**why HDFS as another filesystem is required?**

Filesystems like NTFS, FAT, FAT32, Ext2, Ext3, Ext4 etc. are local to a particular node or machine. Information stored in one of node in NTFS or Ext will not know what information is stored in another nodes NTFS or Ext filesystem. Apache Hadoop is an open-source software framework that allows to store and process big data in a distributed environment across clusters of computers. Now Hadoop to work seamlessly in a distributed environment HDFS was introduced which works on top of your local filesystem.

**BWhat are the key features of HDFS?**

HDFS is highly fault-tolerant, with high throughput, suitable for applications with large data sets, streaming access to file system data and can be built out of commodity hardware.

**BWhat is Fault Tolerance?**

Suppose you have a file stored in a system, and due to some technical problem that file gets destroyed. Then there is no chance of getting the data back present in that file. To avoid such situations, Hadoop has introduced the feature of fault tolerance in HDFS. In Hadoop, when we store a file, it automatically gets replicated at two other locations also. So even if one or two of the systems collapse, the file is still available on the third system.

**BWhat is a heartbeat in HDFS?**

In general a heartbeat is a signal indicating that it is alive. A datanode sends heartbeat to Namenode. If the Namenode does not receive heart beat then they will decide that there is some problem in datanode.

**BWhat is a ‘block’ in HDFS?**

A ‘block’ is the minimum amount of data that can be read or written. In HDFS, the default block size is 64 MB as contrast to the block size of 8192 bytes in Unix/Linux. Files in HDFS are broken down into block-sized chunks, which are stored as independent units. HDFS blocks are large as compared to disk blocks, particularly to minimize the cost of seeks. If a particular file is 50 mb, will the HDFS block still consume 64 mb as the default size? No, not at all! 64 mb is just a unit where the data will be stored. In this particular situation, only 50 mb will be consumed by an HDFS block and 14 mb will be free to store something else. It is the MasterNode that does data allocation in an efficient manner.

**BWhat is a block and block scanner in HDFS?**

Block - The minimum amount of data that can be read or written is generally referred to as a “block” in HDFS. The default size of a block in HDFS is 64MB. Block Scanner - Block Scanner tracks the list of blocks present on a DataNode and verifies them to find any kind of checksum errors. Block Scanners use a throttling mechanism to reserve disk bandwidth on the datanode

**IHow do you define “block” in HDFS? What is the block size in Hadoop 1 and in Hadoop 2? Can it be changed?**

A “block” is the minimum amount of data that can be read or written. Files in HDFS are broken down into block-sized chunks, which are stored as independent units. Hadoop 1 default block size: 64 MB Hadoop 2 default block size: 128 MB Yes, blocks can be configured. The dfs.block.size parameter can be used in the hdfs-site.xml file to set the size of a block in a Hadoop environment.

**IWhat are the benefits of block transfer?**

A file can be larger than any single disk in the network. There’s nothing that requires the blocks from a file to be stored on the same disk, so they can take advantage of any of the disks in the cluster. Making the unit of abstraction a block rather than a file simplifies the storage subsystem. Blocks provide fault tolerance and availability. To insure against corrupted blocks and disk and machine failure, each block is replicated to a small number of physically separate machines (typically three). If a block becomes unavailable, a copy can be read from another location in a way that is transparent to the client.

**IWhy do we use HDFS for applications having large data sets and not when there are lot of small files?**

HDFS is more suitable for large amount of data sets in a single file as compared to small amount of data spread across multiple files. This is because Namenode is a very expensive high performance system, so it is not prudent to occupy the space in the Namenode by unnecessary amount of metadata that is generated for multiple small files. So, when there is a large amount of data in a single file, name node will occupy less space. Hence for getting optimized performance, HDFS supports large data sets instead of multiple small files.

