SNS COLLEGE OF TECHNOLOGY

COIMBATORE-35

DEPARTMENT OF INFORMATION TECHNOLOGY

19ITE305 – BIG DATA ANALYTICS

UNIT II: INTRODUCTION TO TECHNOLOGY LANDSCAPE

Topic 5: Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size. We describe the architecture of HDFS and report on experience using HDFS to manage 40 petabytes of enterprise data at Yahoo.

Features:

a. Rack awareness allows consideration of a node's physical location, when allocating storage and scheduling tasks

b. Minimal data motion. MapReduce moves compute processes to the data on HDFS and not the other way around. Processing tasks can occur on the physical node where the data resides. This significantly reduces the network I/O patterns and keeps most of the I/O on the local disk or within the same rack and provides very high aggregate read/write bandwidth.

c. Utilities diagnose the health of the files system and can rebalance the data on different nodes

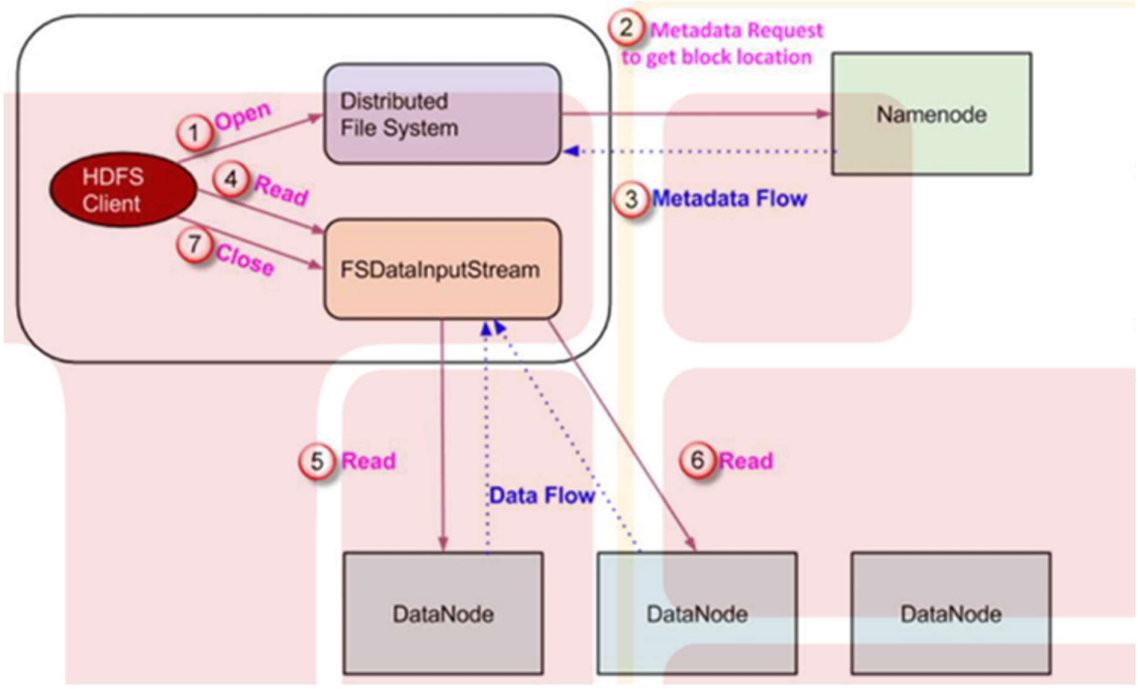
d. Rollback allows system operators to bring back the previous version of HDFS after an upgrade, in case of human or system errors

e. Standby NameNode provides redundancy and supports high availability

f. Highly operable. Hadoop handles different types of cluster that might otherwise require operator intervention. This design allows a single operator to maintain a cluster of 1000s of nodes.

HDFS is a distributed file system for storing very large data files, running on clusters of commodity hardware. It is fault tolerant, scalable, and extremely simple to expand. Hadoop comes bundled with HDFS (Hadoop Distributed File Systems). When data exceeds the capacity of storage on a single physical machine, it becomes essential to divide it across a number of separate machines. A file system that manages storage specific operations across a network of machines is called a distributed file system. HDFS is one such software.

Read Operation In HDFS Data read request is served by HDFS, Name Node, and Data Node. Let's call the reader as a 'client'. Below diagram depicts file read operation in Hadoop.



1.A client initiates read request by calling 'open()' method of File System object; it is an object of type Distributed File System.

2. This object connects to name node using RPC and gets metadata information such as the locations of the blocks of the file. Please note that these addresses are of first few blocks of a file.

3. In response to this metadata request, addresses of the Data Nodes having a copy of that block is returned back.

4. Once addresses of Data Nodes are received, an object of type FS Data Input Stream is returned to the client. FS Data Input Stream contains DFS Input Stream which takes care of interactions with Data Node and Name Node. In step 4 shown in the above diagram, a client invokes 'read()' method which causes DFS Input Stream to establish a connection with the first Data Node with the first block of a file.

