HADOOP INTERVEW QUESTION

1. Explain Hadoop Architecture

Hadoop is an open-source framework for distributed storage and processing of large-scale data sets. It provides a reliable, scalable, and fault-tolerant platform for big data processing. The architecture of Hadoop consists of several key components working together to handle storage and processing tasks:

Hadoop Distributed File System (HDFS):

- HDFS is the primary storage component of Hadoop.
- It is designed to store large files across a cluster of machines, providing fault tolerance and high throughput.

YARN (Yet Another Resource Negotiator):

- YARN is the resource management and job scheduling framework of Hadoop.
- It manages resources in a Hadoop cluster and schedules tasks for execution.

MapReduce:

- MapReduce is a programming model and processing framework for parallel processing of large data sets.

Hadoop Common:

- Hadoop Common provides the common utilities and libraries used by other Hadoop components.

Hadoop Ecosystem:

- Hadoop has a rich ecosystem of additional components and tools that work together with the core components.
- These components include Apache Hive (data warehousing), Apache Pig (data analysis), Apache HBase (distributed database), Apache Spark (in-memory processing), Apache Kafka (distributed messaging), and many others.

2. Configuration files used during hadoop installation

During the installation and configuration of Hadoop, several configuration files are used to customize the behavior and settings of various Hadoop components.

`core-site.xml`:

- This file contains core configuration properties that are shared across Hadoop components.

`hdfs-site.xml`:

- This file contains configuration properties specific to the Hadoop Distributed File System (HDFS).

`mapred-site.xml`:

- This file contains configuration properties related to the MapReduce framework (deprecated in newer Hadoop versions in favor of YARN).

`yarn-site.xml`:

- This file contains configuration properties specific to the YARN resource management and job scheduling framework.

`hadoop-env.sh` (or **`hadoop-env.cmd`** on Windows):

- This file defines environment variables and options that affect Hadoop's execution environment.

`workers` (or **`slaves`**):

- This file contains a list of worker nodes (DataNodes and NodeManagers) in the Hadoop cluster.

`masters`:

- This file specifies the machine that acts as the master node (NameNode and ResourceManager) in the Hadoop cluster.

3. Difference between Hadoop fs and hdfs dfs

Both **`hadoop fs`** and **`hdfs dfs`** are command-line utilities provided by Hadoop for interacting with the Hadoop Distributed File System (HDFS). However, there is no functional difference between the two; they are essentially the same command with different names.

In earlier versions of Hadoop (1.x and earlier), the command to interact with HDFS was **`hadoop fs`**.

With the introduction of Hadoop 2.x and the separation of resource management from the file system, the command was renamed to **`hdfs dfs`** to align with the specific functionality of the Hadoop Distributed File System (HDFS).

4. Difference between Hadoop 2 and Hadoop 3.

Hadoop 2 and Hadoop 3 are major versions of the Apache Hadoop framework. Here are some key differences between the two:

YARN (Yet Another Resource Negotiator):

- Hadoop 2 introduced YARN as a resource management and job scheduling framework. It decoupled the resource management functionality from the MapReduce processing framework, allowing Hadoop to support alternative processing models like Apache Spark and Apache Flink.

- Hadoop 3 further enhances YARN with several improvements, including better resource allocation, support for GPU resources, and support for distributed scheduling.

Scalability and Performance:

- Hadoop 3 includes several performance improvements and optimizations, such as better memory management, faster data replication, and reduced memory overhead in YARN. These enhancements improve the overall scalability and performance of Hadoop applications.

Erasure Coding:

- Hadoop 3 introduces native support for erasure coding in HDFS. Erasure coding is an alternative to traditional replication, offering higher storage efficiency by using less storage overhead compared to replication while maintaining data durability and fault tolerance.

Federation and ViewFS:

- Hadoop 2 introduced HDFS Federation, allowing multiple independent HDFS clusters to be managed as a single namespace. This allows scaling HDFS horizontally and provides better fault tolerance.
- Hadoop 3 enhances Federation with the introduction of ViewFS, which provides a unified view of multiple federated HDFS clusters. ViewFS allows applications to access multiple federated clusters transparently.

Containerization and Docker Support:

- Hadoop 3 includes improved support for containerization, allowing Hadoop components to run in container environments like Docker. It provides better isolation, flexibility, and resource management when running Hadoop applications in containerized environments.

