**YARN (Yet another resource negotiator)**

* In Hadoop1, the job execution was controlled by 2 processes.
  + Master: job tracker
  + Slave: task tracker

**Job Tracker:**

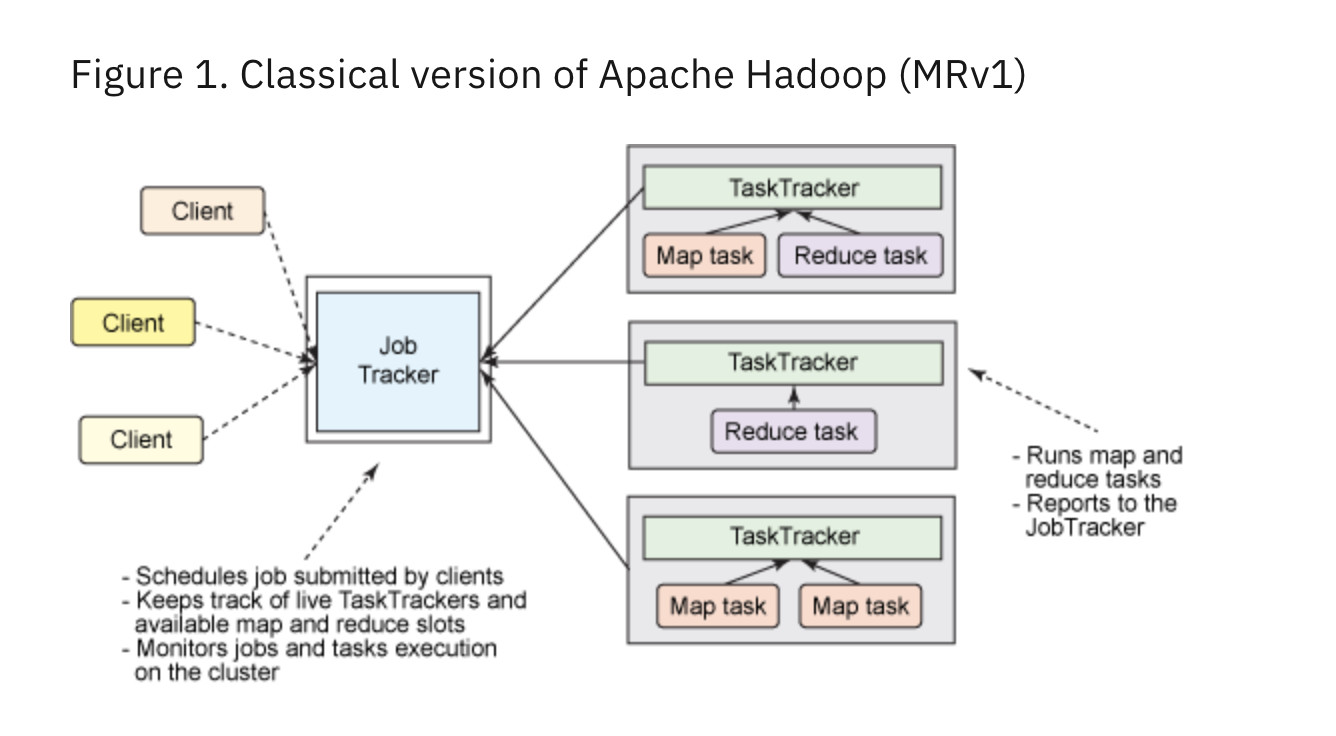
* Scheduling
  + Which job is more prioritize
  + Which resources need to be given to a job
* Monitoring
  + Tracking the progress of a job
  + If a task fails, it re-runs the task
  + If the task is slow then based of speculative execution start on another on another machine

**Task Tracker**

* This task tracker the tasks on each data node and informs the job tracker.

Table

Description automatically generated with low confidence

****

So, you will notice that the job tracker has to mange alone monitoring of

Thousand of task trackers, 100s of jobs, thousands of map and reduce tasks. Which make it overwhelmed when there is a big cluster.

**Limitations of MR1**

* + **Scalability :** When cluster size goes beyond 4K Datanodes then the job tracker used to become bottlenecks.
  + **Resource Utilization:** In MR1 there used to be fix numbers of map and reduce slots.

