**1. What is HDFS?**

[***Hadoop***](https://www.edureka.co/blog/hadoop-tutorial/) Distributed file system or [***HDFS***](https://www.edureka.co/blog/hdfs-tutorial) is a Java based distributed file system that allows us to store Big data across multiple nodes in a Hadoop cluster. So, if I install Hadoop, I will get HDFS as an underlying storage system for storing the huge data sets in the distributed environment.

**2. What are the key features of HDFS?**

♣**Tip**: You should also explain the features briefly while listing different HDFS features.

Some of the prominent features of HDFS are as follows:

* ***Cost effective and Scalable:*** HDFS, in general, is deployed on a commodity hardware. So, it is very economical in terms of the cost of ownership of the project. Also, one can scale the cluster by adding more nodes.
* ***Variety and Volume of Data:*** HDFS is all about storing huge data i.e. Terabytes & Petabytes of data and different kinds of data. So, I can store any type of data into HDFS, be it structured, unstructured or semi structured.
* ***Reliability and Fault Tolerance:*** HDFS divides the given data into data blocks, replicates it and stores it in a distributed fashion across the Hadoop cluster. This makes HDFS very reliable and fault tolerant.
* ***High Throughput:***Throughput is the amount of work done in a unit time. HDFS provides high throughput access to application data.

**3. Explain the HDFS Architecture and list the various HDFS daemons in HDFS cluster?**

While listing various HDFS daemons, you should also talk about their roles in brief. Here is how you should answer this question:

Apache Hadoop [***HDFS Architecture***](https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/) follows a Master/Slave topology where a cluster comprises a single NameNode (Master node or daemon) and all the other nodes are DataNodes (Slave nodes or daemons).  Following daemon runs in HDFS cluster:

* **NameNode:**It is the master daemon that maintains and manages the data block present in the DataNodes.
* **DataNode:** DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is a commodity hardware, that is responsible of storing the data as blocks.
* **Secondary NameNode:**The Secondary NameNode works concurrently with the primary NameNode as a *helperdaemon*. It performs checkpointing.

**4. What is checkpointing in Hadoop?**

Checkpointing is the process of combining the Edit Logs with the FsImage (File system Image). It is performed by the Secondary NameNode.

**5. What is a NameNode in Hadoop?**

The [***NameNode***](https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/) is the master node that manages all the DataNodes (slave nodes). It records the metadata information regarding all the files stored in the cluster (on the DataNodes), e.g. The location of blocks stored, the size of the files, permissions, hierarchy, etc.

**6. What is a DataNode?**

[***DataNodes***](https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/)are the slave nodes in HDFS. It is a commodity hardware that provides storage for the data. It serves the read and write request of the HDFS client.

**7. Is Namenode machine same as DataNode machine as in terms of hardware?**

Unlike the DataNodes, a NameNode is a highly available server that manages the File System Namespace and maintains the metadata information. Therefore, NameNode requires higher RAM for storing the metadata information corresponding to the millions of HDFS files in the memory, whereas the DataNode needs to have a higher disk capacity for storing huge data sets.

**8. What is the difference between NAS (Network Attached Storage) and HDFS?**

Here are the key differences between NAS and HDFS:

* Network-attached storage (NAS) is a file-level computer data storage server connected to a computer network providing data access to a heterogeneous group of clients. NAS can either be a hardware or software which provides a service for storing and accessing files. Whereas Hadoop Distributed File System (HDFS) is a distributed file system to store data using commodity hardware.
* In HDFS, data blocks are distributed across all the machines in a cluster. Whereas in NAS, data is stored on a dedicated hardware.
* HDFS is designed to work with MapReduce paradigm, where computation is moved to the data. NAS is not suitable for MapReduce since data is stored separately from the computations.
* HDFS uses commodity hardware which is cost effective, whereas a NAS is a high-end storage devices which includes high cost.

