**HDFS INTERVIEW QUESTIONS**

**Blogs:**

<https://mindmajix.com/hadoop-interview-questions>

<https://www.edureka.co/blog/interview-questions/hadoop-interview-questions-hdfs-2/>

<https://intellipaat.com/blog/interview-question/hdfs-interview-questions/>

<https://data-flair.training/blogs/top-hdfs-interview-questions-and-answers/>

<https://acadgild.com/blog/hdfs-interview-questions-and-answers-2019>

<https://www.whizlabs.com/blog/hdfs-interview-questions/>

[***Hadoop Shell Commands***](https://www.edureka.co/blog/hdfs-commands-hadoop-shell-command)

**Questions:**

**RDBMS vs Hadoop**

|  |  |  |
| --- | --- | --- |
| **Name** | **RDBMS** | **Hadoop** |
| Data volume | RDBMS cannot store and process a large amount of data | Hadoop works better for large amounts of data. It can easily store and process a large amount of data compared to RDBMS. |
| Throughput | RDBMS fails to achieve a high Throughput | Hadoop achieves high Throughput |
| Data variety | Schema of the data is known in RDBMS and it always depends on the structured data. | It stores any kind of data. Whether it could be structured, unstructured or semi-structured. |
| Data processing | RDBMS supports OLTP(Online Transactional Processing) | Hadoop supports OLAP(Online Analytical Processing) |
| Read/Write Speed | Reads are fast in RDBMS because the schema of the data is already known. | Writes are fast in Hadoop because no schema validation happens during HDFS write. |
| Schema on reading Vs Write | RDBMS follows schema on write policy | Hadoop follows the schema on reading policy |
| Cost | RDBMS is a licensed software | Hadoop is a free and open-source framework |

**Explain Big data and its characteristics.**

Big Data refers to a large amount of data that exceeds the processing capacity of conventional database systems and requires a special parallel processing mechanism. This data can be either structured or unstructured data.

**Characteristics of Big Data:**

* + - **Volume -** It represents the amount of data which is increasing at an exponential rate i.e. in gigabytes, Petabytes, Exabytes, etc.
    - **Velocity -** Velocity refers to the rate at which data is generated, modified, and processed. At present, Social media is a major contributor to the velocity of growing data.
    - **Variety -** It refers to data coming from a variety of sources like audios, videos, CSV, etc. It can be either structured, unstructured, or semi-structured.
    - **Veracity -** Veracity refers to imprecise or uncertain data.
    - **Value -** This is the most important element of big data. It includes data on how to access and deliver quality analytics to the organization. It provides a fair market value on the used technology.

**What are the core components of Hadoop?**

|  |  |
| --- | --- |
| **Hadoop is an open-source framework used for storing large data sets and runs applications across clusters of commodity hardware. It offers extensive storage for any type of data and can handle endless parallel tasks.** | |
| **Component** | **Description** |
| ***HDFS (NN & DN)*** | [***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. |
| ***YARN (NM & RM)*** | YARN is the processing framework in Hadoop that allows multiple data processing engines to manage data stored on a single platform and provide Resource management. |

### What is YARN and explain its components?

Yet Another Resource Negotiator (YARN) is one of the core components of Hadoop and is responsible for managing resources for the various applications operating in a Hadoop cluster, and also schedules tasks on different cluster nodes.

**YARN components:**

* + - **Resource Manager -** It runs on a master daemon and controls the resource allocation in the cluster.
    - **Node Manager -** It runs on a slave daemon and is responsible for the execution of tasks for each single Data Node.
    - **Application Master -** It maintains the user job lifecycle and resource requirements of individual applications. It operates along with the Node Manager and controls the execution of tasks.
    - **Container -** It is a combination of resources such as Network, HDD, RAM, CPU, etc., on a single node.

**What is the difference between a regular file system and HDFS?**

|  |  |
| --- | --- |
| **Regular File Systems** | **HDFS** |
| A small block size of data (like 512 bytes) | Large block size (orders of 64mb/128mb) |
| Multiple disks seek large files | Reads data sequentially after single seek |

### What are the Hadoop daemons and explain their roles in a Hadoop cluster?

Generally, the daemon is nothing but a process that runs in the background. Hadoop has five such daemons. They are:

* + - **NameNode -**  It is the Master node responsible to store the meta-data for all the directories and files.
    - **DataNode -** It is the Slave node responsible to store the actual data.
    - **Secondary NameNode -** It is responsible for the backup of NameNode and stores entire metadata of data nodes like data node properties, address, and block report of each data node.
    - **JobTracker -**  It is used for creating and running jobs. It runs on data nodes and allocates the job to TaskTracker.
    - **TaskTracker -** It operates on the data node. It runs the tasks and reports the tasks to JobTracker.

