Hadoop Interview Questions:

1. Explain Hadoop Architecture

Hadoop architecture consists of the following components: HDFS for distributed file storage, YARN for resource management, and MapReduce for distributed data processing. Hadoop Common provides shared utilities, and the ecosystem includes additional tools like Hive, Pig, Spark, and HBase for higher-level data processing and analytics. Hadoop enables scalable and fault-tolerant processing of big data in a distributed computing environment.

1. Configuration files used during Hadoop installation

During Hadoop installation, important configuration files are used to customise various aspects of the framework. Some common files include: “core-site.xml” for core Hadoop settings, “hdfs-site.xml” for HDFS-specific configurations, “yarn-site.xml” for YARN settings, “mapred-site.xml” for MapReduce configurations and “hadoop-env.sh” for environment variables. Additionally, “workers” or “slaves” lists slave node hosts, and “masters” specifies the master node host. These files allow administrators to tailor Hadoop settings for their specific cluster setup.

1. Difference between Hadoop fs and hdfs dfs

“hadoop fs” and “hdfs dfs” are command-line utilities used to interact with the Hadoop Distributed File System (HDFS). Both commands provide similar functionality for performing operations like file listing, copying, deletion, creation, permission changes, etc. The main difference is that “hadoop fs” is a general-purpose command that supports multiple file systems, whereas “hdfs dfs” is specifically designed for working exclusively with HDFS.

1. Difference between Hadoop 2 and Hadoop 3

Hadoop 3 introduces improvements in YARN resource management, scalability, and performance. It enhances HDFS with features like erasure coding and multiple standby NameNodes. Hadoop 3 also embraces containerisation and supports platforms like Kubernetes. It maintains compatibility with Hadoop 2 and benefits from an expanded ecosystem of tools and frameworks.

1. What is replication factor ? why it’s important

The replication factor in Hadoop determines the number of copies of each data block stored across different nodes in the cluster. It is important for many reasons as follows:

1. Fault Tolerance: Replication provides data redundancy, ensuring that data remains available even if some nodes fail.

2. Data Reliability: Multiple replicas increase data reliability by reducing the risk of data loss.

3. Performance: Replication improves read performance by allowing data to be accessed from multiple nodes in parallel.

4. Load Balancing: Replication distributes the data load across nodes, improving overall cluster performance.

5. Data Access Parallelism: Replication enables parallel processing of data by allowing multiple nodes to access replicas simultaneously.

Choosing an optimal replication factor involves considering factors like fault tolerance requirements, storage capacity, and network bandwidth.

1. What if DataNode fails?

If a DataNode fails in a Hadoop cluster, the cluster automatically replicates the data stored on the failed node to maintain the desired replication factor. The NameNode identifies the failed DataNode and triggers the replication process on other nodes to create additional replicas. This ensures data availability and fault tolerance in case of DataNode failures.

1. What if NameNode fails?

In Hadoop 3, if the active NameNode fails, the standby NameNode takes over its responsibilities in a High Availability (HA) setup. This ensures minimal downtime and uninterrupted cluster operation. Manual intervention and recovery are still required, and backup procedures are important for data restoration. Following HA best practices and disaster recovery plans help mitigate the impact of NameNode failures in Hadoop 3.

1. 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 specific requirements. Increasing the block size can improve performance for large files, while decreasing the block size can be beneficial for smaller files and frequent data access. However, changing the block size should consider factors such as storage efficiency, memory requirements, and data locality.

As aforementioned, changing the block size in Hadoop can impact performance, storage efficiency, and data locality. Increasing the block size can improve performance for large files but may lead to inefficiencies for smaller files. Decreasing the block size can enhance storage efficiency for small files but might increase metadata overhead.

1. Small file problem

The "small file problem" refers to the challenge of efficiently managing and processing a large number of small files in distributed file systems like Hadoop. It can result in increased metadata overhead, reduced NameNode performance, inefficient storage utilisation, and compromised data locality. Techniques such as file concatenation, archiving, and custom input formats can help mitigate the small file problem.

