**1. Hadoop Interview Questions**

In This Hadoop Interview Questions and Answers blog, we are going to cover top 100 Hadoop Interview questions along with their detailed answers. We will be covering Hadoop scenario based interview questions, Hadoop interview questions for freshers as well as Hadoop interview questions and answers for experienced. You will also see How to explain Hadoop project in interview which carries a lot of weightage in the interview. These Hadoop real time interview questions will help you a lot to crack Hadoop Interview easily.

In this Hadoop Interview Questions blog we will explain the latest Hadoop interview questions and answers based on the different components of the Hadoop Ecosystem such as [**HDFS**](http://data-flair.training/blogs/comprehensive-hdfs-guide-introduction-architecture-data-read-write-tutorial/), **[MapReduce](http://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/)**, [**YARN**](http://data-flair.training/blogs/hadoop-yarn-tutorial/), [**Hive**](http://data-flair.training/blogs/apache-hive-tutorial/), **[HBase](http://data-flair.training/blogs/hbase-tutorial-beginners-guide/)** etc.

*Top 100 Hadoop Interview Questions*

We have categorized all these Hadoop Interview Questions and Answers according to the Hadoop Ecosystem Components-

* [Hadoop Interview Questions](https://data-flair.training/blogs/top-100-hadoop-interview-questions-and-answers/#basic)
* [HDFS Hadoop Interview Questions](https://data-flair.training/blogs/top-100-hadoop-interview-questions-and-answers/#hdfs)
* [MapReduce Hadoop Interview Questions](https://data-flair.training/blogs/top-100-hadoop-interview-questions-and-answers/#mapreduce)

**2. Hadoop Interview Questions and Answers**

In this section of Hadoop Interview Questions, we will cover the Basic Hadoop Interview Questions and Answers for freshers and experienced. After this general Hadoop Interview Questions sections we will have different sections on the Hadoop Interview Questions on HDFS and **[MapReduce](http://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/)**.

**1) What is Apache Hadoop?**

[**Hadoop**](http://data-flair.training/blogs/hadoop-introduction-tutorial-quick-guide/) emerged as a solution to the “[**Big Data**](http://data-flair.training/blogs/why-learn-big-data-use-cases/)” problems. It is a part of the Apache project sponsored by the *Apache Software Foundation (ASF)*. It is an open source software framework for distributed storage and distributed processing of large data sets. Open source means it is freely available and even we can change its source code as per our requirements. Apache Hadoop makes it possible to run applications on the system with thousands of commodity hardware nodes. It’s distributed file system has the provision of rapid data transfer rates among nodes. It also allows the system to continue operating in case of node failure. Apache Hadoop provides:

* **Storage layer**– [**HDFS**](http://data-flair.training/blogs/comprehensive-hdfs-guide-introduction-architecture-data-read-write-tutorial/)
* **Batch processing engine**– **[MapReduce](http://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/)**
* **Resource Management Layer**– [**YARN**](http://data-flair.training/blogs/hadoop-yarn-tutorial/)

**2) Why do we need Hadoop?**

The picture of Hadoop came into existence to deal with Big Data challenges. The challenges with Big Data are-

* **Storage –** Since data is very large, so storing such huge amount of data is very difficult.
* **Security –** Since the data is huge in size, keeping it secure is another challenge.
* **Analytics –** In Big Data, most of the time we are unaware of the kind of data we are dealing with. So analyzing that data is even more difficult.
* **Data Quality –** In the case of Big Data, data is very messy, inconsistent and incomplete.
* **Discovery –** Using a powerful algorithm to find patterns and insights are very difficult.

Hadoop is an open-source software framework that supports the storage and processing of large data sets. Apache Hadoop is the best solution for storing and processing Big data because:

* Apache Hadoop stores huge files as they are (raw) without specifying any schema.
* **High scalability –** We can add any number of nodes, hence enhancing performance dramatically.
* **Reliable –** It stores data reliably on the cluster despite machine failure.
* **High availability –** In Hadoop data is highly available despite hardware failure. If a machine or hardware crashes, then we can access data from another path.
* **Economic –**Hadoop runs on a cluster of commodity hardware which is not very expensive

Here is detailed explanation of [**features of Hadoop**](https://data-flair.training/blogs/hadoop-features-design-principles-tutorial/)

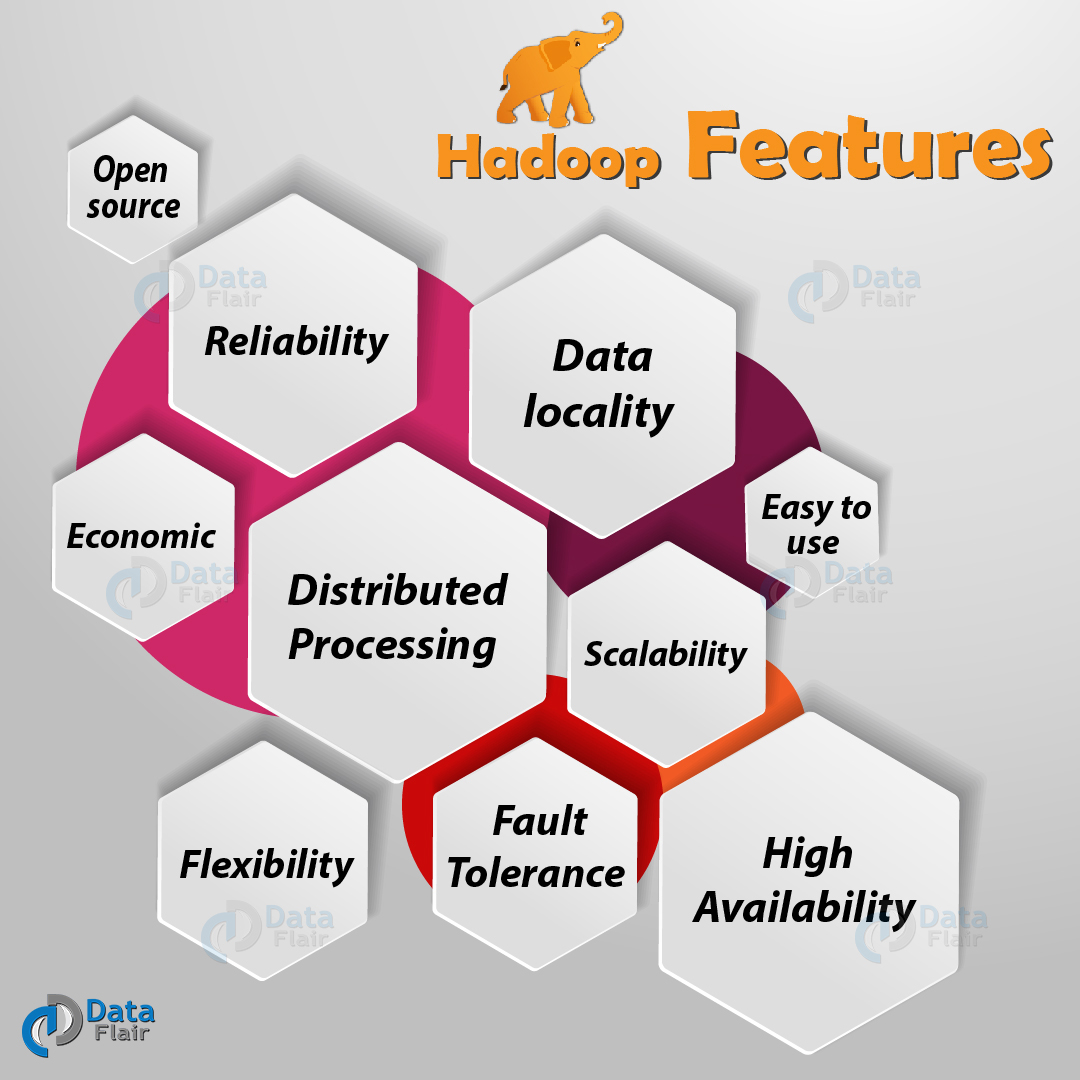
**3) What are the core components of Hadoop?**

**Hadoop** is an open-source software framework for distributed storage and processing of large datasets. Apache Hadoop core components are HDFS, MapReduce, and YARN.

* **HDFS-** Hadoop Distributed File System (HDFS) is the primary storage system of Hadoop. HDFS store very large files running on a cluster of commodity hardware. It works on the principle of storage of less number of large files rather than the huge number of small files. HDFS stores data reliably even in the case of hardware failure. It provides high throughput access to an application by accessing in parallel.
* **MapReduce- [MapReduce](http://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/)**is the data processing layer of Hadoop. It writes an application that processes large structured and unstructured data stored in HDFS. MapReduce processes a huge amount of data in parallel. It does this by dividing the job (submitted job) into a set of independent tasks (sub-job). In Hadoop, MapReduce works by breaking the processing into phases: **Map** and **Reduce.** The Map is the first phase of processing, where we specify all the complex logic code. Reduce is the second phase of processing. Here we specify light-weight processing like aggregation/summation.
* **YARN-** YARN is the processing framework in Hadoop. It provides Resource management and allows multiple data processing engines. For example real-time streaming, data science, and batch processing.

[**Read: Hadoop Ecosystem Components in detail.**](http://data-flair.training/blogs/hadoop-ecosystem-components/)

**4) What are the Features of Hadoop?**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/Hadoop-Features-01.jpg)

*Hadoop Interview Questions – Features of Hadoop*

The various Features of Hadoop are:

* **Open Source –** Apache Hadoop is an open source software framework. Open source means it is freely available and even we can change its source code as per our requirements.
* **Distributed processing –** As *HDFS* stores data in a distributed manner across the cluster. MapReduce process the data in parallel on the cluster of nodes.
* **Fault Tolerance –** Apache Hadoop is highly Fault-Tolerant. By default, each block creates 3 replicas across the cluster and we can change it as per needment. So if any node goes down, we can recover data on that node from the other node. Framework recovers failures of nodes or tasks automatically.
* **Reliability –** It stores data reliably on the cluster despite machine failure.
* [**High Availability**](https://data-flair.training/blogs/hadoop-high-availability-tutorial/)**–** Data is highly available and accessible despite hardware failure. In Hadoop, when a machine or hardware crashes, then we can access data from another path.
* **Scalability –** Hadoop is highly scalable, as one can add the new hardware to the nodes.
* **Economic-** Hadoop runs on a cluster of commodity hardware which is not very expensive. We do not need any specialized machine for it.
* **Easy to use –** No need of client to deal with distributed computing, the framework take care of all the things. So it is easy to use.