## **What are the key features of HDFS?**

* **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.

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

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 helper daemon. It performs checkpointing.

### What is the function of namenode?

It contains two important information  
– Hadoop file system tree and metadata  
– In-memory mapping of blocks and corresponding data node  
Name node is a service which contains the metadata about the HDFS. Name node contains the file system tree of Hadoop. It contains the metadata like permissions of a file, replication factor of a file, block size, creation time, owner of the file, and the mapping of blocks of file in the data nodes like which block is present in which node.  
**Fsimage**  
It contains all directory structure of HDFS, replication level of file, modification and access time of files, access permissions of files and directories, block size of files, the blocks constituting a file.  
**Edit Logs**  
– When any operation take place in HDFS, the directory structure gets modified.  
– These modifications are stored in memory as well as in edits files (edits files are stored on hard disk) All these changes are added in append only fashion in the namenode.

## **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.

## **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.

**The default block size in Unix and Linux is 4KB, then why HDFS block size is set to 64MB or 128MB?**

ABlock is the smallest unit of data that is stored in a file system. Hence if we consider the default block size of Linux/Unix for data storing in Hadoop then for a massive set of data (petabytes) it will take a large number of blocks. Consequently, the metadata amount will increase significantly causing performance issue of NameNode. So, in Hadoop 1.x the default block size is 64MB and in Hadoop 2.x it is set to 128MB.

**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.

## **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.

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

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.

Any updates done to HDFS are entered in the editlogs. As the number of entries increases the file size grows automatically, however, the file size for the fsimage file remains the same. When the server gets restarted, the contents of the editlogs file are written into the fsimage file which is then loaded into main memory which is time-consuming. The more the editlogs file size, the more time it will take to load into fsimage causing an extended downtime.

To avoid such prolonged downtime, a helper node for NameNode which is known as Secondary NameNode is used which periodically copies the contents from editlogs to fsimage and copy the new fsimage file back to the NameNode.

**What is Safe mode in Hadoop?**

Safe mode indicates the maintenance state of the NameNode. During the safe mode, the HDFS cluster becomes read-only. Hence, no modification is allowed in the filesystem. Also, you cannot delete or replicate any data block in this mode.

**What happens when the NameNode starts?**

When the NameNode starts it performs the following operations:

* From last saved FsImage and the editlog file, it loads the file system namespace into its main memory.
* Creates the new fsimage file by merging the previous fsimage and editslog file to create new file system namespace.
* Receives information about block locations from all DataNodes.

**How does NameNode handle DataNode failure?**

HDFS architecture is designed in a way that every DataNodes periodically send heartbeat to the NameNode to assure it is in working mode. When the NameNode does not receive any heartbeat from a particular DataNode, it considers that DataNode as dead or non-functional and transfer all the respective DataBlock to some other active DataNode which is already replicated with it.

### What is HDFS namenode federation?

In this, one single namenode will contain the metadata, whereas multiple name nodes will contain the metadata about the block mapping of files and directories of subsets of the entire HDFS.  
For example, if HDFS contains two directories inside a directory, there may be two namenodes which maintain the two different directories so that the load will not be more and if one namenode fails, then the other can take over.  
The list of sub-directories maintained by a name node is called a namespace volume.  
Blocks for files belonging to a namespace is called block pool.  
For these reasons, namenode will not become a single point of failure.

### What is name node high availability?

The problem with the federation is that if one name node goes down, you cannot access the portion of the data that the namenode is taking care of.  
In HDFS high availability, you will maintain two namenodes: one of which is active and the other stand by each namenode and contain the file system tree, block mapping of the entire HDFS, and the edits are shared across both the name nodes. In case of failure, other name node will take the charge.  
Architectural changes:  
-The namenode must use high available shared storage to share the edit log. Edit logs are read by StandbyNameNode when it takes the responsibility of ActiveNameNode  
-Data nodes should send block report to both the namenodes  
-Check pointing is done by standby namenode

