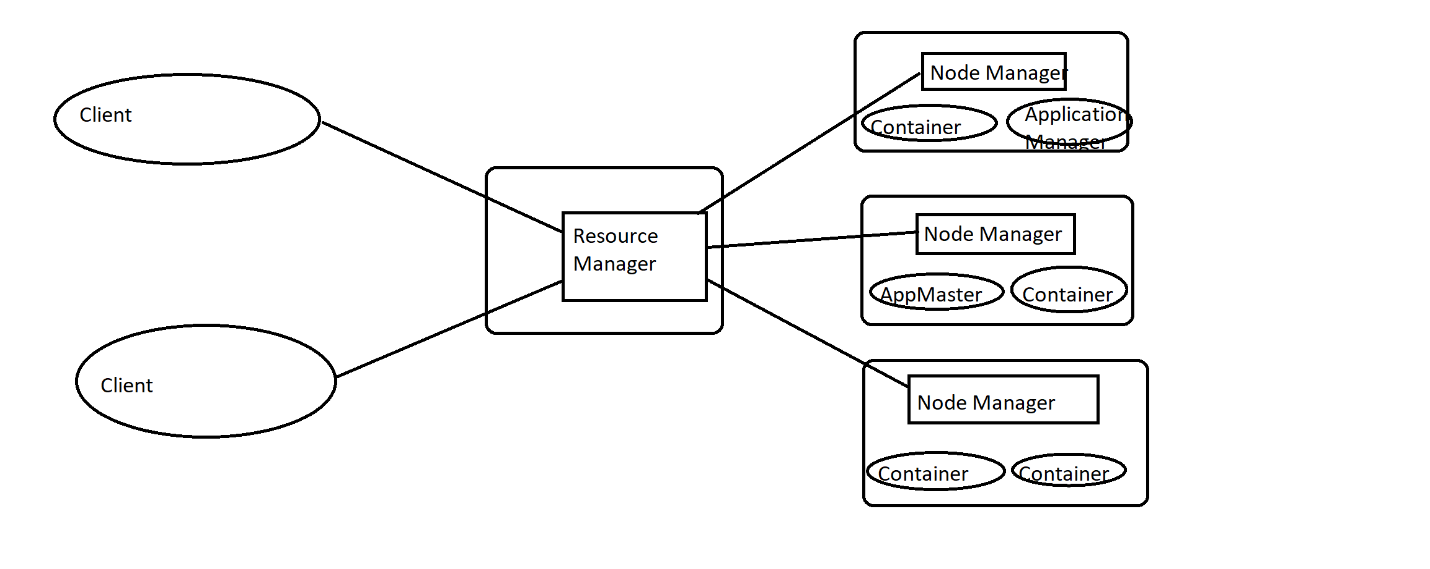
|  |  |  |
| --- | --- | --- |
| **UMBC – CSEE Department** |  | Dr. Waleed Youssef |
| **Data Science Program** |  | youssef1@umbc.edu |
| **Spring 2023** |  |  |
| **DATA 603 – Big Data Platforms** | | |
|  |  |  |
| **Homework #7 - YARN** | | |
|  |  |  |

**Questions:**

1. **[10 Points]** Describe the YARN Architecture, and discuss functions of its components?



**Client:**

Submits MapReduce jobs.

**Resource Manager:**

It is a global resource scheduler. It manages resources allocation in cluster.

**Application Master:**

Manages resource needs of individual application. Also runs application specific process (Map and Reduce) in those clusters. Manages application life cycle and task scheduling.

**Load Manager**:

It is a generalized task tracker.

**Node Manager:**

Provides computational resources in the form of containers and manages processes running in those containers.

**Container:**

Container executes an application specific process. It contains resources including RAM, CPU, Network, HDD, etc.

1. **[10 Points]** Describe Zookeeper and discuss its consistency guarantees?

Distributed applications require coordination. For which we can develop your own service or use robust coordination system like ZooKeeper.

Zookeeper is a distributed centralized coordination service that is opensource and ready for use.

It can be used to help you tackle many of the common challenges distributed application phase.

It is useful for:

**Maintaining configuration information:**

We can store configuration data in zookeeper and share that data across all nodes in your distributed system.

**Naming:**

It can be used as a neighboring service there by allowing one node to find a specific machine in a cluster of thousands of servers.

**Providing distributed synchronization:**

It provides building blocks for locks, barriers, and Queues.

**Providing group services:**

E.g.: Leader election.

Zookeeper provides the building blocks for all above scenarios and is distributed, reliable, fast and simple.

**Consistency Guarantees:**

It provides 6 consistency guarantees they are:

**Sequential Consistency:**

Updates from a client to the zookeeper service are applied in the order they are sent.

**Atomicity:**

Updates in zookeeper either succeeds or fail. Partial updates are not allowed.

**Single System Image**:

Client sees the same view of zookeeper service regardless of the server in the ensemble that it is connected to.

**Reliability:**

If an update succeeds in zookeeper, then it will persist and not be rolled back. The update will be overridden when another client performs a new update.

**Timeliness:**

Clients view of the system is guaranteed to be up to date within a certain time bound. Generally, within tens of seconds. If a client does not see system change within a time bound then the clients assume a service outage and will connect to a different server in the ensemble.

1. **[10 Points]** Discuss the different types of schedulers that Hadoop YARN can support?

Scheduler is one of the major components of resource manager. It is totally dedicated to scheduling the jobs, it cannot track the status of the application.

There are mainly 3 types of Schedulers in Hadoop:

* FIFO (First In First Out) Scheduler
* Capacity Scheduler
* Fair Scheduler

**First In First Out Scheduler**:

As the name suggests FIFO i.e. First In First Out the tasks are applications tht comes first will be served first. This is the default Scheduler. Here the tasks are placed in a queue and the tasks are performed in their submission order. Once the job is scheduled, no intervention is allowed. So high-priority process might have to wait for long time since the priority of task does not matter in this method.

**Capacity Scheduler:**

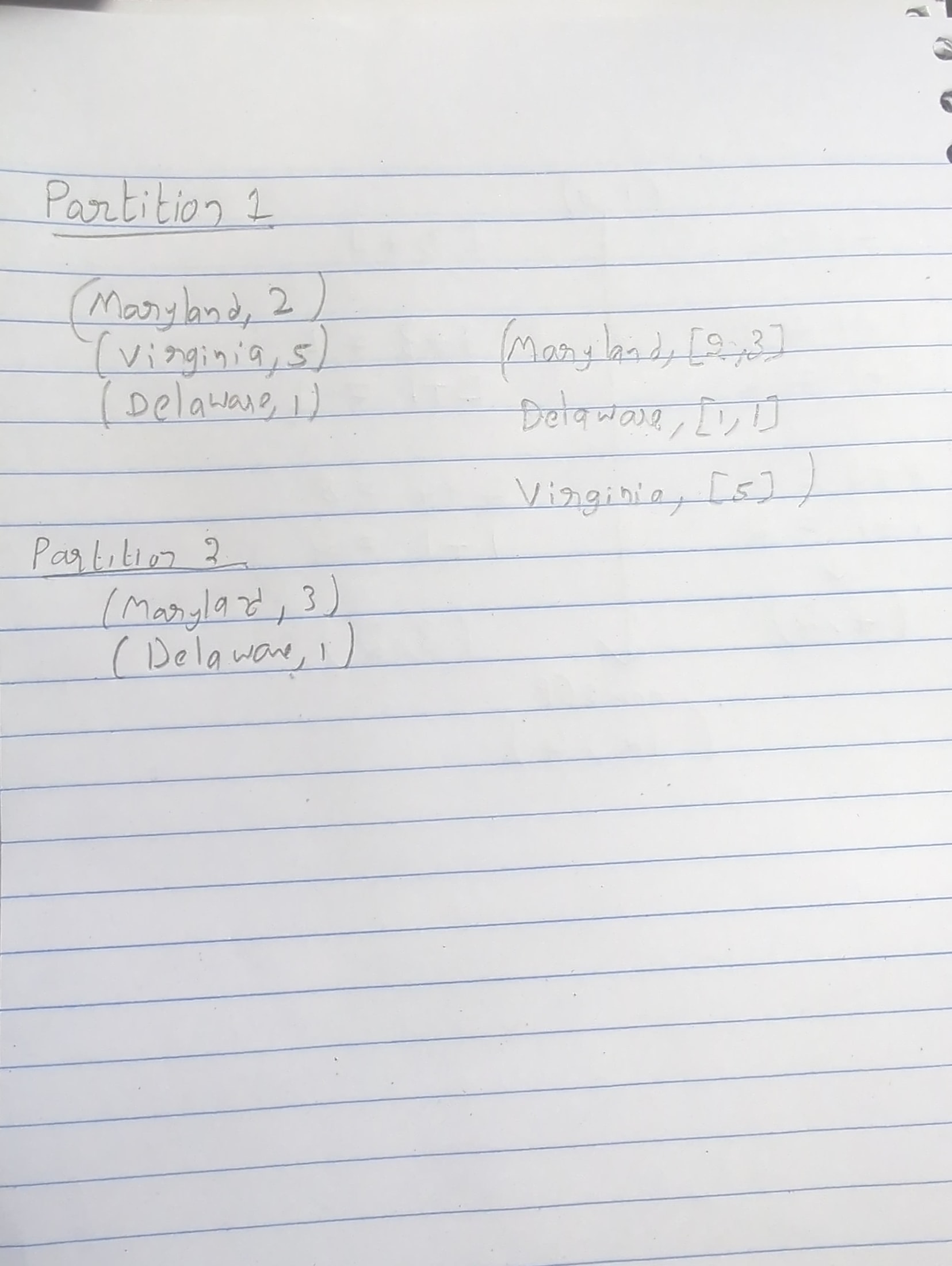
This enables multiple users to share resource of a large cluster. Here there are multiple job queues. Available resources are divided among these queues. Each queue can access the resources of other queue if they are not being used. But when a new task enters a queue, the running jobs in its own slot is replaced with new task of its own queue.

It also provides abstraction to ensure that no single user or application takes up an unfair number of resources.

**Fair Scheduler**:

It allows all applications get , on average, an equal share of resources over time. It evenly balances resources between all running tasks and jobs. It take decision based on the memory.

1. **[20 points]** Simulate the implementation of the “CombineByKey” for the data below using the data and the cluster shown below:
   * Partition 1 – ((Maryland, 2), (Virginia, 5), (Delaware, 1))
   * Partition 2 – ((Maryland, 3), (Delaware, 1))



A picture containing screenshot

Description generated with very high confidence

* + *Please note that no implementation required for this problem*