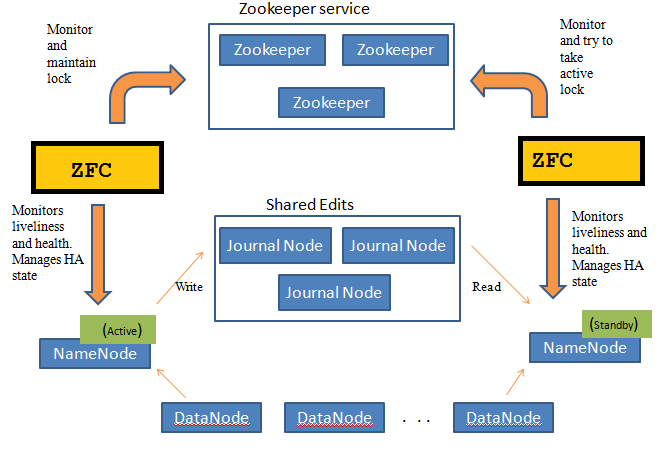
**High\_Availablity**



1:- Introduction to HDFS NameNode

HDFS is a FileSystem of Hadoop designed for storing very large files.

HDFS architecture follows master /slave topology in which master is NameNode and slaves is DataNode. Namenode stores meta-data i.e. number of blocks, their location, replicas and other details. This meta-data is available in memory in the master for faster retrieval of data.

Before Hadoop 2.0.0, the NameNode was a single point of failure (SPOF) in an HDFS cluster. Each cluster had a single NameNode, and if NameNode fails, the cluster as a whole would be out services. The cluster will be unavailable until the NameNode restarts or brought on a separate machine.

NameNode SPOF problem limit availability in following ways:

Planned maintenance activities like software or hardware upgrades on the Namenode would result in downtime of the Hadoop cluster.

If any unplanned event triggers, like machine crash, then cluster would be Unavailable unless an operator restarted the new namenode.

################################

Hadoop 2.0 overcomes this SPOF by providing support for many NameNode. HDFS NameNode High Availability architecture provides the option of running two redundant NameNodes in the same cluster in an active/passive configuration with a hot standby.

Active NameNode – It handles all client operations in the cluster.

Passive NameNode – It is a standby namenode, which has similar data as active NameNode. It acts as a slave, and fast failover, if necessary.

If Active NameNode fails, then passive NameNode takes all the responsibility of active node and cluster continues to work.

Issues in maintaining consistency in the HDFS High Availability cluster are as follows:

Active and Standby NameNode should always be in sync with each other, i.e. they should have the same metadata.

There should be only one NameNode active at a time. Otherwise, two NameNode will lead to corruption of the data. We call this scenario as a “Split-Brain Scenario”, where a cluster gets divided into the smaller cluster. Each one believes that it is the only active cluster. “Fencing” avoids such scenarios. Fencing is a process of ensuring that only one NameNode remains active at a particular time.

In High Availability Architecture, two NameNodes run at the same time. We can Implement the Active and Standby NameNode configuration in following two ways:

1:-Using Quorum Journal Nodes

2:- Using Shared Storage

1:- Using Quorum Journal Nodes

QJM is an HDFS implementation. It is designed to provide edit logs. It allows sharing these edit logs between the active namenode and standby namenode.

For High Availability, standby namenode communicates and synchronizes with the active namenode. It happens through a group of nodes or daemons called “Journal nodes”. The QJM runs as a group of journal nodes. There should be at least three journal nodes.

For N journal nodes, the system can tolerate at most (N-1)/2 failures and continue to function. So, for three journal nodes, the system can tolerate the failure of one {(3-1)/2} of them.

When an active node performs any modification, it logs modification to all journal nodes.

The standby node reads the edits from the journal nodes and applies to its own namespace in a constant manner. In the case of failover, the standby will ensure that it has read all the edits from the journal nodes before promoting itself to the Active state. This ensures that the namespace state is completely synchronized before a failure occurs.

To provide a fast failover, the standby node must have up-to-date information about the location of data blocks in the cluster. For this to happen, IP address of both the namenode is available to all the datanodes and they send block location information and heartbeats to both NameNode.

Fencing of NameNode

For the correct operation of an HA cluster, only one of the namenodes should active at a time. Otherwise, the namespace state would deviate between the two namenodes. So, fencing is a process to ensure this property in a cluster.

The journal nodes perform this fencing by allowing only one namenode to be the writer at a time.

The active namenode takes the responsibility of writing to the journal nodes .

Using Shared Storage

Standby and active namenode synchronize with each other by using “shared storage device”. For this implementation, both active namenode and standby namenode must have access to the particular directory on the shared storage device (.i.e. Network file system).

When active namenode perform any namespace modification, it logs a record of the modification to an edit log file stored in the shared directory. The standby namenode watches this directory for edits, and when edits occur, the standby namenode applies them to its own namespace. In the case of failure, the standby namenode will ensure that it has read all the edits from the shared storage before promoting itself to the Active state. This ensures that the namespace state is completely synchronized before failover occurs.

To prevent the “split-brain scenario” in which the namespace state deviates between the two namenode, an administrator must configure at least one fencing method for the shared storage.