**ASSIGNMENT-3.4**

1. **HDFS federation and High availability**

***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.
* In order to scale the name service horizontally, federation uses multiple independent namenodes/namespaces.
* The namenodes are federated, that is, the namenodes are independent and don’t require coordination with each other.
* The datanodes are used as common storage for blocks by all the namenodes. Each datanode registers with all the namenodes in the cluster.
* Datanodes send periodic heartbeats and block reports and handles commands from the namenodes.
* **Key Benefits**
* **Scalability and isolation**
* **Generic storage service**
* **Design simplicity**

***HDFS Availability:***

* The HDFS NameNode High Availability feature enables you to run redundant NameNodes in the same cluster in an Active/Passive configuration with a hot standby
* This eliminates the NameNode as a potential single point of failure (SPOF) in an HDFS cluster.
* Formerly, if a cluster had a single NameNode, and that machine or process became unavailable, the entire cluster would be unavailable until the NameNode was either restarted or started on a separate machine.
* This situation impacted the total availability of the HDFS cluster in two major ways:
* In the case of an unplanned event such as a machine crash, the cluster would be unavailable until an operator restarted the NameNode.
* Planned maintenance events such as software or hardware upgrades on the NameNode machine would result in periods of cluster downtime.
* HDFS NameNode HA avoids this by facilitating either a fast failover to the new NameNode during machine crash, or a graceful administrator-initiated failover during planned maintenance.

1. **How HDFS handles failures while writing data**

* The pipeline is closed and any packets in the ack queue are added to the front of the data queue.
* The current block on the good DataNodes is given a new identity, which is communicated to the NameNode.
* The failed DataNode is removed from the pipeline, and a new pipeline is constructed from the two good DataNodes.
* The DataNode is identified as failed one as follows:
* Data node passes a heartbeat signal to Name node in an interval of specified time (Usually 2 minutes), which helps the Name node to determine that the data node is alive & functional.
* When Name node does not receive heartbeat signals from Data node, it assumes that the data node is either dead or non-functional.
* The remainder of the block’s data is written to the good DataNodes in the pipeline.
* The NameNode notices that the block is under-replicated, and it arranges for a further replica to be created on another node.
* As long as dfs.namenode.replication.min replicas (which defaults to 1) are written, the write will succeed.
* The block will be asynchronously replicated across the cluster until its target replication factor is reached (dfs.replication, which defaults to 3).
* In the process of transferring the data blocks from dead name node to other data nodes, Name node is not involved.