



HDFS High Availability (HA)

Let's learn how HDFS achieves high availability.

We'll cover the following



- HDFS high availability architecture
 - QJM
 - Zookeeper
- Failover and fencing
 - Fencing

HDFS high availability architecture#

Although NameNode's metadata is copied to multiple file systems to protect against data loss, it still does not provide high availability of the filesystem. If the NameNode fails, no clients will be able to read, write, or list files, because the NameNode is the sole repository of the metadata and the file-to-block mapping. In such an event, the whole Hadoop system would effectively be out of service until a new NameNode is brought online.

To recover from a failed NameNode scenario, an administrator will start a new primary NameNode with one of the filesystem metadata replicas and configure DataNodes and clients to use this new NameNode. The new NameNode is not able to serve requests until it has

1. loaded its namespace image into memory,
2. replayed its EditLog, and
3. received enough block reports from the DataNodes.



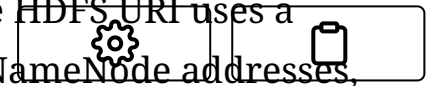
On large clusters with many files and blocks, it can take half an hour or more to perform a cold start of a NameNode. Furthermore, this long recovery time is a problem for routine maintenance. In fact, because an unexpected failure of the NameNode is rare, the case for planned downtime is actually more important in practice.

To solve this problem, Hadoop, in its 2.0 release, added support for HDFS High Availability (HA). In this implementation, there are two (or more) NameNodes in an active-standby configuration. At any point in time, exactly one of the NameNodes is in an active state, and the others are in a Standby state. The active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a follower of the active, maintaining enough state to provide a fast failover when required.

For the Standby nodes to keep their state synchronized with the active node, HDFS made a few architectural changes:

- The NameNodes must use highly available shared storage to share the EditLog (e.g., a Network File System (NFS) mount from a Network Attached Storage (NAS)).
- When a standby NameNode starts, it reads up to the end of the shared EditLog to synchronize its state with the active NameNode, and then continues to read new entries as the active NameNode writes them.
- DataNodes must send block reports to all the NameNodes because the block mappings are stored in a NameNode's memory, and not on disk.
- Clients must be configured to handle NameNode failover, using a mechanism that is transparent to users. Client failover is handled transparently by the client library. The simplest implementation uses


client-side configuration to control failover. The HDFS URI uses a logical hostname which is mapped to multiple NameNode addresses, and the client library tries each NameNode address until the operation succeeds.



There are two choices for the highly available shared storage: an NFS filer (as described above), or a **Quorum Journal Manager (QJM)**.

QJM#

The sole purpose of the QJM is to provide a highly available EditLog. The QJM runs as a group of journal nodes, and each edit must be written to a quorum (or majority) of the journal nodes. Typically, there are three journal nodes, so that the system can tolerate the loss of one of them. This arrangement is similar to the way ZooKeeper (<https://zookeeper.apache.org/>) works, although it is important to realize that the QJM implementation does not use ZooKeeper.

 **Note:** HDFS High Availability does use ZooKeeper for electing the active NameNode. More details on this later. QJM process runs on all NameNodes and communicates all EditLog changes to journal nodes using RPC.

