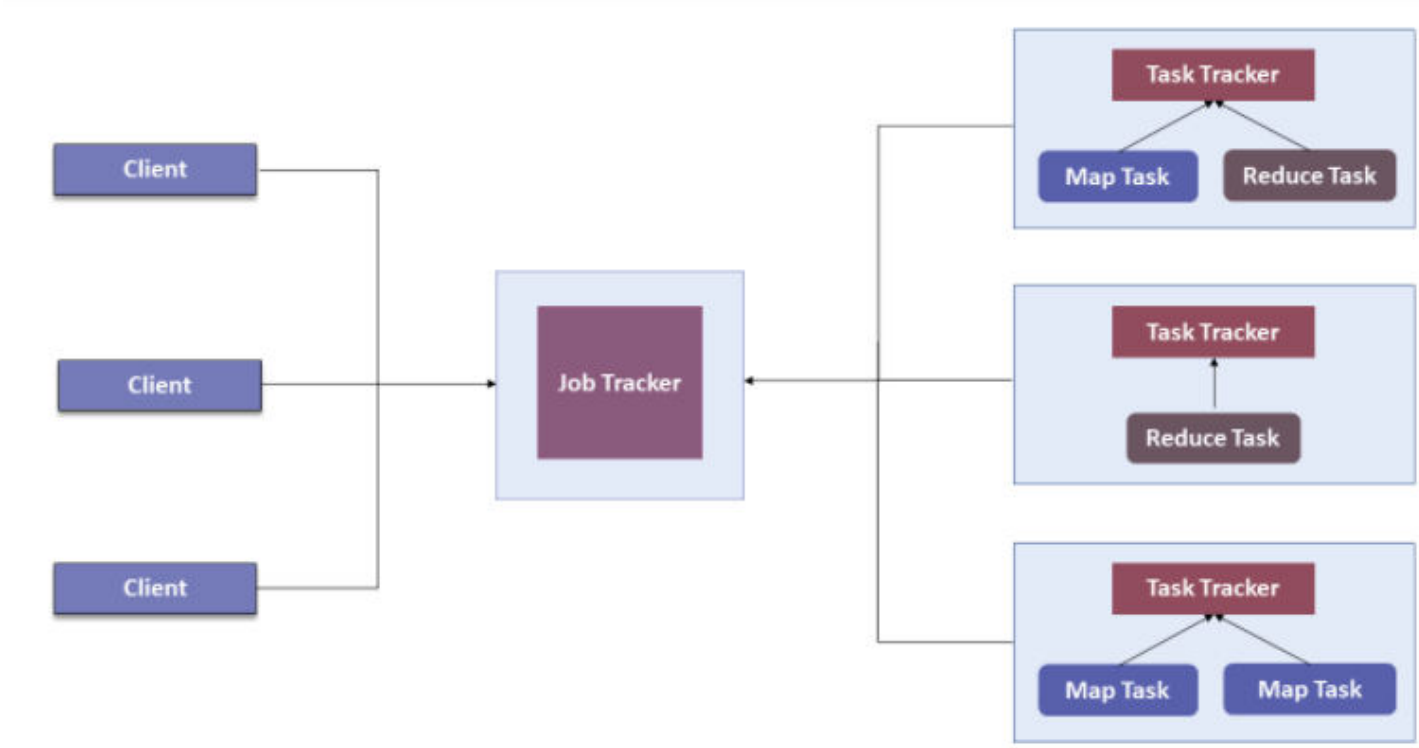
YARN:

Hadoop YARN knits the storage unit of Hadoop i.e. HDFS (Hadoop Distributed File System) with the various processing tools. For those of you who are completely new to this topic, YARN stands for "**Y**et **A**nother **R**esource **N**egotiator".

Why YARN?

