BIG DATA ANALYTICS FRAMEWORK

UNIT III MAPREDUCE-II AND SPARK

MapReduce

MapReduce is a big data processing technique, and a model for how to programmatically implement that technique.

Its goal is to sort and filter massive amounts of data into smaller subsets, then distribute those subsets to computing nodes, which process the filtered data in parallel.

Failures

- ✓In the real world, user code is buggy, processes crash, and machines fail.
- ✓One of the major benefits of using Hadoop is its ability to handle such failures and allow your job to complete successfully.
- ✓We need to consider the failure of any of the following entities:
  - ✓The task
  - ✓The application master
  - ✓The node manager, and
  - ✓The resource manager.

Task Failure

The most common occurrence of this failure is when user code in the map or reduce task throws a runtime exception.

If this happens,the task JVM reports the error back to its parent application master before it exits.

The error ultimately makes it into the user logs. The application master marks the task attempt as failed, and frees up the container so its resources are available for another task.

Another failure mode is the sudden exit of the task JVM—perhaps there is a JVM bug that causes the JVM to exit for a particular set of circumstances exposed by the MapReduce user code. In this case, the node manager notices that the process has exited and informs the application master so it can mark the attempt as failed.

Hanging tasks are dealt with differently. The application master notices that it hasn’t received a progress update for a while and proceeds to mark the task as failed. The task JVM process will be killed automatically after this period.

The timeout period after which tasks are considered failed is normally 10 minutes and can be configured on a per-job basis (or a cluster basis) by setting the mapreduce.task.timeout property to a value in milliseconds.

Application Master Failure

- Just like MapReduce tasks are given several attempts to succeed (in the face of hardware or network failures), applications in YARN are retried in the event of failure.
- The maximum number of attempts to run a MapReduce application master is controlled by the mapreduce.am.max-attempts property.
- The default value is 2, so if a MapReduce application master fails twice it will not be tried again and the job will fail.
- The way recovery works is :.
- An application master sends periodic heartbeats to the resource manager, and in the event of application master failure, the resource manager will detect the failure and start a new instance of the master running in a new container (managed by a node manager).

Node Manager Failure

- If a node manager fails by crashing or running very slowly, it will stop sending heartbeats to the resource manager (or send them very infrequently).
- The resource manager will notice a node manager that has stopped sending heartbeats if it hasn’t received one for 10 minutes (this is configured, in milliseconds, via the yarn.resourcemanager.nm.liveness-monitor.expiry-interval-ms property) and remove it from its pool of nodes to schedule containers on.

Resource Manager Failure

- Failure of the resource manager is serious, because without it, neither jobs nor task containers can be launched.
- In the default configuration, the resource manager is a single point of failure, since in the (unlikely) event of machine failure, all running jobs fail—and can’t be recovered.
- To achieve high availability (HA), it is necessary to run a pair of resource managers in an active-standby configuration. If the active resource manager fails, then the standby can take over without a significant interruption to the client.
- Information about all the running applications is stored in a highly available state store (backed by ZooKeeper or HDFS), so that the standby can recover the core state of the failed active resource manager

Shuffle and Sort

- MapReduce makes the guarantee that the input to every reducer is sorted by key.
- The process by which the system performs the sort—and transfers the map outputs to the reducers as inputs—is known as the shuffle.
- The shuffle is an area of the codebase where refinements and improvements are continually being made
- In many ways, the shuffle is the heart of MapReduce and is where the “magic” happens.

- YARN stands for “Yet Another Resource Negotiator“. It was introduced in Hadoop 2.0 to remove the bottleneck on Job Tracker which was present in Hadoop 1.0.