## **Report on Hadoop Cluster with High Availability**

Ibrahim Ghali - Hamza Zarai

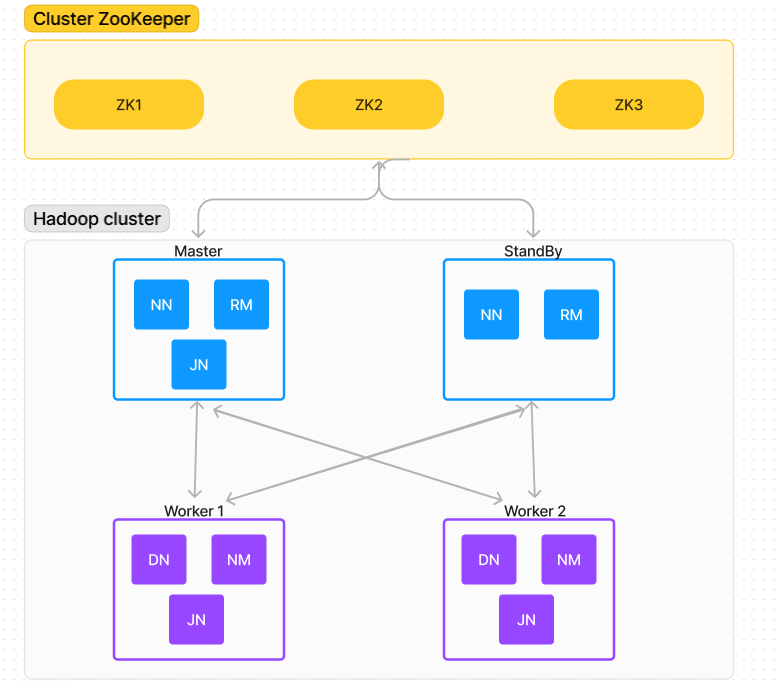
### **Introduction**

Hadoop is a robust framework designed for the storage and processing of large datasets. In a traditional Hadoop cluster, the NameNode serves as a critical component, and its failure can disrupt the entire system. High Availability (HA) addresses this issue by ensuring continuous operation despite hardware or software failures. This report details how HA provides fault tolerance in Hadoop cluster architecture and discusses essential configuration details for both ZooKeeper and Hadoop.

### **Cluster Architecture**

The Hadoop cluster's architecture, featuring a 3-node ZooKeeper cluster and a Hadoop setup with an Active NameNode, two DataNodes, and a Standby NameNode, forms the foundation for HA. Below is a detailed breakdown of the key components:

* ZooKeeper Cluster (3 Nodes): This distributed coordination service acts as the central nervous system for HA. It maintains the cluster state, manages leader election for the NameNode, and coordinates failover.
* Active NameNode: This primary node manages the HDFS (Hadoop Distributed File System) namespace, plays a crucial role in data storage and retrieval, and interacts with clients.
* Standby NameNode: This secondary node remains synchronized with the edits made to the HDFS metadata by the Active NameNode. It takes over seamlessly in the event of an Active NameNode failure.
* DataNodes (2): These nodes store the actual data blocks across the cluster. Replication ensures data availability even if a DataNode fails.



### **Resource Allocation**

Distributing resources across two physical machines enhances HA and fault tolerance:

* Machine 1: Hosts the ZooKeeper cluster and the Standby NameNode.
* Machine 2: Runs the Active NameNode and the two DataNodes.

### **Benefits of High Availability in Hadoop**

* Minimized Downtime: HA ensures the Hadoop cluster remains operational during hardware or software failures. The Standby NameNode swiftly takes over, minimizing service disruption.
* Increased Fault Tolerance: Multiple NameNodes and a ZooKeeper cluster enhance the cluster's ability to withstand failures without compromising data integrity or accessibility.
* Improved Scalability: The HA architecture allows for the addition of more DataNodes to the cluster for increased storage capacity without affecting overall availability.

### **ZooKeeper Configuration**

A robust and secure ZooKeeper cluster requires careful configuration:

1. Software Installation:
   * Download and install ZooKeeper on each of the 3 machines.
   * Create a directory (e.g., Zookeeper\_node1) for each machine.
2. ZooKeeper Configuration (zoo.cfg file):
   * tickTime: Defines heartbeat frequency for election and synchronization.
   * dataDir: Directory to store transaction logs and snapshots.
   * clientPort: Port for clients to connect to ZooKeeper.
   * maxClientCnxns: Maximum client connections.
   * initLimit and syncLimit: Parameters for the leader election process.
   * Server Definitions: Specify each ZooKeeper server in the cluster.
3. Unique Server ID:
   * Create a file named /tmp/zookeeper/myid on each machine.
   * Write a unique number (1, 2, or 3) into the myid file for each machine.
4. Starting ZooKeeper Server:
   * Navigate to the ZooKeeper bin directory and run ./zkServer.sh start on each machine.
5. Verifying the Cluster (Optional):
   * Use the command echo stat | nc zk\_ip 2181 to check the leader/follower status of each server.
   * Use ./zkCli.sh to connect to the cluster and interact with ZooKeeper.