**9. What is the difference between traditional RDBMS and Hadoop?**

This question seems to be very easy, but in an interview these simple questions matter a lot. So, here is how you can answer the very question:

|  |  |  |
| --- | --- | --- |
|  | **RDBMS** | **Hadoop** |
| **Data Types** | RDBMS relies on the structured data and the schema of the data is always known. | Any kind of data can be stored into Hadoop i.e. Be it structured, unstructured or semi-structured. |
| **Processing** | RDBMS provides limited or no processing capabilities. | Hadoop allows us to process the data which is distributed across the cluster in a parallel fashion. |
| **Schema on Read Vs. Write** | RDBMS is based on ‘schema on write’ where schema validation is done before loading the data. | On the contrary, Hadoop follows the schema on read policy. |
| **Read/Write Speed** | In RDBMS, reads are fast because the schema of the data is already known. | The writes are fast in HDFS because no schema validation happens during HDFS write. |
| **Cost** | Licensed software, therefore, I have to pay for the software. | Hadoop is an open source framework. So, I don’t need to pay for the software. |
| **Best Fit Use Case** | RDBMS is used for OLTP (Online Trasanctional Processing) system. | Hadoop is used for Data discovery, data analytics or OLAP system. |

**10. What is throughput? How does HDFS provides good throughput?**

Throughput is the amount of work done in a unit time. HDFS provides good throughput because:

* The HDFS is based on Write Once and Read Many Model, it simplifies the data coherency issues as the data written once can’t be modified and therefore, provides high throughput data access.
* In Hadoop, the computation part is moved towards the data which reduces the network congestion and therefore, enhances the overall system throughput.

**11. What is Secondary NameNode? Is it a substitute or back up node for the NameNode?**

Here, you should also mention the function of the Secondary NameNode while answering the later part of this question so as to provide clarity:

A [***Secondary NameNode***](https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/) is a helper daemon that performs checkpointing in HDFS. No, it is not a backup or a substitute node for the NameNode. It periodically, takes the edit logs (meta data file) from NameNode and merges it with the FsImage (File system Image) to produce an updated FsImage as well as to prevent the Edit Logs from becoming too large.

**12. What do you mean by meta data in HDFS? List the files associated with metadata.**

The metadata in HDFS represents the structure of HDFS directories and files. It also includes the various information regarding HDFS directories and files such as ownership, permissions, quotas, and replication factor.

♣**Tip:** While listing the files associated with metadata, give a one line definition of each metadata file.

There are two files associated with metadata present in the NameNode:

* **FsImage:** It contains the complete state of the file system namespace since the start of the NameNode.
* **EditLogs:** It contains all the recent modifications made to the file system with respect to the recent FsImage.

**13. What is the problem in having lots of small files in HDFS?**

As we know, the NameNode stores the metadata information regarding file system in the RAM. Therefore, the amount of memory produces a limit to the number of files in my HDFS file system. In other words, too much of files will lead to the generation of too much meta data and storing these meta data in the RAM will become a challenge. As a thumb rule, metadata for a file, block or directory takes 150 bytes.

**14. What is a heartbeat in HDFS?**

Heartbeats in HDFS are the signals that are sent by DataNodes to the NameNode to indicate that it is functioning properly (alive). By default, the heartbeat interval is 3 seconds, which can be configured using dfs.heartbeat.interval in hdfs-site.xml.

**15. How would you check whether your NameNode is working or not?**

There are many ways to check the status of the NameNode. Most commonly, one uses the **jps command** to check the status of all the daemons running in the HDFS. Alternatively, one can visit the NameNode’s Web UI for the same.

**16. What is a block?**

You should begin the answer with a general definition of a block. Then, you should explain in brief about the blocks present in HDFS and also mention their default size.