In Hadoop version 1.0 which is also referred to as MRV1 (MapReduce Version 1), MapReduce performed both processing and resource management functions. It consisted of a Job Tracker which was the single master. The Job Tracker allocated the resources, performed scheduling and monitored the processing jobs. It assigned map and reduce tasks on a number of subordinate processes called the Task Trackers. The Task Trackers periodically reported their progress to the Job Tracker.



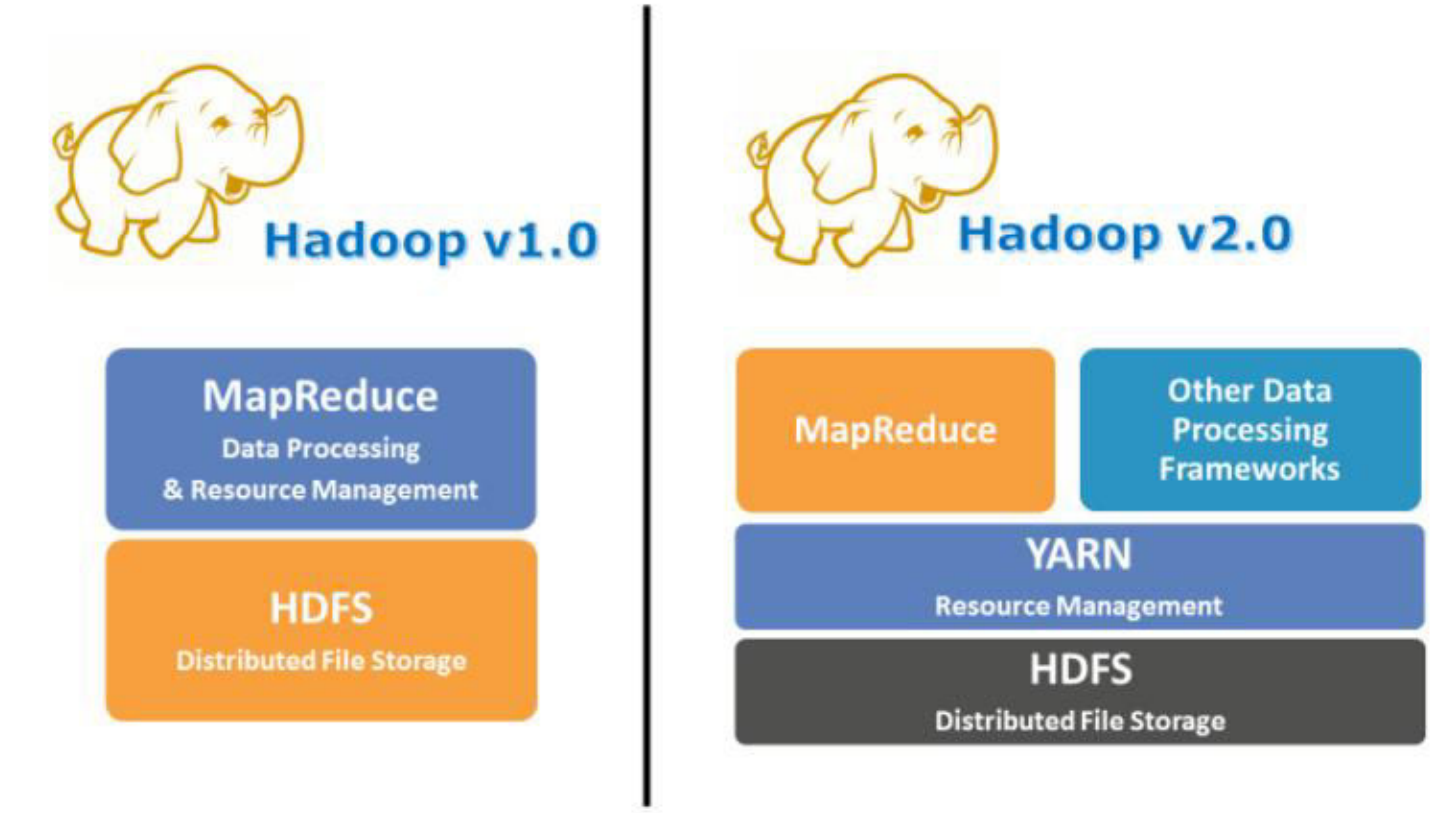
This design resulted in scalability bottleneck due to a single Job Tracker. IBM mentioned in its article that according to Yahoo!, the practical limits of such a design are reached with a cluster of 5000 nodes and 40,000 tasks running concurrently. Apart from this limitation, the utilization of computational resources is inefficient in MRV1. Also, the Hadoop framework became limited only to MapReduce processing paradigm.

