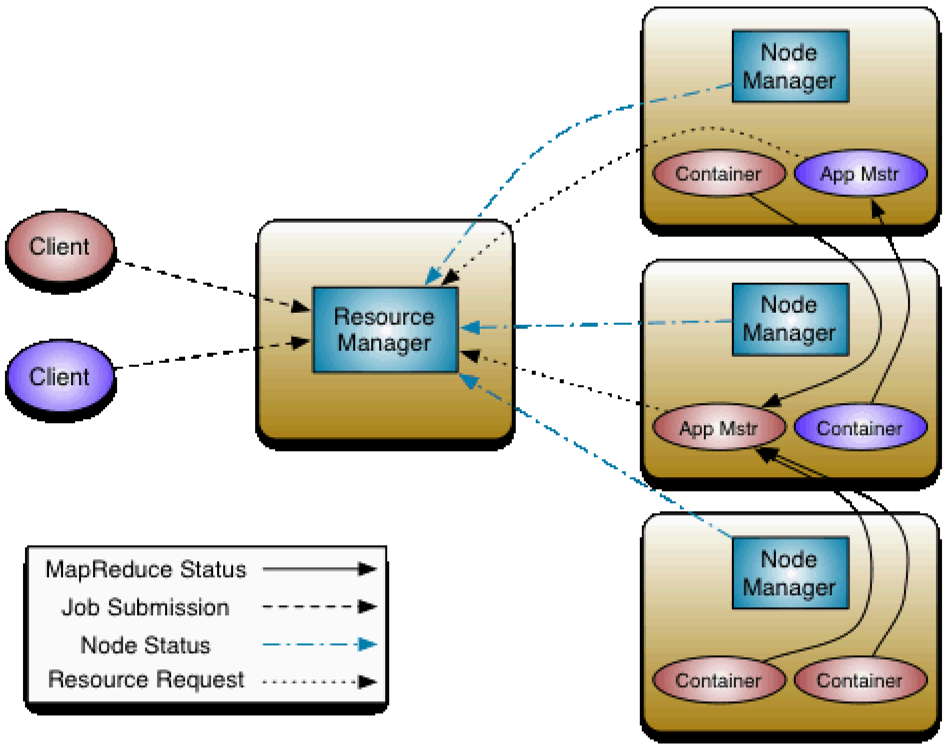
Apache Hadoop YARN

The fundamental idea of YARN is to split up the functionalities of resource management and job scheduling/monitoring into separate daemons. The idea is to have a global ResourceManager (*RM*) and per-application ApplicationMaster (*AM*). An application is either a single job or a DAG of jobs.

The ResourceManager and the NodeManager form the data-computation framework. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The NodeManager is the per-machine framework agent who is responsible for containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager/Scheduler.

The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks.



The ResourceManager has two main components: Scheduler and ApplicationsManager.

The Scheduler is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues etc. The Scheduler is pure scheduler in the sense that it performs no monitoring or tracking of status for the application. Also, it offers no guarantees about restarting failed tasks either due to application failure or hardware failures. The Scheduler performs its scheduling function based on the resource requirements of the applications; it does so based on the abstract notion of a resource *Container* which incorporates elements such as memory, cpu, disk, network etc.

The Scheduler has a pluggable policy which is responsible for partitioning the cluster resources among the various queues, applications etc. The current schedulers such as the CapacityScheduler and the FairScheduler would be some examples of plug-ins.

The ApplicationsManager is responsible for accepting job-submissions, negotiating the first container for executing the application specific ApplicationMaster and provides the service for restarting the ApplicationMaster container on failure. The per-application ApplicationMaster has the responsibility of negotiating appropriate resource containers from the Scheduler, tracking their status and monitoring for progress.

MapReduce in hadoop-2.x maintains **API compatibility** with previous stable release (hadoop-1.x). This means that all MapReduce jobs should still run unchanged on top of YARN with just a recompile.

YARN supports the notion of **resource reservation** via the ReservationSystem, a component that allows users to specify a profile of resources over-time and temporal constraints (e.g., deadlines), and reserve resources to ensure the predictable execution of important jobs.The *ReservationSystem* tracks resources over-time, performs admission control for reservations, and dynamically instruct the underlying scheduler to ensure that the reservation is fulfilled.

In order to scale YARN beyond few thousands nodes, YARN supports the notion of **Federation** via the YARN Federation feature. Federation allows to transparently wire together multiple yarn (sub-)clusters, and make them appear as a single massive cluster. This can be used to achieve larger scale, and/or to allow multiple independent clusters to be used together for very large jobs, or for tenants who have capacity across all of them.