1. Explain what is High availability of Namenode.

Prior to Hadoop 2.0, the Namenode was considered the Single Point Of failure(SPOF) in the HDFS cluster. In the event of Namenode going down or becoming unavailable for any given reason, the entire cluster would be unavailable until the Namenode was brought back or recovered on a different hardware.

HDFS High Availability feature addresses the SPOF problem by providing the option of running two redundant NameNodes in the same cluster in an Active/Passive configuration with a hot standby.

**NameNode HA Architecture:**

In a Namenode HA cluster setup, we have two different machines (which has exact hardware configuration) configured as Namenodes. One of the Namenodes in Active stste while the other is in Standby stste. The active Namenode performs all the client operations which includes serving the read and write requests. The standby Namenode maintains its state in order to ensure a fast failover in the event the active Namenode goes down.

The architectute design allows all of the DataNodes to send their block reports and heartbeat to both namenodes. This is possible with the HDFS configuration wherein a logical name service URI and it’s corresponding namenodes are provided. This design ensure that both the Namenodes receives exact same blocks.

The ZooKeeper Failover Controller (ZKFC) is responsible for HA Monitoring of the Namenode service. It also triggers an automatic failover when the Active Namenode is unavailable. Each Namenode has a ZKFC processes. ZKFC uses the Zookeeper Service for coordination in determining which is the active Namenode and in determining when to failover to the standby Namenode.

There are two major implementations in Namenode HA:

1. Quoram-based storage
2. Shared storage

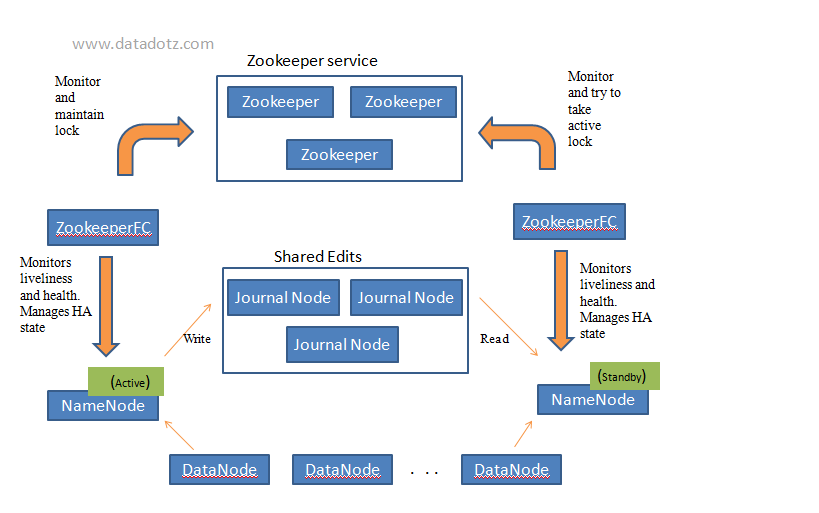
**Quoram-based storage:**

The Quoram-based implementation of Namenode HA basically uses Quoram Journal Manager (QJM).

This is currently the most common and reliable implementation for Namenode HA.

The standby Namenode communicates and synchronizes with the active Namenode through a group of hosts called **JournalNodes.** When there is any change in the namespace by Active Namenode, it makes a log record ina majority of journalNodes. JournalNodes receives these transaction edits or filesystem journal from the active Namenode on a constant basis. The standby Namenode reads these journals and updates its namespace to be in sync with the active Namenod. This ensures the standby Namenode is “completely” in sync with the active Namenode.

If the active Namenode stops responding or is unavailable, the Standby Namenode ensures that it will read all the edits from JournalNodes before it presents itself to be elected as Active Namenode.



**Shared Storage:**

The Shared sorage implementation of Namenode HA uses a shared storage space which is mounted on the Namenodes through NFS protocol. Both the Namenodes must have access to the directory which can be a network shared mounted on them.

Anytime there is a change in the namespace, the Active Namenode makes a log record in this shared directory. The standby Namenode constsntly monitors this shared directory for edits and applies to its namespace when there is a new edit.

If the Active Namenode stops responding or is unavailable, the standby Namenode ensures that it will read all the edits from the shared directory before it presents itself to be elected as Active Namenode.

This method however, is unreliable since a network failure or the shared directory location failure can be catastrophic.

1. Explain what is check pointing and how it is useful.

Checkpoint node in Hadoop is a new implementation of the Secondary Namenode to solve the drawbacks of Secondary Namenode.

Main function of the Checkpoint node in Hadoop is to create periodic checkpoints of filesystem metadata by merging ***edits*** file with ***fsImage*** file. Usually the new fsimage from merge operation is called as a **Checkpoint.**

The Checkpoint node periodically creates checkpoints of the namespace. It downloads fsimage and edits from the active NameNode, merges them locally, and uploads the new image back to the active NameNode. The Checkpoint node usually runs on a different machine than the NameNode since its memory requirements are on the same order as the NameNode. The Checkpoint node is started by bin/hdfs namenode -checkpoint on the node specified in the configuration file.

The start of the checkpoint process on the Checkpoint node is controlled by two configuration parameters.

1. dfs.namenode.checkpoint.period, set to 1 hour by default, specifies the maximum delay between two consecutive checkpoints
2. dfs.namenode.checkpoint.txns, set to 1 million by default, defines the number of uncheckpointed transactions on the NameNode which will force an urgent checkpoint, even if the checkpoint period has not been reached.

The Checkpoint node stores the latest checkpoint in a directory that is structured the same as the NameNode's directory. This allows the checkpointed image to be always available for reading by the NameNode if necessary. See Import checkpoint.

Multiple checkpoint nodes may be specified in the cluster configuration file.

3.Explain what is HDFS federation.

HDFS Federation improves the existing HDFS architecture through a clear separation of namespace and storage, enabling generic block storage layer. It enables support for multiple namespaces in the cluster to improve scalability and isolation. Federation also opens up the architecture, expanding the applicability of HDFS cluster to new implementations and use cases.

HDFS has two main layers:

Namespace manages directories, files and blocks. It supports file system operations such as creation, modification, deletion and listing of files and directories. Block Storage has two parts:

1) Block Management - it maintains the membership of datanodes in the cluster. It supports block-related operations such as creation, deletion, modification and getting location of the blocks. It also takes care of replica placement and replication.

2) Physical Storage- it stores the blocks and provides read/write access to it.

4.What are the configuration files that are to be edited for sure while installing a hadoop cluster?

The four files that need to be configured explicitly while setting up a single node Hadoop cluster are:

1. Core-site.xml
2. HDFS-site.xml
3. YARN-site.xml
4. xml