//Jatin Lamba //A20362075

Ans1: The six steps to complete the HDFS read are:

Step 1. The client with the help of open( ) will open the file which it needs to read on the filesystem object. Which is basically DistributedFileSystem for theHDFS.

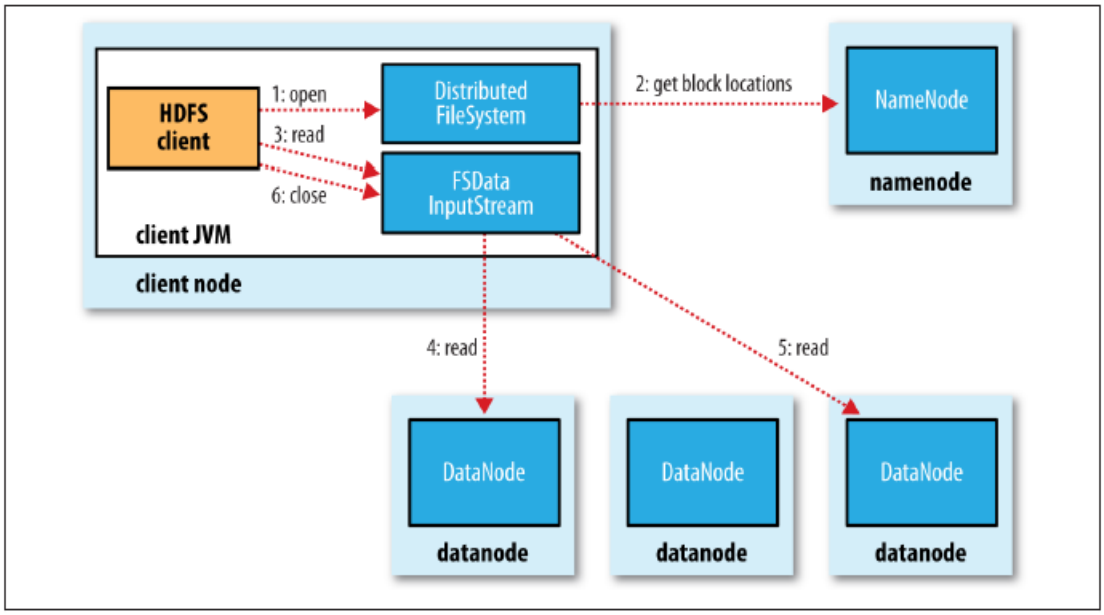
Step 2. Now the DistributedFileSystem will call the namenode using RPCs to get the location of the first few blocks in the file. For each block namenode will be returning with the addresses of the datanode which consist of the copies of that block. The DistributedFileSystem seeks an input stream that will support the file seeks by returning a FSDataInputStream. Which in turn wraps a DFSInputStream which controls the communication between the datanode and namenode.

Step 3. Now the client will call for the read( ) on the stream. So the addresses which are stored for the first few blocks will first connect to the nearest datanode for first file block.

Step 4. Now the data will be streamed back to the client which will again call read( ).

Step 5. As it reaches to the end of the block then the DFSInputstream will close the connection to the datanode and will start it search for the new datanode for the next block.

Step 6. Finally, when the client is done with reading it will call close( ) on the FSDataInputStream.



Ans2: Hadoop makes a simple approach to find the distance between two nodes i.e it is the sum of their distances to their closest ancestor. Hadoop uses a formula to determine distance:

a. Marking points on the graph for calculation of distance.
b. Where d= datacenter
   r= rack