[Read Hadoop Features in detail](http://data-flair.training/blogs/hadoop-features-design-principles-tutorial/)

**5) Compare Hadoop and RDBMS?**

Apache Hadoop is the future of the database because it stores and processes a large amount of data. Which will not be possible with the traditional database. There is some difference between **Hadoop** and **RDBMS** which are as follows:

* **Architecture –**Traditional RDBMS have**ACID** properties. Whereas Hadoop is distributed computing framework having two main components: Distributed**file system (HDFS)** and **MapReduce.**
* **Data acceptance –**RDBMS accepts only structured data. While Hadoop can accept both structured as well as unstructured data. It is a great feature of hadoop, as we can store everything in our database and there will be no data loss.
* **Scalability –**RDBMS is a traditional database which provides vertical scalability. So if the data increases for storing then we have to increase particular system configuration. While Hadoop provides horizontal scalability. So we just have to add one or more node to the cluster if there is any requirement for an increase in data.
* **OLTP (Real-time data processing) and OLAP –**Traditional RDMS support OLTP (Real-time data processing). OLTP is not supported in Apache Hadoop. Apache Hadoop supports large scale Batch Processing workloads (OLAP).
* **Cost –**Licensed software, therefore we have to pay for the software. Whereas Hadoop is open source framework, so we don’t need to pay for software.

**Read:**[**50+ Hadoop MapReduce Interview Questions and Answers**](https://data-flair.training/blogs/hadoop-mapreduce-interview-questions-and-answers/)

If you have any doubts or queries regarding Hadoop Interview Questions at any point you can ask that Hadoop Interview question to us in comment section and our support team will get back to you.

**3. Hadoop Interview Questions for Freshers:**

The following Hadoop Interview Questions are for freshers and students, however, Experienced can also refer them for revising the basics

**6) What are the modes in which Hadoop run?**

Apache Hadoop runs in three modes:

* **Local (Standalone) Mode –**Hadoop by default run in a single-node, non-distributed mode, as a single Java process. Local mode uses the local file system for input and output operation. It is also used for debugging purpose, and it does not support the use of HDFS. Further, in this mode, there is no custom configuration required for configuration files.
* **Pseudo-Distributed Mode –**Just like the Standalone mode, Hadoop also runs on a single-node in a Pseudo-distributed mode. The difference is that each daemon runs in a separate Java process in this Mode. In Pseudo-distributed mode, we need configuration for all the four files mentioned above. In this case, all daemons are running on one node and thus, both Master and Slave node are the same.
* **Fully-Distributed Mode –**In this mode, all daemons execute in separate nodes forming a multi-node cluster. Thus, it allows separate nodes for Master and Slave.

**Read:**[**Know what is Rack Awareness in Hadoop**](https://data-flair.training/blogs/rack-awareness-hadoop-hdfs/)

**7) What are the features of Standalone (local) mode?**

By default, Hadoop run in a single-node, non-distributed mode, as a single Java process. Local mode uses the local file system for input and output operation. One can also use it for debugging purpose. It does not support the use of [**HDFS**](https://data-flair.training/blogs/comprehensive-hdfs-guide-introduction-architecture-data-read-write-tutorial/). Standalone mode is suitable only for running programs during development for testing. Further, in this mode, there is no custom configuration required for configuration files. Configuration files are:

* core-site.xml
* hdfs-site.xml files.
* mapred-site.xml
* yarn-default.xml

**8) What are the features of Pseudo mode?**

Just like the Standalone mode, Hadoop can also run on a*single-node* in this mode. The difference is that each Hadoop daemon runs in a separate Java process in this Mode. In Pseudo-distributed mode, we need configuration for all the four files mentioned above. In this case, all daemons are running on one node and thus, both **Master** and **Slave** node are the same.

The pseudo mode is suitable for both for development and in the testing environment. In the Pseudo mode, all the daemons run on the same machine.

**Read:**[**How Hadoop MapReduce Works**](https://data-flair.training/blogs/how-hadoop-mapreduce-works/)

**9) What are the features of Fully-Distributed mode?**

In this mode, all daemons execute in separate nodes forming a multi-node cluster. Thus, we allow separate nodes for *Master and Slave*.

We use this mode in the production environment, where ‘**n**’ number of machines forming a cluster. Hadoop daemons run on a cluster of machines. There is one host onto which NameNode is running and the other hosts on which DataNodes are running. Therefore, NodeManager installs on every DataNode. And it is also responsible for the execution of the task on every single DataNode.

The *ResourceManager* manages all these*NodeManager.* ResourceManager receives the processing requests. After that, it passes the parts of the request to corresponding NodeManager accordingly.

**10) What are configuration files in Hadoop?**

**Core-site.xml –**It contain configuration setting for Hadoop core such as I/O settings that are common to HDFS & MapReduce. It use Hostname and port .The most commonly used port is 9000.

1. <configuration>
2. <property>
3. <name>fs.defaultFS</name>
4. <value>hdfs://localhost:9000</value>
5. </property>
6. </configuration>

**hdfs-site.xml –** This file contains the configuration setting for HDFS daemons. hdfs-site.xml also specify default block replication and permission checking on HDFS.

1. <configuration>
2. <property>
3. <name>dfs.replication</name>
4. <value>1</value>
5. </property>
6. </configuration>

**mapred-site.xml –**In this file, we specify a framework name for MapReduce. we can specify by setting the mapreduce.framework.name.

1. <configuration>
2. <property>
3. <name>mapreduce.framework.name</name>
4. <value>yarn</value>
5. </property>
6. </configuration>

**yarn-site.xml –**This file provide configuration setting for [NodeManager](http://data-flair.training/blogs/hadoop-yarn-node-manager-tutorial-guide/) and [ResourceManager](http://data-flair.training/blogs/hadoop-yarn-resource-manager-guide-tutorial/).

1. <configuration>
2. <property>
3. <name>yarn.nodemanager.aux-services</name>
4. <value>mapreduce\_shuffle</value>
5. </property>
6. <property>
7. <name>yarn.nodemanager.env-white**list**</name> <value>
8. JAVA\_HOME,HADOOP\_COMMON\_HOME,HADOOP\_HDFS\_HOME,HADOOP\_CONF\_DIR,CLASSPATH\_PREPEND\_DISTCACHE,HADOOP\_YARN\_HOME,HADOOP\_MAPRED\_HOME</value> </property>
9. </configuration>

**11) What are the limitations of Hadoop?**

Various [**limitations of Hadoop**](https://data-flair.training/blogs/limitations-of-hadoop/) are:

* **Issue with small files –** Hadoop is not suited for small files. Small files are the major problems in HDFS. A small file is significantly smaller than the HDFS block size (default 128MB). If you are storing these large number of small files, HDFS can’t handle these lots of files. As HDFS works with a small number of large files for storing data sets rather than larger number of small files. If one use the huge number of small files, then this will overload the namenode. Since namenode stores the namespace of HDFS.  
  HAR files, Sequence files, and Hbase overcome small files issues.
* **Processing Speed –** With parallel and distributed algorithm, MapReduce process large data sets. MapReduce performs the task: ***Map***and ***Reduce***. MapReduce requires a lot of time to perform these tasks thereby increasing latency. As data is distributed and processed over the cluster in MapReduce. So, it will increase the time and reduces processing speed.
* **Support only Batch Processing –** Hadoop supports only batch processing. It does not process streamed data and hence, overall performance is slower. MapReduce framework does not leverage the memory of the cluster to the maximum.
* **Iterative Processing –** Hadoop is not efficient for iterative processing. As hadoop does not support cyclic data flow. That is the chain of stages in which the input to the next stage is the output from the previous stage.
* **Vulnerable by nature –** Hadoop is entirely written in Java, a language most widely used. Hence java been most heavily exploited by cyber-criminal. Therefore it implicates in numerous security breaches.
* **Security-** Hadoop can be challenging in managing the complex application. Hadoop is missing encryption at storage and network levels, which is a major point of concern. Hadoop supports Kerberos authentication, which is hard to manage.

[**Read more on Hadoop Limitations**](http://data-flair.training/blogs/limitations-of-hadoop/)

**4. Hadoop Interview Questions for Experienced:**

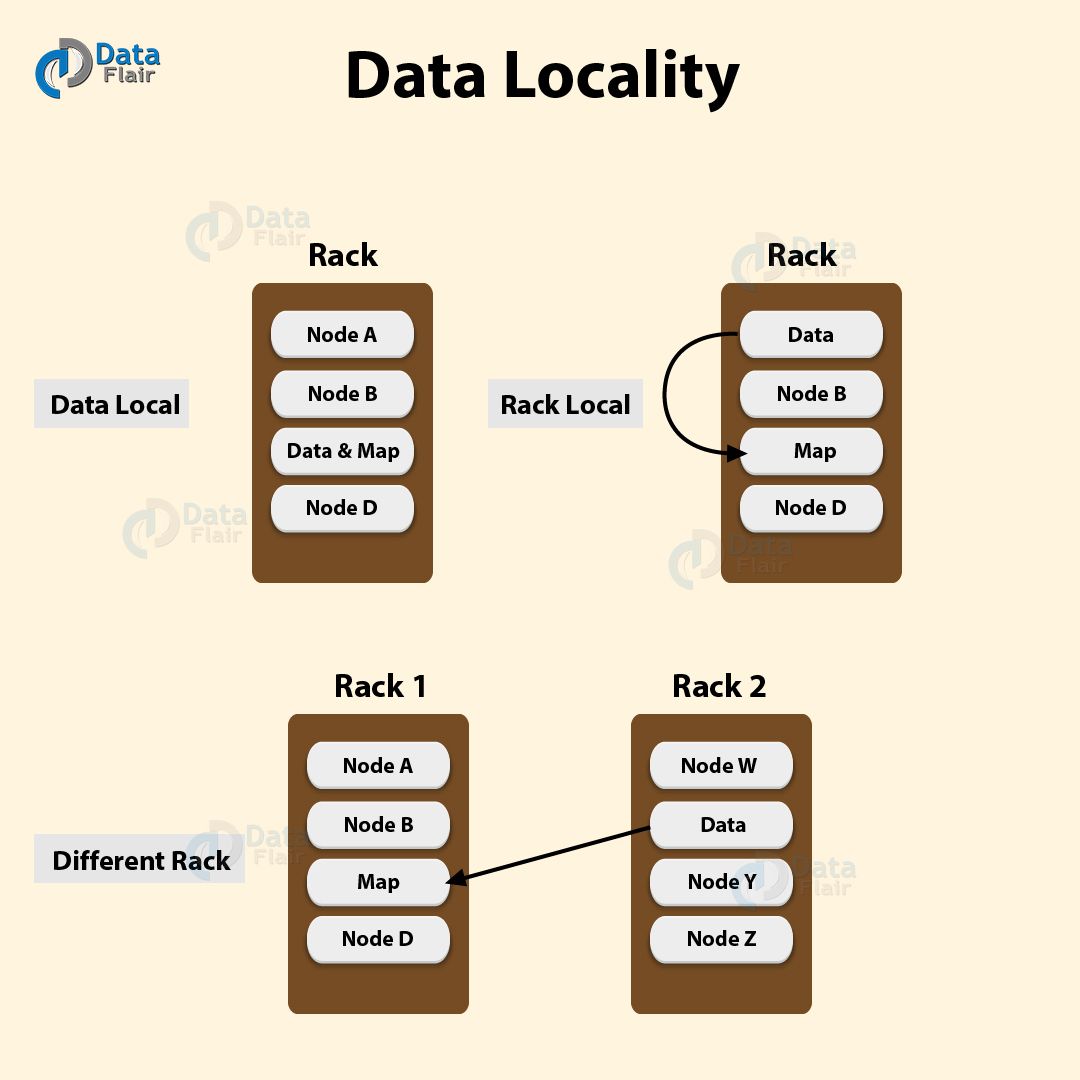
The core Hadoop Interview Questions are for experienced, but freshers and Students can also read and refer them for advanced understanding