High Availability Improvements:

- Hadoop 3 includes various improvements in high availability, such as faster and more efficient failover mechanisms for NameNode and ResourceManager, reducing the downtime during failover scenarios.

Security Enhancements:

- Hadoop 3 introduces several security enhancements, including improved encryption, better authentication mechanisms, support for more secure protocols, and integration with external authentication systems like Kerberos.

5. What is replication factor ? why its important
In the context of Hadoop Distributed File System (HDFS), the replication factor refers to the number of times a data block is replicated and stored across different DataNodes in the Hadoop cluster. It determines the level of data redundancy and fault tolerance in HDFS.
The replication factor is important for Data Reliability and Fault Tolerance, Data Locality and Performance, Load Balancing and Data Retrieval and Availability.

6. What if Datanode fails?
When a DataNode fails in the Hadoop Distributed File System (HDFS), the system is designed to handle the failure and ensure data availability and fault tolerance.
The fault-tolerant design of HDFS ensures that the failure of a DataNode does not result in data loss. The replication mechanism and the coordination between the NameNode and other DataNodes enable the recovery and redistribution of replicas to maintain data availability, fault tolerance, and balanced cluster operations.

7. What if Namenode fails?
When the NameNode fails in the Hadoop Distributed File System (HDFS), the entire HDFS cluster becomes unavailable for normal operations since the NameNode is responsible for managing the file system metadata. However, Hadoop is designed to handle NameNode failures through a mechanism called High Availability (HA) or NameNode Failover.
Secondary NameNode:

- In traditional Hadoop versions (prior to Hadoop 2), there was a Secondary NameNode that periodically merged the edits log with the file system image to create a new checkpoint.
- If the NameNode fails, the Secondary NameNode cannot serve as an immediate replacement. It can only help to speed up the recovery process by providing a more up-to-date checkpoint.

High Availability (HA):

- Hadoop 2 introduced High Availability to overcome the single point of failure of the NameNode. It ensures continuous availability of the file system by using multiple NameNodes in an active-passive setup.
- In an HA configuration, two NameNodes are running simultaneously: the active NameNode and the standby NameNode.
- The active NameNode handles all client requests, manages the file system metadata, and coordinates data operations.
- The standby NameNode remains synchronized with the active NameNode by continuously applying edits received from the active NameNode.
- If the active NameNode fails, the standby NameNode automatically takes over as the new active NameNode, ensuring uninterrupted access to the file system.

ZooKeeper:

- HA in Hadoop relies on ZooKeeper, a distributed coordination service, for leader election and maintaining the state of the active and standby NameNodes.
- ZooKeeper helps manage the failover process and ensures that only one NameNode is active at a time.
- ZooKeeper also coordinates the updates to the cluster-wide metadata, such as block locations, while a failover occurs.

Manual Recovery (Non-HA Setup):

- In a non-HA setup or in the absence of a standby NameNode, recovering from a NameNode failure involves manual steps.
- The failed NameNode needs to be repaired or replaced, and its metadata needs to be restored from the latest checkpoint and edits logs.
- Once the NameNode is back online, it may require additional time to process the edits logs and become fully operational. During this recovery process, the HDFS cluster remains unavailable.

8. Why is block size 128 MB ? what if I increase or decrease the block size

The default block size in Hadoop is 128 MB, but it can be adjusted based on the specific requirements and characteristics of the data and the Hadoop cluster. The block size represents the unit in which data is stored and transferred within Hadoop Distributed File System (HDFS).

If you have large files, you can get the following benefit by increasing the block size:

- Increasing Block Size: Increasing the block size can improve storage efficiency and reduce metadata overhead. However, it may result in increased latency for small files, as the entire block needs to be processed even if the file is smaller than the block size. It can also impact data locality if processing smaller subsets of data is desired.
- Decreasing Block Size: Decreasing the block size can improve data locality and reduce latency for small files. However, it may increase metadata overhead, reduce storage efficiency, and impact the performance of network transfers and data processing, as more metadata operations and network transfers are required.

9. Small file problem

The "small file problem" is a common challenge encountered when working with Hadoop and Hadoop Distributed File System (HDFS). It refers to the inefficiency and performance degradation caused by storing and processing a large number of small files in HDFS

10. What is Rack awareness?

Rack awareness is a feature in Hadoop that aims to improve data locality and network efficiency by taking into account the physical network topology of the cluster. It helps optimize data placement within the cluster by considering the proximity of DataNodes in terms of their network connectivity and reducing network traffic.

