Fault Tolerance

Let's explore what techniques HDFS uses for fault tolerance.

We'll cover the following

How does HDFS handle DataNode failures?#

Replication#

When a DataNode dies, all of its data becomes unavailable. HDFS handles this data unavailability through replication. As stated earlier, every block written to HDFS is replicated to multiple (default three) DataNodes. Therefore, if one DataNode becomes inaccessible, its data can be read from other replicas.

HeartBeat#

The NameNode keeps track of DataNodes through a heartbeat mechanism. Each DataNode sends periodic heartbeat messages (every few seconds) to the NameNode. If a DataNode dies, the heartbeats will stop, and the NameNode will detect that the DataNode has died. The NameNode will then mark the DataNode as dead and will no longer forward any read/write request to that DataNode. Because of replication, the blocks stored on that DataNode have additional replicas on other DataNodes. The NameNode performs regular status checks on the file system to discover under-replicated blocks and performs a **cluster rebalance** process to replicate blocks that have less than the desired number of replicas.

What happens when the NameNode fails?#

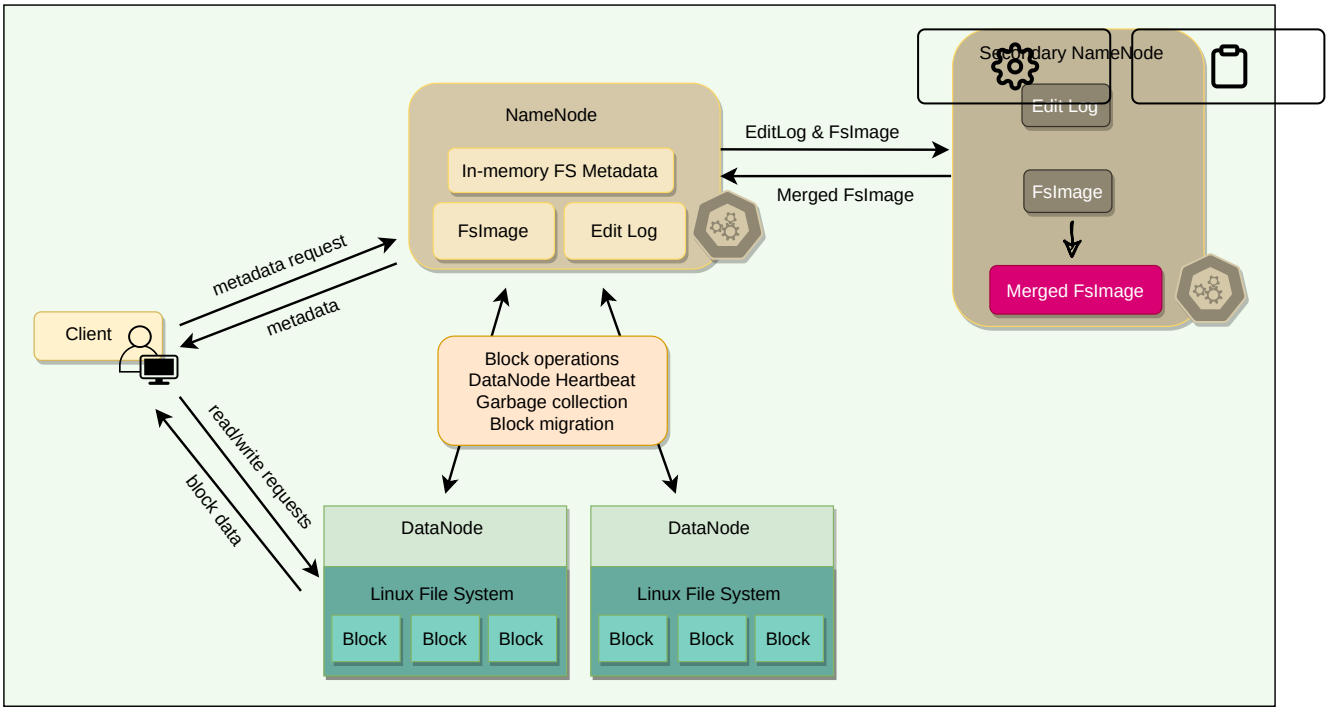
FsImage and EditLog#

The NameNode is a **single point of failure (SPOF)**. A NameNode failure will bring the entire file system down. Internally, the NameNode maintains two on-disk data structures that store the file system's state: an **FsImage** file and an **EditLog**. FsImage is a checkpoint (or the image) of the file system metadata at some point in time, while the EditLog is a log of all of the file system metadata transactions since the image file was last created. All incoming changes to the file system metadata are written to the EditLog. At periodic intervals, the EditLog and FsImage files are merged to create a new image file snapshot, and the edit log is cleared out.

Metadata backup#

On a NameNode failure, the metadata would be unavailable, and a disk failure on the NameNode would be catastrophic because the file metadata would be lost since there would be no way of knowing how to reconstruct the files from the blocks on the DataNodes. For this reason, it is crucial to make the NameNode resilient to failure, and HDFS provides two mechanisms for this:

1. The first way is to back up and store multiple copies of FsImage and EditLog. The NameNode can be configured to maintain multiple copies of the files. Any update to either the FsImage or EditLog causes each copy of the FsImages and EditLogs to get updated synchronously and atomically. A common configuration is to maintain one copy of these files on a local disk and one on a remote Network File System (NFS) mount. This synchronous updating of multiple copies of the FsImage and EditLog may degrade the rate of namespace transactions per second that a NameNode can support. However, this degradation is acceptable because even though HDFS applications are very data-intensive, they are not metadata-intensive.
2. Another option provided by HDFS is to run a **Secondary NameNode**, which despite its name, is not a backup NameNode. Its main role is to help primary NameNode in taking the checkpoint of the filesystem. Secondary NameNode periodically merges the namespace image with the EditLog to prevent the EditLog from becoming too large. The secondary NameNode runs on a separate physical machine because it requires plenty of CPU and as much memory as the NameNode to perform the merge. It keeps a copy of the merged namespace image, which can be used in the event of the NameNode failure. However, the state of the secondary NameNode lags behind that of the primary, so in the event of total failure of the primary, data loss is almost inevitable. The usual course of action, in this case, is to copy the NameNode's metadata files that are on NFS to the secondary and run it as the new primary.



Role of primary and secondary NameNode

← Back

Data Integrity & Caching

Next →

HDFS High Availability (HA)

Mark as Completed

Report an Issue