To overcome all these issues, YARN was introduced in Hadoop version 2.0 in the year 2012 by Yahoo and Hortonworks. The basic idea behind YARN is to relieve MapReduce by taking over the responsibility of Resource Management and Job Scheduling. YARN started to give Hadoop the ability to run non-MapReduce jobs within the Hadoop framework.

With the introduction of YARN, the Hadoop ecosystem was completely revolutionalized. It became much more flexible, efficient and scalable. When Yahoo went live with YARN in the first quarter of 2013, it aided the company to shrink the size of its Hadoop cluster from 40,000 nodes to 32,000 nodes. But the number of jobs doubled to 26 million per month.

Introduction to Hadoop YARN:

Now that I have enlightened you with the need for YARN, let me introduce you to the core component of Hadoop v2.0, *YARN*. YARN allows different data processing methods like graph processing, interactive processing, stream processing as well as batch processing to run and process data stored in HDFS. Therefore YARN opens up Hadoop to other types of distributed applications beyond MapReduce.

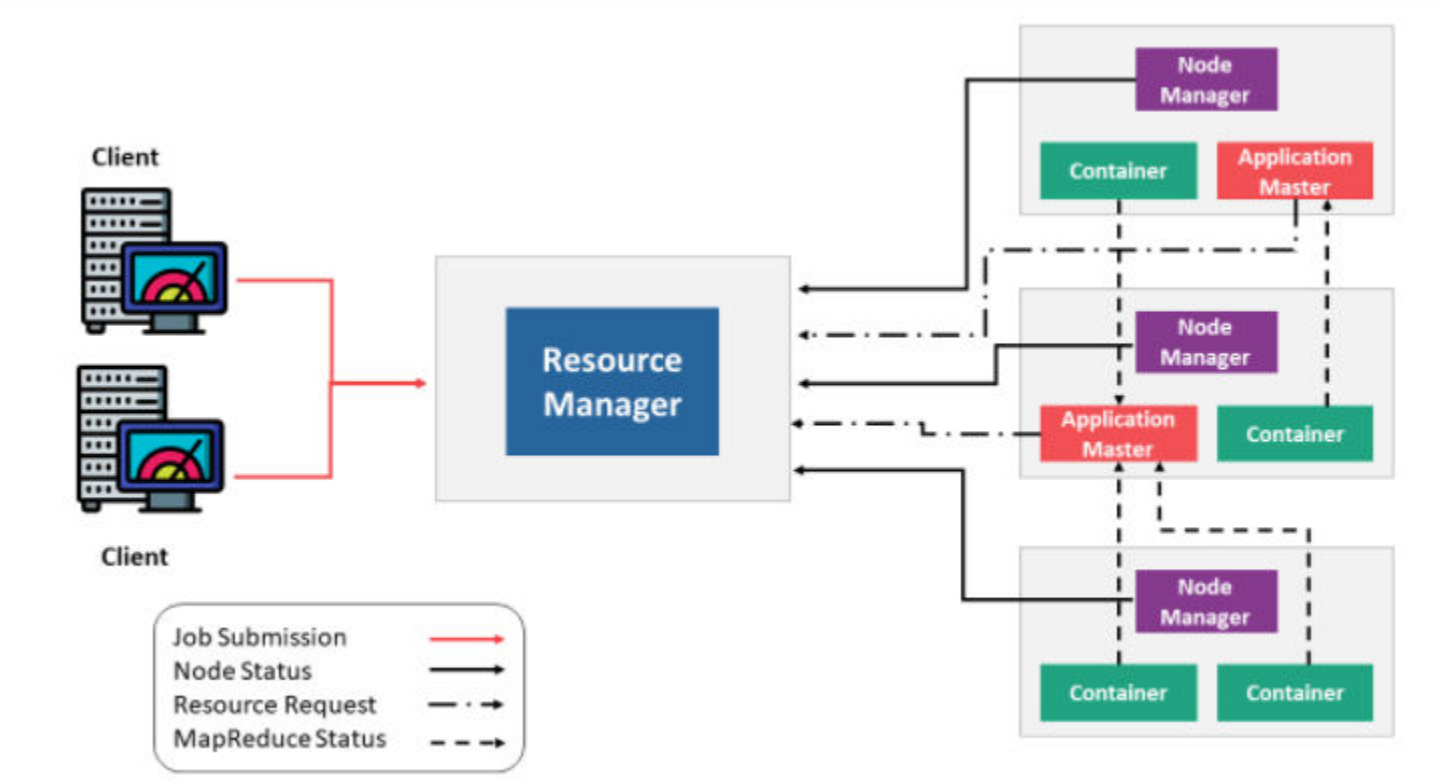


YARN enabled the users to perform operations as per requirement by using a variety of tools like Spark for real-time processing, **Hive** for SQL, **HBase** for NoSQL and others.
Apart from Resource Management, YARN also performs Job Scheduling. YARN performs all your processing activities by allocating resources and scheduling tasks. Apache Hadoop YARN Architecture consists of the following main components:

- **Resource Manager:** Runs on a master daemon and manages the resource allocation in the cluster.
- **Node Manager:** They run on the slave daemons and are responsible for the execution of a task on every single Data Node.

- **Application Master:** Manages the user job lifecycle and resource needs of individual applications. It works along with the Node Manager and monitors the execution of tasks.
- **Container:** Package of resources including RAM, CPU, Network, HDD etc on a single node.

Components of YARN:
You can consider YARN as the brain of your Hadoop Ecosystem. The image below represents the YARN Architecture.



The **first component** of YARN Architecture is,
Resource Manager:

- It is the ultimate authority in resource allocation.