### How can you control block size and replication factor at file level?

You can change the block size and replication factors and many other configurations at the cluster level by setting the properties in the configuration files like core-site.xml, hdfs-site.xml, mapred-site.xml, yarn-site.xml.  
If you want to upload a file into HDFS with some specific block size and with some specific replication factor, you can do that by providing the configuration and its value while writing the file into HDFS.  
Changing block size  
hadoop fs -Ddfs.block.size=1048576 -put file.txt /user/acadgild  
hadoop fs -Ddfs.blocksize=1048576 -put file.txt /user/acadgild  
Changing replication factor  
hadoop fs -setrep -w2 /my/file  
or  
hadoop fs -Ddfs.replication=2 /my/file

**How data/file read operation is performed in HDFS?**

HDFS NameNode is the placeholder for all the file information and their actual locations in the slave nodes. The below steps are followed in the read operation of a file:

* When a file needs to be read, the file information is retrieved from NameNode by DistributedFileSystem instance.
* NameNode checks whether that particular file exists and the user has the access for the file
* Once the above-mentioned criteria are met, the NameNode provides the token to the client, for authentication to get the file from DataNode.
* NameNode provides the list of all Block detail and related data nodes of the file
* DataNodes are then sorted as per their proximity to the client.
* DistributedFileSystem returns an input stream to the client called as FSDataInputStream so that client can read data from it.
* FSDataInputStream works as a wrapper to the DFSInputStream, which is responsible for managing NameNode and DataNode and I/O.
* As the Client calls read () on the stream, the DFSInputStream connects to the closet DataNode block and data is returned to the client via stream. The read () operation is repeatedly called till the end of the first block is completely read.
* Once the first block is completely read, the connection with that DataNode is closed.
* Next, the DFSInputStream again connects to the next possible DataNode for the next block, and it continues until the file is completely read.
* Once the entire file is read, FSDataInputStream calls the close () operation to close the connection.

**Is concurrent write into HDFS file possible?**

No, HDFS does not allow concurrent writing. Because when one client receives permission by NameNode for writing on data node block, the particular block gets locked till the finish of the write operation. Hence, no other client can write on the same block.

**What are the challenges in existing HDFS architecture?**

Existing HDFS architecture consists of only one NameNode which contains the single Namespace and multiple DataNodes that hold the actual data. This architecture works well with limited cluster size. However, if we try to increase the cluster size, we come across few challenges.

* As the Namespace and Blocks are tightly coupled, other services cannot easily utilize the storage capacity of Blocks efficiently.
* With a single Namenode, if we want to add more DataNodes in the cluster, it will create huge metadata. Here we can scale DataNodes horizontally. However, we cannot scale up Namenode in the same manner. This is a Namespace Scalability issue.
* The current HDFS file system has a performance limitation related to the throughput. Because a single name node supports only 60000 concurrent tasks.
* We cannot get isolated namespace for a single application as HDFS deployments happen on a multi-tenant environment and multiple applications or organizations share a single cluster.

**Compare HDFS (Hadoop Distributed File System) and NAS (Network Attached Storage)?**

|  |  |
| --- | --- |
| **HDFS** | **NAS** |
| It is a distributed file system used for storing data by commodity hardware. | It is a file-level computer data storage server connected to a computer network, provides network access to a heterogeneous group of clients. |
| It includes commodity hardware which will be cost-effective | NAS is a high-end storage device which includes a high cost. |
| It is designed to work for the MapReduce paradigm. | It is not suitable for MapReduce. |

### What are the limitations of Hadoop 1.0?

* + - **NameNode:** No Horizontal Scalability and No High Availability
    - **Job Tracker:** Overburdened.
    - **MRv1:** It can only understand Map and Reduce tasks

### How to commission (adding) the nodes in the Hadoop cluster?

* + - Update the network addresses in the dfs.include and mapred.include
    - Update the NameNode: Hadoop dfsadmin -refreshNodes
    - Update the Jobtracker: Hadoop mradmin-refreshNodes
    - Update the slave file.
    - Start the DataNode and NodeManager on the added Node.

### How to decommission (removing) the nodes in the Hadoop cluster?

* + - Update the network addresses in the dfs.exclude and mapred.exclude
    - Update the Namenode: $ Hadoop dfsadmin -refreshNodes
    - Update the JobTracker: Hadoop mradmin -refreshNodes
    - Cross-check the Web UI it will show “Decommissioning in Progress”
    - Remove the Nodes from include file and then run: Hadoop dfsadmin-refreshNodes, Hadoop mradmin -refreshNodes.
    - Remove the Nodes from the slave file.

