

03 HDFS vs Cloud Data Storage

◆ HDFS Overview

HDFS (Hadoop Distributed File System) is the storage layer of Hadoop. It is designed to store large datasets across multiple machines in a distributed and fault tolerant way.

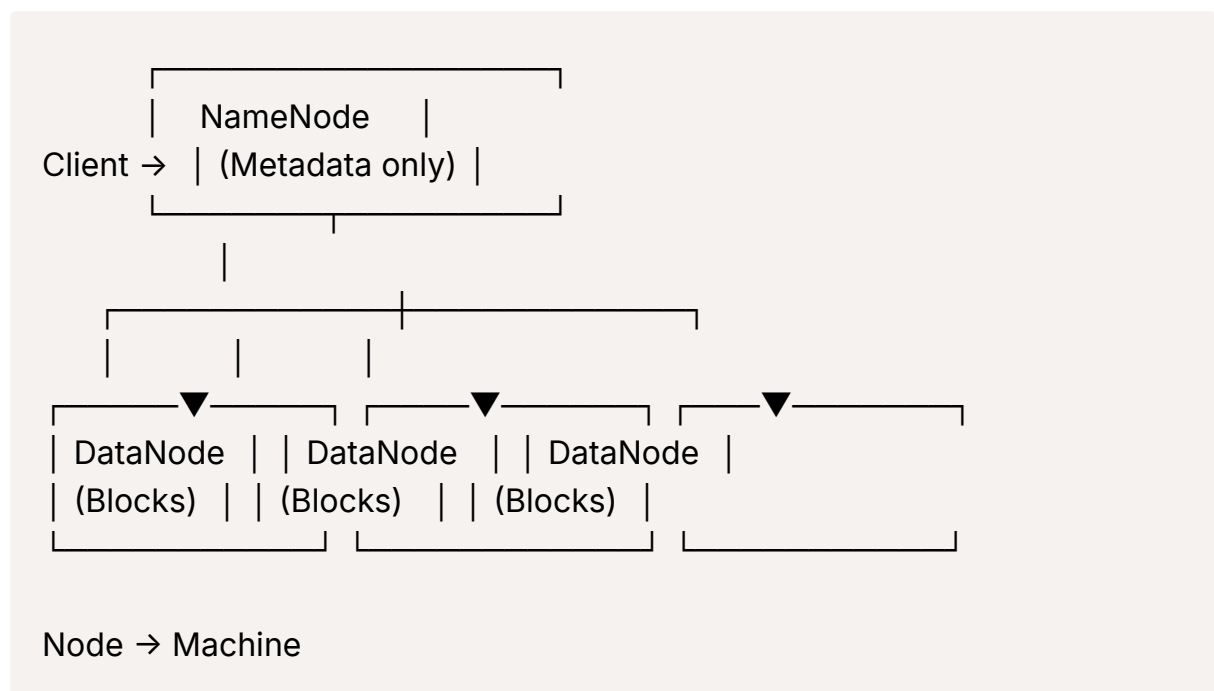
Instead of storing a file on one server, HDFS splits the file into blocks and distributes those blocks across a cluster. This approach allows the system to handle huge datasets while providing durability and high availability.

◆ HDFS Architecture

HDFS follows a **master and worker** architecture.

- **NameNode (Master)**
- **DataNodes (Workers)**

🧠 High-level View



📌 NameNode

- Stores **metadata** about files, directories, and block locations.

- Does **not** store the actual file data.
- Acts as the central controller for the filesystem.

DataNodes

- Store the **actual data blocks**.
- Perform block read/write operations.
- Send heartbeat and block reports to the NameNode.

Replication

- Each block is copied to multiple DataNodes (default replication factor = 3).
- Ensures data is safe even if a machine fails.