In a Hadoop cluster, racks represent physical network switches or cabinets that house multiple machines (nodes). Each rack contains multiple DataNodes. The rack awareness feature allows Hadoop to have knowledge of the rack structure and use it to make intelligent decisions about data placement.

Rack awareness is especially useful in large-scale Hadoop clusters spanning multiple racks and helps optimize data placement, reduce network traffic, and improve overall performance by considering the physical network topology of the cluster.

11. What is SPOF ? how its resolved ?
In the context of Hadoop, a Single Point of Failure (SPOF) refers to a component or a situation that, if it fails, can cause the entire Hadoop system to fail or significantly impact its operation. There are several potential SPOFs in a Hadoop cluster, and it is crucial to address them to ensure system reliability and availability. Here are some common SPOFs in Hadoop and their resolutions:

NameNode as SPOF:

- The NameNode in Hadoop's HDFS is a critical component responsible for managing the file system metadata.
- Resolution: Hadoop 2 introduced High Availability (HA) for the NameNode. It allows for the deployment of multiple NameNodes in an active-passive configuration, ensuring automatic failover in case of NameNode failure. The standby NameNode takes over as the active NameNode, providing uninterrupted access to the file system

DataNode as SPOF:

- DataNodes store the actual data blocks in HDFS. If a DataNode fails, the data blocks stored on that node become unavailable.
- Resolution: HDFS handles DataNode failures through data replication. By default, HDFS replicates each data block three times across different DataNodes. If a DataNode fails, the replicas stored on other nodes can be accessed, ensuring data availability and fault tolerance. Additionally, adding more DataNodes to the cluster increases fault tolerance and reduces the impact of individual DataNode failures.

Network Switch or Rack Failure:

- If a network switch or an entire rack fails, it can disrupt the connectivity and communication between nodes in the cluster.
- Resolution: Rack awareness and data replication in HDFS help address this issue. Hadoop takes into account the physical network topology (racks) and ensures data replication across racks. By storing replicas on different racks, Hadoop mitigates the risk of data loss due to network switch or rack failures.

ResourceManager as SPOF (in YARN):

- The ResourceManager in Hadoop's YARN is responsible for managing the cluster's resources and scheduling jobs.
- Resolution: YARN supports ResourceManager High Availability (RM HA) by deploying multiple ResourceManager nodes in an active-passive configuration. ZooKeeper is used for leader election and to maintain the state of active and standby ResourceManagers. In case of failure, the standby ResourceManager takes over to ensure uninterrupted job scheduling and resource management.

Secondary NameNode as SPOF (in traditional Hadoop versions):

- The Secondary NameNode performs periodic checkpoints to merge edits logs with the file system image but cannot serve as an immediate replacement for the NameNode in case of failure.

- Resolution: Secondary NameNode has been deprecated in Hadoop 2 in favor of NameNode HA. With HA, the standby NameNode takes over in case of NameNode failure, providing a more reliable failover mechanism.

12. Explain zookeeper?

In Hadoop, Apache ZooKeeper plays a crucial role in providing coordination and synchronization services for distributed components, ensuring their reliable and consistent operation.

ZooKeeper's strong consistency guarantees, reliable leader election, and distributed coordination capabilities make it an essential component in Hadoop. It ensures the availability, fault tolerance, and proper functioning of critical Hadoop services, such as the NameNode, ResourceManager, HBase, and Kafka. By providing a unified coordination framework, ZooKeeper simplifies the development and management of distributed systems in the Hadoop ecosystem.

13. Difference between -put and -CopyFromLocal?

In the context of Hadoop's command-line interface, both **`-put`** and **`-copyFromLocal`** are used to copy files from the local file system to the Hadoop Distributed File System (HDFS)

`-put`:

- The **`-put`** command is used to copy files or directories from the local file system to HDFS.
- It supports recursive copying, meaning it can copy entire directories and their contents.
- If the destination path in HDFS already exists, the source files are copied into that directory.
- If the destination path does not exist, a new directory is created with the specified path, and the source files are copied into it.
- The command syntax for **`-put`** is: **`hadoop fs -put <source> <destination>`**.

`-copyFromLocal`:

- The **`-copyFromLocal`** command is also used to copy files or directories from the local file system to HDFS.
- It behaves similarly to **`-put`**, allowing recursive copying and creating new directories if necessary.
- The key difference is that **`-copyFromLocal`** explicitly specifies the source as being from the local file system.
- If the destination path in HDFS already exists, the source files are copied into that directory.
- If the destination path does not exist, a new directory is created with the specified path, and the source files are copied into it.