**12) Compare Hadoop 2 and Hadoop 3?**

* In Hadoop 2, minimum supported version of Java is **Java 7**, while in Hadoop 3 is **Java 8**.
* Hadoop 2, handle fault tolerance by replication (which is wastage of space). While Hadoop 3 handle it by Erasure coding.
* For data balancing Hadoop 2 uses HDFS balancer. While Hadoop 3 uses Intra-datanode balancer.
* In Hadoop 2 some default ports are Linux ephemeral port range. So at the time of startup, they will fail to bind. But in Hadoop 3 these ports have been moved out of the ephemeral range.
* In hadoop 2, HDFS has 200% overhead in storage space. While Hadoop 3 has 50% overhead in storage space.
* Hadoop 2 has features to overcome**SPOF** (single point of failure). So whenever NameNode fails, it recovers automatically. Hadoop 3 recovers SPOF automatically no need of manual intervention to overcome it.

[**Read about more Comparison between Hadoop 2 and Hadoop 3**](http://data-flair.training/blogs/hadoop-2-x-vs-hadoop-3-x-comparison/)

**13) Explain Data Locality in Hadoop?**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/Data-Locality-01.jpg)

*Hadoop Interview Questions and Answers – Data Locality*

Hadoop major drawback was cross-switch network traffic due to the huge volume of data. To overcome this drawback, ***Data locality*** came into the picture. It refers to the ability to move the computation close to where the actual data resides on the node, instead of moving large data to computation. Data locality increases the overall throughput of the system.  
In Hadoop, HDFS stores datasets. Datasets are divided into blocks and stored across the datanodes in[**Hadoop cluster**.](http://data-flair.training/blogs/installation-hadoop-3-x-ubuntu-pseudo-distributed-mode/) When a user runs the MapReduce job then NameNode sends this MapReduce code to the datanodes on which data is available related to MapReduce job.  
**Data locality has three categories:**

* **Data local –** In this category data is on the same node as the mapper working on the data. In such case, the proximity of the data is closer to the computation. This is the most preferred scenario.
* **Intra –** Rack- In this scenarios mapper run on the different node but on the same rack. As it is not always possible to execute the mapper on the same datanode due to constraints.
* **Inter-Rack –** In this scenarios mapper run on the different rack. As it is not possible to execute mapper on a different node in the same rack due to resource constraints.

**14) What is Safemode in Hadoop?**

Safemode in Apache Hadoop is a maintenance state of **[NameNode](https://data-flair.training/blogs/limitations-of-hadoop/)**. During which NameNode doesn’t allow any modifications to the file system. During Safemode, HDFS cluster is in read-only and doesn’t replicate or delete blocks. At the startup of NameNode:

* It loads the file system namespace from the last saved FsImage into its main memory and the edits log file.
* Merges edits log file on FsImage and results in new file system namespace.
* Then it receives block reports containing information about block location from all datanodes.

In SafeMode NameNode perform a collection of block reports from datanodes. NameNode enters safemode automatically during its start up. NameNode leaves Safemode after the DataNodes have reported that most blocks are available. Use the command:

*hadoop dfsadmin –safemode get: To know the status of Safemode*  
*bin/hadoop dfsadmin –safemode enter: To enter Safemode*  
*hadoop dfsadmin -safemode leave: To come out of Safemode*  
NameNode front page shows whether safemode is on or off.

**15) What is the problem with small files in Hadoop?**

Hadoop is not suited for small data. Hadoop HDFS lacks the ability to support the random reading of small files. Small file in HDFS is smaller than the HDFS block size (default 128 MB). If we are storing these huge numbers of small files, HDFS can’t handle these lots of files. HDFS works with the small number of large files for storing large datasets. It is not suitable for a large number of small files. A large number of many small files overload NameNode since it stores the namespace of HDFS.

**Solution:**

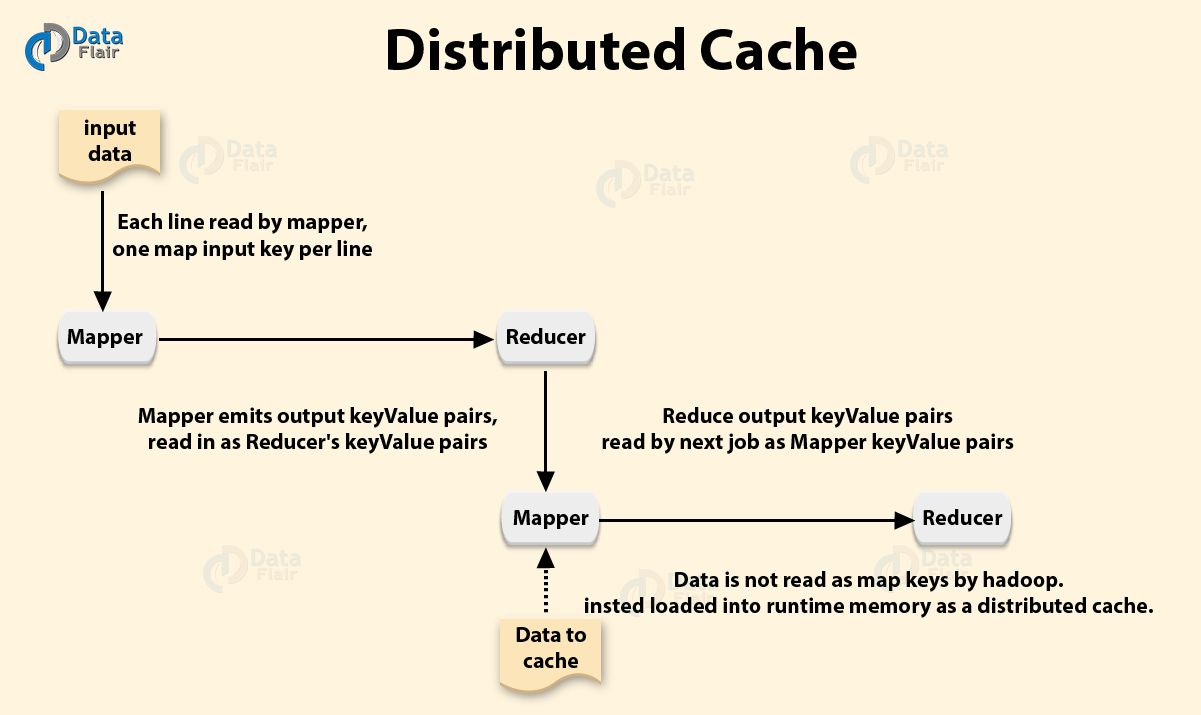
*HAR (Hadoop Archive)* Files deal with small file issue. HAR has introduced a layer on top of HDFS, which provide interface for file accessing. Using Hadoop archive command we can create HAR files. These file runs a MapReduce job to pack the archived files into a smaller number of HDFS files. Reading through files in as HAR is not more efficient than reading through files in HDFS. Since each HAR file access requires two index files read as well the data file to read, this makes it slower.

Sequence Files also deal with small file problem. In this, we use the filename as key and the file contents as the value. If we have 10,000 files of 100 KB, we can write a program to put them into a single sequence file. And then we can process them in a streaming fashion.

**Read: [Hadoop Mapper – 4 Steps Learning to MapReduce Mapper](https://data-flair.training/blogs/hadoop-mapper-in-mapreduce/)**

**16) What is a “Distributed Cache” in Apache Hadoop?**

In Hadoop, data chunks process independently in parallel among DataNodes, using a program written by the user. If we want to access some files from all the DataNodes, then we will put that file to distributed cache.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/Distributed-Cache-01.jpg)

*Big Data Interview Questions For Freshers – Distributed Cache*

MapReduce framework provides [**Distributed Cache**](https://data-flair.training/blogs/hadoop-distributed-cache/) to caches files needed by the applications. It can cache read-only text files, archives, jar files etc.

Once we have cached a file for our job. Then, Hadoop will make it available on each datanodes where map/reduce tasks are running. Then, we can access files from all the datanodes in our map and reduce job.

An application which needs to use distributed cache should make sure that the files are available on URLs. URLs can be either **hdfs:// or http://.** Now, if the file is present on the mentioned URLs. The user mentions it to be cache file to distributed cache. This framework will copy the cache file on all the nodes before starting of tasks on those nodes. By default size of distributed cache is 10 GB. We can adjust the size of distributed cache using local.cache.size.

[**Read Hadoop distributed cache in detail**](http://data-flair.training/blogs/hadoop-distributed-cache/)

**17) How is security achieved in Hadoop?**

Apache Hadoop achieves security by using **Kerberos**.

