1. What are the basic differences between relational database and HDFS?

Here are the key differences between HDFS and relational database:

RDBMS vs. Hadoop

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

2. Explain "Big Data" and what are five V's of Big Data?

"Big data" is the term for a collection of large and complex data sets, that makes it difficult to process using relational database management tools or traditional data processing applications. It is difficult to capture, curate, store, search, share, transfer, analyze, and visualize Big data. Big Data has emerged as an opportunity for companies. Now they can successfully derive value from their data and will have a distinct advantage over their competitors with enhanced business decisions making capabilities.

♣ *Tip: It will be a good idea to talk about the 5Vs in such questions, whether it is asked specifically or not!*

- **Volume**: The volume represents the amount of data which is growing at an exponential rate i.e. in Petabytes and Exabytes.
- **Velocity**: Velocity refers to the rate at which data is growing, which is very fast. Today, yesterday's data are considered as old data. Nowadays, social media is a major contributor to the velocity of growing data.
- **Variety**: Variety refers to the heterogeneity of data types. In another word, the data which are gathered has a variety of formats like videos, audios, csv, etc. So, these various formats represent the variety of data.
- **Veracity**: Veracity refers to the data in doubt or uncertainty of data available due to data inconsistency and incompleteness. Data available can sometimes get messy and may be

difficult to trust. With many forms of big data, quality and accuracy are difficult to control. The volume is often the reason behind for the lack of quality and accuracy in the data.

- **Value**: It is all well and good to have access to big data but unless we can turn it into a value it is useless. By turning it into value I mean, Is it adding to the benefits of the organizations? Is the organization working on Big Data achieving high ROI (Return On Investment)? Unless, it adds to their profits by working on Big Data, it is useless.

As we know Big Data is growing at an accelerating rate, so the factors associated with it are also evolving. To go through them and understand it in detail, I recommend you to go through *Big Data Tutorial* blog.

3. What is Hadoop and its components.

When "Big Data" emerged as a problem, Apache Hadoop evolved as a solution to it. Apache Hadoop is a framework which provides us various services or tools to store and process Big Data. It helps in analyzing Big Data and making business decisions out of it, which can't be done efficiently and effectively using traditional systems.

♣ *Tip: Now, while explaining Hadoop, you should also explain the main components of Hadoop,* i.e.:

- *Storage unit*– HDFS (NameNode, DataNode)
- *Processing framework*– YARN (ResourceManager, NodeManager)

4. What are HDFS and YARN?

HDFS (Hadoop Distributed File System) is the storage unit of Hadoop. It is responsible for storing different kinds of data as blocks in a distributed environment. It follows master and slave topology.

♣ *Tip: It is recommended to explain the HDFS components too i.e.*

- *NameNode***:** NameNode is the master node in the distributed environment and it maintains the metadata information for the blocks of data stored in HDFS like block location, replication factors etc.
- *DataNode***:** DataNodes are the slave nodes, which are responsible for storing data in the HDFS. NameNode manages all the DataNodes.

YARN (Yet Another Resource Negotiator) is the processing framework in Hadoop, which manages resources and provides an execution environment to the processes.

♣ *Tip: Similarly, as we did in HDFS, we should also explain the two components of YARN:*

- *ResourceManager***:** It receives the processing requests, and then passes the parts of requests to corresponding NodeManagers accordingly, where the actual processing takes place. It allocates resources to applications based on the needs.
- *NodeManager***:** NodeManager is installed on every DataNode and it is responsible for the execution of the task on every single DataNode.

List the difference between Hadoop 1 and Hadoop 2.

This is an important question and while answering this question, we have to mainly focus on two points i.e. Passive NameNode and YARN architecture.