How Data is Stored in HDFS

- A large file is **broken into blocks**.
 - Blocks are **distributed** across different DataNodes.
 - Each block is **replicated** multiple times for fault tolerance.
 - Metadata about file-to-block mapping is stored in the **NameNode**.
-

Example

File size: **300 MB**

Block size: **128 MB**

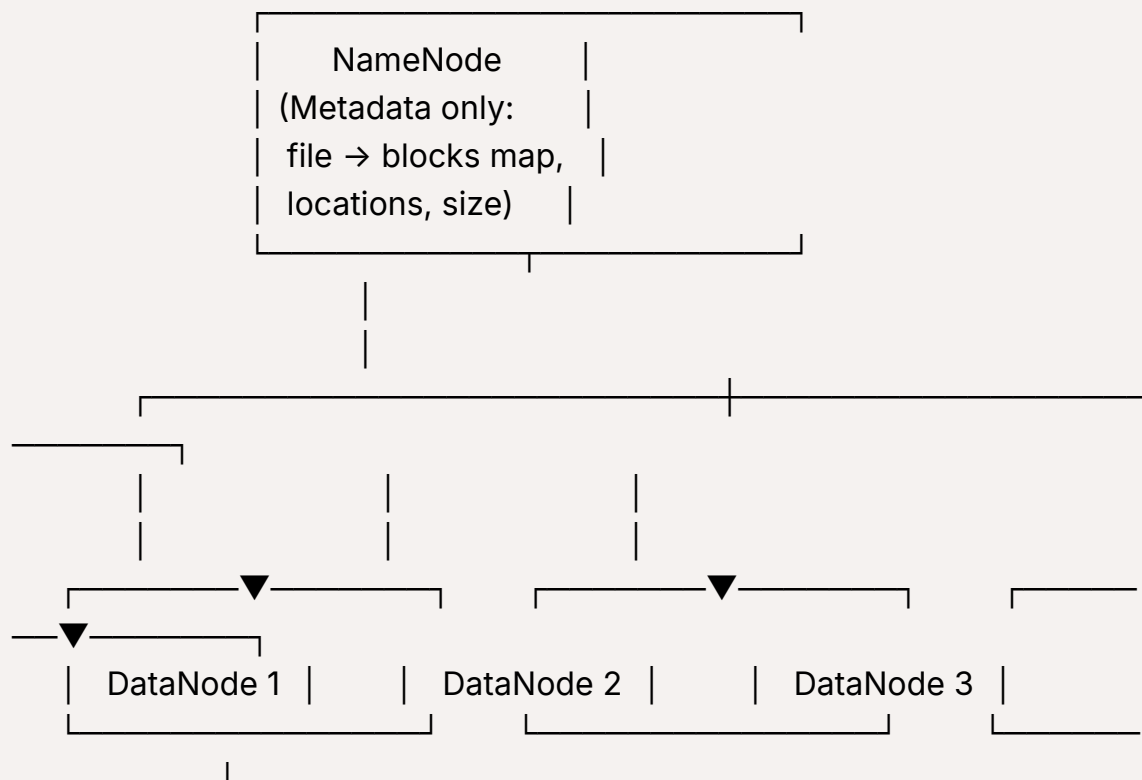
File is split into **3 blocks**:

Block	Size
Block 1	128 MB
Block 2	128 MB
Block 3	44 MB (remaining data)

Replication factor: **3**

So each block will have **3 copies** across different DataNodes.

Visual Representation



Block Distribution Example:

DataNode 1:	[Block1 Copy1]	[Block2 Copy2]	[Block3 Copy3]
DataNode 2:	[Block1 Copy2]	[Block2 Copy3]	[Block3 Copy1]
DataNode 3:	[Block1 Copy3]	[Block2 Copy1]	[Block3 Copy2]

Challenges with HDFS

While HDFS played an important role early in the Big Data world, it comes with several limitations:

- **Scaling requires physical hardware**
 - To store more data, companies must buy and manage more servers.
- **Not persistent**
 - Storage is tightly coupled with compute . If cluster goes down then you can not access data.
- **Single point of failure (NameNode)**

