Task 1 Explain the difference between FIFO and Capacity scheduler

The FIFO Scheduler places applications in a queue and runs them in the order of submission.

->Requests for the first application in the queue are allocated first;

once its requests have been satisfied, it serves to the next application in the queue and so on.

->The FIFO Scheduler’s simple to understand and doesn’t require any configuration, but it’s not suitable for shared clusters.

->All the resources are used by large applications in a cluster, so each application has to wait for its turn.

On a shared cluster, so it’s better to use the Capacity Scheduler or the Fair Scheduler.

With the Capacity Scheduler, a separate dedicated queue allows the small job to start as soon as it is submitted.

->it’s at the cost of overall cluster utilization since the queue capacity is reserved for jobs in that queue.

->If queues are not designed or used properly, some queues may be overloaded while some may be underutilised.

Q.2 Explain the difference between FIFO and Fair scheduler

Fair scheduling :- it is a method which is used to assign resources to jobs such that all jobs get, on average, an equal share of resources over time. The entire cluster is used by the job which is running . When other jobs are submitted, tasks slots that free up are assigned to the new jobs, so that each job gets roughly the same amount of CPU time. Unlike the default Hadoop scheduler, which forms a queue of jobs, this lets short jobs finish in reasonable time while not starving long jobs .It is also a reasonable way to share a cluster between a number of users. Finally, fair sharing can also work with job priorities the priorities are used as weights to determine the fraction of total compute time that each job should get. The original scheduling algorithm that was integrated within the Job Tracker was called FIFO. In FIFO scheduling, a Job Tracker pulled jobs from a work queue, oldest job first.

Q.3 Explain the difference between Capacity and Fair scheduler

Fair scheduling :- it‘s a method which is used to assign resources to jobs such that all jobs get, on average, an equal share of resources over time. When there is a single job running, that job uses the entire cluster. When other jobs are submitted, tasks slots that free up are assigned to the new jobs, so that each job gets roughly the same amount of CPU time. Unlike the default Hadoop scheduler, which forms a queue of jobs, this lets short jobs finish in reasonable time while not starving long jobs. It is also a reasonable way to share a cluster between a number of users. Finally, fair sharing can also work with job priorities – the priorities are used as weights to determine the fraction of total compute time that each job should get. The Capacity Scheduler is designed to allow sharing a large cluster while giving each organization a minimum capacity guarantee. The central idea is that the available resources in the Hadoop Map-Reduce cluster are partitioned among multiple organizations who collectively fund the cluster based on computing needs. There is an added benefit that an organization can access any excess capacity no being used by others.

Q.4 What are the limitations of Hadoop 1.x and how they were overcome in Hadoop 2.x

Ans.

-> Hadoop 1.x supports only one namespace for managing HDFS file system whereas Hadoop 2.x supports multiple namespaces.

-> Hadoop 1.x supports one and only one programming model: Map Reduce. Hadoop 2.x supports multiple programming models with

YARN Component like Map Reduce, Interactive, Streaming, Graph, Spark, Storm etc.

-> Hadoop 1.x has lot of limitations in Scalability. Hadoop 2.x has overcome that limitation with new architecture.

-> Hadoop 2.x has Multi-tenancy Support, but Hadoop 1.x doesn’t.

-> Hadoop 1.x HDFS uses fixed-size Slots mechanism for storage purpose whereas Hadoop 2.x uses variable-sized Containers.

-> Hadoop 1.x supports maximum 4,000 nodes per cluster where Hadoop 2.x supports more than 10,000 nodes per cluster.

-> Hadoop 1.x works on concepts of slots – slots can run either a Map task or a Reduce task only while Hadoop 2.x Works on

concepts of containers. Using containers can run generic tasks.