- In Hadoop 1.x, “NameNode” is the single point of failure. In Hadoop 2.x, we have Active and Passive “NameNodes”. If the active “NameNode” fails, the passive “NameNode” takes charge. Because of this, high availability can be achieved in Hadoop 2.x.
- Also, in Hadoop 2.x, YARN provides a central resource manager. With YARN, you can now run multiple applications in Hadoop, all sharing a common resource. MRV2 is a particular type of distributed application that runs the MapReduce framework on top of YARN. Other tools can also perform data processing via YARN, which was a problem in Hadoop 1.x.

Hadoop 1.x vs. Hadoop 2.x

	Hadoop 1.x	Hadoop 2.x
Passive NameNode	NameNode is a Single Point of Failure	Active & Passive NameNode
Processing	MRV1 (Job Tracker & Task Tracker)	MRV2/YARN (ResourceManager & NodeManager)

8. What are active and passive “NameNodes”?

In HA (High Availability) architecture, we have two NameNodes – Active “NameNode” and Passive “NameNode”.

- Active “NameNode” is the “NameNode” which works and runs in the cluster.
- Passive “NameNode” is a standby “NameNode”, which has similar data as active “NameNode”.

When the active “NameNode” fails, the passive “NameNode” replaces the active “NameNode” in the cluster. Hence, the cluster is never without a “NameNode” and so it never fails.

9. Why does one remove or add nodes in a Hadoop cluster frequently?

One of the most attractive features of the Hadoop framework is its *utilization of commodity hardware*. However, this leads to frequent “DataNode” crashes in a Hadoop cluster. Another striking feature of Hadoop Framework is the *ease of scale* in accordance with the rapid growth in data volume. Because of these two reasons, one of the most common task of a Hadoop administrator is to commission (Add) and decommission (Remove) “Data Nodes” in a Hadoop Cluster.

Read this blog to get a detailed understanding on **commissioning and decommissioning nodes** in a Hadoop cluster.

10. What happens when two clients try to access the same file in the HDFS?

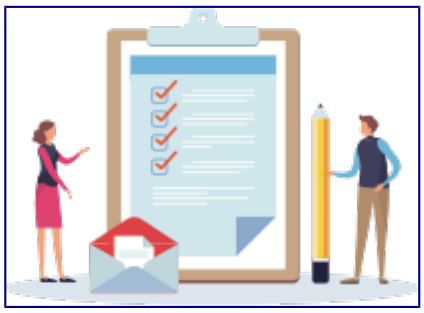
HDFS supports exclusive writes only.

When the first client contacts the “NameNode” to open the file for writing, the “NameNode” grants a lease to the client to create this file. When the second client tries to open the same file for writing,

the “NameNode” will notice that the lease for the file is already granted to another client, and will reject the open request for the second client.

11. How does NameNode tackle DataNode failures?

NameNode periodically receives a Heartbeat (signal) from each of the DataNode in the cluster, which implies DataNode is functioning properly.



Big Data Hadoop Certification Training

- Instructor-led Sessions
- Real-life Case Studies
- Assessments
- Lifetime Access

A block report contains a list of all the blocks on a DataNode. If a DataNode fails to send a heartbeat message, after a specific period of time it is marked dead.

The NameNode replicates the blocks of dead node to another DataNode using the replicas created earlier.

12. What will you do when NameNode is down?

The NameNode recovery process involves the following steps to make the Hadoop cluster up and running:

1. Use the file system metadata replica (FsImage) to start a new NameNode.
2. Then, configure the DataNodes and clients so that they can acknowledge this new NameNode, that is started.
3. Now the new NameNode will start serving the client after it has completed loading the last checkpoint FsImage (for metadata information) and received enough block reports from the DataNodes.

Whereas, on large Hadoop clusters this NameNode recovery process may consume a lot of time and this becomes even a greater challenge in the case of the routine maintenance. Therefore, we have HDFS High Availability Architecture which is covered in the **HA architecture** blog.

13. What is a checkpoint?

In brief, “Checkpointing” is a process that takes an FsImage, edit log and compacts them into a new FsImage. Thus, instead of replaying an edit log, the NameNode can load the final in-memory state

directly from the FsImage. This is a far more efficient operation and reduces NameNode startup time. Checkpointing is performed by Secondary NameNode.

