Solution for Assignment 2 - Day 2 (Tuesday) – HDFS & YARN

---

1) For NameNode, why it's not necessary to store block locations persistently?
ANS:
Because this information is reconstructed from the data nodes when the system starts.

2) Why is it important to make the NameNode resilient to failures?
ANS:
Without the NN the filesystem cannot be used. If the NN is gone then there would be no way of knowing how to reconstruct the files from the blocks on the DNs. So it's imp to make the NN resilient to failures.

3) What details are there in the FsImage file?
ANS:
An fsimage file contains the complete state of the file system at a point in time. This information includes metadata like list of files, list of blocks for each file, list of DataNodes for each block, file permissions, modification, access times, replication factor, etc.) Every file system modification is assigned a unique, monotonically increasing transaction ID. An fsimage file represents the file system state after all modifications up to a specific transaction ID.

4) What is the purpose of the secondary name-node?
ANS:
The term "secondary name-node" is somewhat misleading. It is not a name-node in the sense that data-nodes cannot connect to the secondary name-node, and in no event, it can replace the primary name-node in case of its failure.
The only purpose of the secondary name-node is to perform periodic checkpoints. The secondary name-node periodically downloads current name-node image and edits log files, joins them into new image and uploads the new image back to the (primary and the only) name-node.
So, if the name-node fails and you can restart it on the same physical node then there is no need to shutdown data-nodes, just the name-node need to be restarted. If you cannot use the old node anymore you will need to copy the latest image somewhere else. The latest image can be found either on the node that used to be the primary before failure if available; or on the secondary name-node. The latter will be the latest checkpoint without subsequent edits logs, that means the most recent name space modifications may be missing there. You will also need to restart the whole cluster in this case.

5) Does the NameNode stay in safe mode till all under-replicated files are fully replicated? Why or why not?
ANS:
No. During safe mode replication of blocks is prohibited. The name-node awaits when all or majority of data-nodes report their blocks.
Depending on how safe mode parameters are configured the name-node will stay in safe mode until a specific percentage of blocks of the system are minimally replicated. If the safe mode threshold is set to 1 then all blocks of all files should be minimally replicated.
Minimal replication does not mean full replication. Some replicas may be missing and to replicate them the name-node needs to leave safe mode.

6) What are the core changes in Hadoop 2.x compared to Hadoop 1.x? Or in other words, state the differences between Hadoop 1 and Hadoop 2.
ANS:

Hadoop 2.x provides an upgrade to Hadoop 1.x in terms of resource management, scheduling and the manner in which execution occurs. In Hadoop 2.x the cluster resource management capabilities work in isolation from the MapReduce specific programming logic. This helps Hadoop to share resources dynamically between multiple parallel processing frameworks like Hive, Spark and the core MapReduce component.

In Hadoop 1.x, MapReduce is responsible for both processing and cluster management whereas in Hadoop 2.x processing is taken care of by other processing models and YARN is responsible for cluster management.

In short, following are the changes in Hadoop 2.x.

1. NN Single point of failure rectified by adding High availability feature (addition of StandBy NameNode) and HDFS federation.

2. Nodes limitation (scalability issue) rectified with Federation which allows horizontal scalability and by splitting up JobTracker's job in YARN.

3. JobTracker bottleneck rectified by allocating Application Master for each job.

4. MapReduce slots are changed from static to dynamic with the use of separate containers for Map & Reduce tasks.

5. Allows other applications like interactive, graph iterative algorithms also to integrate with HDFS.

7) What is the difference between MR1 in Hadoop 1.0 and MR2 in Hadoop2.0?
ANS:
In Hadoop 2, MapReduce is split into two components: The cluster resource management capabilities have become YARN, while the MapReduce-specific capabilities remain MapReduce.

In MR1 architecture, the cluster was managed by a service called the JobTracker. TaskTracker services lived on each node and would launch tasks on behalf of jobs. The JobTracker would serve information about completed jobs.

In MR2, the functions of the JobTracker are divided into three services. The *ResourceManager* is a persistent YARN service that receives and runs applications (a MapReduce job is an application) on the cluster. It contains the scheduler, which, as in MR1, is pluggable.

The MapReduce-specific capabilities of the JobTracker have moved into the MapReduce ApplicationMaster, one of which is started to manage each MapReduce job and terminated when the job completes. The JobTracker's function of serving information about completed jobs has been moved to the JobHistoryServer.

The TaskTracker has been replaced with the NodeManager, a YARN service that manages resources and deployment on a node. NodeManager is responsible for launching containers, each of which can house a map or reduce task.

8) What is HDFS Federation? What advantage does it provide?
ANS:
HDFS federation is mainly added to take care of NameNode SPOF and Scalability issue related to that. With HDFS Federation it is possible to add multiple NameNodes to a cluster. Each NameNode is responsible to manage a portion of the filesystem there by sharing the workload of the cluster.

For instance, let's say we have 2 teams – Marketing and Research in our company funding the Hadoop cluster. You can create a Namespace called /marketing which will be managed by one NameNode and another Namespace under /research which will be managed by another NameNode.

The advantage of this is that you don't have to run two different Hadoop clusters. You can run a single Hadoop infrastructure but one NameNode will manage all the files under /marketing and another NameNode will manage all the files under /research.

Each NameNode is only responsible for it's assigned namespace and will not share metadata or information between them and also will not communicate with one another. When a NameNode managing /marketing goes down it will affect all the files under /marketing and users will still be able to access HDFS and files under /research since NameNode managing /research is fully functional.

9) What is NameNode High Availability and how is it achieved in Hadoop 2?
ANS:
Standby NameNode is added to Hadoop 2 as a High Availability feature.
In the event of the failure of the active NameNode, the standby takes over its duties to continue servicing client requests without a significant interruption. A few architectural changes are needed to allow this to happen:

The NameNodes must use highly available shared storage to share the edit log. When a standby NameNode comes up, it reads up to the end of the shared edit log to synchronize its state with the active NameNode, and then continues to read new entries as they are written by the active NameNode.

Datanodes must send block reports to both NameNodes because the block mappings are stored in a NameNode's memory, and not on disk. Clients must be configured to handle NameNode failover, using a mechanism that is transparent to users. The Secondary NameNode's role is subsumed by the standby, which takes periodic checkpoints of the active NameNode's namespace.

There are two choices for the highly available shared storage: an NFS filer, or a quorum journal manager (QJM). The QJM is a dedicated HDFS implementation, designed for the sole purpose ofproviding a highly available edit log, and is the recommended choice for most HDFS installations. The QJM runs as a group of journal nodes, and each edit must be written to a majority of the journal nodes. Typically, there are three journal nodes, so the system can tolerate the loss of one of them. This arrangement is similar to the way ZooKeeper works, although it is important to realize that the QJM implementation does not use ZooKeeper.

If the active NameNode fails, the standby can take over very quickly (in a few tens of seconds) because it has the latest state available in memory: both the latest edit log entries and an up-to-date block mapping. The actual observed failover time will be longer in practice (around a minute or so), because the system needs to be conservative in deciding that the active NameNode has failed. In the unlikely event of the standby being down when the active fails, the administrator can still start the standby from cold. This is no worse than the non-HA case.

10) What is the role of Application Master in YARN application execution?
ANS:
There is one ApplicationMaster per application and has short life. It's a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks.

To run an application, a client contacts the ResourceManager and asks it to run an ApplicationMaster process. It is responsible for requesting containers to perform application-specific work.

Precisely what the AM does once it is running depends on the application. It could simply run a computation in the container it is running in and return the result to the client. Or, it could request more containers from the resource manager and use them to run a distributed computation (MapReduce does this).