- If the NameNode goes down, the entire system becomes inaccessible.
 - **High operational overhead**
 - Needs dedicated teams to manage clusters, failures, replication, balancing, upgrades, etc.
 - **Not cost-effective at large scale**
 - Replication (usually x3) means storage cost triples.
 - **Not ideal for real-time or interactive workloads**
 - Designed for large sequential reads, not fast low-latency access.
 - **On-Premises locked**
 - Scaling depends on physical procurement, rack space, networking, and maintenance cycles.
-

◆ Shift Toward Cloud Storage Systems

Modern cloud storage systems like **Amazon S3, Azure Data Lake Storage, and Google Cloud Storage** address many of the limitations of HDFS.

◆ Advantages of Cloud-Based Storage

- **Elastic Storage**
 - Scale storage instantly without buying hardware.
- **No infrastructure maintenance**
 - No need to manage disks, replication, rack failures, or hardware refresh cycles.
- **Lower cost**
 - Pay only for storage used; no upfront capital expense.
- **High durability and availability**
 - Cloud providers handle replication across regions and zones.
- **Separation of storage and compute**
 - Compute engines (Spark, Snowflake, Databricks, BigQuery) can scale independently.

- **Supports multiple processing engines**
 - SQL engines, Spark, AI/ML tools, pipelines , all can read from the same data lake.

◆ Why Companies Are Moving to the Cloud

Organizations are adopting cloud platforms because they offer **scalability, flexibility, lower operational costs, and faster innovation**. Unlike on-premises Hadoop environments, cloud platforms provide serverless or managed services that reduce operational complexity and adapt easily to growing data workloads. Cloud storage also supports modern analytics patterns such as real-time streaming, AI/ML workflows, and data lakehouse architectures making it more suitable for evolving business needs.



Data Lake vs Data Warehouse vs Lakehouse

Modern data architectures have evolved over time. Organizations started with **data warehouses**, then moved to **data lakes** for flexibility, and now use **lakehouses** to combine the strengths of both.



Data Warehouse

- Stores **structured and cleaned data**
- Used for **analytics, reporting, BI**
- Schema-on-write (data must be modeled before storing)

A data warehouse is a centralized storage system optimized for analytics. Data loaded into a warehouse is already cleaned, transformed, and structured. This makes querying fast and reliable, especially for dashboards and business reporting. Technologies include **Snowflake, Redshift, BigQuery, and Teradata**.

Data Lake

- Stores **raw data of any format**
- Supports structured, semi-structured, and unstructured data
- Schema-on-read (structure applied at query time)

A data lake allows organizations to store all their data in its native form without enforcing a strict schema upfront. This makes it ideal for storing logs, clickstreams, audio, video, images, IoT data, and more. Data lakes are flexible and useful for machine learning and exploratory analytics. Examples include **Amazon S3, Azure Data Lake Storage (ADLS), and Google Cloud Storage (GCS)**.

Lakehouse

- Combines the best of data lake + data warehouse
- Supports **raw data storage + analytics + ACID transactions**
- Allows streaming + batch + ML workloads in one system

A Lakehouse solves the limitations of data lakes (lack of reliability and governance) by adding structure, ACID compliance, indexing, and performance optimization similar to a warehouse. It allows BI and ML workloads from the same storage layer. Popular examples include **Databricks Delta Lake, Apache Iceberg, and Apache Hudi**.

Comparison Table

Feature	Data Lake	Data Warehouse	Lakehouse
Data Type	Structured + Semi + Unstructured	Structured only	All data types
Schema	Schema-on-read	Schema-on-write	Both
Cost	Low	Higher	Medium
Performance	Medium	High	High
ACID Transactions	✗ No	✓ Yes	✓ Yes
ML Support	✓ Strong	Limited	✓ Strong

Feature	Data Lake	Data Warehouse	Lakehouse
Ideal Use Case	Data science, raw data storage	Business analytics	Unified analytics (BI + ML)