**Compare Hadoop 1.x and Hadoop 2.x**

|  |  |  |
| --- | --- | --- |
| **Name** | **Hadoop 1.x** | **Hadoop 2.x** |
| 1. NameNode | In Hadoop 1.x, NameNode is the single point of failure | In Hadoop 2.x, we have both Active and passive NameNodes. |
| 2. Processing | MRV1 (Job Tracker & Task Tracker) | MRV2/YARN (ResourceManager & NodeManager) |

### What is the difference between active and passive NameNodes?

* + - Active NameNode works and runs in the cluster.
    - Passive NameNode has similar data as active NameNode and replaces it when it fails.

### How will you resolve the NameNode failure issue?

The following steps need to be executed to resolve the NameNode issue and make Hadoop cluster up and running:

* + - Use the **FsImage** (file system metadata replica) to start a new NameNode.
    - Now, configure DataNodes and clients, so that they can acknowledge the new NameNode, that is started.
    - The new NameNode will start serving the client once it has completed loading the last checkpoint FsImage and enough block reports from the DataNodes.

### List the different types of Hadoop schedulers.

* + - Hadoop FIFO scheduler
    - Hadoop Fair Scheduler
    - Hadoop Capacity Scheduler

### How to keep an HDFS cluster balanced?

However, it is not possible to limit a cluster from becoming unbalanced. In order to give a balance to a certain threshold among data nodes, use the Balancer tool. This tool tries to subsequently even out the block data distribution across the cluster.

### What is DistCp?

* + - **DistCp** is the tool used to copy large amounts of data to and from Hadoop file systems in parallel.
    - It uses MapReduce to effect its distribution, reporting, recovery  and error handling.

### What is rack-aware replica placement policy?

* + - Rack Awareness is the algorithm used for improving the network traffic while reading/writing HDFS files to Hadoop cluster by NameNode.
    - NameNode chooses the Datanode which is closer to the same rack or nearby rack for reading/Write request. The concept of choosing closer data nodes based on racks information is called Rack Awareness.
    - Consider replication factor is 3 for data blocks on HDFS it means for every block of data two copies are stored on the same rack, while the third copy is stored on a different rack. This rule is called Replica Placement Policy.

### What is the main purpose of Hadoop fsck command?

Hadoop fsck command is used for checking the HDFS file system.

There are different arguments that can be passed with this command to emit different results.

* + - **Hadoop fsck / -files:** Displays all the files in HDFS while checking.
    - **Hadoop fsck / -files** -blocks: Displays all the blocks of the files while checking.
    - **Hadoop fsck / -files** -blocks -locations: Displays all the files block locations while checking.
    - **Hadoop fsck / -files** -blocks -locations -racks: Displays the networking topology for data-node locations.
    - **Hadoop fsck -delete:** Deletes the corrupted files in HDFS.
    - **Hadoop fsck -move:** Moves the corrupted files to a particular directory.

### What is the purpose of a DataNode block scanner?

* + - The purpose of the DataNode block scanner is to operate and periodically check all the blocks that are stored on the DataNode.
    - If bad blocks are detected it will be fixed before any client reads.

### What is the purpose of dfsadmin tool?

* + - **dfsadmin tool** is used for examining the HDFS cluster status.
    - **dfsadmin –** report command produces useful information about basic statistics of the cluster such as DataNodes and NameNode status, disk capacity configuration, etc.
    - It performs all the administrative tasks on the HDFS.

### What is the command used for printing the topology?

hdfs dfsadmin -point topology is used for printing the topology. It displays the tree of racks and DataNodes attached to the tracks.

### What is RAID?

RAID (redundant array of independent disks) is a data storage virtualization technology used for improving performance and data redundancy by combining multiple disk drives into a single entity.

### Does Hadoop requires RAID?

* + - In DataNodes, RAID is not necessary as storage is achieved by replication between the Nodes.
    - In NameNode’s disk RAID is recommended.

### List the various site-specific configuration files available in Hadoop?

* + - conf/Hadoop-env.sh
    - conf/yarn-site.xml
    - conf/yarn-env.sh
    - conf/mapred-site.xml
    - conf/hdfs-site.xml
    - conf/core-site.xml

### Which command is used to format the NameNode?

$ hdfs namenode -format

### How a client application interacts with the NameNode?

* + - Client applications associate the Hadoop HDFS API with the NameNode when it has to copy/move/add/locate/delete a file.
    - The NameNode returns to the successful requests by delivering a list of relevant DataNode servers where the data is residing.
    - The client can talk directly to a DataNode after the NameNode has given the location of the data

## **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.

## **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.

## **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.

## **What is a block?**

[***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.

**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.

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

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.

## **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.

## **How data or a file is written into HDFS?**

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.

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

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.

**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.

## **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

## **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>