- On receiving the processing requests, it passes parts of requests to corresponding node managers accordingly, where the actual processing takes place.
- It is the arbitrator of the cluster resources and decides the allocation of the available resources for competing applications.
- Optimizes the cluster utilization like keeping all resources in use all the time against various constraints such as capacity guarantees, fairness, and SLAs.
- It has two major components: **a) Scheduler b) Application Manager**

a) Scheduler

- The scheduler is responsible for allocating resources to the various running applications subject to constraints of capacities, queues etc.
- It is called a pure scheduler in ResourceManager, which means that it does not perform any monitoring or tracking of status for the applications.
- If there is an application failure or hardware failure, the Scheduler does not guarantee to restart the failed tasks.
- Performs scheduling based on the resource requirements of the applications.
- It has a pluggable policy plug-in, which is responsible for partitioning the cluster resources among the various applications. There are two such plug-ins: **Capacity Scheduler** and **Fair Scheduler**, which are currently used as Schedulers in ResourceManager.

b) Application Manager

- It is responsible for accepting job submissions.
- Negotiates the first container from the Resource Manager for executing the application specific Application Master.
- Manages running the Application Masters in a cluster and provides service for restarting the Application Master container on failure.

Coming to the **second component** which is:

Node Manager:

- It takes care of individual nodes in a Hadoop cluster and manages user jobs and workflow on the given node.
- It registers with the Resource Manager and sends heartbeats with the health status of the node.
- Its primary goal is to manage application containers assigned to it by the resource manager.
- It keeps up-to-date with the Resource Manager.
- Application Master requests the assigned container from the Node Manager by sending it a Container Launch Context(CLC) which includes everything the application needs in order to run. The Node Manager creates the requested container process and starts it.
- Monitors resource usage (memory, CPU) of individual containers.
- Performs Log management.
- It also kills the container as directed by the Resource Manager.

