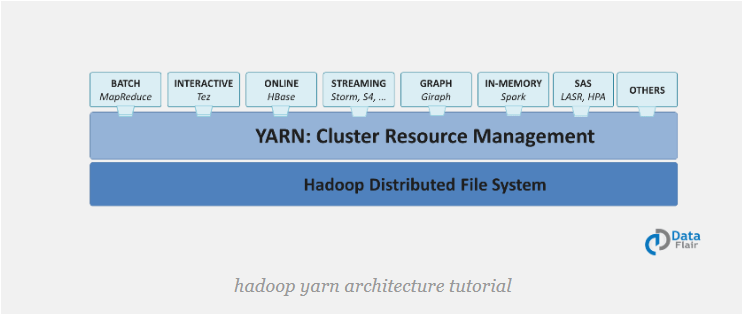
1. **What is YARN (Yet Another Resource Negotiator)**

Apache Yarn – “Yet Another Resource Negotiator” is the resource management layer of Hadoop along with resource management it also does job scheduling.



Even though YARN is part of Hadoop framework it is not only restricted for Mapreduce data processing programming model.

It allows other data processing systems like spark. So other frameworks can also run in the same hardware where Hadoop is deployed.

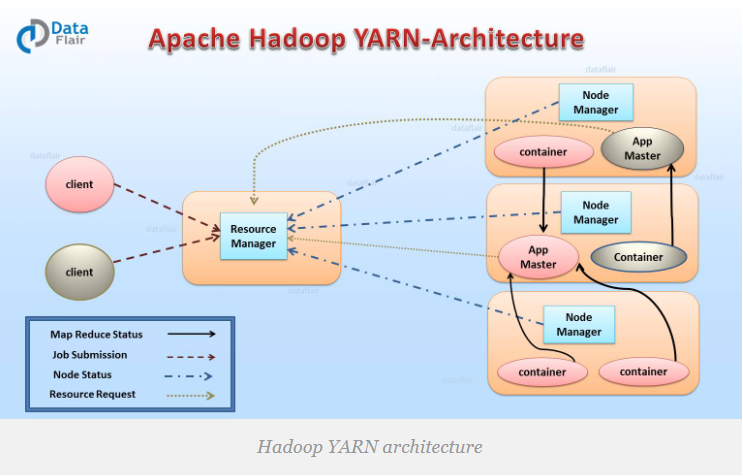
1. **When was the YARN introduced?**

Yarn was introduced in Hadoop 2.x

1. **What are the different data processing engine supported by YARN?**

Batch processing, Stream processing, graph processing and interactive processing engines are supported to run and process the data stored in HDFS.

1. **Explain the architecture of YARN? IMP**



Apache Yarn Framework consists of a master daemon known as “Resource Manager”, slave daemon called node manager (one per slave node) and Application Master (one per application).

1. **Resource Manager(RM):**

Resource Manager decides and assigns resources (CPU and Memory) to all competing application based on availability.

Can refer below link for detailed understanding of RM

<https://data-flair.training/blogs/hadoop-yarn-resource-manager/>

Resource Manager has two Main components

**Scheduler**: The scheduler is responsible for allocating the resources to the running application.

The scheduler is pure scheduler it means that it performs no monitoring no tracking for the application and even doesn’t guarantees about restarting failed tasks either due to application failure or hardware failures.

**Application Manager:** It manages running Application Masters in the cluster, i.e., it is responsible for starting application masters and for monitoring and restarting them on different nodes in case of failures.

1. **Node Manager (NM):**

Node Manager is the slave daemon of Yarn. It is responsible for monitoring containers, their resource usage and reporting the same to the Resource Manager.

Manage the user process on that machine.

Yarn NodeManager also tracks the health of the node on which it is running.

Node Manager is designed such that we can plug in auxiliary (Helping) services to it. These services will be application specific services which can be configured and loaded to NM during start up.

For example shuffle is typical auxiliary (helping) service provided by Node Manager for Map reduce applications on YARN.

1. **Application master: (AM)**

One application master runs per application. It manages the application life cycle.

The Application Master acquires containers from the RM’s Scheduler (negotiates) and then contact the corresponding NMs to start the application’s individual tasks**.**

1. **What is container in YARN?**
2. **Types of Resource manager restart?**

Resource manageris central authorityformanaging resources and schedules applications running on YARN. So Resource manager is potentially single point of contact for apache yarn cluster.

There are 2 types of restart for Resource Manager

1. **Non-work-preserving RM restart:**

In this restart RM persist application/attempt state to storage (This storage is pluggable).

When RM is restarted it will reload the same information from the star-store and re-kicks the previously running application by itself and client does not need to resubmit the application. (State of previously running application will not be restored)

When RM is down Node Manager (NM) and client will keep polling RM until RM comes up. Once RM is up it will send re-sync command to all NM and AM it was talking before via hear beats. Node Manager will kill all containers and re registers with RM.

1. **Work-preserving RM restart:**

In this case when RM restarts it tries to reconstruct original running state by combining *“container status from Node manager” and “container requests from the application master”.*

In this case previously running applications will not be stopped when RM is restarted, so applications will not lose its processed data.

Node manager will not kill containers when it re syncs with restarted Resource Manager, instead it continues to manage containers and send their status

1. **Explain YARN Resource Manager (RM) high availability?**

Before to Hadoop v2.4, the master (RM) was the SPOF (single point of failure).

So to avoid this High Availability **(HA)** feature is introduced by adding Active/Standby Resource Manager pair but this adds redundancy.

Resource Manager HA is realized through an Active/Standby architecture – at any point in time, one in the masters is Active, and other Resource Managers are in Standby mode, they are waiting to take over when anything happens to the Active.

Admin can identify which is active and which is stand by via command line commands or by using integrated failover-controller when automatic failover is enabled.

There are 2 types of HA configurations:

1. **Manual transition and failover**

In this case admin has to manually transit one of the resource manager to active state, when active resource manager is fails another resource manager is automatically selected as active and active resource manager will be made as stand by.

1. **Automatic Failover:**

In this case **ActiveStandbyElector** (Zookeeper based) is embedded to decide which resource manager should be active. So when active fails another resource manager is automatically selected as active

Note: There is no need to run separate zookeeper demon because ActiveStandbyElector embedded in Resource Managers acts as a failure detector and a leader elector instead of a separate ZKFC daemon.