[***Blocks***](https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/) are the smallest continuous location on your hard drive where data is stored. HDFS stores each file as blocks, and distribute it across the Hadoop cluster. The default size of a block in HDFS is 128 MB (Hadoop 2.x) and 64 MB (Hadoop 1.x) which is much larger as compared to the Linux system where the block size is 4KB. The reason of having this huge block size is to minimize the cost of seek and reduce the meta data information generated per block.

**17. Suppose there is file of size 514 MB stored in HDFS (Hadoop 2.x) using default block size configuration and default replication factor. Then, how many blocks will be created in total and what will be the size of each block?**

Default block size in Hadoop 2.x is 128 MB. So, a file of size 514 MB will be divided into 5 blocks ( 514 MB/128 MB) where the first four blocks will be of 128 MB and the last block will be of 2 MB only. Since, we are using the default replication factor i.e. 3, each block will be replicated thrice. Therefore, we will have 15 blocks in total where 12 blocks will be of size 128 MB each and 3 blocks of size 2 MB each.

**18. How to copy a file into HDFS with a different block size to that of existing block size configuration?**

♣**Tip:** You should start the answer with the command for changing the block size and then, you should explain the whole procedure with an example. This is how you should answer this question:

Yes, one can copy a file into HDFS with a different block size by using ‘-Ddfs.blocksize=block\_size’ where the block\_size is specified in Bytes.

Let me explain it with an example: Suppose, I want to copy a file called test.txt of size, say of 120 MB, into the HDFS and I want the block size for this file to be 32 MB (33554432 Bytes) instead of the default (128 MB). So, I would issue the following command:

*hadoop fs -Ddfs.blocksize=33554432 -copyFromLocal /home/edureka/test.txt /sample\_hdfs*

Now, I can check the HDFS block size associated with this file by:

*hadoop fs -stat %o /sample\_hdfs/test.txt*

Else, I can also use the NameNode web UI for seeing the HDFS directory.

**♣Tip:** You can go through the blog on [***Hadoop Shell Commands***](https://www.edureka.co/blog/hdfs-commands-hadoop-shell-command) where you will find various Hadoop commands, explained with an example.

**19. Can you change the block size of HDFS files?**

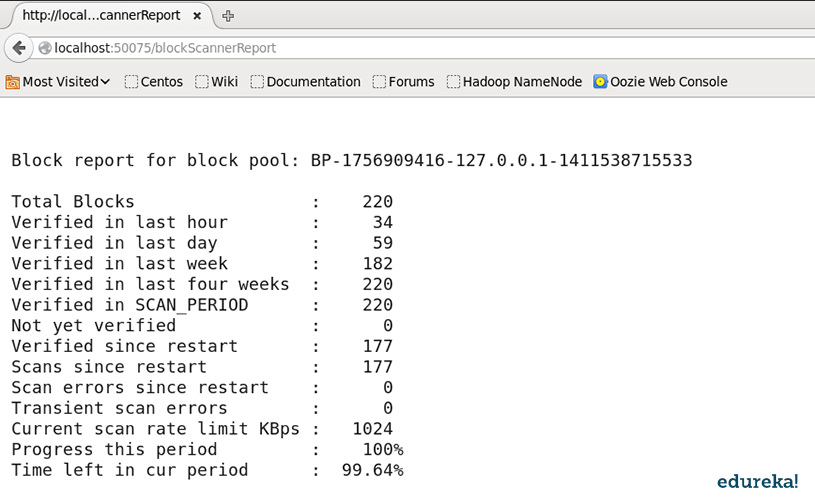
Yes, I can change the block size of HDFS files by changing the default size parameter present in hdfs-site.xml. But, I will have to restart the cluster for this property change to take effect.

**20. What is a block scanner in HDFS?**

Block scanner runs periodically on every DataNode to verify whether the data blocks stored are correct or not. The following steps will occur when a corrupted data block is detected by the block scanner:

* First, the DataNode will report about the corrupted block to the NameNode.
* Then, NameNode will start the process of creating a new replica using the correct replica of the corrupted block present in other DataNodes.
* The corrupted data block will not be deleted until the replication count of the correct replicas matches with the replication factor (3 by default).