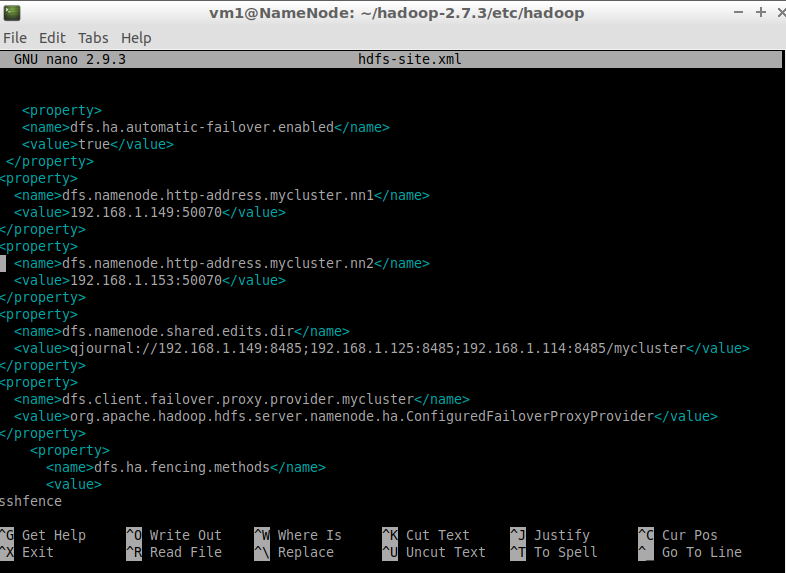
### **Hadoop Configuration**

Configuring Hadoop for HA involves updating multiple configuration files:

#### **HDFS High Availability (HA)**

hdfs-site.xml:

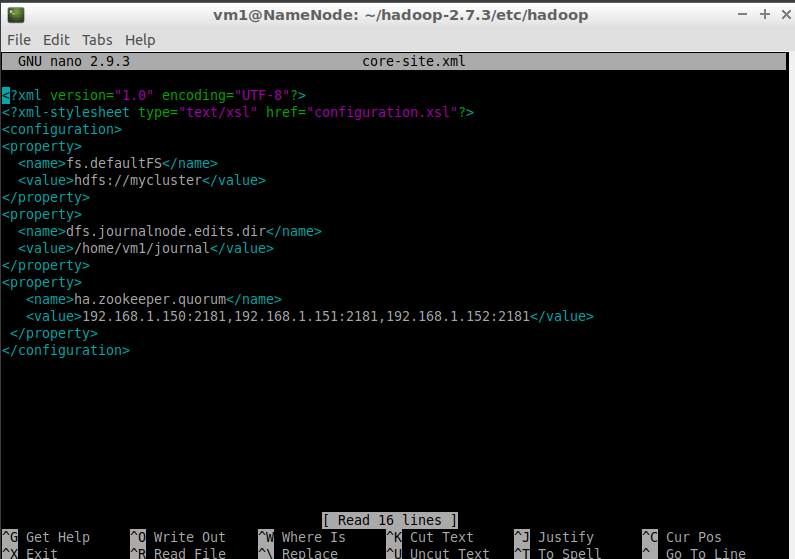
* Replication Factor: Set to 2 to ensure data blocks are replicated across multiple DataNodes.
* NameNode Directories: Paths for storing the NameNode metadata and the edits log.
* Nameservices: Defines a logical name for the cluster, enabling clients to interact with the HA setup.
* NameNode Identification: Specifies the IDs and RPC addresses for the active and standby NameNodes.
* Automatic Failover: Enabled to allow automatic transition between the active and standby NameNodes.
* HTTP Addresses: Specifies HTTP addresses for web-based NameNode status monitoring.
* Shared Edits Directory: Uses a Quorum Journal Manager (QJM) to share the edits log between NameNodes for synchronization.
* Failover Proxy Provider: Configures the client failover handling.
* Fencing Methods: Defines mechanisms to prevent split-brain scenarios by ensuring that only one NameNode is active at any time.
* SSH Configuration: Specifies the SSH private key files for fencing operations.



#### **Core Configuration**

core-site.xml:

* Default FileSystem (fs.defaultFS): Sets the default filesystem to the logical name of the HDFS cluster (e.g., hdfs://mycluster), allowing clients to connect to the HA cluster.
* JournalNode Edits Directory: Specifies the directory where JournalNodes store their edits logs, crucial for maintaining HA.
* ZooKeeper Quorum: Lists the IP addresses and ports of the ZooKeeper servers that manage the coordination and failover processes.



#### **YARN High Availability (HA)**

yarn-site.xml:

* NodeManager Local and Log Directories: Specifies directories for NodeManager's local storage and logs.
* Log Server URL: Sets the URL for the log server where NodeManager logs can be accessed.
* Auxiliary Services: Configures the auxiliary services required by YARN, such as the MapReduce shuffle service.
* Disk Health Checker: Configures the maximum disk utilization for health checks.
* ResourceManager HA Enabled: Enables HA for the ResourceManager.
* Cluster ID and ResourceManager IDs: Defines a cluster ID and identifies the active and standby ResourceManagers.
* ResourceManager Hostnames and WebApp Addresses: Specifies hostnames and web application addresses for both active and standby ResourceManagers.
* Recovery Enabled: Ensures that the ResourceManager can recover its state after a failure.
* State Store Class: Uses ZooKeeper-based state store for ResourceManager state management.
* ZooKeeper Quorum: Lists the IP addresses and ports of ZooKeeper servers used by YARN for coordination.

