# HDFS (Hadoop Distributed File System)

It is a distributed file system designed to run on affordable, low-cost hardware. One of the key assumptions in the design of HDFS is that **"Hardware Failure" is not considered a rare event** rather it is assumed to occur normally. Therefore, it aims to perform detections and quick and automatic recovery from faults. Its emphasis is on high throughput (batch processing) instead of low latency (user interactions). Its use case follows **"Write Once Read Many"** access model for files. This simplifies the data coherency issues as no multiple simultaneous mutations to data can occur. Since the size of data involved is huge as compared to the computation instructions, HDFS provides interfaces for applications to move closer to data.

HDFS consists of a single **Name Node (master)** and a number of **Data Nodes**. The former manages the filesystem namespace and maps data blocks to data nodes (collectively called **FsImage**) and the latter manage storage attached to the nodes they run on.

Each file is divided into one or more blocks which are stored in a set of Data Nodes. The blocks are replicated to other nodes by a replication factor (stored in name node) for fault tolerance. The data is pipelined from one Data Node to the next.

Data nodes directly serve read/write requests from client and perform operations upon instruction from Name node. This ensures that the user data never flows through the Name Node and prevents it from becoming a bottleneck.

Name Node uses a **EditLog** to persistently record changes that occur to **FsImage.** On startup the NameNode reads the FsImage and EditLog from disk, applies all the transactions from the EditLog to the in-memory representation of the FsImage, and flushes out this new version into a new FsImage on disk and truncates the EditLog. This process is called **a checkpoint**.

When a Data Node starts up, it scans through its local file system, generates a list of all HDFS data blocks that correspond to each of these local files and sends this report to the Name Node (Block Report).

**Data Node Failure**:

- The Name Node detects this condition by the absence of a Heartbeat message from the node and marks the node as dead.
- The Name Node tracks which blocks need to be replicated as a result of the Data Node failure and replicates them accordingly from other copies.

**Note:** Checksum of each block of file is stored in the namespace to check for block corruption.

**Name Node Failure:**

- A corruption of FsImage is dealt by maintaining multiple copies of FsImage and EditLog.
- However, the Name Node Machine is a single point of failure of an HDFS cluster. If it fails, manual Intervention may be necessary if no secondary name node exists.

# GFS (Google File System) and its differences from HDFS

Like HDFS, GFS is a scalable distributed file system for large distributed data-intensive applications.

Similar to HDFS's name node the master node in GFS maintains the metadata and performs system-wide activities such as chunk lease management, garbage collection of orphaned chunks, and chunk migration between chunk servers. Files are divided into fixed-size chunks (blocks in HDFS) and stored on chunk servers (data nodes). For reliability, each chunk is replicated on multiple chunk servers. Also all the data transfers occur directly between client and the chunk servers.

Most of the assumptions and implementations of GFS are similar to HDFS. However, there are some differences which are as follows.

- **HDFS doesn't support** Snapshots and append operations while **GFS does**.
- In HDFS clients can only read in parallel while in **GFS clients can read, write and append in parallel.**
- Another difference is that in **HDFS the entire FsImage is stored persistently** in the storage while in **GFS only saves the namespace and mappings** but not the chunk location information persistently. Instead, it asks each chunk server about its chunks at master startup and whenever a chunk server joins the cluster.
- Similar to HDFS's EditLog the GFS maintains an Operation Log that contains a historical record of critical metadata changes. **GFS however replicates its Operation log on multiple remote machines.** Therefore, in case of Log corruption GFS master can recover its file system by replaying the operation log. This was not the case with the HDFS as there are no replicas of the FsImage.
- While in HDFS **the data node periodically sends a Heartbeat** (signifying its alive) and a block report to the name node, In **GFS the master periodically communicates with each chunk server** in Heartbeat messages to give it instructions and collect its state.
- Unlike HDFS, **GFS assumes multiple simultaneous concurrent mutations that is writes/appends can occur on the same data**. Therefore, **GFS uses leases** to ensure consistent mutation order across all replicas of a file that is being mutated concurrently by multiple clients.
- Master grants a chunk lease to one of the replicas called primary. The primary decides the order in which the mutations should be applied. **There is no concept of lease or primary in HDFS.**
- HDFS uses distributed cache a facility provided by MapReduce framework. **In GFS** on the other hand **neither the chunk servers nor the clients cache file data**. Chunks are stored as local files so Linux's buffer cache already keeps frequently access data in memory.
- In **GFS** the **master state is replicated for reliability**. Its operation log and checkpoints are replicated on multiple machines. This is not the case in HDFS.