Definition of Distributed File System (DFS):

A **Distributed File System (DFS)** is a file system that allows users to access and process data stored on remote servers as if it were on their local machines. It enables **multiple users and systems** to share files over a network in a **transparent, reliable, and scalable** way. The main goal of DFS is to provide a **centralized file management system** while physically storing data across multiple nodes or locations.

Key Characteristics:

- **Transparency:** Users are unaware of where data is physically stored.
- **Scalability:** Supports growing amounts of data and users.
- **Fault Tolerance:** Continues to operate even if parts of the system fail.
- **Concurrency:** Multiple users can access the same data simultaneously.
- **Security & Consistency:** Ensures proper access control and consistent data views.

Types of Distributed File Systems:

Type	Description
1. Client-Server DFS	Traditional DFS model where clients request file operations from centralized or replicated servers.
2. Peer-to-Peer DFS	All nodes act as both client and server; files can be shared directly (e.g., BitTorrent).
3. Cluster-Based DFS	DFS built on clusters of machines to provide high availability and parallel access (e.g., HDFS).
4. Cloud-Based DFS	Offered via cloud services; provides global file access (e.g., Google Drive, Dropbox, OneDrive).
5. Parallel DFS	Allows multiple clients to read/write simultaneously for high-performance applications.
6. Object-Based DFS	Stores data as objects with metadata; supports scalability and easy data retrieval.

Popular Examples:

- **HDFS** – Hadoop Distributed File System (Big Data)
- **NFS** – Network File System (UNIX/Linux)
- **GFS** – Google File System (Proprietary)
- **CephFS** – Highly scalable open-source file system

- **Azure Blob Storage / Amazon S3** – Cloud-based DFS

HDFS (Hadoop Distributed File System)

🔹 **Overview:**

HDFS is a **Java-based open-source** distributed file system developed by the **Apache Hadoop Project**. It is designed to store and manage **large datasets across clusters of commodity hardware**.

🔹 **Key Features:**

- **Fault Tolerant:** Data is replicated (typically 3 copies) across nodes.
- **High Throughput:** Optimized for batch processing and streaming data access.
- **Large Block Size:** Default block size is 128MB (or 64MB), reducing metadata overhead.
- **Write-Once, Read-Many:** Files are written once and read multiple times — optimizing performance.
- **Scalable Architecture:** Easily scales to thousands of nodes and petabytes of data.

🔹 **Architecture:**

- **NameNode:** Manages metadata and namespace.
- **DataNode:** Stores actual data blocks.
- **Secondary NameNode:** Periodically merges namespace with edit logs (not a backup node).

🔹 **Use Cases:**

- Big Data processing using **MapReduce**, **Apache Spark**, etc.
- Data lakes and archival systems.
- Log analysis and data warehousing.

GFS (Google File System)

🔹 **Overview:**

GFS is a **proprietary distributed file system** developed by **Google** to support their data-intensive applications like search indexing and web crawling.

🔹 **Key Features:**

- **Fault Tolerance:** Designed to recover quickly from disk and node failures.
- **Chunk-Based Storage:** Data is divided into large **64MB chunks**, stored on multiple chunkservers.
- **Replication:** Each chunk is replicated (typically 3 times) for reliability.
- **Append-Only Writes:** Optimized for appending rather than overwriting data.
- **Centralized Control:** A single **Master server** manages metadata and chunk locations.

🔹 **Architecture:**

- **Master Server:** Stores metadata (chunk names, locations, versions).
- **Chunk Servers:** Store actual file chunks and handle read/write requests.
- **Clients:** Communicate with Master to locate chunks and then interact with ChunkServers directly.

🔹 **Use Cases:**

- Internal infrastructure at Google for storing:
  - Web indexing data
  - YouTube video metadata
  - Logs and crawling results

✅ **Comparison: HDFS vs GFS**

Feature	HDFS	GFS
Developer	Apache Hadoop community	Google
Open Source	Yes	No (proprietary)
Block/Chunk Size	128MB (configurable)	64MB (fixed)
Metadata Server	NameNode	Master Server
Replication Factor	Default 3	Default 3
Write Pattern	Write-once, read-many	Append-only
Use Cases	General-purpose big data applications	Google-specific large-scale applications