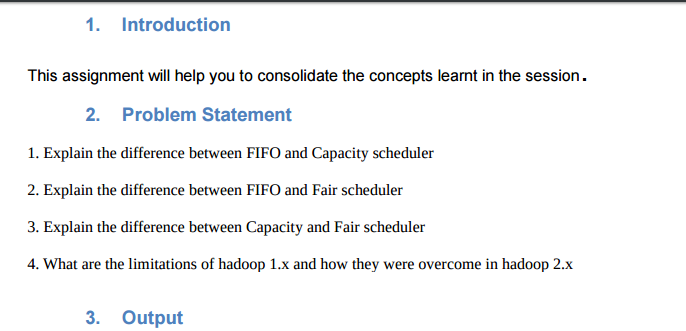
**Assignment 8.4**

****

1. **Explain the di1fference between FIFO and Capacity scheduler**

|  |  |  |
| --- | --- | --- |
|  | **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). | With the Capacity Scheduler, a separate dedicated queue allows the small job to start as soon as it is submitted. |
| 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** |
|  | * The FIFO Scheduler places applications in a queue and runs them in the order of submission (first in, first out). * Large applications will use all the resources in a cluster, so each application has to wait its turn * The FIFO Scheduler has the merit of being simple to understand and not needing any configuration * Not suitable for Shared cluster | * 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. |

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

|  |  |  |
| --- | --- | --- |
|  | **CAPACITY SCHEDULER** | **FAIR SCHEDULER** |
| 1 | The Capacity Scheduler 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**

* **NameNode**  - Single Point Of Failure-overcome by High Availability
* **Cluster limit** is upto 4000 nodes - Overcome by Federation in case of Hadoop 2.x it can scale upto 10000 nodes.

**High Availability**

In Hadoop 1.x there is a main disadvantage related to NameNode that is, 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 Standby Name node becomes active NameNode.

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 when the failed name node gets up it will now be a standby and the process goes on.

* **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 such that each name node maintains the METADATA of each department separately but all the name nodes can access all DATA NODES but they will only look at related information.