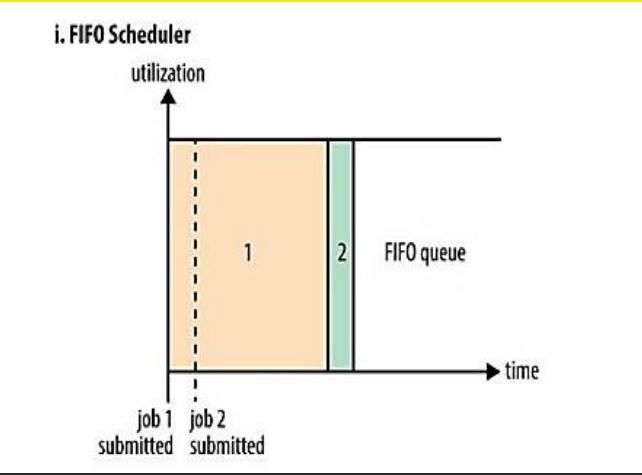
**Problem Statement**

1. **Explain the difference between FIFO and Capacity scheduler**

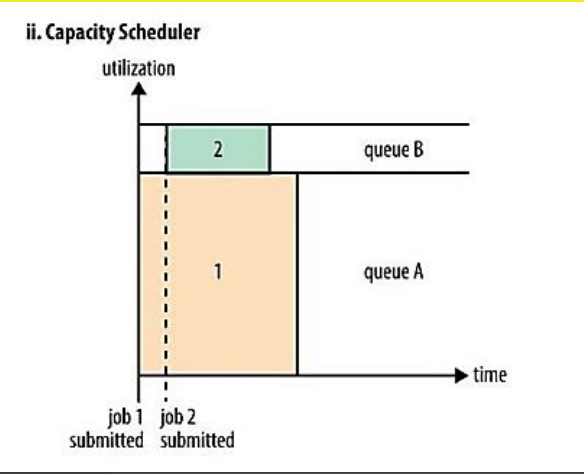
**FIFO Scheduler-**

* 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-**

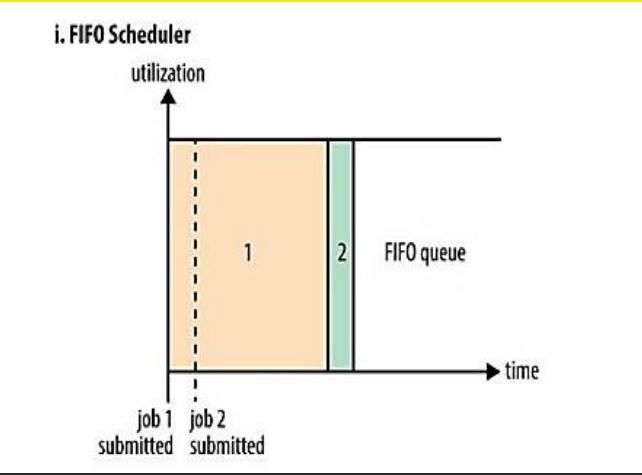
* With the Capacity Scheduler, a separate dedicated queue allows the small job to start as soon as it is submitted.
* This is at the cost of overall cluster utilization since the queue capacity is reserved for jobs in that queue.
* If queues are not designed or used properly, some queues may be overloaded while some may be underutilized.
* Large job finishes late when compared with using the FIFO Scheduler.

****

1. **Explain the difference between FIFO and Fair scheduler**

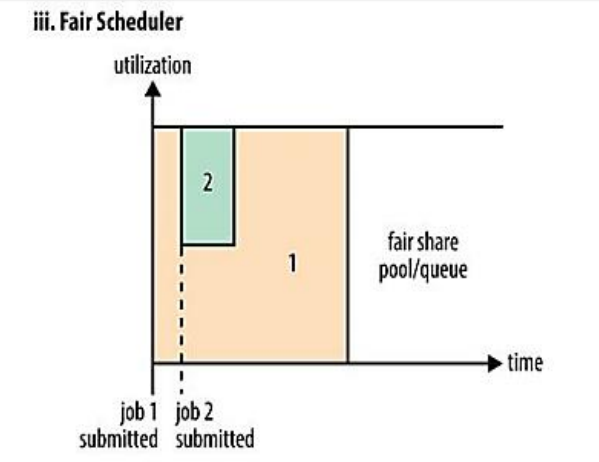
**FIFO Scheduler-**

* 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.