1. What is Rack awareness?

Rack awareness is a feature in Hadoop that considers the network topology of the cluster, specifically the physical racks, when making data placement and task scheduling decisions. It improves data locality, enhances fault tolerance by distributing replicas across racks, and optimises network traffic within the cluster.

1. What is SPOF ? how its resolved ?

A SPOF (Single Point of Failure) is a component or resource in a system that, if it fails, can cause the entire system to fail. To resolve a SPOF, redundancy can be introduced, load balancing techniques can be employed, failover mechanisms can be implemented, robust monitoring and alerting systems can be used, high availability architectures can be designed, and comprehensive disaster recovery plans can be put in place. These measures help minimise the impact of failures and ensure system reliability and availability.

1. Explain zookeeper?

ZooKeeper is a distributed coordination service that helps manage configuration information, provides distributed synchronisation, and offers group services in a distributed system. It allows processes to share and synchronise data, provides distributed locks, sequential node ordering, reliable notifications, and leader election. ZooKeeper ensures high availability and fault tolerance through a replicated cluster of servers. It is commonly used in distributed systems like Hadoop, Kafka, and HBase for coordination and consistency.

1. Difference between -put and -CopyFromLocal?

The “-put” command is used to copy files or directories from the local file system to HDFS, while the “-copyFromLocal” command is specifically used to copy individual files from the local file system to HDFS.

1. What is erasure coding?

Erasure coding is a technique used in data storage systems to provide fault tolerance and efficient storage utilisation. It divides data into smaller fragments called shards and generates additional parity shards for redundancy. It offers improved fault tolerance, reduced storage overhead, and cost savings compared to traditional replication methods. However, it requires additional processing overhead and higher network bandwidth for data reconstruction.

1. What is speculative execution?

Speculative execution is a technique used in distributed computing to improve job completion time and resource utilisation. It involves running duplicate tasks for slow-running tasks and considering the one that completes first as the successful output. It reduces job completion time, maximises resource efficiency, and provides fault tolerance.

1. Explain Yarn Architecture

YARN (Yet Another Resource Negotiator) is the cluster management technology in Hadoop that separates resource management and job execution. It consists of the ResourceManager, NodeManager, and ApplicationMaster components. The ResourceManager allocates resources and schedules jobs, the NodeManager manages resources on individual nodes, and the ApplicationMaster manages the execution of specific applications. YARN enables multi-tenancy, dynamic resource allocation, and efficient utilisation of cluster resources.

1. How does ApplicationManager and Application Master  differ

The ResourceManager manages resources and scheduling for the entire cluster, while the ApplicationMaster manages the execution of a specific application or job within the cluster.

1. Explain MapReduce working?

MapReduce is a programming model and processing framework used in Hadoop for large-scale data processing. It works by dividing input data into smaller chunks and applying a Map function to each chunk independently. The intermediate results are then sorted, grouped, and passed to Reduce functions for final processing. MapReduce enables parallel processing and efficient handling of big data tasks across a distributed cluster.

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

The number of mappers created for a 1 GB file in Hadoop depends on the block size configured for the Hadoop cluster. By default, the block size is set to 128 MB in Hadoop.

In Hadoop, a file is divided into blocks, and each block is processed by a separate mapper. So, the number of mappers created for a file is determined by the file size divided by the block size.

For example, if the file size is 1 GB (1024 MB) and the block size is 128 MB, the number of mappers created would be 1024 MB / 128 MB = 8 mappers.

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

By default, if not explicitly specified, Hadoop assigns a single reducer for a job processing a 1 GB file. However, the number of reducers can be explicitly set by the user or configured in the job settings. The recommended number of reducers depends on various factors, such as the size and nature of the data, available resources, and the desired level of parallelism.

1. What is combiner?

A combiner is a function in Hadoop that performs a local aggregation of intermediate data on the mapper nodes before sending it to the reducer. It helps reduce data transfer, optimise network bandwidth, and lighten the workload on the reducer.

1. What is partitioner?

A partitioner in Hadoop's MapReduce framework determines which reducer receives each intermediate key-value pair generated by the mappers. It ensures even distribution of data among reducers for efficient processing and load balancing.