What is HDFS

Hadoop comes with a distributed file system called HDFS. In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application.

It is cost effective as it uses commodity hardware. It involves the concept of blocks, data nodes and node name.

Where to use HDFS

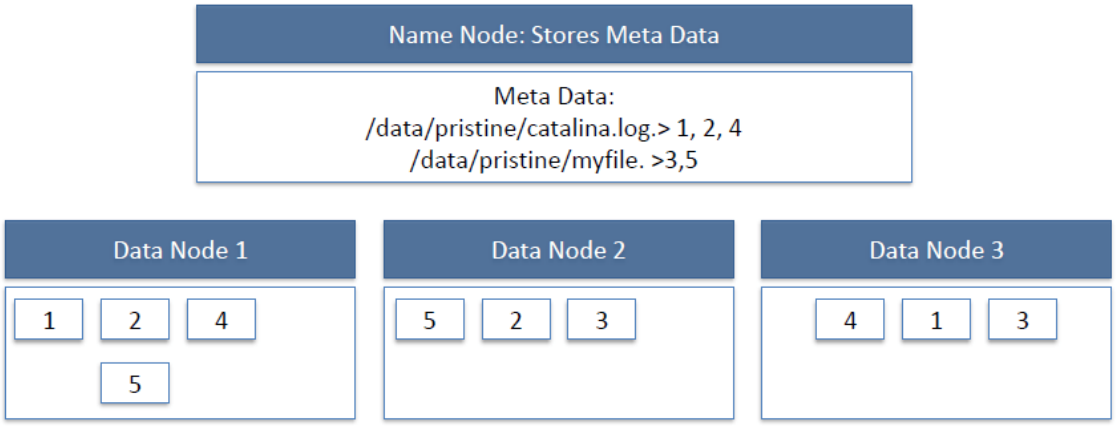
- **Very Large Files:** Files should be of hundreds of megabytes, gigabytes or more.
- **Streaming Data Access:** The time to read whole data set is more important than latency in reading the first. HDFS is built on write-once and read-many-times pattern.
- **Commodity Hardware:**It works on low cost hardware.

Where not to use HDFS

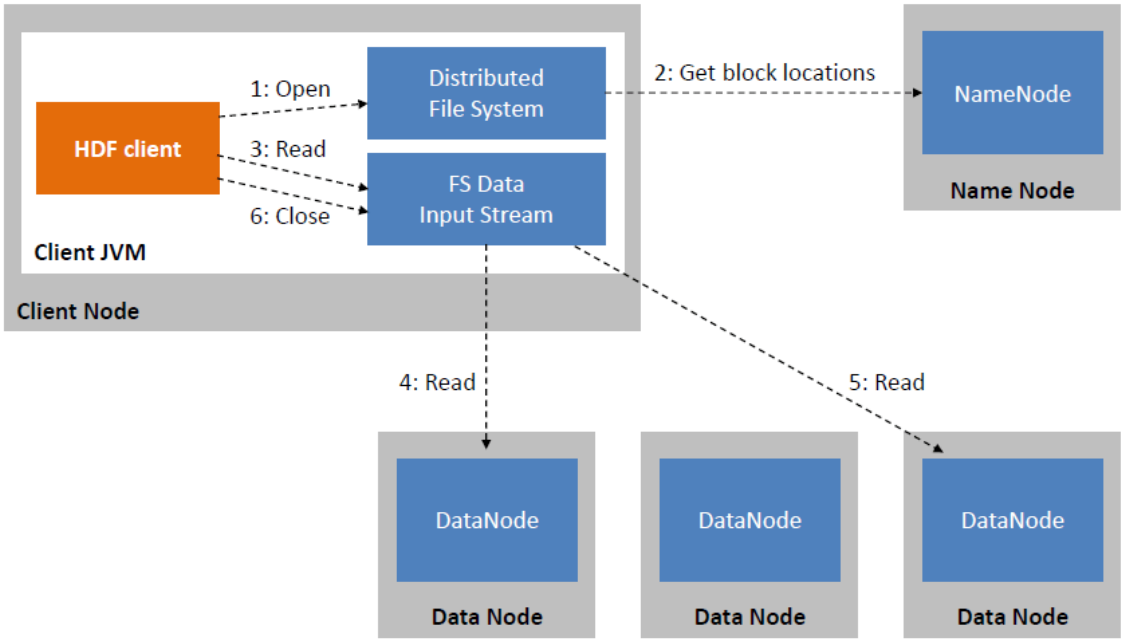
- **Low Latency data access:** Applications that require very less time to access the first data should not use HDFS as it is giving importance to whole data rather than time to fetch the first record.
- **Lots Of Small Files:**The name node contains the metadata of files in memory and if the files are small in size it takes a lot of memory for name node's memory which is not feasible.
- **Multiple Writes:**It should not be used when we have to write multiple times.

