**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 underutilised.

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

**Limitations of hadoop 1.x and how they were overcome in hadoop 2.x**

* In Hadoop2.x with the help of YARN  architecture, we can run larger clusters than Hadoop v1. Hadoop v1 hits scalability bottlenecks in the region of 4,000 nodes and 40,000 tasks, deriving from the fact that the job tracker has to manage both jobs and tasks. YARN overcomes these limitations by virtue of its split resource manager/application master architecture: It is designed to scale up to 10,000 nodes and 100,000 tasks.
* In contrast to the jobtracker, each instance of an application  – here, a MapReduce job – has a dedicated application master, which runs for the duration of the application. This model is actually closer to the original GFS paper, which describes how a master process is started to coordinate map and reduce tasks running on a set of workers.
* n Hadoop1.x, we can only run MapReduce framework jobs to process the data which is stored in HDFS. We couldn’t had the opportunity to run other applications than MapReduce in the HDFS cluster. Thus, Hadoop2.x came up with new framework YARN which provides the ability to run non-MapReduce jobs like Spark, Hama, Giraph, Message Passing Interface) MPI & HBase coprocessors.
* Previously, in Hadoop1.x we had single namenode which maintained a directory tree of HDFS files and tracked where data was stored in the cluster.  If the Namenode is down due to some unplanned event such as a machine crash, the whole Hadoop cluster will be down as well.
* Hadoop2.x comes with the solution for this problem, which allows users to configure clusters with redundant namenodes, removing the chance that a lone namenode will become a single point of failure within a cluster
* Hadoop was originally developed to support the UNIX family of operating systems. With Hadoop2, the Windows operating system is natively supported. This extends the reach of Hadoop significantly to a sizable Windows Server market.