At a high level, there are three steps that a client must take to access a service when using Kerberos. Thus, each of which involves a message exchange with a server.

* **Authentication –**The client authenticates itself to the authentication server. Then, receives a timestamped *Ticket-Granting Ticket (TGT)*.
* **Authorization –**The client uses the TGT to request a service ticket from the Ticket Granting Server.
* **Service Request –**The client uses the service ticket to authenticate itself to the server.

**18) Why does one remove or add nodes in a Hadoop cluster frequently?**

The most important features of the Hadoop is its utilization of Commodity hardware. However, this leads to frequent Datanode crashes in a Hadoop cluster.

Another striking feature of Hadoop is the ease of scale by the rapid growth in data volume.  
Hence, due to above reasons, administrator Add/Remove DataNodes in a Hadoop Cluster.

**Read: [Hadoop HDFS Architecture Explanation and Assumptions](https://data-flair.training/blogs/hadoop-hdfs-architecture/)**

**19) What is throughput in Hadoop?**

The amount of work done in a unit time is Throughput. Because of bellow reasons HDFS provides good throughput:

* The HDFS is Write Once and Read Many Model. It simplifies the data coherency issues as the data written once, one can not modify it. Thus, provides high throughput data access.
* Hadoop works on [**Data Locality**](https://data-flair.training/blogs/data-locality-in-hadoop-mapreduce/) principle. This principle state that moves computation to data instead of data to computation. This reduces network congestion and therefore, enhances the overall system throughput.

**20) How to restart NameNode or all the daemons in Hadoop?**

By following methods we can restart the NameNode:

* You can stop the NameNode individually using ***/*sbin*/hadoop-daemon.sh*** stop namenode command. Then start the NameNode using ***/sbin/hadoop-daemon.sh***start namenode.
* Use ***/sbin/stop-all.sh*** and the use ***/sbin/start-all.sh***, command which will stop all the demons first. Then start all the daemons.  
  The sbin directory inside the Hadoop directory store these script files.

**21) What does jps command do in Hadoop?**

The jbs command helps us to check if the Hadoop daemons are running or not. Thus, it shows all the Hadoop daemons that are running on the machine. Daemons are *Namenode, Datanode, ResourceManager, NodeManager etc*.

**Read:**[**Top 10 Hadoop Hdfs Commands with Examples and Usage**](https://data-flair.training/blogs/top-hadoop-hdfs-commands-tutorial/)

**22) What are the main hdfs-site.xml properties?**

**hdfs-site.xml –** This file contains the configuration setting for HDFS daemons. hdfs-site.xml also specify default block replication and permission checking on HDFS.  
The three main hdfs-site.xml properties are:

1. ***dfs.name.dir*** gives you the location where NameNode stores the metadata (FsImage and edit logs). And also specify where DFS should locate – on the disk or in the remote directory.
2. ***dfs.data.dir*** gives the location of DataNodes where it stores the data.
3. ***fs.checkpoint.dir*** is the directory on the file system. On which secondary NameNode stores the temporary images of edit logs. Then this EditLogs and FsImage will merge for backup.

**23) What is fsck?**

[](https://data-flair.training/blogs/category/quiz/hadoop-quiz/)

**fsck** is the **File System Check**. Hadoop[HDFS](https://data-flair.training/blogs/hadoop-hdfs-tutorial/) use the fsck (filesystem check) command to check for various inconsistencies. It also reports the problems with the files in HDFS. For example, missing blocks for a file or under-replicated blocks. It is different from the traditional fsck utility for the native file system. Therefore it does not correct the errors it detects.

Normally NameNode automatically corrects most of the recoverable failures. Filesystem check also ignores open files. But it provides an option to select all files during reporting. The HDFS fsck command is not a Hadoop shell command. It can also run as *bin/hdfs fsck*. Filesystem check can run on the whole file system or on a subset of files.

**Usage:**

hdfs fsck <path>  
[-list-corruptfileblocks |  
[-move | -delete | -openforwrite]  
[-files [-blocks [-locations | -racks]]]  
[-includeSnapshots]  
Path- Start checking from this path  
-delete- Delete corrupted files.  
-files- Print out the checked files.  
-files –blocks- Print out the block report.  
-files –blocks –locations- Print out locations for every block.  
-files –blocks –rack- Print out network topology for data-node locations  
-includeSnapshots- Include snapshot data if the given path indicates or include snapshottable directory.  
-list -corruptfileblocks- Print the list of missing files and blocks they belong to.

**24) How to debug Hadoop code?**

First, check the list of MapReduce jobs currently running. Then, check whether orphaned jobs is running or not; if yes, you need to determine the location of RM logs.

* First of all, Run: “**ps –ef| grep –I ResourceManager**” and then, look for log directory in the displayed result. Find out the job-id from the displayed list. Then check whether error message associated with that job or not.
* Now, on the basis of RM logs, identify the worker node which involves in the execution of the task.
* Now, login to that node and run- “**ps –ef| grep –I NodeManager**”
* Examine the NodeManager log.
* The majority of errors come from user level logs for each amp-reduce job.

**Read: [Hadoop HDFS Commands with Examples and Usage](https://data-flair.training/blogs/most-used-hdfs-commands-tutorial-examples/)**

**25) Explain Hadoop streaming?**

Hadoop distribution provides generic *application programming interface (API)*. This allows writing Map and Reduce jobs in any desired programming language. The utility allows creating/running jobs with any executable as [Mapper](https://data-flair.training/blogs/mapper-in-hadoop-mapreduce/)/[Reducer.](https://data-flair.training/blogs/reducer-in-hadoop-mapreduce/)  
For example:

hadoop jar hadoop-streaming-3.0.jar \  
-input myInputDirs \  
-output myOutputDir \  
-mapper /bin/cat \  
-reducer /usr/bin/wc

In the example, both the Mapper and reducer are executables. That read the input from stdin (line by line) and emit the output to stdout. The utility allows creating/submitting Map/Reduce job, to an appropriate cluster. It also monitors the progress of the job until it completes. Hadoop Streaming uses both streaming command options as well as generic command options. Be sure to place the generic options before the streaming. Otherwise, the command will fail.

The general line syntax shown below:

Hadoop command [genericOptions] [streamingOptions]

**26) What does hadoop-metrics.properties file do?**

Statistical information exposed by the Hadoop daemons is Metrics. Hadoop framework uses it for monitoring, performance tuning and debug.

By default, there are many metrics available. Thus, they are very useful for troubleshooting.

Hadoop framework use **hadoop-metrics.properties** for ‘**Performance Reporting**’ purpose. It also controls the reporting for Hadoop. The API provides an abstraction so we can implement on top of a variety of metrics client libraries. The choice of client library is a configuration option. And different modules within the same application can use different metrics implementation libraries.  
This file is present inside*/etc/hadoop.*

**Also Read [Hadoop Tutorial with Hadoop Daemons](https://data-flair.training/blogs/hadoop-tutorial-for-beginners/)**

**27) How Hadoop’s CLASSPATH plays a vital role in starting or stopping in Hadoop daemons?**

**CLASSPATH** includes all directories containing jar files required to start/stop Hadoop daemons.

For example- *HADOOP\_HOME/share/hadoop/common/lib* contains all the utility jar files. We cannot start/ stop Hadoop daemons if we don’t set CLASSPATH.

We can set CLASSPATH inside*/etc/hadoop/hadoop-env.sh* file. The next time you run hadoop, the CLASSPATH will automatically add. That is, you don’t need to add CLASSPATH in the parameters each time you run it.

**28) What are the different commands used to startup and shutdown Hadoop daemons?**

• To start all the hadoop daemons use: ./sbin/start-all.sh.  
Then, to stop all the Hadoop daemons use:./sbin/stop-all.sh  
• You can also start all the dfs daemons together using ./sbin/start-dfs.sh. Yarn daemons together using ./sbin/start-yarn.sh. MR Job history server using /sbin/mr-jobhistory-daemon.sh start history server. Then, to stop these daemons we can use  
./sbin/stop-dfs.sh  
./sbin/stop-yarn.sh  
/sbin/mr-jobhistory-daemon.sh stop historyserver.  
• Finally, the last way is to start all the daemons individually. Then, stop them individually:  
./sbin/hadoop-daemon.sh start namenode  
./sbin/hadoop-daemon.sh start datanode  
./sbin/yarn-daemon.sh start resourcemanager  
./sbin/yarn-daemon.sh start nodemanager  
./sbin/mr-jobhistory-daemon.sh start historyserver

**Read – [Hadoop Commands with Examples and Usage Part](https://data-flair.training/blogs/hdfs-hadoop-commands/)**

**29) What is configured in /etc/hosts and what is its role in setting Hadoop cluster?**

***./etc/hosts*** file contains the hostname and their *IP address* of that host. It also, maps the IP address to the hostname. In hadoop cluster, we store all the hostnames (master and slaves) with their IP address in**./etc/hosts**. So, we can use hostnames easily instead of IP addresses.

**30) How is the splitting of file invoked in Hadoop framework?**

Input file store data for Hadoop MapReduce task’s, and these files typically reside in HDFS. *InputFormat* defines how these input files split and read. It is also responsible for creating **[InputSplit](https://data-flair.training/blogs/inputsplit-in-hadoop-mapreduce/)**, which is the logical representation of data. **[InputFormat](https://data-flair.training/blogs/hadoop-inputformat-types/)**also divides split into records. Then, mapper will process each record (which is a key-value pair). Hadoop framework invokes Splitting of the file by running *getInputSplit()*method. This method belongs to InputFormat class (like FileInputFormat) defined by the user.

**[Learn Hadoop from Industry Experts](https://data-flair.training/big-data-hadoop/" \t "_blank)**

**31) Is it possible to provide multiple input to Hadoop? If yes then how?**

Yes, it is possible by using *MultipleInputs* class.  
For example:  
If we had weather data from the UK Met Office. And we want to combine with the NCDC data for our maximum temperature analysis. Then, we can set up the input as follows:

1. MultipleInputs.addInputPath(job,ncdcInputPath,TextInput**For**mat.**class**,MaxTemperatureMapper.**class**);
2. MultipleInputs.addInputPath(job,metofficeInputPath,TextInput**For**mat.**class**, MetofficeMaxTemperatureMapper.**class**);

The above code replaces the usual calls to*FileInputFormat.addInputPath()* and *job.setmapperClass()*. Both the Met Office and NCDC data are text based. So, we use TextInputFormat for each. And, we will use two different mappers, as the two data sources have different line format. The *MaxTemperatureMapperr* reads NCDC input data and extracts the year and temperature fields. The *MetofficeMaxTemperatureMappers* reads Met Office input data. Then, extracts the year and temperature fields.