HDFS Concepts

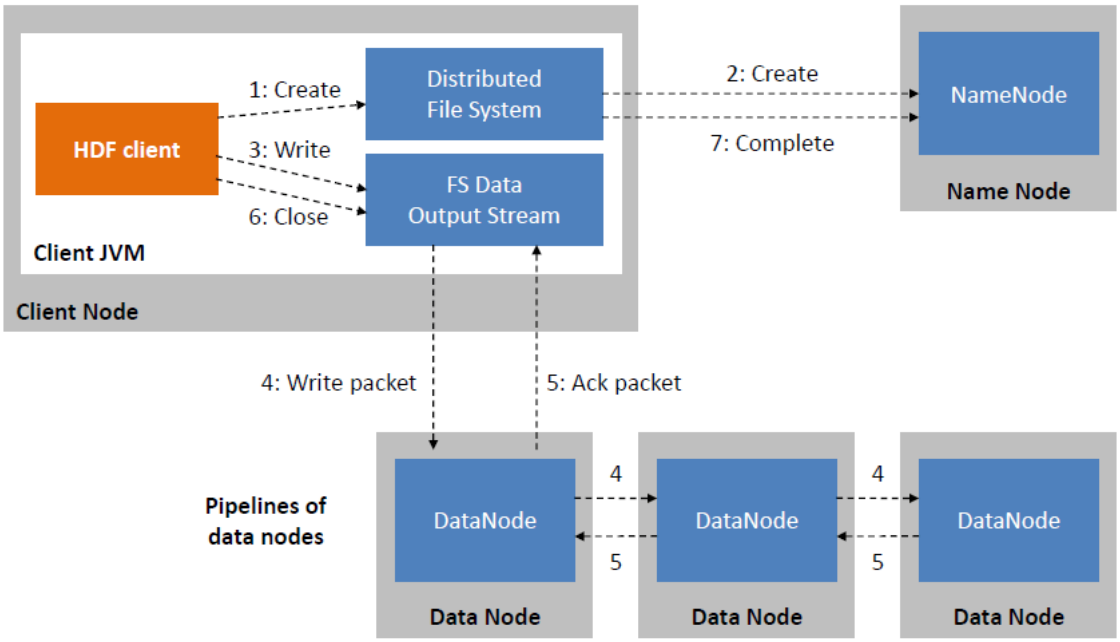
1. **Blocks:** A Block is the minimum amount of data that it can read or write.HDFS blocks are 128 MB by default and this is configurable.Files n HDFS are broken into block-sized chunks,which are stored as independent units.Unlike a file system, if the file is in HDFS is smaller than block size, then it does not occupy full block?s size, i.e. 5 MB of file stored in HDFS of block size 128 MB takes 5MB of space only.The HDFS block size is large just to minimize the cost of seek.
2. **Name Node:** HDFS works in master-worker pattern where the name node acts as master.Name Node is controller and manager of HDFS as it knows the status and the metadata of all the files in HDFS; the metadata information being file permission, names and location of each block.The metadata are small, so it is stored in the memory of name node,allowing faster access to data. Moreover the HDFS cluster is accessed by multiple clients concurrently,so all this information is handled bya single machine. The file system operations like opening, closing, renaming etc. are executed by it.
3. **Data Node:** They store and retrieve blocks when they are told to; by client or name node. They report back to name node periodically, with list of blocks that they are storing. The data node being a commodity hardware also does the work of block creation, deletion and replication as stated by the name node.



HDFS Read Image:



HDFS Write Image:



Since all the metadata is stored in name node, it is very important. If it fails the file system can not be used as there would be no way of knowing how to reconstruct the files from blocks present in data node. To overcome this, the concept of secondary name node arises.

Secondary Name Node: It is a separate physical machine which acts as a helper of name node. It performs periodic check points.It communicates with the name node and take snapshot of meta data which helps minimize downtime and loss of data.

Starting HDFS

The HDFS should be formatted initially and then started in the distributed mode. Commands are given below.

To Format **hdfs namenode -format**

To Start **start-dfs.cmd**

HDFS Basic File Operations

1. Putting data to HDFS from local file system
    - First create a folder in HDFS where data can be put form local file system.

$ hadoop fs -mkdir /user/test

    - Copy the file "data.txt" from a file kept in local folder /usr/home/Desktop to HDFS folder /user/ test

$ hadoop fs -copyFromLocal /usr/home/Desktop/data.txt /user/test

    - Display the content of HDFS folder

$ Hadoop fs -ls /user/test

2. Copying data from HDFS to local file system
    - $ hadoop fs -copyToLocal /user/test/data.txt /usr/bin/data_copy.txt
3. Compare the files and see that both are same
    - $ md5 /usr/bin/data_copy.txt /usr/home/Desktop/data.txt

Recursive deleting

- hadoop fs -rmr <arg>

Example:

- hadoop fs -rmr /user/sonoo/

HDFS Other commands

The below is used in the commands

"<path>" means any file or directory name.

"<path>..." means one or more file or directory names.

"<file>" means any filename.

"<src>" and "<dest>" are path names in a directed operation.

"<localSrc>" and "<localDest>" are paths as above, but on the local file system

- put <localSrc><dest>

Copies the file or directory from the local file system identified by localSrc to dest within the DFS.

- copyFromLocal <localSrc><dest>

Identical to -put

- copyFromLocal <localSrc><dest>

Identical to -put

- moveFromLocal <localSrc><dest>

Copies the file or directory from the local file system identified by localSrc to dest within HDFS, and then deletes the local copy on success.

- get [-crc] <src><localDest>

Copies the file or directory in HDFS identified by src to the local file system path identified by localDest.

- cat <filen-ame>

Displays the contents of filename on stdout.

- moveToLocal <src><localDest>

Works like -get, but deletes the HDFS copy on success.

- setrep [-R] [-w] rep <path>

Sets the target replication factor for files identified by path to rep. (The actual replication factor will move toward the target over time)

- touchz <path>

Creates a file at path containing the current time as a timestamp. Fails if a file already exists at path, unless the file is already size 0.

- test -[ezd] <path>

Returns 1 if path exists; has zero length; or is a directory or 0 otherwise.

- stat [format] <path>

Prints information about path. Format is a string which accepts file size in blocks (%b), filename (%n), block size (%o), replication (%r), and modification date (%y, %Y).