14. How is HDFS fault tolerant?

When data is stored over HDFS, NameNode replicates the data to several DataNode. The default replication factor is 3. You can change the configuration factor as per your need. If a DataNode goes down, the NameNode will automatically copy the data to another node from the replicas and make the data available. This provides fault tolerance in HDFS.

15. Can NameNode and DataNode be a commodity hardware?

The smart answer to this question would be, DataNodes are commodity hardware like personal computers and laptops as it stores data and are required in a large number. But from your experience, you can tell that, NameNode is the master node and it stores metadata about all the blocks stored in HDFS. It requires high memory (RAM) space, so NameNode needs to be a high-end machine with good memory space.

16. Why do we use HDFS for applications having large data sets and not when there are a lot of small files?

HDFS is more suitable for large amounts of data sets in a single file as compared to small amount of data spread across multiple files. As you know, the NameNode stores the metadata information regarding the 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 many files will lead to the generation of too much metadata. And, storing these metadata in the RAM will become a challenge. As a thumb rule, metadata for a file, block or directory takes 150 bytes.

17. How do you define “block” in HDFS? What is the default block size in Hadoop 1 and in Hadoop 2? Can it be changed?

Blocks are the nothing but the smallest continuous location on your hard drive where data is stored. HDFS stores each as blocks, and distribute it across the Hadoop cluster. Files in HDFS are broken down into block-sized chunks, which are stored as independent units.

- Hadoop 1 default block size: 64 MB
- Hadoop 2 default block size: 128 MB

Yes, blocks can be configured. The dfs.block.size parameter can be used in the hdfs-site.xml file to set the size of a block in a Hadoop environment.

18. What does ‘jps’ command do?

The ‘jps’ command helps us to check if the Hadoop daemons are running or not. It shows all the Hadoop daemons i.e namenode, datanode, resourcemanager, nodemanager etc. that are running on the machine.

19. How do you define “Rack Awareness” in Hadoop?

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

To know rack awareness in more detail, refer to the **HDFS architecture** blog.

20. What is “speculative execution” in Hadoop?

If a node appears to be executing a task slower, the master node can redundantly execute another instance of the same task on another node. Then, the task which finishes first will be accepted and the other one is killed. This process is called “speculative execution”.

21. How can I restart “NameNode” or all the daemons in Hadoop?

This question can have two answers, we will discuss both the answers. We can restart NameNode by following methods:

1. You can stop the NameNode individually using. */sbin /hadoop-daemon.sh stop namenode* command and then start the NameNode using. */sbin/hadoop-daemon.sh start namenode* command.
2. To stop and start all the daemons, use**.** */sbin/stop-all.sh* and then use *./sbin/start-all.sh* command which will stop all the daemons first and then start all the daemons.

5. Tell me about the various Hadoop daemons and their roles in a Hadoop cluster.

Generally approach this question by first explaining the HDFS daemons i.e. NameNode, DataNode and Secondary NameNode, and then moving on to the YARN daemons i.e. ResorceManager and NodeManager, and lastly explaining the JobHistoryServer.

- **NameNode:** It is the master node which is responsible for storing the metadata of all the files and directories. It has information about blocks, that make a file, and where those blocks are located in the cluster.
- **Datanode:** It is the slave node that contains the actual data.
- **Secondary NameNode:** It periodically merges the changes (edit log) with the FsImage (Filesystem Image), present in the NameNode. It stores the modified FsImage into persistent storage, which can be used in case of failure of NameNode.
- **ResourceManager:** It is the central authority that manages resources and schedule applications running on top of YARN.
- **NodeManager:** It runs on slave machines, and is responsible for launching the application’s containers (where applications execute their part), monitoring their resource usage (CPU, memory, disk, network) and reporting these to the ResourceManager.

- **JobHistoryServer:** It maintains information about MapReduce jobs after the Application Master terminates.