Q1. Explain what is checksum and the importance of checksum and how hadoop performs checksum?

Answer:

1. Checksum is a numeric value which is calculated using an algorithm and appended to the data packets before transmission.

2. The Checksum is verified at the receiving end to ensure data integrity.

3. If the data is tampered, the checksum will be different.

4. Hadoop uses CRC-32(Cyclic Redundancy Check). HDFS transparently checksums all data written to it and by default verifies checksums when reading data.

5. The default is 512 bytes, and because a CRC-32 checksum is 4 bytes long, the storage overhead is less than 1%.

Q2-Explain the anatomy of file write to HDFS for writing a file the hdfs client calls a function called as create which when, this is called the DFS(Distributed file system )?

Answer:

1. The client creates the file by calling create() on DistributedFileSystem.

2. DistributedFileSystem makes an RPC call to the namenode to create a new file in the filesystem’s namespace, with no blocks associated with it.

3. 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; otherwise, file creation fails and the client is thrown an IOException. The DistributedFileSystem returns an FSDataOutputStream for the client.

4. As the client writes data, DFSOutputStream splits it into packets, which it writes to an internal queue, called the data queue.

5. The data queue is consumed by the Data Streamer, whose responsibility it is to ask the namenode to allocate new blocks by picking a list of suitable datanodes to store the replicas.

5. The list of datanodes forms a pipeline—we’ll assume the replication level is three, so there are three nodes in the pipeline. The DataStreamer streams the packets to the first datanode in the pipeline, which stores the 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

6. 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.

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

Q3-Explain how hdfs handles when write fails?

Answer:

If a datanode fails while data is being written to it, then the following actions are taken, which are transparent to the client writing the data:

1. First the pipeline is closed, and any packets in the ack queue are added to the front of the data queue so that datanodes that are downstream from the failed node will not miss any packets.

2. The current block on the good datanodes is given a new identity, which is communicated to the namenode, so that the partial block on the failed datanode will be deleted if the failed datanode recovers later on.

3. The failed datanode is removed from the pipeline and the remainder of the block’s data is written to the two good datanodes in the pipeline.

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

5. Subsequent blocks are then treated as normal.