This whole process allows HDFS to maintain the integrity of the data when a client performs a read operation. One can check the block scanner report using the DataNode’s web interface- **localhost:50075/blockScannerReport** as shown below:



**Fig.** – *Block Scanner Report – Hadoop HDFS Interview Question*

**21. HDFS stores data using commodity hardware which has higher chances of failures. So, How HDFS ensures the Fault Tolerance capability of the system?**

♣**Tip:**Basically, this question is regarding replication of blocks in Hadoop and how it helps in providing fault tolerance.

HDFS provides fault tolerance by replicating the data blocks and distributing it among different DataNodes across the cluster. By default, this replication factor is set to 3 which is configurable. So, if I store a file of 1 GB in HDFS where the replication factor is set to default i.e. 3, it will finally occupy a total space of 3 GB because of the replication. Now, even if a DataNode fails or a data block gets corrupted, I can retrieve the data from other replicas stored in different DataNodes.

**22. Replication causes data redundancy and consume a lot of space, then why is it pursued in HDFS?**

Replication is pursued in HDFS to provide the fault tolerance. And, yes, it will lead to the consumption of a lot of space, but one can always add more nodes to the cluster if required. By the way, in practical clusters, it is very rare to have free space issues as the very first reason to deploy HDFS was to store huge data sets. Also, one can change the replication factor to save HDFS space or use different codec provided by Hadoop to compress the data. 

**23. Can we have different replication factor of the existing files in HDFS?**

♣**Tip:** You should always answer such type of questions by taking an example to provide clarity.

Yes, one can have different [***replication factor***](https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/) for the files existing in HDFS. Suppose, I have a file named test.xml stored within the sample directory in my HDFS with the replication factor set to 1. Now, the command for changing the replication factor of text.xml file to 3 is:

*hadoop fs -setrwp -w 3 /sample/test.xml*

Finally, I can check whether the replication factor has been changed or not by using following command:

*hadoop fs -ls /sample*

or

*hadoop fsck /sample/test.xml -files*

**24. What is a rack awareness algorithm and why is it used in Hadoop?**

[***Rack Awareness***](https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/) algorithm in Hadoop ensures that all the block replicas are not stored on the same rack or a single rack. Considering the replication factor is 3, the Rack Awareness Algorithm says that the first replica of a block will be stored on a local rack and the next two replicas will be stored on a different (remote) rack but, on a different DataNode within that (remote) rack. There are two reasons for using Rack Awareness:

* **To improve the network performance:**In general, you will find*greater network bandwidth* between machines in the same rack than the machines residing in different rack. So, the Rack Awareness helps to reduce write traffic in between different racks and thus provides a better write performance.
* **To prevent loss of data:**I don’t have to worry about the data even if an entire rack fails because of the switch failure or power failure. And if one thinks about it, it will make sense, as it is said that *never put all your eggs in the same basket.*

**25. How data or a file is written into HDFS?**

The best way to answer this question is to take an example of a client and list the steps that will happen while performing the write without going into much of the details:

Suppose a client wants to write a file into HDFS. So, the following steps will be performed internally during the whole HDFS write process:

* The client will divide the files into blocks and will send a write request to the NameNode.
* For each block, the NameNode will provide the client a list containing the IP address of DataNodes (depending on replication factor, 3 by default) where the data block has to be copied eventually.
* The client will copy the first block into the first DataNode and then the other copies of the block will be replicated by the DataNodes themselves in a sequential manner.

♣**Tip:**I would suggest you to go through the blog on [***HDFS Read/Write Architecture***](https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/) where the whole process of HDFS Read/Write has been explained in detail with images.