//Jatin Lamba //A20362075

n= node

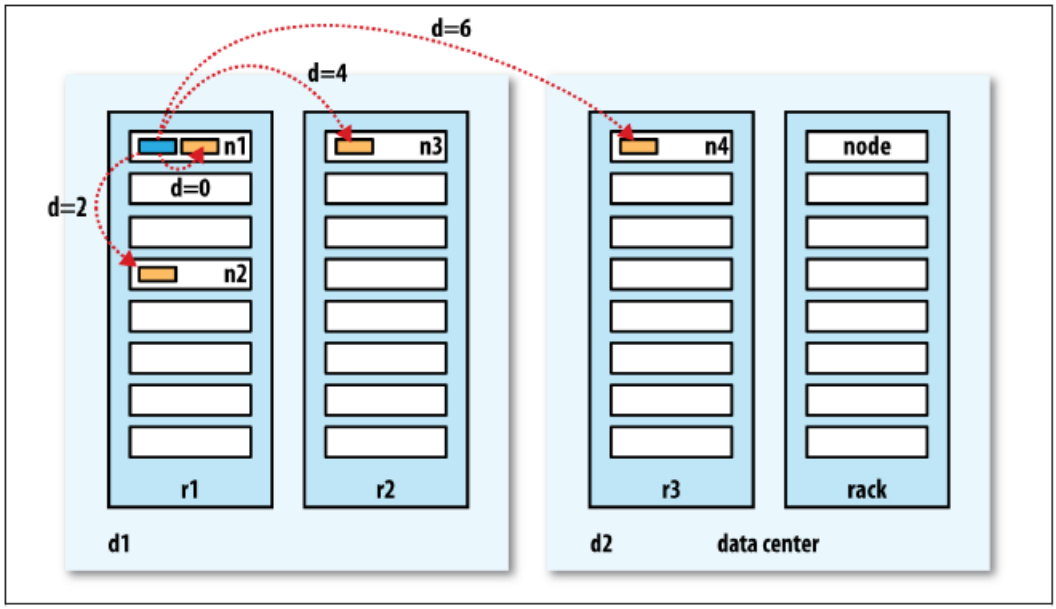
For Example,

a. The distance between (d1/r1/n1) and (d1/r1/n2) is equal to 2 (it is because we have different nodes on the same rack).

b. The distance between (d1/r1/n1) and (d1/r1/n1) is equal to 0 (it is because these processes are on the same system).

c. The distance between (d1/r1/n1) and (d1/r2/n3) is equal to 4 (it is because we have different nodes on different racks).

d. The distance between (d1/r1/n1) and (d2/r3/n6) is equal to 6 (it is because nodes are in different data centers).



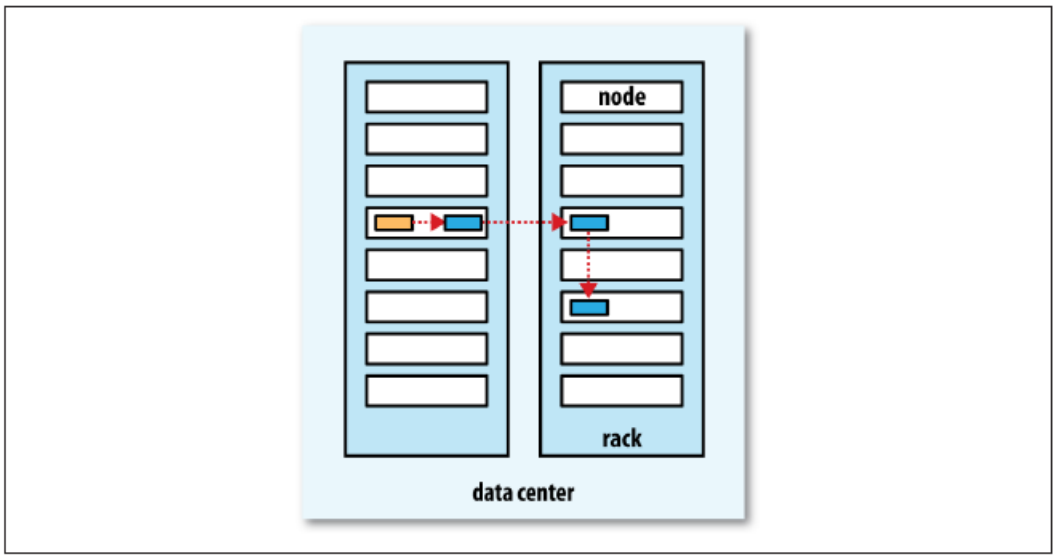
Ans3: The process of namenode replication goes in the following way:

a. First replica will go to the same node. If the client is in remote location, then the replica can go to any random node.

b. Second replica will be placed on a different rack which will be chosen randomly.

c. Third replica will be placed on the same rack as that of second replica but will be placed on a different node which will be chosen randomly.

d. Further replicas will be placed on the randomly chosen nodes in the cluster even the system tries to avoid too many replications on the same rack.



Ans4: The seven steps to complete the HDFS write are:

Step 1. The client will call the create( ) to create the file on the DistributedFileSystem.

Step 2. The DistributedFileSystem will make an RPC call with the namenode for the creation of new file in the system namespace without any blocks associated with it. It also checks to make sure that the file with the name doesn't already exists in the system with the help of various checking process. The DistributedFileSystem seeks an output stream that will support the file seeks by returning a FSDataOutputStream. Which in turn wraps a DFSOutputStream which controls the communication between the datanode and namenode.

Step 3. Now as the client starts to write the data then the DFSOutputstream starts splitting the data into packets which will write it to the internal queue known as data queue. It gets consumed by the datastream which is responsible for requesting the namenode for the allocation of new blocks by a list of datanodes to store the replicas.

Step 4. So the streaming of packets goes in this way that the data streaming streams the packets to the first datanode of the pipeline which will store packets and forward it for replication and the process continues till the last datanode in the pipeline.

Step 5. The queue between the datanodes is also known as ack queue as it refers to the internal queue of packets which are also maintained by the DataOutputStream.

Step 6. Finally, when the client is done with writing data it will call close( ) on the FSDataInput Stream.

Step 7. After the close( ) call it destroys all the remaining packets to the datanode pipeline and waits for the conformation before contacting the main namenode to send the signal of file completion.

//Jatin Lamba //A20362075