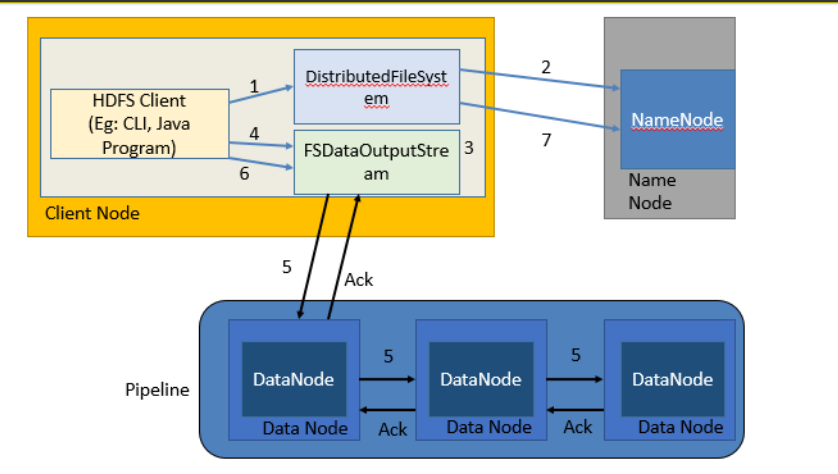
Ans 1. CHECKSUM

Checksum checks the data before storing it to the datanode. HDFS checksums the data written in the node and also read before storing un the node. A separate checksum is created for every dfs.bytes-per-checksum bytes of data. The default value of checksum 512 bytes. When the clients read data from datanodes, they verify checksums as well, comparing them with the ones stored at the datanodes. Provides a protection from corruption, handling failure,scalability.

It works in below way.

1. The HDFS client software implements checksum checker. When a client creates an HDFS file, it computes a checksum of each block of the file and stores these checksums in a separate hidden file in the same HDFS namespace.
2. When a client retrieves file contents, it verifies that the data it received from each DataNode matches the checksum stored in the associated checksum file.
3. If not, then the client can opt to retrieve that block from another DataNode that has a replica of that block.
4. If checksum of another Data node block matches with checksum of hidden file, system will serve these data blocks.

Ans 2.



1.Clients create() in DistributedFileSystem .

2. DistributedFileSystem contacts namenode to create a new file in the filesystem’s namespace, with no blocks associated with it. The namenode performs various checks to make sure the file doesn’t already exist and that the client has the right permissions to create the file. If these checks pass, the namenode makes a record of the new file.

3. DistributedFileSystem returns an FSDataOutputStream for the client to start writing data. FSDataOutputStream uses DFSOutputStream, which handles communication with the datanodes and namenode.

4. Client signals write() method on FSDataOutputStream.

5. DFSOutputStream splits data into packets and writes it to an internal queue called the data queue.

6. When the client has finished writing data, it calls close() on the stream. This flushes all the remaining packets to the datanode pipeline and waits for acknowledgments.

7. DistributedFileSystem contacts the namenode to signal that the file write activity is complete.

Ans 3.

Hdfs failure handling

1. The pipeline is closed and any packets in the ack queue are added to the front of the data queue.

2. The current block on the good datanodes is given a new identity, which is communicated to the namenode

3. The failed datanode is removed from the pipeline, and a new pipeline is constructed from the two good datanodes.

4. The remainder of the block’s data is written to the good datanodes in the pipeline.

5. The namenode notices that the block is under-replicated, and it arranges for a further replica to be created on another node.

6. As long as dfs.namenode.replication.min replicas are written, the write will succeed.

7. The block will be asynchronously replicated across the cluster until its target replication factor is reached.