*Do you prefer watching a video tutorial to understand & prepare yourself for your Hadoop interview? Here is our video on the top 50 Hadoop interview questions.*

**26. Can you modify the file present in HDFS?**

No, I cannot modify the files already present in HDFS, as HDFS follows Write Once Read Many model. But, I can always append data into the existing HDFS file.

**27. Can multiple clients write into an HDFS file concurrently?**

No, multiple clients can’t write into an HDFS file concurrently. HDFS follows single writer multiple reader model. The client which opens a file for writing is granted a lease by the NameNode. Now suppose, in the meanwhile, some other client wants to write into that very file and asks NameNode for the write permission. At first, the NameNode will check whether the lease for writing into that very particular file has been granted to someone else or not. Then, it will reject the write request of the other client if the lease has been acquired by someone else, who is currently writing into the very file.

**29. Does HDFS allow a client to read a file which is already opened for writing?**

Basically, the intent of asking this question is to know about the constraints associated with reading a file which is currently being written by some client. You may answer this question in following manner:

Yes, one can read the file which is already opened. But, the problem in reading a file which is currently being written lies in the consistency of the data i.e. HDFS does not provide the surety that the data which has been written into the file will be visible to a new reader before the file has been closed. For this, one can call the hflush operation explicitly which will push all the data in the buffer into the write pipeline and then the hflush operation will wait for the acknowledgements from the DataNodes. Hence, by doing this the data that has been written into the file before the hflush operation will be visible to the readers for sure.

**30. Define Data Integrity? How does HDFS ensure data integrity of data blocks stored in HDFS?**

Data Integrity talks about the correctness of the data. It is very important for us to have a guarantee or assurance that the data stored in HDFS is correct. However, there is always a slight chance that the data will get corrupted during I/O operations on the disk. HDFS creates the checksum for all the data written to it and verifies the data with the checksum during read operation by default. Also, each DataNode runs a block scanner periodically, which verifies the correctness of the data blocks stored in the HDFS.

**31. What do you mean by the High Availability of a NameNode? How is it achieved?**

NameNode used to be single point of failure in Hadoop 1.x where the whole Hadoop cluster becomes unavailable as soon as NameNode is down. In other words, [***High Availability***](https://www.edureka.co/blog/how-to-set-up-hadoop-cluster-with-hdfs-high-availability/) of the NameNode talks about the very necessity of a NameNode to be active for serving the requests of Hadoop clients.

To solve this Single Point of Failure problem of NameNode, HA feature was intorduced in Hadoop 2.x where we have two NameNode in our HDFS cluster in an active/passive configuration. Hence, if the active NameNode  fails, the other passive NameNode can take over the responsibility of the failed NameNode and keep the HDFS up and running.

**32. Define Hadoop Archives? What is the command for archiving a group of files in HDFS.**

Hadoop Archive was introduced to cope up with the problem of increasing memory usage of the NameNode for storing the metadata information because of too many small files. Basically, it allows us to pack a number of small HDFS files into a single archive file and therefore, reducing the metadata information. The final archived file follows the .har extension and one can consider it as a layered file system on top of HDFS.

The command for archiving a group of files:

hadoop archive –archiveName edureka\_archive.har /input/location /output/location

## ****33. How will you perform the inter cluster data copying work in HDFS?****

One can perform the inter cluster data copy by using distributed copy command given as follows:

hadoop distcp hdfs://<source NameNode> hdfs://<target NameNode>

I hope you find this blog on Hadoop HDFS Interview questions very helpful and informative. For becoming a Hadoop expert, you need to work on various Hadoop related projects as well, apart from having sound theoretical knowledge. We have designed a curriculum which covers all the aspects of the Hadoop framework along with lots of hands on experience. You will be taught by the Industry Experts who will be sharing their precious experience of various Hadoop related projects.

Last but not the least, I would like to say that HDFS is only one of the many components present in Hadoop framework. We have created dedicated blogs for important interview questions corresponding to each Hadoop component. Therefore, I would suggest you to follow the links provided below and enjoy the readings: