**Q1. 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:

• In Capacity scheduling, there are some queues. Each queue has its own assigned resources and uses FIFO strategy in itself.

•In order to prevent some user take too much of resources in one queue, the scheduler can limit the resources for the jobs from each user.

•When scheduling, all queues are monitored, if a queue does not use its allocated capacity, the spare capacity will be assigned to other queues.

•Jobs with a higher priority can access to resources sooner than lower priority jobs.

**Q2. 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.

**Q3. 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 underutilised.

• 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.

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

Hadoop 1.x has single NameNode which is single point of failure.  In case of Namenode failure, needs manual intervention to overcome.

Hadoop 2.x has feature to overcome SPOF with a standby Namenode and in case of Namenode failure, it is configured for automatic recovery.

The scaling capacity of Hadoop 1.x is increased in Hadoop 2.x upto 10000 nodes per cluster.

The MapReduce performs the resource management as well as processing in Hadoop 1.x.In Hadoop 2.x YARN is used for resource management and it has other processing models for processing.

A Namenode failure affects the stack in Hadoop 1.x.To solve this problem in Hadoop 2.x stack – Hive, Pig, HBase etc. are all equipped to handle Namenode failure.

Hadoop 2.x has support for Microsoft Windows which is not there in Hadoop 1.x.

In this way the limitations of Hadoop 1.x are overcome in Hadoop 2.x