**Ex.** Lets say we have 100 map slots and 50 reduce slots. You want to execute a map reduce job which required 150 mappers. So, though we have 150 slots but we can run 100 at a time due to a fix size allotted to them.

* + Only map reduce jobs are supported.

**To resolve these limitations of MR1 YARN came in picture.**

YARN components

Resource Manger

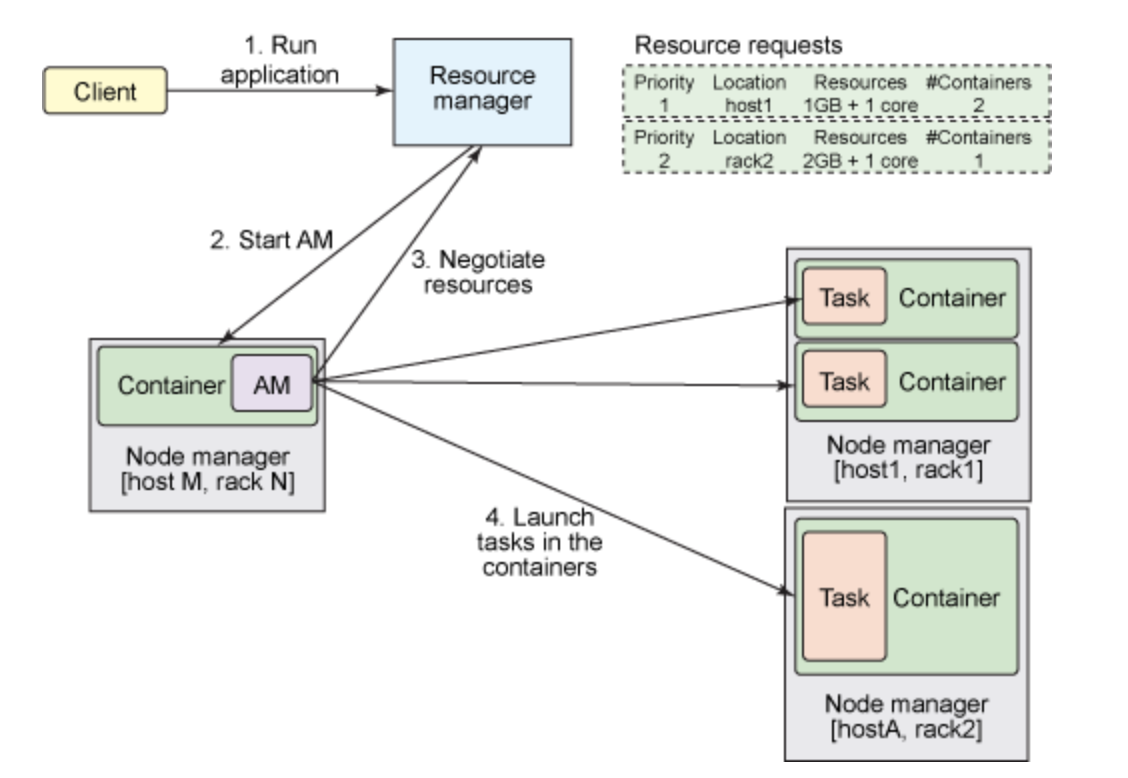
(Master)

Application Master

Node Manager

(Slave)

* + In updated MR architecture when YARN is introduced, resource manager (which a job tracker) does only scheduling. Monitoring part has taken from it.
  + Task tracker is replaced with Node managers (which does the same thing)
  + With YARN our resource manager creates a **‘container’** in one of the node manager. Once container is created then it will launch ‘**application master’** in that.
  + This application master will do monitoring.
  + This application ask the resources in the form of containers (2 gb + 1 core).
  + Resource manger allocates the resources in the form of containers, and it will send container id and the host name to the application master.



**How Limitations are handled in MR2**

* **Scalability** 
  + Scheduling – resource manager
  + Monitoring – App master
* In MR2 we can run other jobs also like Spark, Tez apart from map reduce.
* There is no longer fix map and reduce slots
* **Uberization:**
  + If there is a new job comes. And this job is a very small which does not require much resources in that case ‘Application master’ will not allocate new resources but use existing job resource to handle this job. This is call uberization.