**Read More on [InputSplit in Hadoop MapReduce](https://data-flair.training/blogs/inputsplit-in-hadoop-mapreduce/)**

**32) Is it possible to have hadoop job output in multiple directories? If yes, how?**

Yes, it is possible by using following approaches:

**a. Using MultipleOutputs class-**

This class simplifies writing output data to multiple outputs.

MultipleOutputs.addNamedOutput(job,”OutputFileName”,Output**For**matClass,keyClass,valueClass);

The API provides two overloaded write methods to achieve this

MultipleOutput.write(‘OutputFileName”, **new** Text (key), **new** Text(value));

Then, we need to use overloaded write method, with an extra parameter for the base output path. This will allow to write the output file to separate output directories.

MultipleOutput.write(‘OutputFileName”, **new** Text (key), **new** Text(value), baseOutputPath);

Then, we need to change your baseOutputpath in each of our implementation.

**b. Rename/Move the file in driver class-**

This is the easiest hack to write output to multiple directories. So, we can use MultipleOutputs and write all the output files to a single directory. But the file names need to be different for each category.

We have categorized the above Hadoop Scenario based Interview Questions for Hadoop developers, Hadoop adminstrators, Hadoop architect etc for freshers as well as for experienced candidates.

* Hadoop Interview Questions and Answers for Freshers – Q. No.- 5 -11.
* Hadoop Interview Questions and Answers for Experienced – Q. No.- 11- 32.

[**Click here**](http://data-flair.training/blogs/50-hadoop-interview-questions-and-answers/)for more frequently asked Hadoop real time interview Questions and Answers for Freshers and Experienced.

**5. HDFS Hadoop Interview Questions and Answers**

In this section, we have covered top HDFS Hadoop Interview Questions and Answers. In the below Hadoop Interview questions we have covered what is HDFS, HDFS components, indexing, read-write operations, blocks, replication etc. So, Lets get started with the Hadoop Interview Questions on HDFS.

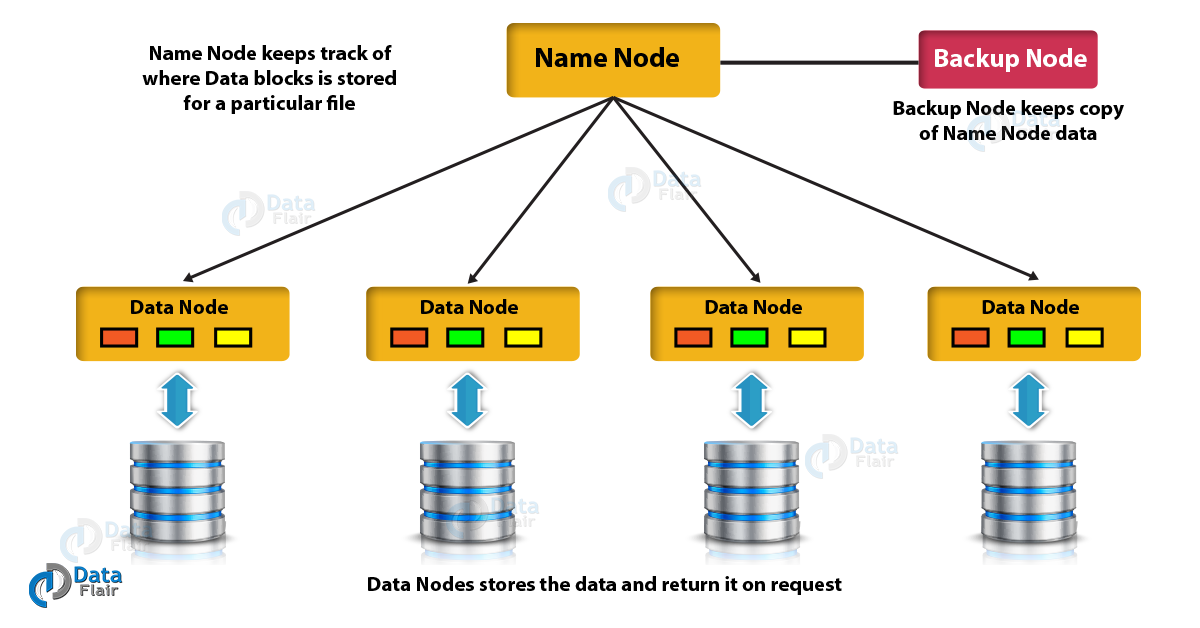
**1) What is HDFS- Hadoop Distributed File System?**

*Hadoop distributed file system (HDFS)* is the primary storage system of Hadoop. HDFS stores very large files running on a cluster of commodity hardware. It works on the principle of storage of less number of large files rather than the huge number of small files. HDFS stores data reliably even in the case of hardware failure. It provides high throughput access to the application by accessing in parallel.  
**Components of HDFS:**

* **NameNode –** It is also known as Master node. Namenode stores meta-data i.e. number of blocks, their replicas and other details.
* **DataNode –** It is also known as Slave. In Hadoop HDFS, DataNode is responsible for storing actual data. DataNode performs read and write operation as per request for the clients in HDFS.

[**Read Hadoop HDFS in detail**](http://data-flair.training/blogs/comprehensive-hdfs-guide-introduction-architecture-data-read-write-tutorial/)

**2) Explain NameNode and DataNode in HDFS?**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/Name-Node.png)

*Hadoop Interview Questions – Name Node and Data Node*

**I. NameNode –** It is also known as *Master node*. Namenode stores meta-data i.e. number of blocks, their location, replicas and other details. This meta-data is available in memory in the master for faster retrieval of data. NameNode maintains and manages the slave nodes, and assigns tasks to them. It should be deployed on reliable hardware as it is the centerpiece of HDFS.  
***Task of NameNode***

* Manage file system namespace.
* Regulates client’s access to files.
* In HDFS, NameNode also executes file system execution such as naming, closing, opening files and directories.

**II. DataNode –** It is also known as *Slave*. In Hadoop HDFS, DataNode is responsible for storing actual data in HDFS. DataNode performs read and write operation as per request for the clients. One can deploy the DataNode on commodity hardware.  
***Task of DataNode***

* In HDFS, DataNode performs various operations like block replica creation, deletion, and replication according to the instruction of NameNode.
* DataNode manages data storage of the system.

**Read:**[**HDFS Blocks & Data Block Size**](https://data-flair.training/blogs/data-block/)

**3) Why is block size set to 128 MB in Hadoop HDFS?**

Block is a continuous location on the hard drive which stores the data. In general, FileSystem store data as a collection of blocks. HDFS stores each file as blocks, and distributes it across the [Hadoop](http://data-flair.training/blogs/install-hadoop-2-x-ubuntu-hadoop-multi-node-cluster/)**[cluster](http://data-flair.training/blogs/install-hadoop-2-x-ubuntu-hadoop-multi-node-cluster/)**. In HDFS, the default size of data block is 128 MB, which we can configure as per our requirement. Block size is set to 128 MB:

* To reduce the disk seeks (IO). Larger the block size, lesser the file blocks and less number of disk seek and transfer of the block can be done within respectable limits and that to parallelly.
* HDFS have huge data sets, i.e. terabytes and petabytes of data. If we take 4 KB block size for HDFS, just like[Linux](http://data-flair.training/blogs/frequent-linux-commands-beginners-tutorial/) file system, which have 4 KB block size, then we would be having too many blocks and therefore too much of metadata. Managing this huge number of blocks and metadata will create huge overhead and traffic which is something which we don’t want. So, the block size is set to 128 MB.

On the other hand, block size can’t be so large that the system is waiting a very long time for the last unit of data processing to finish its work.

**4) How data or file is written into HDFS?**

When a client wants to write a file to HDFS, it communicates to namenode for metadata. The Namenode responds with details of a number of blocks, replication factor. Then, on basis of information from NameNode, client split files into multiple blocks. After that client starts sending them to first DataNode. The client sends block A to Datanode 1 with other two Datanodes details.

When Datanode 1 receives block A sent from the client, Datanode 1 copy same block to Datanode 2 of the same rack. As both the Datanodes are in the same rack so block transfer via rack switch. Now Datanode 2 copies the same block to Datanode 3. As both the Datanodes are in different racks so block transfer via an out-of-rack switch.

After the Datanode receives the blocks from the client. Then Datanode sends write confirmation to Namenode. Now Datanode sends write confirmation to the client. The Same process will repeat for each block of the file. Data transfer happen in parallel for faster write of blocks.

[**Read HDFS file write operation workflow in detail**](http://data-flair.training/blogs/hdfs-data-write-operation/)

**[Fascinated by Big Data? Enroll Now!](https://data-flair.training/big-data-hadoop/" \t "_blank)**

**6. HDFS Hadoop Interview Questions for Freshers:**

The HDFS Hadoop Intrview Questions from number 5 – 8 are basic questions for freshers. However Experienced People can also read them for understanding the basics.

**5) Can multiple clients write into an HDFS file concurrently?**

Multiple clients cannot write into an HDFS file at same time. Apache Hadoop HDFS follows single writer multiple reader models. The client which opens a file for writing, the NameNode grant a lease. Now suppose, some other client wants to write into that file. It asks NameNode for the write operation. NameNode first checks whether it has granted the lease for writing into that file to someone else or not. When someone already acquires the lease, then, it will reject the write request of the other client.

**6) How data or file is read in HDFS?**

To read from HDFS, the first client communicates to namenode for metadata. A client comes out of namenode with the name of files and its location. The Namenode responds with details of the number of blocks, replication factor. Now client communicates with Datanode where the blocks are present. Clients start reading data parallel from the Datanode. It read on the basis of information received from the namenodes.

Once client or application receives all the blocks of the file, it will combine these blocks to form a file. For read performance improvement, the location of each block ordered by their distance from the client. HDFS selects the replica which is closest to the client. This reduces the read latency and bandwidth consumption. It first read the block in the same node. Then another node in the same rack, and then finally another Datanode in another rack.