- The command syntax for `-copyFromLocal` is: **`hadoop fs -copyFromLocal <source> <destination>`**.

14. What is erasure coding?

Erasure coding is a data protection technique used in distributed storage systems to enhance data durability and fault tolerance while optimizing storage efficiency. It involves breaking data into small fragments, calculating additional parity fragments, and distributing them across different storage nodes in a cluster.

However, it's important to note that erasure coding introduces additional computational overhead during encoding and decoding operations. The trade-off between storage efficiency, fault tolerance, and computational requirements needs to be carefully considered based on the specific needs and resources of the storage system.

15. What is speculative execution?

Speculative execution is a feature in Hadoop that aims to mitigate the impact of slow-running tasks and improve overall job completion time in a distributed computing environment. It is designed to handle situations where certain tasks take longer to complete due to factors such as hardware issues, network congestion, or data skew.

16. Explain Yarn Architecture

YARN (Yet Another Resource Negotiator) is the resource management layer in the Hadoop ecosystem. It separates the responsibilities of cluster resource management and job scheduling from the Hadoop Distributed File System (HDFS), enabling a more flexible and scalable framework for distributed data processing. YARN's architecture consists of the following components:

ResourceManager (RM):

- The ResourceManager is the central authority responsible for overall resource management and job scheduling in the YARN cluster.

NodeManager (NM):

- Each node in the cluster runs a NodeManager, which is responsible for managing resources and executing tasks on that node.

ApplicationMaster (AM):

- The ApplicationMaster is responsible for managing the execution of a specific application or job in the YARN cluster.

Container:

- A container represents a set of allocated resources (CPU, memory) on a specific node in the cluster.

17. How does ApplicationManager and Application Master differ

the ApplicationMaster is a component within an application's execution that interacts with the ResourceManager(ApplicationManager) to acquire resources and manage the execution of tasks. On the other hand, the ResourceManager is responsible for overall resource management and scheduling in the YARN cluster, including managing multiple ApplicationMasters and allocating resources to them.

18. Explain Mapreduce working?

MapReduce is a programming model and computational paradigm designed for processing and analyzing large-scale distributed data sets in a parallel and fault-tolerant manner. It simplifies the development of distributed data processing applications in systems by dividing a large-scale computation into smaller, independent tasks that can be executed in parallel across a cluster of machines. By leveraging parallelism and fault tolerance, MapReduce enables the processing of massive data sets efficiently and reliably

19. How many mappers are created for 1 GB file?

The number of mappers created for a 1 GB file in Hadoop depends on several factors, including the block size, the size of the input file, and the Hadoop configuration settings. By default, Hadoop uses a block size of 128 MB. Here's how the number of mappers is determined:

- **Block Size**: The input file is divided into blocks, with each block typically having a size equal to the block size specified in the Hadoop configuration. In this case, with a default block size of 128 MB, the 1 GB file would be divided into approximately 8 blocks.
- **Mapper per Block:** By default, Hadoop assigns one mapper per block. So, with 8 blocks, there would be 8 mappers created. Each mapper processes a single block of data independently.

20. How many reducers are created for 1 GB file?

The number of reducers created for a Hadoop job is determined by the configuration settings and the desired output requirements rather than the size of the input file. By default, the number of reducers is set to one. However, the number of reducers can be adjusted based on factors such as the processing requirements, available resources, and desired parallelism.

21. What is combiner?

In the context of MapReduce, a combiner is a feature that allows for local aggregation of intermediate key-value pairs on the map side before sending them to the reduce phase. It is an optional component that can be used to optimize the overall performance of a MapReduce job by reducing the amount of data shuffled between the map and reduce tasks.

The combiner function is similar to the reducer function used in the reduce phase, but it operates only at the map task level. It takes the output of the map function, which consists of key-value pairs, and performs a local aggregation of values with the same key.

22. What is partitioner?

In the context of MapReduce, a partitioner is a component that determines which reducer receives which intermediate key-value pairs produced by the map tasks. It divides the intermediate data based on the keys and distributes them to the reducers for further processing.

The primary purpose of a partitioner is to enable parallelism and load balancing during the reduce phase of a MapReduce job. By distributing the intermediate data across multiple reducers, the partitioner ensures that the workload is evenly distributed, allowing for efficient utilization of cluster resources and faster processing.