**AReplication causes data redundancy, then why is it pursued in HDFS?**

HDFS works with commodity hardware (systems with average configurations) that has high chances of getting crashed any time. Thus, to make the entire system highly fault-tolerant, HDFS replicates and stores data in different places. Any data on HDFS gets stored at least 3 different locations. So, even if one of them is corrupted and the other is unavailable for some time for any reason, then data can be accessed from the third one. Hence, there is no chance of losing the data. This replication factor helps us to attain the feature of Hadoop called Fault Tolerant.

**ASince the data is replicated thrice in HDFS, does it mean that any calculation done on one node will also be replicated on the other two?**

No, calculations will be done only on the original data. The master node will know which node exactly has that particular data. In case, if one of the nodes is not responding, it is assumed to be failed. Only then, the required calculation will be done on the second replica.

**AIf we want to copy 10 blocks from one machine to another, but another machine can copy only 8.5 blocks, can the blocks be broken at the time of replication?**

In HDFS, blocks cannot be broken down. Before copying the blocks from one machine to another, the Master node will figure out what is the actual amount of space required, how many block are being used, how much space is available, and it will allocate the blocks accordingly.

**IExplain how indexing in HDFS is done?**

Hadoop has its own way of indexing data. Depending on the block size, HDFS will continue storing the last part of the data. HDFS stores the last part of the data that further points to the address where the next part of data chunk is stored.

**IHow do you do a file system check in HDFS?**

FSCK command is used to do a file system check in HDFS. It is a very useful command to check the health of the file, block names and block locations.

hdfs fsck /dir/hadoop-test -files -blocks –locations

## BWhat is a commodity hardware?

Commodity hardware is a non-expensive system which is not of high quality or high-availability. Commodity hardware includes RAM because there will be some services which will be running on RAM. Hadoop can be installed in any average commodity hardware. We don’t need super computers or high-end hardware to work on Hadoop to execute jobs.

## BWhat is a rack?

Rack is a storage area with all the datanodes put together. These datanodes can be physically located at different places. Rack is a physical collection of datanodes which are stored at a single location. There can be multiple racks in a single location.

## IWhat is rack awareness?

Rack awareness is the way in which the namenode decides how to place blocks based on the rack definitions Hadoop will try to minimize the network traffic between datanodes within the same rack and will only contact remote racks if it has to. The namenode is able to control this due to rack awareness.

## IOn what basis data will be stored on a rack?

When the client is ready to load a file into the cluster, the content of the file will be divided into blocks. Now the client consults the Namenode and gets 3 datanodes for every block of the file which indicates where the block should be stored. While placing the datanodes, the key rule followed is “for every block of data, two copies will exist in one rack, third copy in a different rack“. This rule is known as “Replica Placement Policy“.

## AHow do you define “rack awareness” in Hadoop?

It is the manner in which the “Namenode” decides how blocks are placed, based on rack definitions to minimize network traffic between “DataNodes” within the same rack. Let’s say we consider replication factor 3 (default), the policy is that “for every block of data, two copies will exist in one rack, third copy in a different rack”. This rule is known as the “Replica Placement Policy”.

## IExplain the difference between NAS and HDFS.

* NAS runs on a single machine and thus there is no probability of data redundancy whereas HDFS runs on a cluster of different machines thus there is data redundancy because of the replication protocol.
* HDFS data blocks are distributed across local drives of all machines in a cluster while NAS data is stored on dedicated hardware.
* In NAS data is stored independent of the computation and hence Hadoop MapReduce cannot be used for processing whereas HDFS works with Hadoop MapReduce as the computations in HDFS are moved to data.

## IIs HDFS is a replacement for your local filesystem?

HDFS by no means is a replacement for your filesystem. OS still reply on local filesystem. HDFS still go through the local filesystem for saving each block. HDFS is placed on top of local filesystem.

## AHow a small file is stored in HDFS?

A 1 MB file stored with a block size of 128 MB uses 1 MB of disk space, not 128 MB. In HDFS the block size is more about how a single file is split up / partitioned, not about some reserved part of the file system.

