Task 1:-

Checksums ensures the integrity of a file after it has been transmitted from one storage device to another. This can be across

the Internet or simply between two computers on the same network. if you want to ensure The checksum is calculated using a

hash function and is normally posted along with the download. To verify the integrity of

the file, a user calculates the checksum using a checksum calculator program and then compares the two to make sure they match,

that the transmitted file is exactly the same as the source file, you can use a checksum.

Checksums are used not only to ensure a corrupt-free transmission, but also to ensure that the file has not been tampered with.

When a good checksum algorithm is used, even a tiny change to the file will result in a completely different checksum value.

How Hadoop perform checksum value:-

1. Client –side check summing is performed by Hadoop local file system .

Client used to calculate the checksum for each chunk of data and pass it is passed to data nodes for storage .

2. it means when you write a file called filename, the file system client transparently

It creates a hidden file, .file name.crc, in the same directory containing the checksums for each chunk of the file.

Checksums are verified when the file is read, and if an error is detected, Check sum exception is thrown by local file system .

Task2:- Explain the anatomy of file write to HDFS.

1) Client creates, calls create() on Distributed File System.

2) Distributed File System contacts name node to create a new file in the file system’s namespace, with no blocks associated with it.

The name node 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 name node makes a record of the new file.

3) Distributed File System returns an FS Data Output Stream for the client to start writing data. FS Data Output Stream uses DFS Output Stream,

which handles communication with the data nodes and name node.

4) Client signals write() method on FS Data Output Stream.

5) DFS Output Stream 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 data node pipeline and waits for acknowledgments.

7) Distributed File System contacts the name node to signal that the file write activity is complete.

(1) The name node already knows which blocks the file is made up of ((because DataStreamer had asked for block allocations)),

so it only has to wait for blocks to be minimally replicated before returning successfully.

(2) Closing a file in HDFS performs an implicit hflush().

(3) After a successful return from hflush(), HDFS guarantees that the data written up to that point in

the file has reached all the Data nodes in the write pipeline and is visible to all new readers.

Task 3:- Explain how HDFS handles failures during file write.

HDFS handles failures during file write

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

2) The current block on the good data nodes is given a new identity, that is communicated to the name node

3) The failed data node is removed from the pipeline, and a new pipeline is constructed .

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

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

6) As long as dfs.namenode.replication.minreplicas (which defaults to 1) are written, the write will succeed.