[**Read HDFS file read operation workflow in detail**](http://data-flair.training/blogs/hdfs-data-read-operation/)

**7) Why HDFS stores data using commodity hardware despite the higher chance of failures?**

HDFS stores data using commodity hardware because HDFS is highly fault-tolerant. HDFS provides fault tolerance by replicating the data blocks. And then distribute it among different DataNodes across the cluster. By default, replication factor is 3 which is configurable. Replication of data solves the problem of data loss in unfavorable conditions. And unfavorable conditions are crashing of the node, hardware failure and so on. So, when any machine in the cluster goes down, then the client can easily access their data from another machine. And this machine contains the same copy of data blocks.

**Learn**[**: Automatic Failover in Hadoop**](https://data-flair.training/blogs/hadoop-namenode-automatic-failover/)

**8) How is indexing done in HDFS?**

Hadoop has a unique way of**indexing**. Once Hadoop framework store the data as per the block size. HDFS will keep on storing the last part of the data which will say where the next part of the data will be. In fact, this is the base of HDFS.

**7. HDFS Hadoop Interview Questions for Experienced:**

The Hadoop Interview Questions from 9- 23 are for experienced people but freshers and students can also read them for advanced understanding

**9) What is a Heartbeat in HDFS?**

**Heartbeat** is the signals that NameNode receives from the DataNodes to show that it is functioning (alive). NameNode and DataNode do communicate using Heartbeat. If after the certain time of heartbeat NameNode do not receive any response from DataNode, then that Node is dead. The NameNode then schedules the creation of new replicas of those blocks on other DataNodes.

Heartbeats from a Datanode also carry information about total storage capacity. Also, carry the fraction of storage in use, and the number of data transfers currently in progress.  
The default heartbeat interval is 3 seconds. One can change it by using ***dfs.heartbeat.interval***in ***hdfs-site.xml.***

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

One can copy a file into HDFS with a different [**block size**](https://data-flair.training/blogs/data-block/)by using:

***–Ddfs.blocksize=block\_size****, where block\_size is in bytes.*

So, let us explain it with an example:

Suppose, you want to copy a file called test.txt of size, say of 128 MB, into the hdfs. And for this file, you want the block size to be 32MB (33554432 Bytes) in place of the default (128 MB). So, you would issue the following command:

Hadoop fs –Ddfs.blocksize=33554432 –copyFromlocal/home/dataflair/test.txt/sample\_hdfs

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

[](https://data-flair.training/blogs/category/interview-questions/hadoop-interview-questions/)

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

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

**11) Why HDFS performs replication, although it results in data redundancy?**

In HDFS, Replication provides the fault tolerance. Data replication is one of the most important and unique features of HDFS. Replication of data solves the problem of data loss in unfavorable conditions. Unfavorable conditions are crashing of the node, hardware failure and so on. HDFS by default creates 3 replicas of each block across the cluster in Hadoop. And we can change it as per the need. So, if any node goes down, we can recover data on that node from the other node.

In HDFS, Replication will lead to the consumption of a lot of space. But the user can always add more nodes to the cluster if required. It is very rare to have free space issues in the practical cluster. 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 one can also use different codec provided by the Hadoop to compress the data.

**Read:**[**Namenode High Availability**](https://data-flair.training/blogs/hadoop-high-availability-tutorial/)

**12) What is the default replication factor and how will you change it?**

The default replication factor is 3. One can change this in following three ways:

* By adding this property to hdfs-site.xml:
  1. <property>
  2. <name>dfs.replication</name>
  3. <value>5</value>
  4. <description>Block Replication</description>
  5. </property>
* You can also change the replication factor on per-file basis using the command:hadoop fs –setrep –w 3 / file\_location
* You can also change replication factor for all the files in a directory by using:hadoop fs –setrep –w 3 –R / directoey\_location

**Learn:**[**Read Write Operations in HDFS**](https://data-flair.training/blogs/hadoop-hdfs-data-read-and-write-operations/)

**13) Explain Hadoop Archives?**

Apache Hadoop HDFS stores and processes large (terabytes) data sets. However, storing a large number of small files in HDFS is inefficient, since each file is stored in a block, and block metadata is held in memory by the namenode.

Reading through small files normally causes lots of seeks and lots of hopping from datanode to datanode to retrieve each small file, all of which is inefficient data access pattern.

**Hadoop Archive (HAR)** basically deals with small files issue. HAR pack a number of small files into a large file, so, one can access the original files in parallel transparently (without expanding the files) and efficiently.

Hadoop Archives are special format archives. It maps to a file system directory. Hadoop Archive always has a ***\*.har***extension. In particular, Hadoop MapReduce uses Hadoop Archives as an Input.

**14) What do you mean by the High Availability of a NameNode in Hadoop HDFS?**

In Hadoop 1.0, NameNode is a single point of Failure (**SPOF**), if namenode fails, all clients including MapReduce jobs would be unable to read, write file or list files. In such event, whole Hadoop system would be out of service until new namenode is brought online.

Hadoop 2.0 overcomes this single point of failure by providing support for multiple NameNode.[**High availability**](http://data-flair.training/blogs/hadoop-high-availability-tutorial/) feature provides an extra NameNode (active standby NameNode) to Hadoop architecture which is configured for automatic failover. If active NameNode fails, then Standby Namenode takes all the responsibility of active node and cluster continues to work.  
The initial implementation of HDFS namenode high availability provided for single active namenode and single standby namenode. However, some deployment requires high degree fault-tolerance, this is enabled by new version 3.0, which allows the user to run multiple standby namenode. For instance configuring three namenode and five journal nodes, the cluster is able to tolerate the failure of two nodes rather than one.

[**Read HDFS NameNode High Availability in detail**](http://data-flair.training/blogs/hadoop-hdfs-namenode-high-availability/)

**15) What is Fault Tolerance in HDFS?**

Fault-tolerance in HDFS is working strength of a system in unfavorable conditions ( like the crashing of the node, hardware failure and so on). HDFS control faults by the process of replica creation. When client stores a file in HDFS, then the file is divided into blocks and blocks of data are distributed across different machines present in HDFS cluster. And, It creates a replica of each block on other machines present in the cluster. HDFS, by default, creates 3 copies of a block on other machines present in the cluster. If any machine in the cluster goes down or fails due to unfavorable conditions, then also, the user can easily access that data from other machines in the cluster in which replica of the block is present.

[**Read HDFS Fault Tolerance feature in detail**](http://data-flair.training/blogs/learn-hadoop-hdfs-fault-tolerance/)

**16) What is Rack Awareness?**

Rack Awareness improves the network traffic while reading/writing file. In which NameNode chooses the DataNode which is closer to the same rack or nearby rack. NameNode achieves rack information by maintaining the rack IDs of each DataNode. This concept that chooses Datanodes based on the rack information. In HDFS, NameNode makes sure that all the replicas are not stored on the same rack or single rack. It follows Rack Awareness Algorithm to reduce latency as well as fault tolerance.

Default replication factor is 3, according to Rack Awareness Algorithm. Therefore, the first replica of the block will store on a local rack. The next replica will store on another datanode within the same rack. And the third replica stored on the different rack.

In Hadoop, we need Rack Awareness because it improves:

* Data high availability and reliability.
* The performance of the cluster.
* Network bandwidth.

[**Read about Rack Awareness in Detail**](http://data-flair.training/blogs/rack-awareness-hadoop-hdfs/)

**17) Explain the Single point of Failure in Hadoop?**

In Hadoop 1.0, NameNode is a single point of Failure (SPOF). If namenode fails, all clients would unable to read/write files. In such event, whole Hadoop system would be out of service until new namenode is up.

Hadoop 2.0 overcomes this SPOF by providing support for multiple NameNode. High availability feature provides an extra NameNode to Hadoop architecture. This feature provides [automatic failover.](http://data-flair.training/blogs/hadoop-namenode-automatic-failover/) If active NameNode fails, then Standby-Namenode takes all the responsibility of active node. And cluster continues to work.  
The initial implementation of Namenode high availability provided for single active/standby namenode. However, some deployment requires high degree fault-tolerance. So new version 3.0 enable this feature by allowing the user to run multiple standby namenode. For instance configuring three namenode and five journal nodes. So, the cluster is able to tolerate the failure of two nodes rather than one.

**18) Explain Erasure Coding in Hadoop?**

In Hadoop, by default HDFS replicates each block three times for several purposes. Replication in HDFS is very simple and robust form of redundancy to shield against the failure of datanode. But replication is very expensive. Thus, 3 x replication scheme has 200% overhead in storage space and other resources.

Thus, **[Hadoop 2.x](http://data-flair.training/blogs/install-hadoop-2-x-ubuntu-hadoop-multi-node-cluster/)**introduced **Erasure Coding** a new feature to use in the place of **Replication**. It also provides the same level of fault tolerance with less space store and 50% storage overhead.

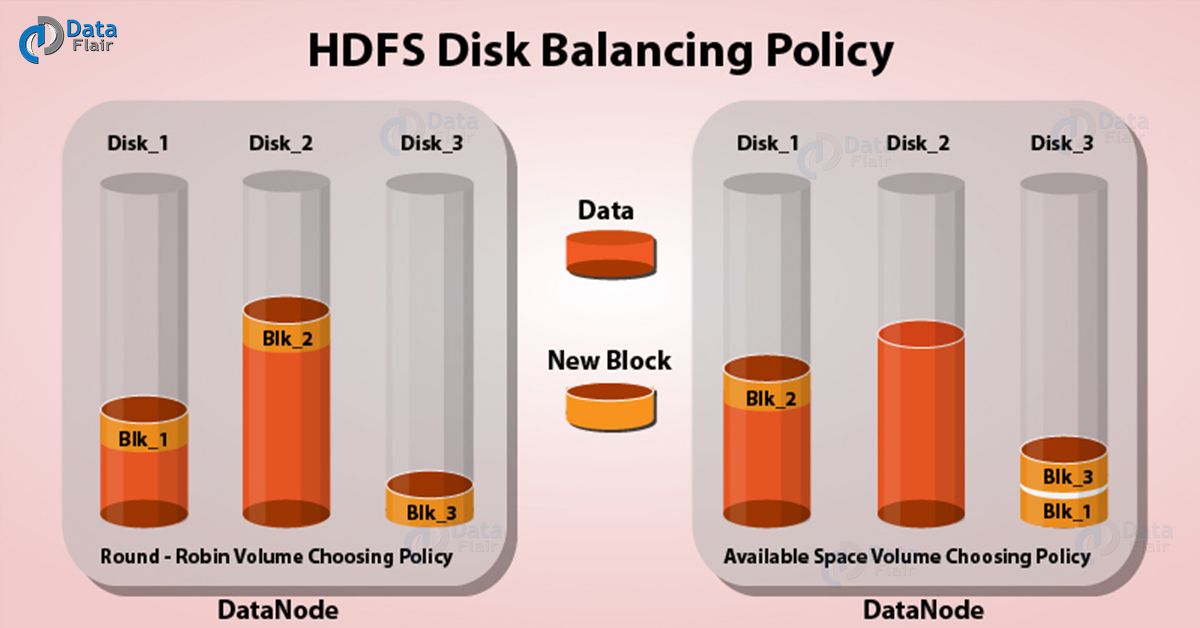
Erasure Coding uses Redundant Array of Inexpensive Disk **(RAID).** RAID implements EC through striping. In which it divide logical sequential data (such as a file) into the smaller unit (such as bit, byte or block). Then, stores data on different disk.

