. Its primary job is to keep-up with the Resource Manager. It registers with the Resource Manager and sends heartbeats with the health status of the node. It monitors resource usage, performs log management and also kills a container based on directions from the resource manager. It is also responsible for creating the container process and start it on the request of Application master.
* **Application Master:**An application is a single job submitted to a framework. The application master is responsible for negotiating resources with the resource manager, tracking the status and monitoring progress of a single application. The application master requests the container from the node manager by sending a Container Launch Context(CLC) which includes everything an application needs to run. Once the application is started, it sends the health report to the resource manager from time-to-time.
* **Container:** It is a collection of physical resources such as RAM, CPU cores and disk on a single node. The containers are invoked by Container Launch Context(CLC) which is a record that contains information such as environment variables, security tokens, dependencies etc.

**Application workflow in Hadoop YARN:**



1. Client submits an application
2. The Resource Manager allocates a container to start the Application Manager
3. The Application Manager registers itself with the Resource Manager
4. The Application Manager negotiates containers from the Resource Manager
5. The Application Manager notifies the Node Manager to launch containers
6. Application code is executed in the container
7. Client contacts Resource Manager/Application Manager to monitor application’s status
8. Once the processing is complete, the Application Manager un-registers with the Resource Manager

**Advantages :**

* **Flexibility:** YARN offers flexibility to run various types of distributed processing systems such as Apache Spark, Apache Flink, Apache Storm, and others. It allows multiple processing engines to run simultaneously on a single Hadoop cluster.
* **Resource Management:** YARN provides an efficient way of managing resources in the Hadoop cluster. It allows administrators to allocate and monitor the resources required by each application in a cluster, such as CPU, memory, and disk space.
* **Scalability:** YARN is designed to be highly scalable and can handle thousands of nodes in a cluster. It can scale up or down based on the requirements of the applications running on the cluster.
* **Improved Performance:**YARN offers better performance by providing a centralized resource management system. It ensures that the resources are optimally utilized, and applications are efficiently scheduled on the available resources.
* **Security:** YARN provides robust security features such as Kerberos authentication, Secure Shell (SSH) access, and secure data transmission. It ensures that the data stored and processed on the Hadoop cluster is secure.

**Disadvantages :**

* **Complexity:**YARN adds complexity to the Hadoop ecosystem. It requires additional configurations and settings, which can be difficult for users who are not familiar with YARN.
* **Overhead:** YARN introduces additional overhead, which can slow down the performance of the Hadoop cluster. This overhead is required for managing resources and scheduling applications.
* **Latency:**YARN introduces additional latency in the Hadoop ecosystem. This latency can be caused by resource allocation, application scheduling, and communication between components.
* **Single Point of Failure:** YARN can be a single point of failure in the Hadoop cluster. If YARN fails, it can cause the entire cluster to go down. To avoid this, administrators need to set up a backup YARN instance for high availability.
* **Limited Support:** YARN has limited support for non-Java programming languages. Although it supports multiple processing engines, some engines have limited language support, which can limit the usability of YARN in certain environments.