Hadoop Architecture and HDFS Homework

Q1. What are HDFS and YARN?
Answer:

Hadoop Distributed File System

HDFS is a Java-based file system that provides scalable and reliable data storage, and it was designed to span large clusters of commodity servers. HDFS has demonstrated production scalability of up to 200 PB of storage and a single cluster of 4500 servers, supporting close to a billion files and blocks.

YARN (Yet Another Resource Negotiator)

YARN is essentially a system for managing distributed applications. It consists of a central Resource manager (RM), which arbitrates all available cluster resources, and a per-node Node Manager (NM), which takes direction from the Resource manager. The Node manager is responsible for managing available resources on a single node.

Q2. What are the various Hadoop daemons and their roles in a Hadoop cluster?

Answer:

Various Hadoop Daemons:

- NameNode
- DataNode
- Secondary Name Node
- Resource Manager
- Node Manager

Roles of the Hadoop Daemons:

The namenode daemon is a master daemon and is responsible for storing all the location information of the files present in HDFS. The actual data is never stored on a namenode.

The datanode daemon acts as a slave node and is responsible for storing the actual files in HDFS.

Each Slave Node in a Hadoop cluster has a single NodeManager Daemon running in it.

Slave Node also sends monitoring information to the resource Manager.

Q3. Why does one remove or add nodes in a Hadoop cluster frequently?

Answer:

In Hadoop, we use commodity hardware for large clusters. The downsides to using commodity hardware are frequent failures and data node crashes. This is why data is replicated to increase the fault tolerance of the system.Hadoop provides us with the advantage of scaling up the data nodes for high volume storage. This is why nodes are constantly added and removed in Hadoop clusters. It entirely depends on the volume and the health of nodes.

Q4. What happens when two clients try to access the same file in the HDFS?

Answer:

Multiple clients can't write into HDFS file at the similar time. When a client is granted a permission to write data on data node block, the block gets locked till the completion of a write operation. If some another client request to write on the same block of the same file then it is not permitted to do so.

Q5. How does NameNode tackle DataNode failures?

Answer:

Data blocks on the failed Datanode are replicated on other Datanodes based on the specified replication factor in hdfs-site.

xml file. Once the failed datanodes comes back the Name node will manage the replication factor again.

This is how Namenode handles the failure of data node.

Q6. What will you do when NameNode is down?

Answer: When the NameNode goes down, the file system goes offline. There is an optional SecondaryNameNode 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.

Q7. How is HDFS fault tolerant?

Answer:

The HDFS is highly fault-tolerant that if any machine fails,

the other machine containing the copy of that data automatically become active. Distributed data storage -

This is one of the most important features of HDFS that makes Hadoop very powerful.

Here, data is divided into multiple blocks and stored into nodes

Q8. Why do we use HDFS for applications having large data sets and not when there are a lot of small files?

Answer:

HDFS is more efficient for a large number of data sets,

maintained in a single file as compared to the small chunks of data stored in multiple files.

211720243010

Q9. How do you define “block” in HDFS? What is the default block size in Hadoop 1 and in Hadoop 2? Can it be changed?

Answer:

a)Blocks are the smallest continuous location on your hard drive where data is stored.

HDFS stores each file as blocks, and distribute it across the Hadoop cluster.

b)The default size of a block in HDFS is 128 MB (Hadoop 2.x) and 64 MB (Hadoop 1.x)

c)Yes we can change the Hadoop block size.