Hadoop – HDFS (Hadoop Distributed File System)

Last Updated : 12 May, 2023

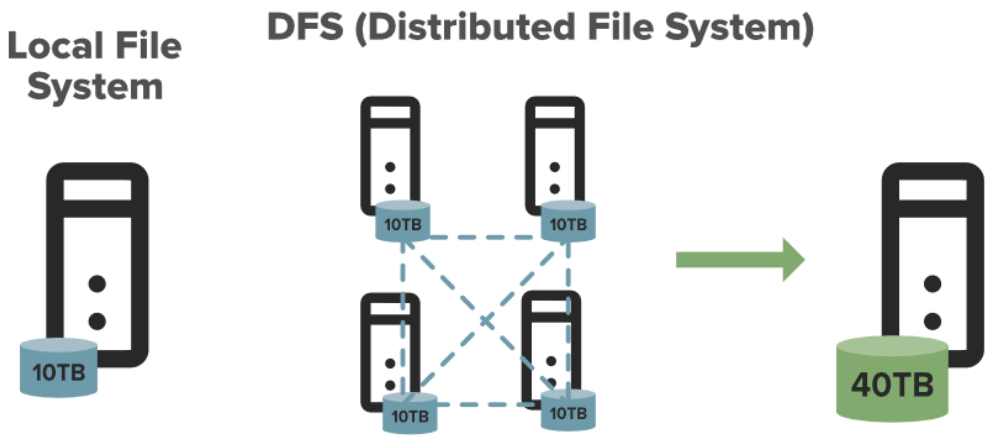
Before head over to learn about the HDFS(Hadoop Distributed File System), we should know what actually the file system is. The file system is a kind of Data structure or method which we use in an operating system to manage file on disk space. This means it allows the user to keep maintain and retrieve data from the local disk.

An example of the windows file system is NTFS(New Technology File System) and FAT32(File Allocation Table 32). FAT32 is used in some older versions of windows but can be utilized on all versions of *windows xp*. Similarly like windows, we have ext3, ext4 kind of file system for Linux OS.

What is DFS?

DFS stands for the distributed file system, it is a concept of storing the file in multiple nodes in a distributed manner. DFS actually provides the Abstraction for a single large system whose storage is equal to the sum of storage of other nodes in a cluster.

Let's understand this with an example. Suppose you have a DFS comprises of 4 different machines each of size 10TB in that case you can store let say 30TB across this DFS as it provides you a combined Machine of size 40TB. The 30TB data is distributed among these Nodes in form of Blocks.

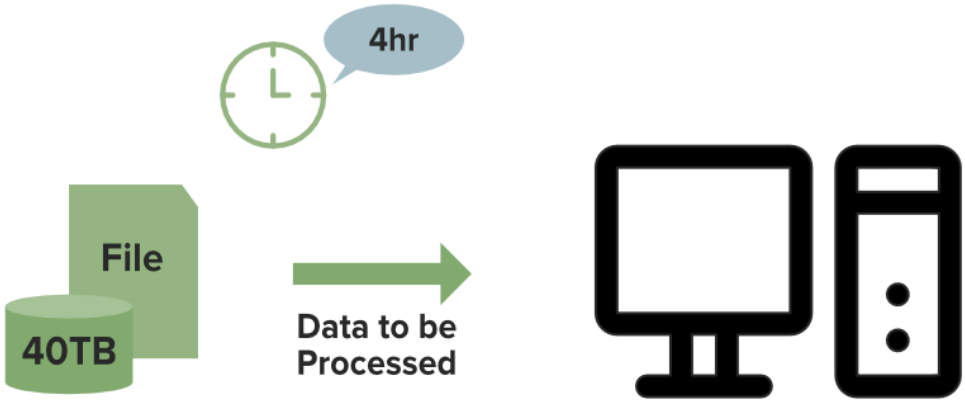


Why We Need DFS?