The **third component** of Apache Hadoop YARN is:

Application Master:

- An application is a single job submitted to the framework. Each such application has a unique Application Master associated with it which is a framework specific entity.
- It is the process that coordinates an application's execution in the cluster and also manages faults.

- Its task is to negotiate resources from the Resource Manager and work with the Node Manager to execute and monitor the component tasks.
- It is responsible for negotiating appropriate resource containers from the ResourceManager, tracking their status and monitoring progress.
- Once started, it periodically sends heartbeats to the Resource Manager to affirm its health and to update the record of its resource demands.

The **fourth component** is:

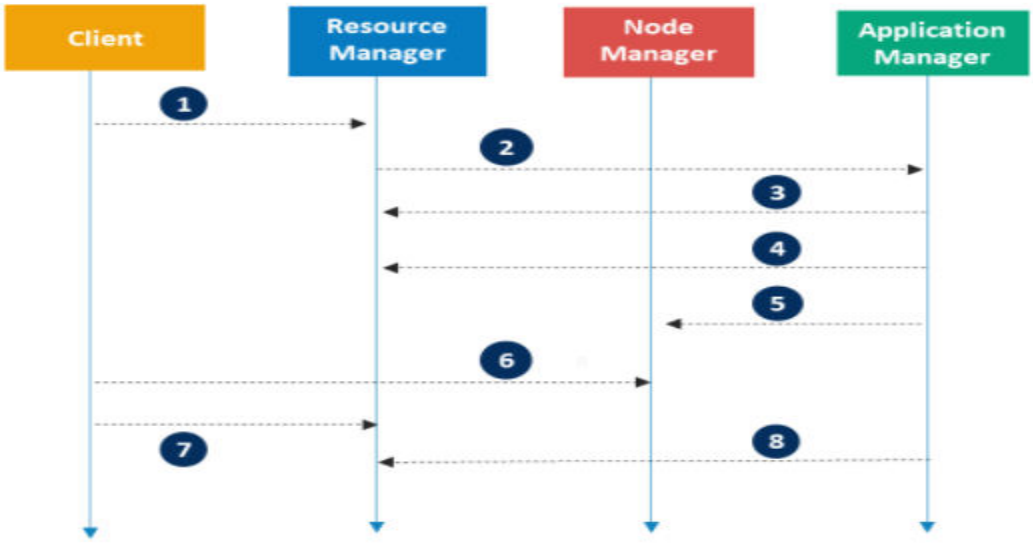
Container:

- It is a collection of physical resources such as RAM, CPU cores, and disks on a single node.
- YARN containers are managed by a container launch context which is container life-cycle (CLC). This record contains a map of environment variables, dependencies stored in a remotely accessible storage, security tokens, payload for Node Manager services and the command necessary to create the process.
- It grants rights to an application to use a specific amount of resources (memory, CPU etc.) on a specific host.

Application Workflow in Hadoop YARN:

Refer to the given image and see the following steps involved in Application workflow of Apache Hadoop YARN:

1) Client submits an application
2) Resource Manager allocates a container to start Application Manager
3) Application Manager registers with Resource Manager
4) Application Manager asks containers from Resource Manager
5) Application Manager notifies Node Manager to launch containers
6) Application code is executed in the container
7) Client contacts Resource Manager/Application Manager to monitor application's status
8) Application Manager unregisters with Resource Manager



Application Submission in YARN:

Refer to the image and have a look at the steps involved in application submission of Hadoop YARN:

1) Submit the job
2) Get Application ID
3) Application Submission Context
4 a) Start Container Launch
  b) Launch Application Master
5) Allocate Resources
6 a) Container
  b) Launch
7) Execute