Q1.Explain the difference between FIFO and Capacity scheduler?

A1.

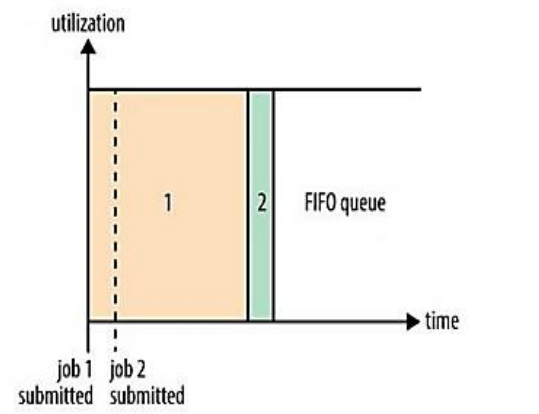
**FIFO scheduler**

The FIFO Scheduler places applications in a queue and runs them in the order of submission (first in, first out).

• Requests for the first application in the queue are allocated first; once its requests have been satisfied, the next application in the queue is served, and so on.

• The FIFO Scheduler has the merit of being simple to understand and not needing any configuration, but it’s not suitable for shared clusters.

• Large applications will use all the resources in a cluster, so each application has to wait its turn. On a shared cluster, it is better to use the Capacity Scheduler or the Fair Scheduler.



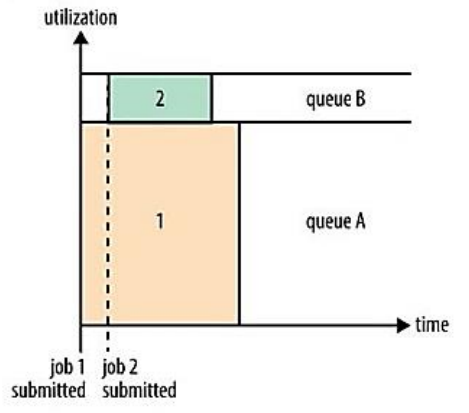
### Capacity scheduler

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

### This is 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 underutilized.

### Large job finishes late when compared with using the FIFO Scheduler.



Q2.Explain the difference between FIFO and Fair scheduler?

A2

**Fair scheduling**

Fair scheduling is a method of assigning 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.

**FIFO scheduling**

The original scheduling algorithm that was integrated within the JobTracker was called FIFO. In FIFO scheduling, a JobTracker pulled jobs from a work queue, oldest job first. This schedule had no concept of the priority or size of the job, but the approach was simple to implement and efficient.

Q3 Explain the difference between Capacity and Fair scheduler?

A3

**Fair scheduling**

Fair scheduling is a method of assigning 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.

**Capacity Scheduler**

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. This provides elasticity for the organizations in a cost-effective manner.

Q4.What are the limitations of hadoop 1.x and how they were overcome in hadoop 2.x?

A4

* Hadoop 1.x supports only one namespace for managing HDFS filesystem whereas Hadoop 2.x supports multiple namespaces.
* Hadoop 1.x supports one and only one programming model: MapReduce. Hadoop 2.x supports multiple programming models with YARN Component like MapReduce, Interative, 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.
* Hadoop 1.x does not support Microsoft Windows while Hadoop 2.x Added support for Microsoft windows