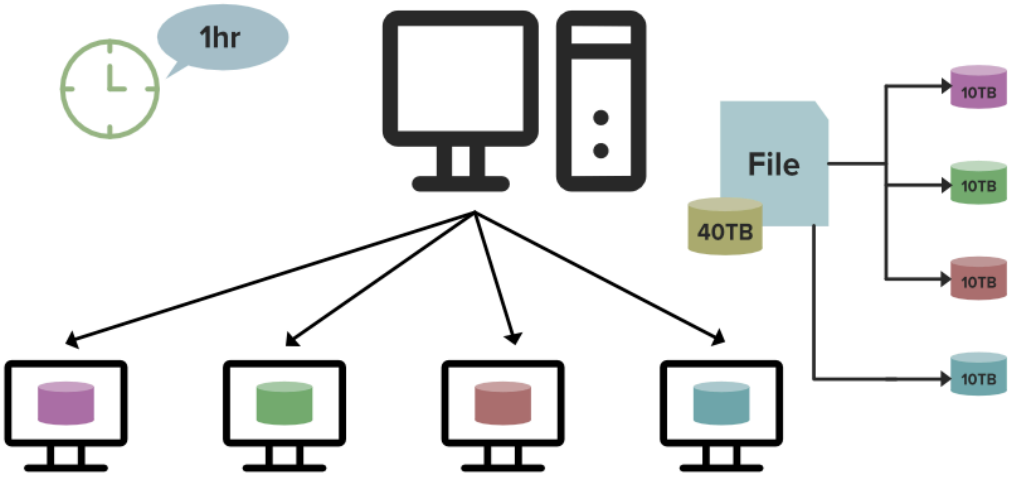
You might be thinking that we can store a file of size 30TB in a single system then why we need this DFS. This is because the disk capacity of a system can only increase up to an extent. If somehow you manage the data on a single system then you'll face the processing problem, processing large datasets on a single machine is not efficient.

Let's understand this with an example. Suppose you have a file of size 40TB to process. On a single machine, it will take suppose 4hrs to process it completely but what if you use a DFS(Distributed File System). In that case, as you can see in the below image the File of size 40TB is distributed among the 4 nodes in a cluster each node stores the 10TB of file. As all these nodes are working simultaneously it will take the only 1 Hour to completely process it which is Fastest, that is why we need DFS.

Local File System Processing:



Distributed File System Processing:



Overview – HDFS

Now we think you become familiar with the term *file system* so let's begin with HDFS. HDFS(Hadoop Distributed File System) is utilized for storage permission is a Hadoop cluster. It mainly designed for working on commodity Hardware devices(devices that are inexpensive), working on a distributed file system design. HDFS is designed in such a way that it believes more in storing the data in a large chunk of blocks rather than storing small data blocks. HDFS in Hadoop provides Fault-tolerance and High availability to the storage layer and the other devices present in that Hadoop cluster.

HDFS is capable of handling larger size data with high volume velocity and variety makes Hadoop work more efficient and reliable with easy access to all its components. HDFS stores the data in the form of the block where the size of each data block is 128MB in size which is configurable means you can change it according to your requirement in *hdfs-site.xml* file in your Hadoop directory.

Some Important Features of HDFS(Hadoop Distributed File System)

- It's easy to access the files stored in HDFS.
- HDFS also provides high availability and fault tolerance.
- Provides scalability to scaleup or scaledown nodes as per our requirement.
- Data is stored in distributed manner i.e. various Datanodes are responsible for storing the data.

- HDFS provides Replication because of which no fear of Data Loss.
- HDFS Provides High Reliability as it can store data in a large range of *Petabytes*.
- HDFS has in-built servers in Name node and Data Node that helps them to easily retrieve the cluster information.
- Provides high throughput.

HDFS Storage Daemon's

As we all know Hadoop works on the MapReduce algorithm which is a master-slave architecture, HDFS has *NameNode* and *DataNode* that works in the similar pattern.

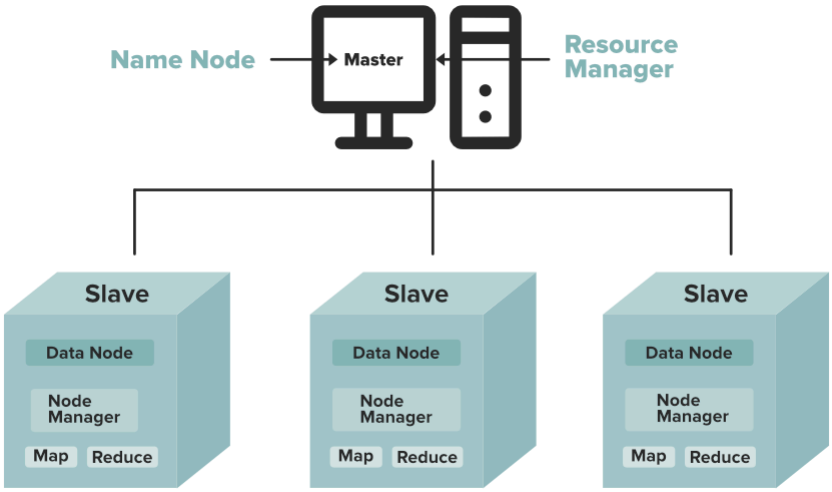
1. NameNode(Master)
2. DataNode(Slave)

1. NameNode: NameNode works as a *Master* in a Hadoop cluster that Guides the Datanode(Slaves). Namenode is mainly used for storing the Metadata i.e. nothing but the data about the data. Meta Data can be the transaction logs that keep track of the user's activity in a Hadoop cluster.

Meta Data can also be the name of the file, size, and the information about the location(Block number, Block ids) of Datanode that Namenode stores to find the closest DataNode for Faster Communication. Namenode instructs the DataNodes with the operation like delete, create, Replicate, etc.