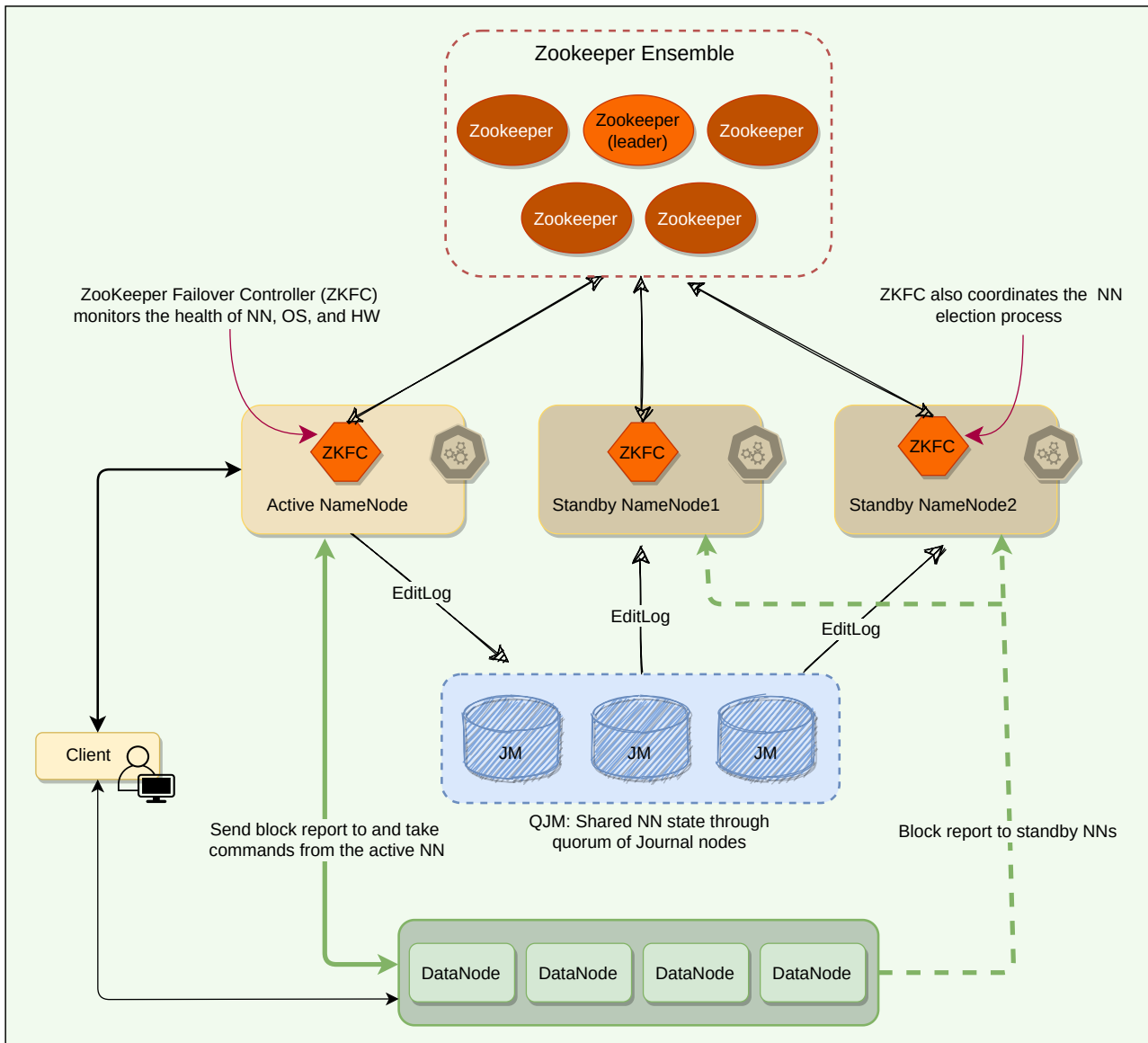
Since the Standby NameNodes have the latest state of the metadata available in memory (both the latest EditLog and an up-to-date block mapping), any standby can take over very quickly (in a few seconds) if the active NameNode fails. However, the actual failover time will be longer in practice (around a minute or so) because the system needs to be conservative in deciding that the active NameNode has failed.

In the unlikely event of the Standbys being down when the active fails, the administrator can still do a cold start of a Standby. This is no worse than the non-HA case.

Zookeeper#



The ZKFailoverController (ZKFC) is a ZooKeeper client that runs on each NameNode and is responsible for coordinating with the Zookeeper and also monitoring and managing the state of the NameNode (more details below).



HDFS high availability architecture

Failover and fencing#

A **Failover Controller** manages the transition from the active NameNode to the Standby. The default implementation of the failover controller uses **ZooKeeper** to ensure that only one NameNode is active. Failover Controller runs as a lightweight process on each NameNode and monitors the NameNode for failures (using Heartbeat), and triggers a failover when the active NameNode fails.

Graceful failover: For routine maintenance, an administrator can manually initiate a failover. This is known as a graceful failover, since the failover controller arranges an orderly transition from the active NameNode to the Standby.

Ungraceful failover: In the case of an ungraceful failover, however, it is impossible to be sure that the failed NameNode has stopped running. For example, a slow network or a network partition can trigger a failover transition, even though the previously active NameNode is still running and thinks it is still the active NameNode.

The HA implementation uses the mechanism of **Fencing** to prevent this “**split-brain**” scenario and ensure that the previously active NameNode is prevented from doing any damage and causing corruption.

Fencing#

Fencing is the idea of putting a fence around a previously active NameNode so that it cannot access cluster resources and hence stop serving any read/write request. To apply fencing, the following two techniques are used:

- **Resource fencing:** Under this scheme, the previously active NameNode is blocked from accessing resources needed to perform essential tasks. For example, revoking its access to the shared storage directory (typically by using a vendor-specific NFS command), or disabling its network port via a remote management command.

- **Node fencing:** Under this scheme, the previously active NameNode is blocked from accessing all resources. A common way of doing this is to power off or reset the node. This is an effective method of keeping it from accessing anything at all. This technique is also called STONIT or “Shoot The Other Node In The Head.”



To learn more about automatic failover, take a look at Apache documentation (https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html#Automatic_Failover).

[← Back](#)

Fault Tolerance

[Next →](#)

HDFS Characteristics

[Mark as Completed](#)[Report an Issue](#)