**Encoding-** In this process, RAID calculates and sort Parity cells for each strip of data cells. And recover error through the parity. Erasure coding extends a message with redundant data for fault tolerance. EC codec operates on uniformly sized data cells. In Erasure Coding, codec takes a number of data cells as input and produces parity cells as the output. Data cells and parity cells together are called an erasure coding group.  
There are two algorithms available for Erasure Coding:

* XOR Algorithm
* Reed-Solomon Algorithm

[**Read about Erasure Coding in detail**](http://data-flair.training/blogs/what-is-erasure-coding-introduction-hadoop-hdfs/)

**19) What is Disk Balancer in Hadoop?**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/HDFS-Disk-Balancing-Policy-01.jpg)

*Hadoop Interview Questions for Experienced – Disk Balancer*

HDFS provides a command line tool called **Diskbalancer.** It distributes data evenly on all disks of a datanode. This tool operates against a given datanode and moves blocks from one disk to another.  
*Disk balancer* works by creating a plan (set of statements) and executing that plan on the datanode. Thus, the plan describes how much data should move between two disks. A plan composes multiple steps. Move step has source disk, destination disk and the number of bytes to move. And the plan will execute against an operational datanode.

By default, disk balancer is not enabled; Hence, to enable disk balancer **dfs.disk.balancer.enabled** must be set true in hdfs-site.xml.

When we write new block in hdfs, then, datanode uses volume choosing the policy to choose the disk for the block. Each directory is the volume in hdfs terminology. Thus, two such policies are:

* **Round-robin:** It distributes the new blocks evenly across the available disks.
* **Available space:** It writes data to the disk that has maximum free space (by percentage).

[**Read Hadoop Disk Balancer in Detail**](http://data-flair.training/blogs/hadoop-hdfs-disk-balancer/)

If you don’t understand any Hadoop Interview Questions and Answers, ask us in the comment section and our support team will get back to you.

**20) How would you check whether your NameNode is working or not?**

There are several ways to check the status of the NameNode. Mostly, one uses the jps command to check the status of all daemons running in the HDFS.

**21) 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. Metadata information is a number of blocks, their location, replicas and other details. It also executes file system execution such as naming, closing, opening files/directories.

Therefore, NameNode requires higher RAM for storing the metadata for millions of files. Whereas, DataNode is responsible for storing actual data in HDFS. It performs[**read and write operation**](http://data-flair.training/blogs/hadoop-hdfs-data-read-and-write-operations/) as per request of the clients. Therefore, Datanode needs to have a higher disk capacity for storing huge data sets.

**Learn: [Namenode High Availability in Hadoop](https://data-flair.training/blogs/hadoop-hdfs-namenode-high-availability/)**

**22) What are file permissions in HDFS and how HDFS check permissions for files or directory?**

For files and directories,*Hadoop distributed file system (HDFS)* implements a permissions model. For each file or directory, thus, we can manage permissions for a set of 3 distinct user classes:  
*The****owner****,****group****, and****others****.*

The 3 different permissions for each user class: **Read (r)**, **write (w)**, and **execute(x)**.

* For files, the **r** permission is to*read* the file, and the **w** permission is to *write* to the file.
* For directories, the **r** permission is to *list the contents* of *the directory.* The **w**permission is to *create or delete the directory.*
* **X** permission is to access a child of the directory.

HDFS check permissions for files or directory:

* We can also check the owner’s permissions if the username matches the owner of the directory.
* If the group matches the directory’s group, then Hadoop tests the user’s group permissions.
* Hadoop tests the “other” permission when the owner and the group names don’t match.
* If none of the permissions checks succeed, the client’s request is denied.

**23) If DataNode increases, then do we need to upgrade NameNode?**

Namenode stores meta-data i.e. number of blocks, their location, replicas. This meta-data is available in memory in the master for faster retrieval of data. NameNode maintains and manages the slave nodes, and assigns tasks to them. It regulates client’s access to files.  
It also executes file system execution such as naming, closing, opening files/directories.  
During **[Hadoop installation](http://data-flair.training/blogs/install-deploy-cloudera-hadoop-cdh5-apache-2-x-centos/)**, framework determines NameNode based on the size of the cluster. Mostly we don’t need to upgrade the NameNode because it does not store the actual data. But it stores the metadata, so such requirement rarely arise.

These Big Data Hadoop interview questions are the selected ones which are asked frequently and by going through these HDFS interview questions you will be able to answer many other related answers in your interview.

We have categorized the above Big Data Hadoop interview questions and answers for HDFS Interview for freshers and experienced.

* Hadoop Interview Questions and Answers for Freshers – Q. No.- 5 – 8.
* Hadoop Interview Questions and Answers for Experienced – Q. No.- 8-23.

Here are few more frequently asked **[Hadoop HDFS interview Questions and Answers for Freshers and Experienced.](http://data-flair.training/blogs/hdfs-interview-questions-and-answers/)**

**8. MapReduce Hadoop Interview Questions and Answers**

In this section, we have discussed top MapReduce hadoop Interview Questions for freshers and experienced. In the below Hadoop Interview Questions on MapReduce we have taken the questions on topics likes What is Hadoop MapReduce, Key – value pair, Mappers, combiners, reducers etc. Lets get started with the MapReduce Interview questions on MapReduce.

The Hadoop Interview questions from 1 – 6 are for freshers but experienced professionals can also refer these Hadoop Interview Questions for basic understanding

**1) What is Hadoop MapReduce?**

**MapReduce** is the data processing layer of Hadoop. It is the framework for writing applications that process the vast amount of data stored in the HDFS. It processes a huge amount of data in parallel by dividing the job into a set of independent tasks (sub-job). In Hadoop, MapReduce works by breaking the processing into phases: *Map and Reduce.*

* **Map-** It is the first phase of processing. In which we specify all the complex logic/business rules/costly code. The map takes a set of data and converts it into another set of data. It also breaks individual elements into tuples (key-value pairs).
* **Reduce-** It is the second phase of processing. In which we specify light-weight processing like aggregation/summation. Reduce takes the output from the map as input. After that, it combines tuples (key-value) based on the key. And then, modifies the value of the key accordingly.

[**Read Hadoop MapReduce in detail**](http://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/)

**2) Why Hadoop MapReduce?**

When we store huge amount of data in HDFS, the first question arises is, how to process this data?  
Transferring all this data to a central node for processing is not going to work. And we will have to wait forever for the data to transfer over the network. Google faced this same problem with its *Distributed Goggle File System (GFS)*. It solved this problem using a **MapReduce** data processing model.  
**Challenges before MapReduce:**

* **Costly –** All the data (terabytes) in one server or as database cluster which is very expensive. And also hard to manage.
* **Time-consuming –** By using single machine we cannot analyze the data (terabytes) as it will take a lot of time.

**MapReduce overcome these challenges:**

* **Cost-efficient –** It distribute the data over multiple low config machines.
* **Time-efficient –** If we want to analyze the data. We can write the analysis code in Map function. And the integration code in Reduce function and execute it. Thus, this MapReduce code will go to every machine which has a part of our data and executes on that specific part. Hence instead of moving terabytes of data, we just move kilobytes of code. So this type of movement is time-efficient.

Read more on Hadoop **[MapReduce](https://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/)**

**3) What is the key- value pair in MapReduce?**

Hadoop MapReduce implements a data model, which represents data as *key-value pairs*. Both input and output to MapReduce Framework should be in Key-value pairs only.

In Hadoop, if the schema is static we can directly work on the column instead of key-value. But, the schema is not static we will work on keys and values. Keys and values are not the intrinsic properties of the data. But the user analyzing the data chooses a key-value pair. A Key-value pair in Hadoop MapReduce generate in following way:

* **InputSplit-** It is the logical representation of data. InputSplit represents the data which individual Mapper will process.
* **RecordReader-** It communicates with the InputSplit (created by InputFormat). And converts the split into records. Records are in form of Key-value pairs that are suitable for reading by the mapper. By Default RecordReader uses TextInputFormat for converting data into a key-value pair.

**Key-** It is the byte offset of the beginning of the line within the file, so it will be unique if combined with the file.  
**Value-** It is the contents of the line, excluding line terminators. For Example file content is- on the top of the crumpetty Tree  
Key- 0  
Value- on the top of the crumpetty Tree

[**Read about Hadoop Key-Value Pair in detail**](http://data-flair.training/blogs/key-value-pairs-hadoop-mapreduce/)

**4) Why MapReduce uses the key-value pair to process the data?**

MapReduce works on unstructured and semi-structured data apart from structured data. One can read the Structured data like the ones stored in **RDBMS** by columns. But handling unstructured data is feasible using *key-value pairs*. And the very core idea of MapReduce work on the basis of these pairs. Framework map data into a collection of key-value pairs by ***[mapper](http://data-flair.training/blogs/mapper-in-hadoop-mapreduce/)*** and[***reducer***](http://data-flair.training/blogs/reducer-in-hadoop-mapreduce/)on all the pairs with the same key. So as stated by Google themselves in their research publication. In most of the computations-  
Map operation applies on each logical “**record**” in our input. This computes a set of intermediate key-value pairs. Then apply reduce operation on all the values that share the same key. This combines the derived data properly.

In conclusion, we can say that key-value pairs are the best solution to work on data problems on MapReduce.

