YARN assigns the resource management and job management duties handled by the Hadoop 1 Job Tracker as follows:

* Resource Manager - manages cluster resources, and tracks resource usage and node health
* Application Master - a framework-specific process that negotiates resources for a single application (a single job or a directed acyclic graph of jobs) which runs in the first container (see below) allocated for the application

The Resource Manager allocates resources among all the applications running the cluster. The Resource Manager includes a pluggable scheduler, which is responsible for allocating resources according to the resource requirements of the running applications. Current MapReduce schedulers, including the Capacity Scheduler and the Fair Scheduler, can be plugged into the YARN scheduler directly.

Each application runs an Application Master to negotiate resources from the Resource Manager. The Application Master works with the Node Managers to execute and monitor tasks.

The duties of the Task Tracker are divided as follows:

* Node Manager - One instance runs on each node, to manage that node's resources.
* Container - An abstraction representing a unit of resources on a node. The NodeManager provides containers to an application.

The Resource Manager and the Node Manager provide the system for distributed management of applications and resources.

**Resource Manager**

The Resource Manager is mainly concerned with arbitrating available resources in the cluster among competing applications, with the goal of maximum cluster utilization. The Resource Manager includes a pluggable scheduler called the Yarn Scheduler, which allows different policies for managing constraints such as capacity, fairness, and service level agreements.

The Resource manager manages resources as follows:

* Each Node Manager takes instructions from the Resource Manager, reporting and handling containers on a single node
* Each Application Master requests resources from the Resource Manager, then works with containers provided by Node Managers

The Resource Manager communicates with application clients via an interface called the Client Service, through which a client can submit or terminate an application and gain information about the scheduling queue or cluster statistics. Administrative requests are served by a separate interface called the Admin Service, through which operators can get updated information about cluster operation. Behind the scenes, the Resource Tracker Service receives node heartbeats from the Node Manager to track new or decommissioned nodes. The NMLiveliness Monitor and Nodes List Manager keep an updated status of which nodes are healthy so that the scheduler and the Resource Tracker Service can allocate work appropriately.

A component called the Application Master Service manages Application Masters on all nodes, keeping the scheduler informed. A component called the AMLiveliness Monitor keeps a list of Application Masters and their last heartbeat times, in order to let the Resource Manager know what applications are healthy on the cluster. Any Application Master that does not heartbeat within a certain interval is marked as dead and re-scheduled to run on a new container.

At the core of the Resource Manager is an interface called the Applications Manager, which maintains a list of applications that have been submitted, are running, or are completed. The Applications Manager accepts job submissions, negotiates the first container for an application (in which the Application Master will run) and restarts the Application Master if it fails.

The Resource Manager and Node Managers communicate via heartbeats.

**Application Master**

The Application Master is an instance of a framework-specific library that negotiates resources from the Resource Manager and works with the Node Manager to execute and monitor the granted resources (bundled as containers) for a given application. An application can be a process or set of processes, a service, or a description of work.

The Application Master is run in a container like any other application. The Applications Manager, part of the Resource Manager, negotiates for the container in which an application’s Application Master runs when the application is scheduled by the Yarn Scheduler.

While an application is running, the Application Master manages the following:

* Application lifecycle
* Dynamic adjustments to resource consumption
* Execution flow
* Faults
* Providing status and metrics

The Application Master is architected to support a specific framework, and can be written in any language since its communication with the Node Managers and the Resource Manager is accomplished using extensible communication protocols. The Application Master can be customized to extend the framework or run any other code. For this reason, the Application Master is not considered trustworthy, and is not run as a trusted service.

An Application Master typically requests resources on multiple nodes to complete a job by sending the Resource Manager requests that include locality preferences and attributes of the containers. When the Resource Manager is able to allocate a resource to the Application Master, it generates a lease that the Application Master pulls on a subsequent heartbeat. A security token associated with the lease guarantees its authenticity when the Application Manager presents the lease to the Node Manager to gain access to the container.

The Application Master heartbeats to the Resource Manager to communicate its changing resource needs, and to let the Resource Manager know it is still alive. In response, the Resource Manager can return a lease on additional containers on other nodes, or cancel the lease on some containers. The Application Master can then adjust its execution strategy to fit the increase or decrease in available resources. When cluster resources become scarce, the Resource Manager can also request that the Application Master relinquish some resources. The Application Master can move work to other running containers in order to give up resources gracefully.

**Node Manager**

The Node Manager runs on each node and manages the following:

* Container lifecycle management
* Container dependencies
* Container leases
* Node and container resource usage
* Node health
* Log management
* Reporting node and container status to the Resource Manager.

When a container is leased to an application, the Node Manager sets up the container’s environment, including the resource constraints specified in the lease and any dependencies such as data or executable files. When instructed by the Resource Manager or the appropriate Application Master, the Node Manager kills containers. A container can be killed when the Resource Manager reports completion of the application to which it was leased, when the scheduler needs the container for another application, when an Application Master requests that the container be killed, or when the Node Manager detects that it has exceeded the restraints of its lease. When a container is killed, all its resources are cleaned up, including memory and running tasks. However, a process can mark some output to be preserved until the application itself exits, in order to preserve data beyond the life of the container.

The Node Manager monitors the health of the node, reporting to the Resource Manager when a hardware or software issue occurs so that the scheduler can divert resource allocations to healthy nodes until the issue is resolved.

The Node Manager also offers a number of services to containers running on the node, for example a log aggregation service. The administrator can configure the Node Manager with additional pluggable services.

The core of the Node Manager is the Container Manager, which maintains a pool of threads in a component called the Containers Launcher for launching containers. A component called the Container Token Secret Manager authenticates incoming requests including container launch requests to ensure that all operations are properly authorized by the Resource Manager. Once a container is launched, a component called the Containers Manager monitors its resource usage in order to prevent runaway containers from affecting other tenants on the cluster.