**Fair Scheduler-**

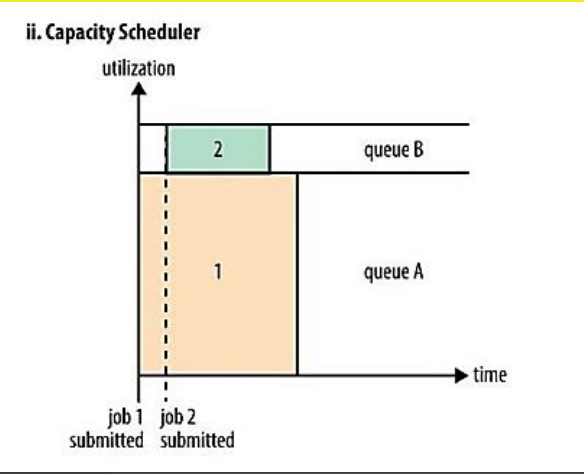
* With the Fair Scheduler, there is no need to reserve a set amount of capacity, since it will dynamically balance resources between all running jobs.
* Just after the first (large) job starts, it is the only job running, so it gets all the resources in the cluster.
* When the second (small) job starts, it is allocated half of the cluster resources, so that each job is using its fair share of resources.
* After the small job completes and no longer requires resources, the large job goes back to using the full cluster capacity again.
* The overall effect is both high cluster utilization and timely small job completion.



1. **Explain the difference between Capacity and Fair scheduler**

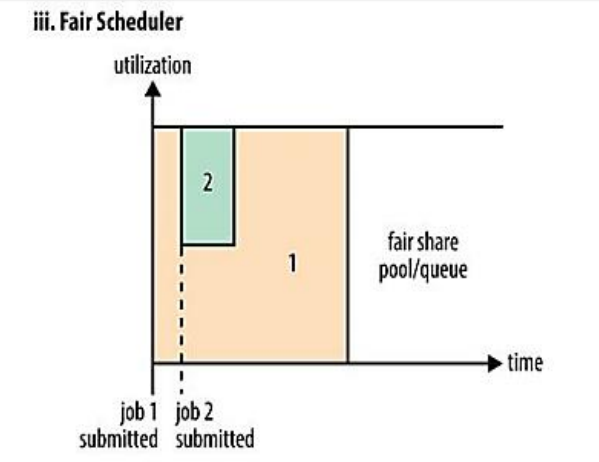
**Capacity Scheduler-**

* 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.

****

**Fair Scheduler-**

* With the Fair Scheduler, there is no need to reserve a set amount of capacity, since it will dynamically balance resources between all running jobs.
* Just after the first (large) job starts, it is the only job running, so it gets all the resources in the cluster.
* When the second (small) job starts, it is allocated half of the cluster resources, so that each job is using its fair share of resources.
* After the small job completes and no longer requires resources, the large job goes back to using the full cluster capacity again.
* The overall effect is both high cluster utilization and timely small job completion.



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

Hadoop 1.x has several issues which were resolved by Hadoop 2.x. Some major issues faced by Hadoop 1.x are-

1. Name Node memory constraint- Amount of meta data, name node can store is restricted to the memory available to Name node daemon in Hadoop 1.x. Also number of nodes that Hadoop 1.x cluster could manage was around 4000 machines.
2. Name node being single point of failure- Name node was turning out to be the single point of failure as of Hadoop 1.x, if name node goes down, the entire cluster would have come to a stand still. A Hadoop admin intervention is required which could mean a cluster downtime of at least 45 minutes to 1 hour.
3. Job Tracker overloading- While processing the data, sometimes the processing daemon becomes overloaded. Processing daemon does too many things like- Job Allocation, Job Scheduling and Monitoring, Resource Allocation etc. Also the processing part of Hadoop 1.x supported only one processing framework.

These all issues were addressed in Hadoop 2.x by

1. Issue 1 was **Name Node memory constraint** was handled by Hadoop 2.x by **HDFS Federation.**  In HDFS Federation we use multiple name nodes each connected with their 4000 machines containing fsimage and edits. In HDFS Federation, there are multiple name nodes, each storing the meta data and block mapping of files and directories contained in particular sub directories.
2. Issue 2 was **Name node being single point of failure**  which was handled by Hadoop 2.x by introducing **High Availability.** In HDFS High Availability, there is a pair of Name Nodes in an active and standby configuration. In the event of failure of an active name node, the standby takes over its duties without a significant interruption.
3. Issue 3 was **Job Tracker Overloading** which was handled by Hadoop 2.x by introducing **YARN.** YARN is Yet another Resource Negotiator provides support for other processing framework like Message Passing Interface, Graph Processing and many more such frameworks. YARN to address the overloading issue of Job Tracker in Hadoop 1.x also divides the job tracker responsibilities of Hadoop 1.x into same separate components.

**Difference between Hadoop 1.x and Hadoop 2.x**

**Hadoop 1.x-** Job Tracker keeps track of Job monitoring and resource allocation.

* Suited for maximum in 4000 nodes and 40000 Tasks.
* Task Tracker configured witch static slots, a map task cannot run on reduce a slot. So cluster utilization is low.
* Supports map reduce processing models.

**Hadoop 2.x-** Resource manager takes care of resource utilization, application master takes care of job monitoring.

* It can scale up to 10000 nodes and 100000 tasks.
* Resources are dynamic and fine grained leads to better cluster utilization.
* Supports processing models other than map reduce.