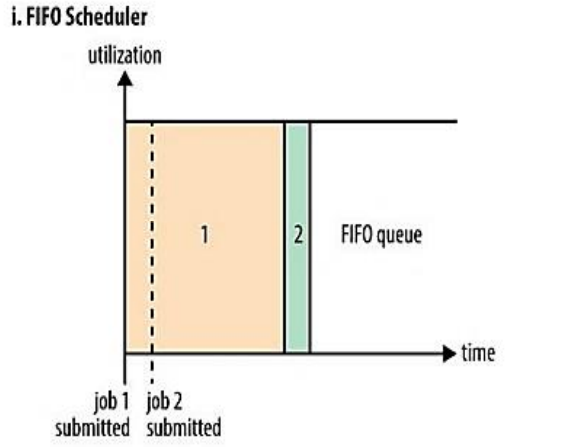
1. **Explain the difference between FIFO and Capacity scheduler**

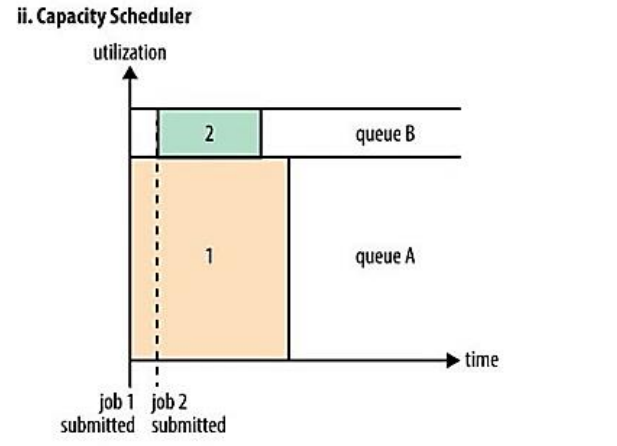
FIFO Scheduler**:**

1. The FIFO Scheduler places applications in a queue and runs them in the order of submission (first in, first out).
2. 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.
3. The FIFO Scheduler has the merit of being simple to understand and not needing any configuration, but it’s not suitable for shared clusters.
4. 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:

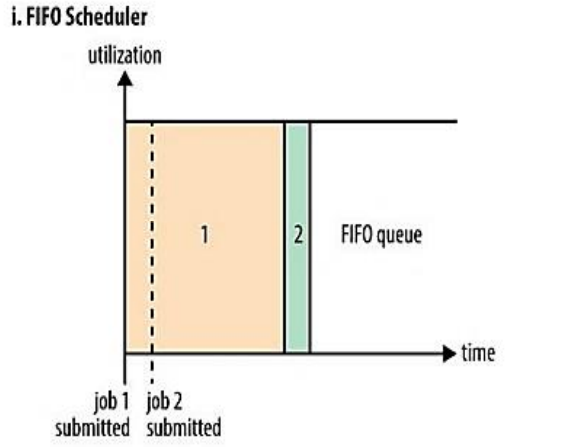
1. With the Capacity Scheduler, a separate dedicated queue allows the small job to start as soon as it is submitted.
2. This is at the cost of overall cluster utilization since the queue capacity is reserved for jobs in that queue.
3. If queues are not designed or used properly, some queues may be overloaded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



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

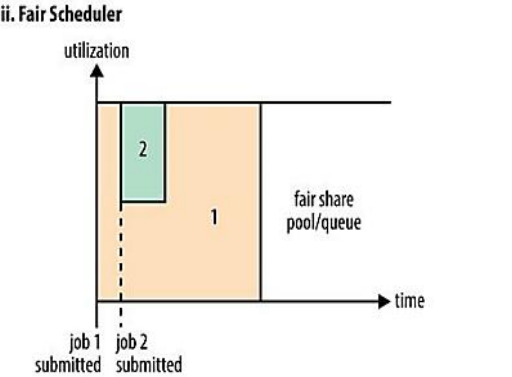
**FIFO Scheduler:**

1. The FIFO Scheduler places applications in a queue and runs them in the order of submission (first in, first out).
2. 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.
3. The FIFO Scheduler has the merit of being simple to understand and not needing any configuration, but it’s not suitable for shared clusters.
4. 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:**

1. 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.
2. Just after the first (large) job starts, it is the only job running, so it gets all the resources in the cluster.
3. 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.
4. After the small job completes and no longer requires resources, the large job goes back to using the full cluster capacity again.
5. The overall effect is both high cluster utilization and timely small job completion.

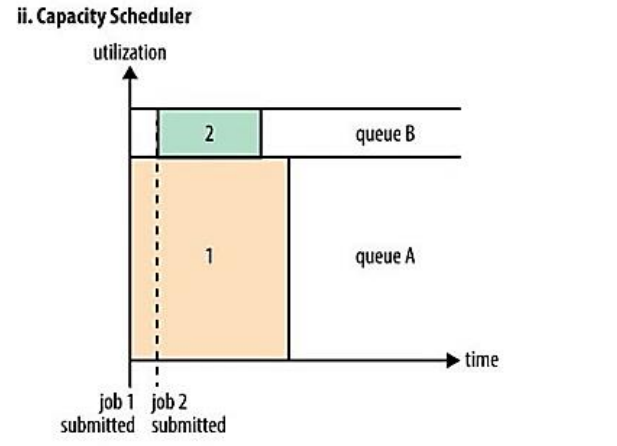


\

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

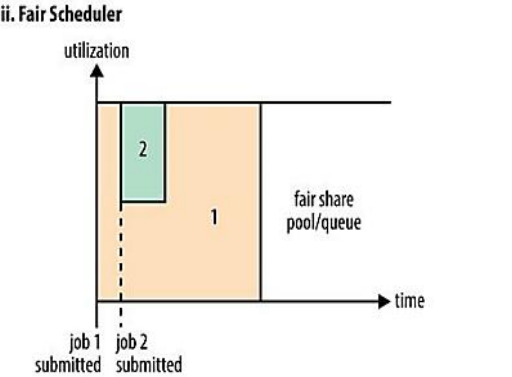
Capacity Scheduler:

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



**Fair scheduler:**

1. 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.
2. Just after the first (large) job starts, it is the only job running, so it gets all the resources in the cluster.
3. 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.
4. After the small job completes and no longer requires resources, the large job goes back to using the full cluster capacity again.
5. 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**

**Limitations of Hadoop 1.x:**

1. It is only suitable for Batch Processing of Huge amount of Data, which is already in Hadoop System.
2. It is not suitable for Real-time Data Processing.
3. It is not suitable for Data Streaming.
4. It supports up to ***4000 Nodes*** per Cluster.
5. It has a single component : JobTracker to perform many activities like Resource Management, Job Scheduling, Job Monitoring, Re-scheduling Jobs etc.
6. JobTracker is the single point of failure.
7. It does not support Multi-tenancy Support.
8. It supports only one Name Node and One Namespace per Cluster.
9. It does not support Horizontal Scalability.
10. It runs only Map/Reduce jobs.
11. It follows Slots concept in HDFS to allocate Resources (Memory, RAM, 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:**

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