As our NameNode is working as a Master it should have a high RAM or Processing power in order to Maintain or Guide all the slaves in a Hadoop cluster. Namenode receives heartbeat signals and block reports from all the slaves i.e. DataNodes.

2. DataNode: DataNodes works as a *Slave* DataNodes are mainly utilized for storing the data in a Hadoop cluster, the number of DataNodes can be from 1 to 500 or even more than that, the more number of DataNode your Hadoop cluster has More Data can be stored. so it is advised that the DataNode should have High storing capacity to store a large number of file blocks. Datanode performs operations like creation, deletion, etc. according to the instruction provided by the NameNode.



Objectives and Assumptions Of HDFS

1. System Failure: As a Hadoop cluster is consists of Lots of nodes with are commodity hardware so node failure is possible, so the fundamental goal of HDFS figure out this failure problem and recover it.

2. Maintaining Large Dataset: As HDFS Handle files of size ranging from GB to PB, so HDFS has to be cool enough to deal with these very large data sets on a single cluster.

3. Moving Data is Costlier then Moving the Computation: If the computational operation is performed near the location where the data is present then it is quite faster and the overall throughput of the system can be increased along with minimizing the network congestion which is a good assumption.

4. Portable Across Various Platform: HDFS Posses portability which allows it to switch across diverse Hardware and software platforms.

5. Simple Coherency Model: A Hadoop Distributed File System needs a model to write once read much access for Files. A file written then closed should not be changed, only data can be appended. This assumption helps us to minimize the data coherency issue. MapReduce fits perfectly with such kind of file model.

6. Scalability: HDFS is designed to be scalable as the data storage requirements increase over time. It can easily scale up or down by adding or removing nodes to the cluster. This helps to ensure that the system can handle large amounts of data without compromising performance.

7. Security: HDFS provides several security mechanisms to protect data stored on the cluster. It supports authentication and authorization mechanisms to control access to data, encryption of data in transit and at rest, and data integrity checks to detect any tampering or corruption.

8. Data Locality: HDFS aims to move the computation to where the data resides rather than moving the data to the computation. This approach minimizes network traffic and enhances performance by processing data on local nodes.

9. Cost-Effective: HDFS can run on low-cost commodity hardware, which makes it a cost-effective solution for large-scale data processing. Additionally, the ability to scale up or down as required means that organizations can start small and expand over time, reducing upfront costs.

10. Support for Various File Formats: HDFS is designed to support a wide range of file formats, including structured, semi-structured, and unstructured data. This makes it easier to store and process different types of data using a single system, simplifying data management and reducing costs.

System?

Similar Reads

Snakebite Python Package For Hadoop HDFS

Prerequisite: Hadoop and HDFS Snakebite is a very popular python package that allows users to access HDFS using some kind of program with python...

3 min read

How Does Namenode Handles Datanode Failure in Hadoop Distributed F...

Hadoop file system is a master/slave file system in which Namenode works as the master and Datanode work as a slave. Namenode is so critical term to...

2 min read

Anatomy of File Read and Write in HDFS

Big data is nothing but a collection of data sets that are large, complex, and which are difficult to store and process using available data management tool...

5 min read

Retrieving File Data From HDFS using Python Snakebite

Prerequisite: Hadoop Installation, HDFS Python Snakebite is a very popular Python library that we can use to communicate with the HDFS. Using the...

3 min read

Difference between Hadoop 1 and Hadoop 2

Hadoop is an open source software programming framework for storing a large amount of data and performing the computation. Its framework is base...

2 min read

Difference Between Hadoop 2.x vs Hadoop 3.x

The Journey of Hadoop Started in 2005 by Doug Cutting and Mike Cafarella. Which is an open-source software build for dealing with the large size Data?...

2 min read

Hadoop - Features of Hadoop Which Makes It Popular

Today tons of Companies are adopting Hadoop Big Data tools to solve their Big Data queries and their customer market segments. There are lots of other...

7 min read

Difference Between HDFS and HBase

HDFS: Hadoop Distributed File System is a distributed file system designed to store and run on multiple machines that are connected to each other as node...

2 min read

Characteristics of HDFS

HDFS is one of the major components of Hadoop that provide an efficient way for data storage in a Hadoop cluster. But before understanding the features o...

6 min read

Why a Block in HDFS is so Large?

A Disk has a block size, which decides how much information or data it can read or write. The disk blocks are generally different than the file system bloc...

5 min read

Article Tags : Hadoop Hadoop

GeeksforGeeks
Sanchhaya Education Private Limited
Corporate & Communications Address:- A-143, 9th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305) | Registered Address:- K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305

Company

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design

Inteview Preparation

Competitive Programming
Top DS or Algo for CP

UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects