Hadoop

1. Explain Hadoop Architecture

The Hadoop Architecture Mainly consists of 4 components. i.e, HDFS (Hadoop Distributed File System), YARN (Yet Another Resource Negotiator), MapReduce, Common Utilities or Hadoop Common.

HDFS is utilized for storage permission. It is mainly designed for working on a distributed file system design. HDFS in Hadoop provides Fault-tolerance and High availability to the storage layer and the other devices present in that Hadoop cluster. Data storage Nodes in HDFS ie, NameNode and DataNode. YARN is used for resource management. 2 operations that are Job scheduling and Resource Management. The Purpose of Job schedular is to divide a big task into small jobs so that each job can be assigned to various slaves in a Hadoop cluster and Processing can be maximized. Resource Manager is to manage all the resources that are made available for running a Hadoop cluster. MapReduce for distributed data processing. MapReduce is a framework conducting distributed and parallel processing of large volumes of data. Written using a number of programming languages, it has two main phases: Map Phase and Reduce Phase. Hadoop Common provides shared utilities, and the ecosystem includes additional tools like Hive, Pig, Spark, and HBase for higher-level data processing and analytics.

1. Configuration files used during hadoop installation

In the hadoop installation mainly used configuration files are, **hadoop-env.sh**: environment variable that are used in the scripts to run hadoop. **core-site.xml** : for core hadoop settings, It common for hdfs and mapreduce. **hdfs-site.xml:** It is one of the important configuration files which is required for runtime environment settings of a Hadoop. It contains the configuration settings for NAMENODE, DATANODE, and SECONDARYNODE.**mapred-site.xml**: It contain configuration settings for MapReduce. yarn-site.xml: It contain configuration settings of ResourceManager and NodeManager .**masters and slaves**: masters is used to determine the master Nodes. It will inform about the location of SECONDARY NAMENODE to Hadoop Daemon. It is used to determine the slave Nodes in Hadoop cluster.The Slave file at Master Node contains a list of hosts, one per line.The Slave file at Slave server contains IP address of Slave nodes.

1. Difference between Hadoop fs and hdfs dfs

The 'hdfs dfs' command is used very specifically for hadoop filesystem (hdfs) data operations while 'hadoop fs' covers a larger variety of data present on external platforms as well. It refers to any file system, it could be local or HDFS

1. Difference between Hadoop 2 and Hadoop 3

Hadoop 2 uses an old timeline service which has scalability issues.  Hadoop 3 improves the timeline service v2 and improves the scalability and reliability of timeline service. In Hadoop 2, there is only one standby NameNode.  Hadoop 3 supports multiple standby NameNodes. If one standby node goes down over the weekend, you have the benefit of other standby NameNodes so the cluster can continue to operate. Hadoop 2 usescontainers, Hadoop 3 containerization brings agility and package isolation story of Docker. Hadoop 2 has a lot more storage overheadthan Hadoop 3.

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

Hadoop Replication Factor is the number of copies of each block that Hadoop creates and stores across the nodes in a cluster. The default replication factor in Hadoop is 3, which means that each block is replicated thrice across the nodes in the cluster. Hadoop Replication Factor is important for two reasons: data availability and fault tolerance. In Data availability, Hadoop Replication Factor ensures that the data is available even if some of the nodes in the cluster go down or become unavailable. If we have a replication factor of 3, then even if two nodes go down, we can still access the data from the third replica. In Fault tolerance, Hadoop Replication Factor ensures that the data is fault-tolerant. If a node goes down or becomes unavailable, then Hadoop can still access the data from the other replicas of the same block.

1. What if Datanode fails?

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?

When the [NameNode](https://cwiki.apache.org/confluence/display/HADOOP2/NameNode) goes down, the file system goes offline. There is an optional [SecondaryNameNode](https://cwiki.apache.org/confluence/display/HADOOP2/NameNode) that can be hosted on a separate machine. It only creates checkpoints of the namespace by merging the edits file into the fsimage file and does not provide any real redundancy

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

The default block size of 128 MB is a good starting point. However, you can change the block size based on the size of your data and the amount of memory available on your machine. If your data is small, you can reduce the block size to improve the performance of your Hadoop jobs. On the other hand, if your data is large, you can increase the block size to reduce the number of blocks and improve the performance of your Hadoop jobs.

1. Small file problem

HDFS is designed to handle large data sets, and the data included is distributed over various machines to help with parallel processing. As the metadata and data are kept in separate components, every created file occupies a minimum amount of memory unit regardless of the size. Small files are typically read as less than 1 HDFS block or 128 MB. However, even with less than 1 KB in size, Files put a massive load on the NameNode and take a metadata storage space equivalent to 128 MB. Practically, smaller file sizes also indicate smaller clusters as there are definite limits on the number of files a single NameNode can manage.

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 it’s resolved?

An SPOF is a single part of a system, which, if it fails, takes down the entire system. To resolve this issue using the mechanisms like **backup and redundant systems, load balancers, backup power, up to date the data security infrastructure.**

1. Explain zookeeper?

Zookeeper is a coordination service for distributed application that enables synchronization across a cluster. Zookeeper in Hadoop can be viewed as centralized repository where distributed applications can put data and get data out of it. It is used to keep the distributed system functioning together as a single unit, using its synchronization, serialization and coordination goals. 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?

HDFS erasure coding is an alternative storage strategy to traditional HDFS three-way block replication. The key advantage of using erasure coding is that your files will consume less space. Erasure Coding in Hadoop 3 is the solution to the expensive 3x default replication. In storage systems, the Redundant Array of Inexpensive Disks (RAID) uses Erasure Coding. RAID implements Erasure Coding by striping that is, dividing logically sequential data such as file into smaller units (bit, byte, or block) and storing consecutive units on different disks. Using Erasure Coding in HDFS improves storage efficiency while providing the same level of fault tolerance and data durability as traditional replication-based HDFS deployment.

1. What is speculative execution?

**Speculative execution** is used when a node is assigned a task and the node appears to be executing the task slower than expected, in which case, the master node assigns another instance of the same task to another executor node. Now, whichever node executes the task first, the output is taken from that node and the concurrent task being executed on the pending node is terminated or killed. It reduces job completion time, maximises resource efficiency, and provides fault tolerance.

1. Explain Yarn Architecture

The Yarn (Yet another Resource Negotiator) architecture consists of multiple components such as Resource Management, Node Management, Containers, and Application Master. These components work together to implement the YARN architecture. Resource Manager and Node Manager are responsible for scheduling and managing Hadoop jobs on the cluster. The Node Manager is responsible for allocating tasks to data nodes in the cluster. The Node Manager will decide which nodes will run the tasks, based on the hardware specifications of each node. The Node Manager will receive task requests from the Resource Manager. The Node Manager will then distribute the tasks to the nodes across the cluster. The ApplicationMaster manages the execution of specific applications. With the scalability of the resource manager of the YARN architecture, Hadoop may manage thousands of nodes and clusters.

1. How does ApplicationManager and Application Master  differ

Application manager is responsible for maintaining a list of submitted application. After application is submitted by the client, application manager firstly validates whether application requirement of resources for its application master can be satisfied or not. If enough resources are available then it forwards the application to scheduler otherwise application will be rejected. It also make sure that no other application is submitted with same application id.

The Application Master is responsible for the execution of a single application. It asks for containers from the Resource Scheduler (Resource Manager) and executes specific programs on the obtained containers. The Application Master knows the application logic and thus it is framework-specific. The MapReduce framework provides its own implementation of an Application Master.

1. Explain Mapreduce working?

MapReduce breaks input data into fragments and distributes them across different machines. The input fragments consist of key-value pairs. Parallel map tasks process the chunked data on machines in a cluster. The mapping output then serves as input for the reduce stage. The reduce task combines the result into a particular key-value pair output and writes the data to HDFS. The Hadoop Distributed File System usually runs on the same set of machines as the MapReduce software. When the framework executes a job on the nodes that also store the data, the time to complete the tasks is reduced significantly.

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

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.

Total size of file = 1GB (1024MB)

Input default split size = 128MB

No of Mappers = 8 (1024 / 128)

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

There is no fixed number of reducers task that can be configured or calculated. It depends on the moment how much of the resources are actually available to allocate. 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?

[Hadoop](https://data-flair.training/blogs/hadoop-tutorial-for-beginners/)**Combiner** is also known as **Mini-Reducer** that summarizes the Mapper output record with the same Key before passing to the Reducer. The main function of a Combiner is to summarize the map output records with the same key. The output (key-value collection) of the combiner will be sent over the network to the actual Reducer task as input.

1. What is partitioner?

A partitioner partitions the key-value pairs of intermediate Map-outputs. The partition works on the mapper output depending on the key value. The same key value goes into the same partition within each mapper. After this process, the final partition is sent to a reducer. The number of partitioners is equal to the number of reducers. That means a partitioner will divide the data according to the number of reducers. Therefore, the data passed from a single partitioner is processed by a single Reducer.