1. Explain the difference between FIFO and Capacity scheduler

|  |  |  |
| --- | --- | --- |
| No | FIFO scheduler | Capacity scheduler |
| 1 | The FIFO Scheduler places applications in a queue and runs them in the order of submission (first in, first out). | The CapacityScheduler is designed to allow sharing a large cluster while giving each organization a minimum capacity guarantee. The central idea is that the available resources in the Hadoop Map-Reduce cluster are partitioned among multiple organizations who collectively fund the cluster based on computing needs. There is an added benefit that an organization can access any excess capacity |
| 2 | Large applications will use all the resources in a cluster, so each application has to wait its turn | No waiting or Less Waiting compared to FIFO.  On a shared cluster, it is better to use the Capacity Scheduler. |
| 3 | The FIFO Scheduler has the merit of being simple to understand and not needing any configuration | Capacity scheduler needs some configuration |
| 4 | Not suitable for Shared cluster | Highly suitable for Shared cluster |

2.Explain the difference between FIFO and Fair scheduler

|  |  |  |
| --- | --- | --- |
|  | FIFO | Fair scheduler |
| 1 | The FIFO Scheduler places applications in a queue and runs them in the order of submission (first in, first out). | Fair scheduling is a method of assigning resources to jobs such that all jobs get, on average,  an equal share of resources over time. When there is a single job running, that job uses the entire cluster. When other jobs are submitted, tasks slots that free up are assigned to the new jobs, so that each job gets roughly the same amount of CPU time |
| 2 | Large applications will use all the resources in a cluster, so each application has to wait its turn | No waiting compared to FIFO.  On a shared cluster, it is better to use the Capacity Scheduler. |
| 3 | The FIFO Scheduler has the merit of being simple to understand and not needing any configuration | Capacity scheduler needs some configuration |
| 4 | Not suitable for Shared cluster | Highly suitable for Shared cluster |

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

|  |  |  |
| --- | --- | --- |
|  | CAPACITY SCHEDULER | FAIR SCHEDULER |
| 1 | The CapacitySched  uler is designed to allow sharing a large cluster while giving each  organization a minimum capacity guarantee. The central idea is that the available resources in the Hadoop Map-Reduce cluster are partitioned among multiple organizations who collectively fund the cluster based on computing needs. There is an added benefit that an organization can access any excess capacity | Fair scheduling is a method of assigning resources to jobs such that all jobs get, on average,  an equal share of resources over time. When there is a single job running, that job uses the entire cluster. When other jobs are submitted, tasks slots that free up are assigned to the new jobs, so that each job gets roughly the same amount of CPU time. |
| 2 | A separate dedicated queue allows the small job to start as soon as it is submitted. | No separate queue instead sharing of resources by tasks in queue  Takes place |
| 3 | If queues are not designed or used properly, some queues may be overloaded while some may be underutilised. | 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

Limitation Of 1.x

1.NameNode-Single Point Of Failure-overcome by High Availability

2.Cluster limit is upto 4000-Overcome by Federation

High Availability

In Hadoop 1.x there is a main disadvantage related to NameNode (ie) Single point of Failure(if name node fails there will be some loss of data despite having a secondary node as the updation from name node to secondary name node is done on hourly basis)

So we come with the solution of HIGH AVAILABILITY OF NODE

In Hadoop 2.x we have two name node (other than secondary name)namely

1.Active NameNode

2.Standby NameNode

Now the Data node will send block reports to both Active NameNode and Standby NameNode

So if active NameNode fails or crashes Stand By Name node becomes active NameNode

Then how do we know how Active NameNode fails .So for that we have Journal Node

This works on the Basis of polling so if there are 3 journal nodes formula for calculated quorom is

(no of journal node + 2)/2(suppose 3 journal node (3+2)/2 =2)) while the expected QUOROM here is 2.

If active name node fails the journal count quorom will be less than 2 and the journal node automatically changes the

Stand by name node as active name node and and when the failed name node gets up it will now be a standby and the process goes on

2. HADOOP FEDERATION

In Hadoop 1.x version there is only one name node which has a memory of 64 gB which could maintain a cluster of 4000 data nodes.

With increase in Data generation we are now running towards a scenario where say 10,000 data nodes may be required.SO MORE MEMORY in NAMENODE is required and the SCALE UP OF A SINGLE NAME NODE ABOVE 64 GB but these results in overhead cost.

So in Hadoop 2.x we have a feature called FEDERATION where instead of a single name node set(i.e active name node,passive name node and secondary name node ) there are Multiple name node set(say 3) such that each name node maintains the METADATA of each department separately(say one name node takes care of marketing data alone and other name node takes care of finance data alone) but all the name nodes can access all DATA NODES but they will only look at related information(ie MARKETING NAME NODE TAKES METADATA ABOUT ONLY MARKEETING AND FINANCE NAME NODE TAKES METADATA ABOUT ONLY FINANCE)