## Aif it stores 1Mb file as 1Mb file then why the concept of block size, come into picture in hadoop?

Because a 1GB file will be stored in blocks of 128MB (for example), with each block being replicated on 3 (typically) nodes in the cluster. The idea being that it's quicker for 8 tasks to process 128MB each, rather than a single task processing all 1G.

## AWhat happens to the memory not allocated in HDFS when the file size is not a multiple of 64 MB or the default file storage block size?

The block sizes are going to be 64MB and 36MB. The second block is not of 64MB size (assuming default size of a block is 64 MB). HDFS isn't going to waste disk space by allocating a 64MB block for a smaller chunk.

The Name Node does not care about the size of the data nodes. A disk file is created on each of the Data Node of the appropriate size. An easier way to think of this is to consider writing a file of 65 MB. HDFS wouldn't allocate two 64 MB chunks to allocate storage of 1MB extra space. It would create a new block of 1MB size.

## Iwhat happens when 1MB file is getting stored in HDFS?

When 1MB file is stored in HDFS, 1MB disk space is consumed eventhough the HDFS block size is 128MB(assuming 128MB block size is configured).

## Iwhat happens when 1KB file is getting stored in HDFS?

When 1KB file is stored in HDFS, 4KB disk space is consumed eventhough the HDFS block size is 128MB(assuming 128MB block size is configured). Why 4KB, why not 1KB because HDFS stores data in underneath local file system. Assuming ext4 has local file system in linux, its block size is 4KB. It consumes full 4KB as block, leaving 4KB as empty.

## Iwhat happens when 130MB file is getting stored in hdfs?

When 130MB file is stored in HDFS, 130MB disk space is consumed consisting of 2 HDFS blocks of size 128MB and 2MB(assuming 128MB block size is configured). Yes remaining 126MB is not wasted as in local file system.

## Awhy block size is 128MB in HDFS, why not 4KB? Why is a Block in HDFS So Large?

Block size is just an indication to HDFS how to split up and distribute the files across the cluster. OS will lay blocks in contiguos locations, as a result read and write in HDFS will be faster because blocks are laid out next to each other. Then disk head doesnt have to seek and position itsef over and over again for blocks. This is a huge benefit because of HDFS design. So to reduce seek time HDFS block size is kept so high.

**What is throughput? How does HDFS get a good throughput?**

Throughput is the amount of work done in a unit time. It describes how fast the data is getting accessed from the system and it is usually used to measure performance of the system. In HDFS, when we want to perform a task or an action, then the work is divided and shared among different systems. So all the systems will be executing the tasks assigned to them independently and in parallel. So the work will be completed in a very short period of time. In this way, the HDFS gives good throughput. By reading data in parallel, we decrease the actual time to read data tremendously.

**AWhat is streaming access?**

As HDFS works on the principle of ‘Write Once, Read Many‘, the feature of streaming access is extremely important in HDFS. HDFS focuses not so much on storing the data but how to retrieve it at the fastest possible speed, especially while analyzing logs. In HDFS, reading the complete data is more important than the time taken to fetch a single record from the data.

**IDo we need to place 2nd and 3rd data in rack 2 only?**

Yes, this is to avoid datanode failure.

**AWhat if rack 2 and datanode fails?**

If both rack2 and datanode present in rack 1 fails then there is no chance of getting data from it. In order to avoid such situations, we need to replicate that data more number of times instead of replicating only thrice. This can be done by changing the value in replication factor which is set to 3 by default.

**AWhat is ‘Key value pair’ in HDFS?**

Key value pair is the intermediate data generated by maps and sent to reduces for generating the final output.

