**Hadoop Interview Questions**

1. **Mention Hadoop distribution**

Cloudera CDH, Hortonworks Data Platform (HDP), MapR, Amazon EMR and Microsoft Azure HDInsigh

2**. Explain Hadoop Architecture**

Hadoop's architecture centers around the Hadoop Distributed File System (HDFS) and the MapReduce programming model. HDFS works by splitings large data into blocks and distributes them across a cluster for redundancy. Whilst MapReduce enables parallel processing. Key components include Resource Manager, Node Manager, and YARN for resource management. This architecture's distributed and fault-tolerant design makes Hadoop ideal for scalable and cost-effective big data processing and storage.

**3. Configuration files used during hadoop installation**

There are several configuration files that are crucial for setting up and management of the distributed data processing framework. These files include core-site.xml, hdfs-site.xml, mapred-site.xml (or yarn-site.xml in Hadoop 2+), and hdfs-site.xml.

* Core-site.xml defines core Hadoop properties like the Hadoop file system's address and port.
* Hdfs-site.xml is responsible for configuring Hadoop Distributed File System (HDFS) properties, such as block size and replication factors.
* Mapred-site.xml (or yarn-site.xml) is used to configure the MapReduce framework's settings.
* Hadoop-env.sh allows you to set environment variables for Hadoop, while yarn-env.sh is for YARN resource manager-specific variables.

These configuration files are essential for tailoring Hadoop's behaviour to suit your specific cluster and application requirements, making them a critical part of a successful Hadoop installation and operation.

**4. Difference between Hadoop fs and hdfs dfs**

Hadoop fs and hdfs dfs are command-line interfaces used for interacting with the Hadoop Distributed File System (HDFS). As hadoop fs can be seen as the original command, while hdfs dfs is seen as more user-friendly and robust replacement.

The main difference lies within their functionality and user experience.

* Hadoop fs primarily works with a URI-based file system, making it somewhat less intuitive for users unfamiliar with Hadoop's intricacies.
* hdfs dfs offers a simplified and more user-friendly interface that abstracts away the complexities of URIs and provides a consistent experience across HDFS commands.
* Additionally, hdfs dfs incorporates features like auto-completion, improved error handling, and more intuitive options, making it the preferred choice for managing files and directories within HDFS for most users. While Hadoop fs is still functional, hdfs dfs is generally recommended for its improved usability and features.

**5. Difference between Hadoop 2 and Hadoop 3**

Hadoop 3 represents the evolution of the Hadoop ecosystem compared to Hadoop 2. One of the notable differences is the enhanced support for erasure coding, this reduces the storage overhead in HDFS (Hadoop Distributed File System) while maintaining data fault tolerance.

Hadoop 3 also offers improved performance and resource utilization through features like enabling administrators to use different storage types for different data. Furthermore, Hadoop 3 has a more efficient and scalable resource management system with the introduction of the YARN (Yet Another Resource Negotiator) federation, which allows for dynamic cluster resizing and resource allocation.

Overall, Hadoop 3 has improved security with support for opportunistic container execution and GPU isolation. It's a more mature and stable version, addressing several limitations of Hadoop 2, making it a better choice for large-scale data processing and analytics workloads.

**6. What is replication factor ? why its important**

Replication Factor in Hadoop helps Determine the number of copies (replicas) of data blocks in Hadoop's Distributed File System (HDFS).

it is important for the following reasons: Importance:

• **Fault Tolerance:** Ensures data availability in case of node failures.

• **Data Locality:** Improves performance by storing data on nodes where it's processed.

• **Load Balancing:** Distributes data access and computation evenly.

• **Parallelism**: Enables concurrent processing of data.

• **Data Recovery:** Facilitates automatic recovery from failures or data corruption.

The choice of replication factor balances fault tolerance with storage efficiency in Hadoop clusters.

**7. What if Datanode fails?**

When a Datanode fails in the Hadoop distributed file system (HDFS), the system handles the situation gracefully to ensure data reliability and availability. Here's what happens:

* Replication: HDFS maintains multiple copies (typically three) of each data block across different Datanodes. If one Datanode fails, there are still copies available on other nodes.
* Replication Maintenance: The HDFS NameNode continuously monitors the health of Datanodes. If it detects a Datanode failure, it triggers replication of the lost blocks to other healthy Datanodes.
* Decommissioning: In the event of a planned Datanode decommission or maintenance, HDFS allows for the safe removal of a Datanode without data loss by redistributing its blocks to other nodes before the decommissioning.
* Block Recovery: HDFS also features block recovery mechanisms to recover lost data blocks in the rare case of a node failure before replication is completed.

