**Assignment 4.5**

**Problem Statement:**

**1. Explain what is checksum and the importance of checksum and how Hadoop performs checksum:**

A checksum is a small-sized datum from a block of digital data. The checksum algorithm is used to calculate the checksum given data as input. A good checksum algorithm will usually output a significantly different value, even for small changes made to the input.

**Importance of checksum:**

The purpose of checksum is detecting errors which may have been introduced during its transmission or storage. Instead of comparing large blocks of data we can compare respective checksums to check if the data is tampered. Checksum plays an important role to verify data integrity. If the computed checksum for the current data input matches the stored value of a previously computed checksum, then there is high probability that the data is not altered. Checksum reduces the time and processing required to check the integrity of the data.

**Hadoop checksums:**

As Hadoop stores big data, it is a challenge to ensure the integrity of such huge bulk of data. Users of Hadoop expect that no data will not be corrupted during storage or processing. However, when the volumes of data flowing through the system are as large, there are chances of data corruption. The checksum is a way to find out if data blocks are tampered in order to ensure the integrity of the data stored in Hadoop.

The HDFS transparently checksums all data written to it and by default verifies checksums when reading data. A separate checksum is created for every io.bytes.per.checksum bytes of data.

When data node receives the data from the client and also during data replication, data nodes verify the data before they store the data and corresponding checksum. The client data is verified again when client wants to read the data from data node. If checksum is changed it means the data is altered.

But as HDFS stores replicas of the data this problem can be overcome by copying the good replica of the data. The Hadoop LocalFileSystem performs client-side check summing. This means that when you write a file called filename, the filesystem client transparently creates a hidden file, .filename.crc, in the same directory containing the checksums for each chunk of the file.

2**. Explain the anatomy of file write to HDFS.**

When Hadoop client wants to write a file into Hadoop, it asks namenode where it can write these files. The namenode checks its metadata and finds the datanodes where it can write this file. This file is divided into blocks if its size is more than the default block size. Then the blocks are copied into various datanodes considering the replication factor. Client request Name node to provide which data nodes to write. After this client writes block to one data node directly. Data node then forwards write request to next data node (replication).Cycle repeats for next block.

The steps can be described as below:

1. The client creates the file on DFS.This is done by calling create () method.

2. DFS makes an RPC call to the namenode to create a new file in the file system’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 file does not exist, then the namenode makes a record of the new file; otherwise, file creation fails and the client is thrown an IOException. The DistributedFileSystem returns an FSDataOutputStream for the client to start writing data to.

3. As the client writes data, the DFSOutputStream splits it into packets, which it writes to an internal queue called the data queue. The data queue is consumed by the DataStreamer, which also gets the list of datanodes to store the replicas from the namenode. The list of datanodes forms a pipeline, and here we’ll assume the replication level is three, so there are three nodes in the pipeline.

4. TheDataStreamer streams the packets to the first datanode in the pipeline, which stores each packet and forwards it to the second datanode in the pipeline. Similarly, the second datanode stores the packet and forwards it to the third (and last) datanode in the pipeline.

5. The DFSOutputStream also maintains an internal queue of packets that are waiting to be acknowledged by datanodes, called the ack queue. A packet is removed from the ack queue only when it has been acknowledged by all the datanodes in the pipeline.

6. When the client has finished writing data, it calls close () on the stream.

7. This action flushes all the remaining packets to the datanode pipeline and waits for acknowledgments before contacting the namenode to signal that the file is complete. The namenode already knows which blocks the file is made up of (because DataStreamer asks for block allocations), so it only has to wait for blocks to be minimally replicated before returning successfully.

**3. Explain how HDFS handles failures during file write.**

Based on the replication factor setting, each block has a specified minimum number of replicas to be maintained. A block is considered safely replicated when the number of replicas of that block has checked in with the name node. If the name node identifies blocks with less than the minimal number of replicas to be maintained, it prepares a list.

After this process, plus an additional few seconds, the name node exits safe mode state. Now the name node replicates these blocks to other data nodes. Data nodes periodically send heartbeat signals and a block-report to the name node.

The name node thus is aware of all the active or functioning data nodes of the cluster and what block each one of them contains.

When data node fails the name node replicates its data to other available data nodes. The name node then updates its file system namespace with the latest information regarding blocks and where they exist now.