**AWhat happens when two clients try to access the same file on the HDFS?**

HDFS supports exclusive writes only. When the first client contacts the “Namenode” to open the file for writing, the “Namenode” grants a lease to the client to create this file. When the second client tries to open the same file for writing, the “Namenode” will notice that the lease for the file is already granted to another client, and will reject the open request for the second client

**AWhy do we sometimes get a “file could only be replicated to 0 nodes, instead of 1” error?**

This happens because the “Namenode” does not have any available DataNodes.

**AHow does one switch off the “SAFEMODE” in HDFS?**

**You use the command: hadoop dfsadmin –safemode leave**

**AWhy is it that in HDFS, ‘Reading‘ is done in parallel and ‘Writing‘ is not in HDFS?**

Using the MapReduce program, the file can be read by splitting its blocks. But while writing as the incoming values are not yet known to the system mapreduce cannot be applied and no parallel writing is possible.

**ACopy a directory from one node in the cluster to another**

Use ‘-distcp’ command to copy,

**AIs there a hdfs command to see available free space in hdfs**

hadoop dfsadmin -report

**AWhat does "file could only be replicated to 0 nodes, instead of 1" mean?**

The namenode does not have any available DataNodes.

**AWhat are Problems with small files and HDFS?**

HDFS is not good at handling large number of small files. Because every file, directory and block in HDFS is represented as an object in the namenode’s memory, each of which occupies approx 150 bytes So 10 million files, each using a block, would use about 3 gigabytes of memory. when we go for a billion files the memory requirement in namenode cannot be met.

**AHow can you overwrite the replication factors in HDFS?**

The replication factor in HDFS can be modified in 2 ways

* Using the Hadoop FS Shell, replication factor can be changed per file basis using the below command

$hadoop fs –setrep –w 2 /my/test\_file

Note: test\_file is the filename whose replication factor will be set to 2

* Using the Hadoop FS Shell, replication factor of all files under a given directory can be modified using the below command-

$hadoop fs –setrep –w 5 /my/test\_dir

Note:test\_dir is the name of the directory and all the files in this directory will have a replication factor set to 5

**AExplain what happens if during the PUT operation, HDFS block is assigned a replication factor 1 instead of the default value 3.**

Replication factor is a property of HDFS that can be set accordingly for the entire cluster to adjust the number of times the blocks are to be replicated to ensure high data availability. For every block that is stored in HDFS, the cluster will have n-1 duplicated blocks. So, if the replication factor during the PUT operation is set to 1 instead of the default value 3, then it will have a single copy of data. Under these circumstances when the replication factor is set to 1 ,if the DataNode crashes under any circumstances, then only single copy of the data would be lost.

**IWhat is the process to change the files at arbitrary locations in HDFS?**

HDFS does not support modifications at arbitrary offsets in the file or multiple writers but files are written by a single writer in append only format i.e. writes to a file in HDFS are always made at the end of the file.

**IWhat happens to a NameNode that has no data?**

There does not exist any NameNode without data. If it is a NameNode then it should have some sort of data in it.

**IExplain what is difference between an Input Split and HDFS Block?**

Logical division of data is known as Split while physical division of data is known as HDFS Block

**IMention what is the best way to copy files between HDFS clusters?**

The best way to copy files between HDFS clusters is by using multiple nodes and the distcp command, so the workload is shared.

**AHow do you overwrite replication factor?**

There are few ways to do this. Look at the below illustration.

hadoop fs -setrep -w 5 -R hadoop-test

hadoop fs -Ddfs.replication=5 -cp hadoop-test/test.csv hadoop-test/test\_with\_rep5.csv

**AHow to configure Replication Factor in HDFS?**

hdfs-site.xml is used to configure HDFS. Changing the dfs.replication property in hdfs-site.xml will change the default replication for all files placed in HDFS. You can also modify the replication factor on a per-file basis using the Hadoop FS Shell:[training@localhost ~]$ hadoopfs –setrep –w 3 /my/fileConversely, you can also change the replication factor of all the files under a directory.

[corejavaguru@localhost ~]$ hadoopfs –setrep –w 3 -R /my/dir