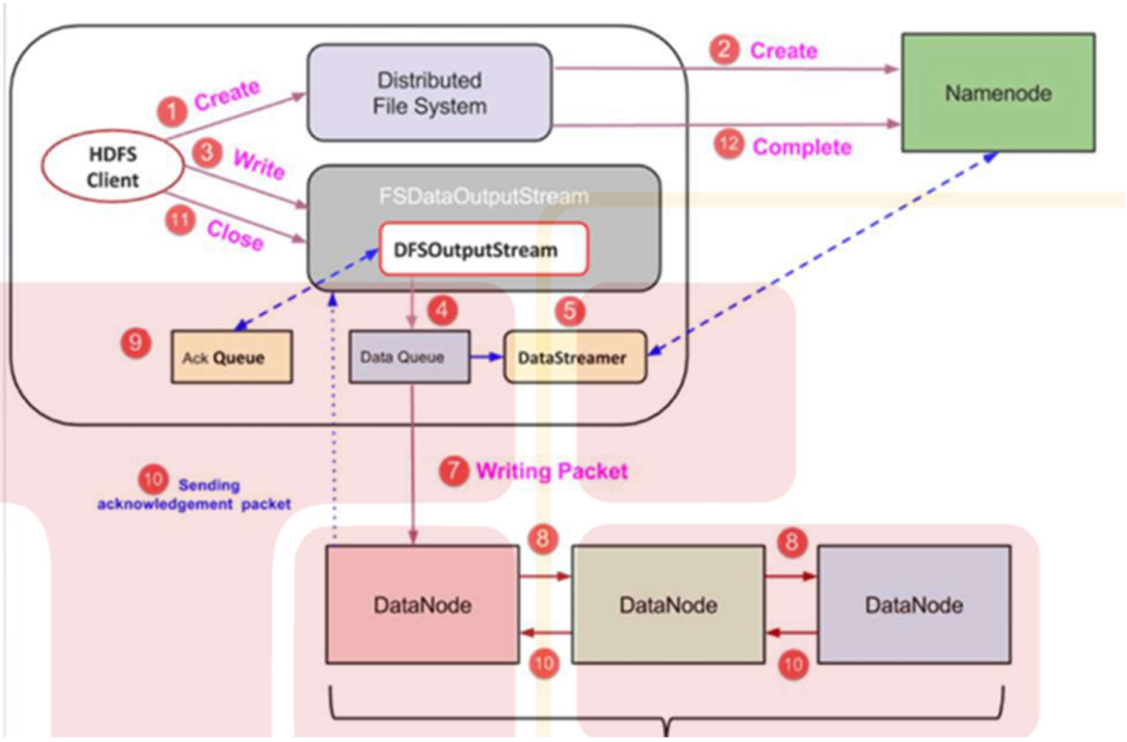
5. Data is read in the form of streams wherein client invokes 'read()' method repeatedly. This process of read() operation continues till it reaches the end of block.

6. Once the end of a block is reached, DFS Input Stream closes the connection and moves on to locate the next Data Node for the next block

7. Once a client has done with the reading, it calls a close() method.

Write Operation

In HDFS Lets understand how data is written into HDFS through files.



1. A client initiates write operation by calling 'create()' method of DistributedFileSystem object which creates a new file - Step no. 1 in the above diagram.

2. Distributed File System object connects to the Name Node using RPC call and initiates new file creation. However, this file creates operation does not associate any blocks with the file. It is the responsibility of Name Node to verify that the file (which is being created) does not exist already and a client has corrected permissions to create a new file. If a file already exists or client does not have sufficient permission to create a new file, then IO Exception is thrown to the client. Otherwise, the operation succeeds and a new record for the file is created by the Name Node.

3. Once a new record in Name Node is created, an object of type FSDataOutputStream is returned to the client. A client uses it to write data into the HDFS. Data write method is invoked (step 3 in the diagram).

4. FS Data Output Stream contains DFSOutputStream object which looks after communication with Data Nodes and Name Node. While the client continues writing data, DFSOutputStream continues creating packets with this data. These packets are enqueued into a queue which is called as Data Queue.

5. There is one more component called Data Streamer which consumes this Data Queue. Data Streamer also asks Name Node for allocation of new blocks thereby picking desirable Data Nodes to be used for replication.

6. Now, the process of replication starts by creating a pipeline using Data Nodes. In our case, we have chosen a replication level of 3 and hence there are 3 Data Nodes in the pipeline.

7. The Data Streamer pours packets into the first Data Node in the pipeline.

8. Every Data Node in a pipeline stores packet received by it and forwards the same to the second Data Node in a pipeline.

9. Another queue, 'Ack Queue' is maintained by DFSOutputStream to store packets which are waiting for acknowledgment from Data Nodes. from

10. Once acknowledgment for a packet in the queue is received from all Data Nodes in the pipeline, it is removed from the 'Ack Queue'. In the event of any Data Node failure, packets this queue are used to reinitiate the operation.

11. After a client is done with the writing data, it calls a close() method (Step 9 in the diagram) Call to close(), results into flushing remaining data packets to the pipeline followed by waiting for acknowledgment.

12. Once a final acknowledgment is received, Name Node is contacted to tell it that the file write operation is complete.