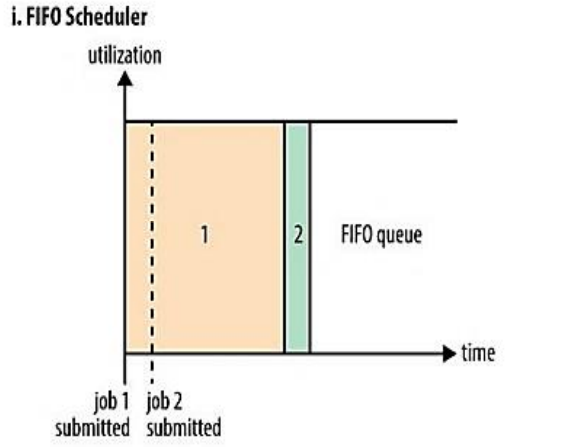
**Assignment 8.4**

**Question:** Explain the difference between FIFO and Capacity scheduler

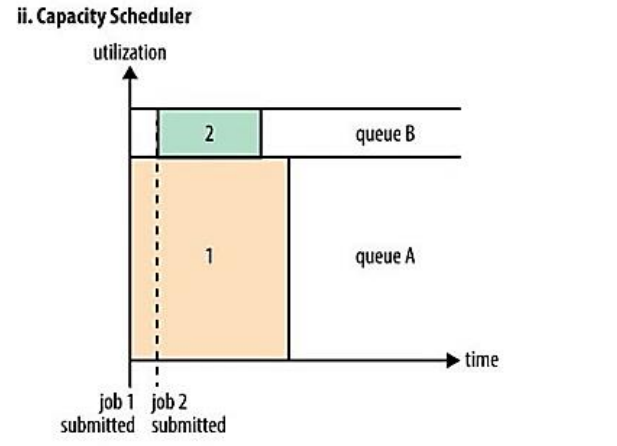
***FIFO Scheduler:***

* 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:***

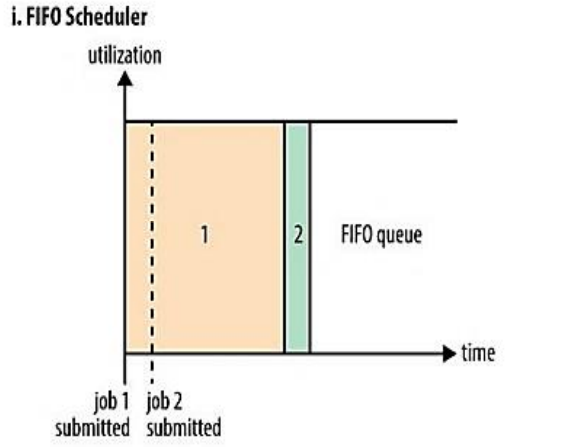
* With the Capacity Scheduler, a separate dedicated queue allows the small job to start as soon as it is submitted.
* This is at the cost of overall cluster utilization since the queue capacity is reserved for jobs in that queue.
* If queues are not designed or used properly, some queues may be overloaded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



**Question:** Explain the difference between FIFO and Fair scheduler

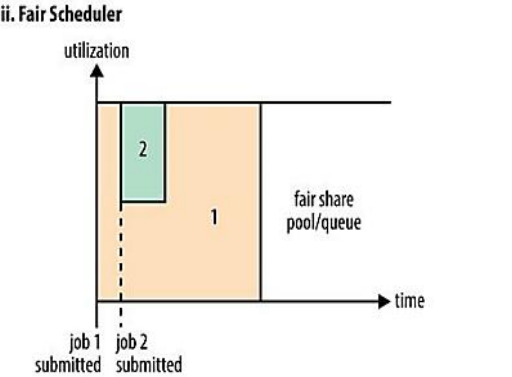
***FIFO Scheduler:***

* 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:***

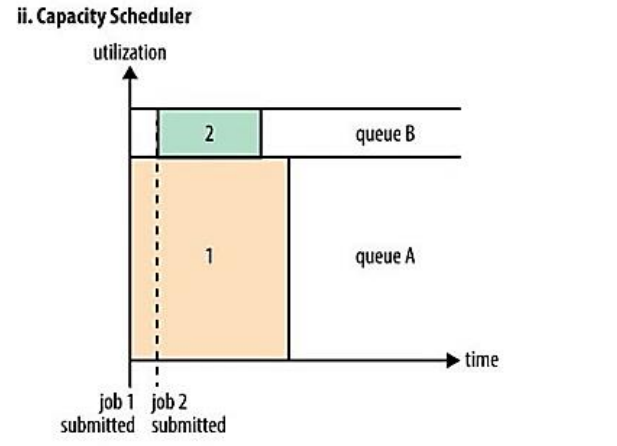
* With the Fair Scheduler, there is no need to reserve a set amount of capacity, since it will dynamically balance resources between all running jobs.
* Just after the first (large) job starts, it is the only job running, so it gets all the resources in the cluster.
* When the second (small) job starts, 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.



**Question:** Explain the difference between Capacity and Fair scheduler

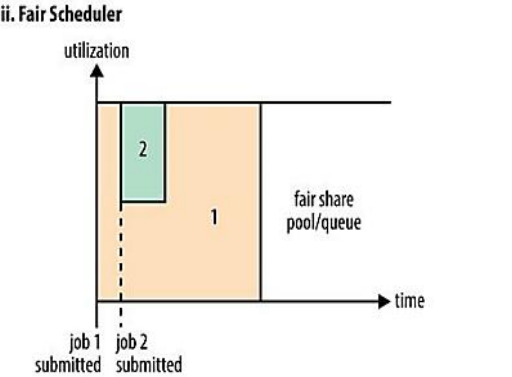
***Capacity Scheduler:***

* 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.



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

***Limitations of Hadoop 1.x:***

* It is only suitable for Batch Processing of Huge amount of Data, which is already in Hadoop System.
* It is not suitable for Real-time Data Processing.
* It is not suitable for Data Streaming.
* It supports up to ***4000 Nodes*** per Cluster.
* It has a single component : JobTracker to perform many activities like Resource Management, Job Scheduling, Job Monitoring, Re-scheduling Jobs etc.
* JobTracker is the single point of failure.
* It does not support Multi-tenancy Support.
* It supports only one Name Node and One Namespace per Cluster.
* It does not support Horizontal Scalability.
* It runs only Map/Reduce jobs.
* It follows Slots concept in HDFS to allocate Resources (Memory, RAM, and CPU). It has static Map and Reduce Slots. That means once it assigns resources to Map/Reduce jobs, it cannot re-use them even though some slots are idle.

***Hadoop 2.x solves limitations of Hadoop 1.x:***

* By decoupling MapReduce component responsibilities into different components.
* By introducing new YARN component for Resource management.
* By decoupling component’s responsibilities, it supports multiple namespace, Multi-tenancy, Higher Availability and Higher Scalability.