1.Explain HDFS federation and High availability:

HDFS FEDERATION:

\*It improves the existing hdfs through the separation of name space and storage.

\*It also expands the architecture of an existing HDFS cluster to allow new implementations and use cases.

\*It enables the block storage layer.

\*The prior HDFS architecture allows only a single namespace for the entire cluster.

\*But HDFS Federation supports multiple namespaces in the cluster to improve scalability and isolation.

\*By adding multiple name nodes we can overcome the bottleneck issues of namespace and performance issues.

\*Hadoop federation allows scaling the name service horizontally.

\*It uses several namenodes or namespaces which are independent of each other. i.e. they don’t require inter coordination.

\*Advantages of this are Scalability and Isolation,Generic Storage Service,Simple Design.

HIGH AVAILABILITY:

\*High availability in Hadoop 2.0 which eliminates the single point of failure (SPOF) in the Hadoop cluster by setting up a secondary NameNode.

\*It provides support multiple namenode .

\* That brings in an extra NameNode i.e Passive Stand by NameNode to the Hadoop Architecture which is configured for automatic failover.

\*But both active namenode and passive namenode contains meta data.

\*Advantages are Stand for Multiple Failures,Self Recovery from a Failure,Ease of Installation

\*Hadoop High Availability feature tackles the namenode failure problem for all the components in the hadoop stack**.**

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

Block recovery:

\*Blocks are checked for corruption whenever they are read; there are little CRC checksum files created for parts of a block which are validated on read() operations.

\* If a block is found to be corrupt on a read, the dfs client will report this to the namenode, and ask for another block, which will be used instead.

\* The namenode then schedules the uncorrupted block for re-replication, as if it was under replicated.

\*The corrupted block doesn't get deleted until that replication succeeds.

Pipeline recovery:

\*Pipeline recovery is initiated when one or more DataNodes in the pipeline encounter an error in any of the three stages while a block is being written.

\*The three stages are Pipeline setup,Pipeline setup,Close.

\*If the pipeline was created for a new block, the client abandons the block and asks the NameNode for a new block and a new list of DataNodes.

\*The pipeline is reinitialized for the new block.

\*If the pipeline was created to append to a block, the client rebuilds the pipeline with the remaining DataNodes and increments the block’s generation stamp.

->The pipeline is closed and any packets in the acknowledgement 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 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 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)