In summary, Hadoop's data replication and fault tolerance mechanisms ensure that data remains available even when a Datanode fails, safeguarding against data loss and downtime.

**8.     What if Namenode fails?**

If the Namenode in a Hadoop cluster fails, it can lead to data loss and unavailability. To address this issue, Hadoop employs several mechanisms:

* Secondary Namenode: Though often misunderstood as a backup, it periodically checkpoints the namespace and helps in faster recovery.
* Hadoop High Availability (HA): Hadoop 2.x onwards supports HA, using multiple active Namenodes. This ensures continuous availability in case of a failure. They use a shared storage system, typically a Quorum Journal Manager (QJM), to maintain consistency.
* Backup and Restore: Regular backups of the Namenode's metadata can be created to recover the filesystem. However, this approach may result in some data loss between backups.
* Data Replication: Hadoop replicates data across DataNodes by default. If a Namenode fails, data is still available on DataNodes.
* Manual Recovery: In worst-case scenarios, you may have to manually recover the Namenode by restoring from a backup or using the Secondary Namenode's checkpoint.

These mechanisms collectively help mitigate the risk of data loss and downtime when the Namenode fails in a Hadoop cluster.

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

* The default Hadoop block size is 128 MB, it is chosen based on a trade-off between multiple factors (like data locality, seek time, and parallelism).
* Data Locality: Larger block sizes improve data locality, reducing network transfer overhead as more data processing can be done where the data resides.
* Seek Time: Smaller blocks mean faster data access, but it increases metadata overhead
* Parallelism: Larger blocks enable efficient parallel processing, but very large blocks may lead to inefficient resource utilization.

- Pros of Increasing block size:

* + Improved data locality and reduced network overhead for large-scale data processing.

  - Cons of Increasing block size:

* + May increase the NameNode's metadata management burden.
  + Smaller files won't utilize the block efficiently, potentially leading to wasted storage.

-pros of you decreasing block size:

* + Faster data access for smaller files and diverse workloads.

  - Cons of you decreasing block size:

* + Reduced data locality and increased network transfer overhead.
  + The NameNode may face increased metadata management, impacting performance with many small blocks.

The choice of block size should consider your specific workload and infrastructure, aiming to balance data locality, seek time, and parallelism.

**10. Small file problem**

The "Small File Problem" in Hadoop refers to the challenge posed by the presence of a large number of small files in the Hadoop Distributed File System (HDFS). This issue can impact Hadoop clusters in several ways:

**NameNode Overhead**: Each file and directory in HDFS is represented as an inode in the NameNode memory. Handling a vast number of small files increases memory overhead, potentially leading to performance degradation and scalability issues.

**Inefficient Data Processing**: Small files lead to inefficient data processing, as Hadoop processes data in blocks. If the data is spread across numerous small files, it results in excessive I/O operations and reduced data locality, slowing down processing.

**Job Performance**: MapReduce jobs in Hadoop, for example, might spend more time opening and closing small files rather than processing data, causing longer job execution times.

To mitigate the Small File Problem, best practices include using sequence files or combining small files into larger ones, which reduces NameNode memory overhead and improves overall Hadoop cluster performance.

**11. What is Rack awareness?**

Rack awareness in Hadoop is a critical concept that improves data reliability and network efficiency in a distributed computing environment. It involves the following key points:

* Hadoop clusters consist of numerous machines organized into racks, where each rack contains multiple data nodes.
* Rack awareness ensures that Hadoop's Hadoop Distributed File System (HDFS) and data processing components are aware of the physical network topology, including the location of data nodes in racks.
* The primary goal of rack awareness is to minimize data transfer across network switches and racks, as cross-rack data transmission can be slower and less reliable.
* HDFS and data processing frameworks like MapReduce take rack information into account when making data placement decisions. Data is preferably stored on nodes within the same rack, improving data locality and reducing network traffic.
* In cases where no local rack storage is available, Hadoop will prioritize placing data on nodes in the same data center before considering nodes in other data centers or remote racks.

Rack awareness is a fundamental optimization in Hadoop clusters, contributing to data resilience, fault tolerance, and efficient data processing.

**12. What is SPOF ? how its resolved ?**

SPOF stands for "Single Point of Failure" in the context of Hadoop. It refers to a critical component or node in the Hadoop cluster that, if it fails, can disrupt the entire system's operation.SPOFs can include the NameNode in HDFS, ResourceManager in YARN, or other central services.