[**Read about MapReduce key-value pair in detail**](http://data-flair.training/blogs/key-value-pairs-hadoop-mapreduce/)

**5) How many Mappers run for a MapReduce job in Hadoop?**

**Mapper** task processes each input record (from**[RecordReader](http://data-flair.training/blogs/recordreader-in-hadoop-mapreduce/)**) and generates a key-value pair. The number of mappers depends on 2 factors:

* The amount of data we want to process along with block size. It depends on the number of InputSplit. If we have the block size of 128 MB and we expect 10TB of input data, thus we will have 82,000 maps. Ultimately ***[InputFormat](http://data-flair.training/blogs/hadoop-inputformat-types/)*** determines the number of maps.
* The configuration of the slave i.e. number of core and RAM available on the slave. The right number of map/node can between 10-100. Hadoop framework should give 1 to 1.5 cores of the processor to each mapper. Thus, for a 15 core processor, 10 mappers can run.  
  In MapReduce job, by changing the block size one can control the number of Mappers. Hence, by Changing block size the number of InputSplit increases or decreases.  
  By using the JobConf’s *conf.setNumMapTasks(int num)* one can increase the number of map tasks manually.  
  Mapper= {(total data size)/ (input split size)}  
  If data size= 1 Tb and input split size= 100 MB  
  Hence, Mapper= (1000\*1000)/100= 10,000

**6) How many Reducers run for a MapReduce job in Hadoop?**

**Reducer** takes a set of an intermediate *key-value pair* produced by the *mapper* as the input. Then runs a reduce function on each of them to generate the output. Thus, the output of the reducer is the final output, which it stores in HDFS. Usually, in the reducer, we do aggregation or summation sort of computation.

With the help of *Job.setNumreduceTasks(int)* the user set the number of reducers for the job. Hence the right number of reducers are set by the formula:

*0.95 Or 1.75 multiplied by (<no. of nodes> \* <no. of the maximum container per node>).*

With 0.95, all the reducers can launch immediately and start transferring map outputs as the map finish.  
With 1.75, faster node finishes the first round of reduces and then launch the second wave of reduces.

By increasing the number of reducers:

* Framework overhead increases
* Increases load balancing
* Lowers the cost of failures

**Read:**[**What is Reducer in MapReduce?**](https://data-flair.training/blogs/reducer-in-hadoop-mapreduce/)

**9. MapReduce Hadoop Interview Questions for Experienced:**

The following Hadoop Interview questions are more technical targeted for experienced professionals, however freshers can also refer them for better understanding.

**7) What is the difference between Reducer and Combiner in Hadoop MapReduce?**

The *Combiner* is Mini-Reducer that perform local reduce task. The Combiner runs on the Map output and produces the output to reducer input. A combiner is usually used for network optimization. Reducer takes a set of an intermediate key-value pair produced by the mapper as the input. Then runs a reduce function on each of them to generate the output. An output of the reducer is the final output.

* Unlike a reducer, the combiner has a limitation. i.e. the input or output key and value types must match the output types of the mapper.
* Combiners can operate only on a subset of keys and values. i.e. combiners can execute on functions that are commutative.
* Combiner functions take input from a single mapper. While reducers can take data from multiple mappers as a result of partitioning.

**8) What happens if the number of reducers is 0 in Hadoop?**

If we set the number of reducer to 0, then no reducer will execute and no aggregation will take place. In such case, we will prefer “**Map-only job**” in Hadoop. In a *map-Only job*, the map does all task with its InputSplit and the reducer does no job. Map output is the final output.

Between **map** and **reduce** phases there is key, sort, and shuffle phase. Sort and shuffle phase are responsible for sorting the keys in ascending order. Then grouping values based on same keys. This phase is very expensive. If reduce phase is not required we should avoid it. Avoiding reduce phase would eliminate sort and shuffle phase as well. This also saves network congestion. As in shuffling an output of mapper travels to the reducer, when data size is huge, large data travel to the reducer.

**9) What do you mean by shuffling and sorting in MapReduce?**

**Shuffling** and **Sorting** takes place after the completion of map task. Shuffle and sort phase in hadoop occurs simultaneously.  
**Shuffling-** It is the process of transferring data from the mapper to reducer. i.e., the process by which the system sorts the key-value output of the map tasks and transfer it to the reducer.  
So, shuffle phase is necessary for reducer, otherwise, they would not have any input. As shuffling can start even before the map phase has finished. So this saves some time and completes the task in lesser time.  
**Sorting-** Mapper generate the intermediate key-value pair. Before starting of reducer, MapReduce framework sort these key-value pairs by the keys.  
Sorting helps reducer to easily distinguish when a new reduce task should start. Thus saves time for the reducer.  
Shuffling and sorting are not performed at all if you specify zero reducers (*setNumReduceTasks(0)*).

Any doubt in the Hadoop Interview Questions and Answers yet? Just Drop  a comment and we will get back to you.

[**Read Shuffling and Sorting in detail**](http://data-flair.training/blogs/shuffling-sorting-hadoop-mapreduce/)

**10) What is the fundamental difference between a MapReduce InputSplit and HDFS block?**

**By definition**

* Block – Block is the continuous location on the hard drive where data HDFS store data. In general, FileSystem store data as a collection of blocks. In a similar way, HDFS stores each file as blocks, and distributes it across the**[Hadoop cluster](http://data-flair.training/blogs/install-deploy-cloudera-hadoop-cdh5-apache-2-x-centos/)**.
* InputSplit-**[InputSplit](http://data-flair.training/blogs/inputsplit-in-hadoop-mapreduce/)**represents the data which individual Mapper will process. Further split divides into records. Each record (which is a key-value pair) will be processed by the map.

**Data representation**

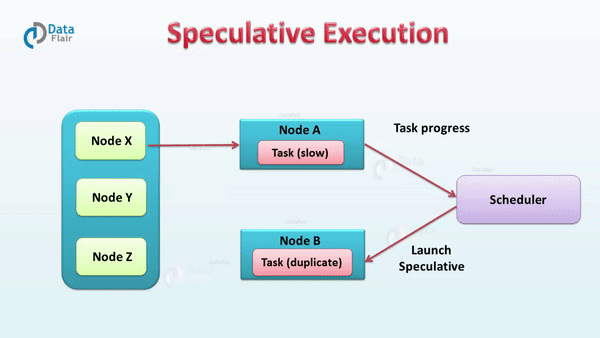
* Block- It is the physical representation of data.
* InputSplit- It is the logical representation of data. Thus, during data processing in MapReduce program or other processing techniques use InputSplit. In MapReduce, important thing is that InputSplit does not contain the input data. Hence, it is just a reference to the data.

**Size**

* Block- The default size of the HDFS block is 128 MB which is configured as per our requirement. All blocks of the file are of the same size except the last block. The last Block can be of same size or smaller. In Hadoop, the files split into 128 MB blocks and then stored into Hadoop Filesystem.
* InputSplit- Split size is approximately equal to block size, by default.

**Example**  
Consider an example, where we need to store the file in HDFS. HDFS stores files as blocks. Block is the smallest unit of data that can store or retrieved from the disk. The default size of the block is 128MB. HDFS break files into blocks and stores these blocks on different nodes in the cluster. We have a file of 130 MB, so HDFS will break this file into 2 blocks.  
Now, if we want to perform MapReduce operation on the blocks, it will not process, as the 2nd block is incomplete. InputSplit solves this problem. InputSplit will form a logical grouping of blocks as a single block. As the InputSplit include a location for the next block. It also includes the byte offset of the data needed to complete the block.  
From this, we can conclude that InputSplit is only a logical chunk of data. i.e. it has just the information about blocks address or location. Thus, during MapReduce execution, Hadoop scans through the blocks and create InputSplits. Split act as a broker between block and mapper.

**11) What is a Speculative Execution in Hadoop MapReduce?**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/Speculative-Execution-in-Spark.gif)

*Hadoop Interview Questions and Answers – Speculative Execution*

***MapReduce*** breaks jobs into tasks and these tasks run parallel rather than sequential. Thus reduces overall execution time. This model of execution is sensitive to slow tasks as they slow down the overall execution of a job. There are various reasons for the slowdown of tasks like hardware degradation. But it may be difficult to detect causes since the tasks still complete successfully. Although it takes more time than the expected time.

Apache Hadoop doesn’t try to diagnose and fix slow running task. Instead, it tries to detect them and run backup tasks for them. This is called Speculative execution in Hadoop. These backup tasks are called Speculative tasks in hadoop. First of all Hadoop framework launch all the tasks for the job in Hadoop MapReduce. Then it launches speculative tasks for those tasks that have been running for some time (one minute). And the task that has not made any much progress, on average, as compared with other tasks from the job. If the original task completes before the speculative task. Then it will kill the speculative task. On the other hand, it will kill the original task if the speculative task finishes before it.

[**Read Hadoop Speculative Execution in detail**](http://data-flair.training/blogs/speculative-execution-in-hadoop-mapreduce/)

**12) How to submit extra files(jars, static files) for MapReduce job during runtime?**

**MapReduce** framework provides [**Distributed Cache**](http://data-flair.training/blogs/hadoop-distributed-cache/) to caches files needed by the applications. It can cache read-only text files, archives, jar files etc.

First of all, an application which needs to use *distributed cache* to distribute a file should make sure that the files are available on URLs. Hence, URLs can be either **hdfs:// or http://.** Now, if the file is present on the hdfs:// or http://urls. Then, user mentions it to be cache file to distribute. This framework will copy the cache file on all the nodes before starting of tasks on those nodes. The files are only copied once per job. Applications should not modify those files.

By default size of the distributed cache is 10 GB. We can adjust the size of distributed cache using local.cache.size.

To seperate the above Interview Questions for freshers and experienced candidates, we have categorized the above Mapreduce Hadoop Interview Questions for freshers and experienced in the following manner.

* Hadoop Interview Questions and Answers for Freshers – Q. No.- 1 – 6.
* Hadoop Interview Questions and Answers for Experienced – Q. No.- 7 – 12.

Here are few more frequently asked **[MapReduce HDFS interview Questions and Answers for Freshers and Experi](http://data-flair.training/blogs/mapreduce-interview-questions/)**[**enced.**](http://data-flair.training/blogs/hdfs-interview-questions-and-answers/)

After going through these top 100 Hadoop Interview questions you will be able to confidently face a interview and will be able to answer Hadoop Interview questions asked in your interview in the best manner. These Hadoop Interview Questions are suggested by the experts at