Resolving SPOFs in Hadoop can be achieved through various strategies, including:

* Implementing High Availability (HA) for services like HDFS and YARN by introducing standby nodes.
* Using data replication in HDFS to ensure data redundancy.
* Employing fault-tolerant hardware and software solutions.

These measures aim to eliminate or mitigate SPOFs and enhance the reliability and availability of the Hadoop cluster.

**13. Explain zookeeper?**

ZooKeeper is a crucial component within the Hadoop ecosystem:

* It's a distributed coordination service that ensures synchronization and management of distributed systems.
* ZooKeeper provides a centralized, reliable platform for maintaining configuration information, naming, distributed synchronization, and group services.
* It plays a vital role in Hadoop by helping to coordinate tasks among various nodes in the Hadoop cluster.
* Hadoop components like HDFS, HBase, and YARN use ZooKeeper to maintain metadata, coordinate leader elections, and manage distributed states.
* ZooKeeper's robustness and consistency make it essential for maintaining the integrity and reliability of Hadoop clusters, ensuring they function smoothly even in the presence of failures.

**14. Difference between -put and -CopyFromLocal?**

* Both `-put` and `-copyFromLocal` are Hadoop commands used to transfer files from the local file system to the Hadoop Distributed File System (HDFS).
* The key difference lies in their usage and the options they offer:

  `-put` is more versatile and allows you to specify source and destination directories. You can use it to upload a single file or multiple files and directories at once.

  - `-copyFromLocal` is simpler and typically used to upload a single file from the local system to HDFS, as it assumes the destination path is in HDFS. It doesn't support multiple files or directories.

* Both commands serve the purpose of moving local files to HDFS, but `-put` provides more flexibility, while `-copyFromLocal` is more straightforward for single file uploads.

**15.  What is erasure coding?**

Erasure coding is a data storage and fault-tolerance technique used in Hadoop and other distributed file systems. It differs from traditional replication in the following ways:

* Reduces Data Duplication: Erasure coding reduces storage overhead compared to traditional replication by breaking data into smaller fragments and generating error-correcting fragments, allowing data to be rebuilt if some fragments are lost.
* Efficient Storage: Erasure coding can offer significant storage savings, making it more cost-effective, especially in large-scale storage environments.
* Improved Fault Tolerance: It provides high fault tolerance without the need for as many replicas, which is important for big data systems like Hadoop where large datasets are common.
* Trade-offs: Erasure coding may have slightly higher computational overhead during data recovery and may not be as suitable for workloads with many small, random read operations.
* Optimized for Different Use Cases: Hadoop allows you to choose between replication and erasure coding, so you can optimize storage and fault tolerance based on your specific use case and hardware resources.

**16.    What is speculative execution?**

Speculative execution in Hadoop is a technique used to enhance job execution reliability and efficiency. Here's a concise explanation:

* It's a feature in Hadoop's MapReduce framework.
* When a task in a job is running significantly slower than other similar tasks, Hadoop launches a speculative (backup) task.
* The backup task runs in parallel with the slow task, aiming to finish faster.
* Whichever task finishes first is used, and the other is killed.
* This minimizes the impact of stragglers (slow tasks) on job completion time.
* Speculative execution ensures that jobs progress at the speed of the fastest tasks, improving overall job runtime.

**17.   Explain Yarn Architecture**

YARN (Yet Another Resource Negotiator) is a key component in Hadoop's architecture designed for resource management and job scheduling. Its architecture can be summarized as follows:

1. **Resource Manager (RM):**

- A global component that manages and allocates cluster resources.

- It has two main components: the Scheduler, which allocates resources, and the ApplicationManager, which manages application lifecycles.

2. **NodeManager (NM**):

- Runs on each node in the Hadoop cluster and is responsible for managing resources on that node.

- Monitors resource usage and reports it to the ResourceManager.

3. **ApplicationMaster (AM):**

- A per-application framework that negotiates resources from the ResourceManager.

- Manages the application's execution and monitors its progress.

4. **Container**:

- A fundamental unit of resource allocation in YARN, representing CPU and memory resources on a node.

- An ApplicationMaster negotiates containers to run its tasks.

YARN provides flexibility and scalability, making it suitable for a wide range of workloads beyond just MapReduce, such as Apache Spark, Tez, and more. Its architecture separates resource management from job execution, enabling efficient and multi-tenant use of Hadoop clusters.