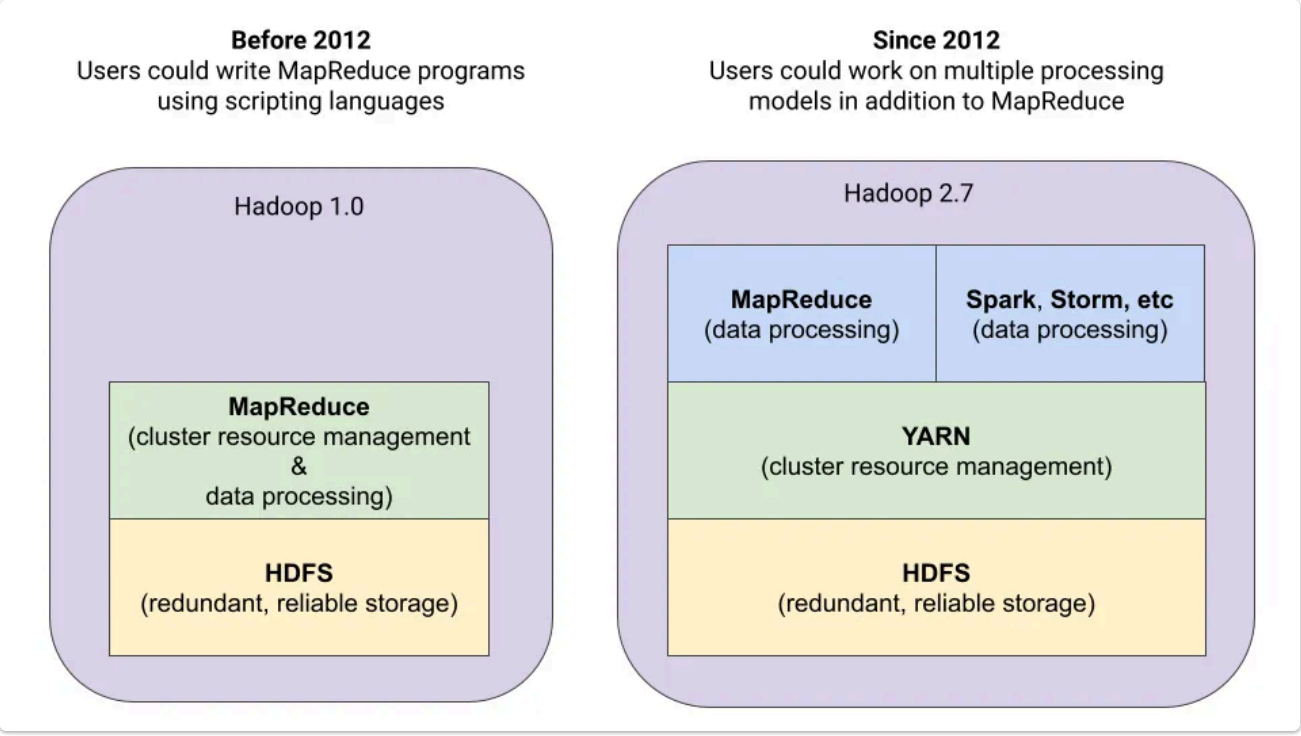
04 Yarn Architecture - KirkYagami

YARN (Yet Another Resource Negotiator)

1. Functions as the cluster resource management layer, responsible for managing and allocating resources such as CPU, memory, and storage for distributed applications running on a Hadoop cluster.
2. Global resource Manager: Decouples the resource management and Data Processing, thus supporting data processing applications such as MapReduce, Spark, Storm Tez etc.



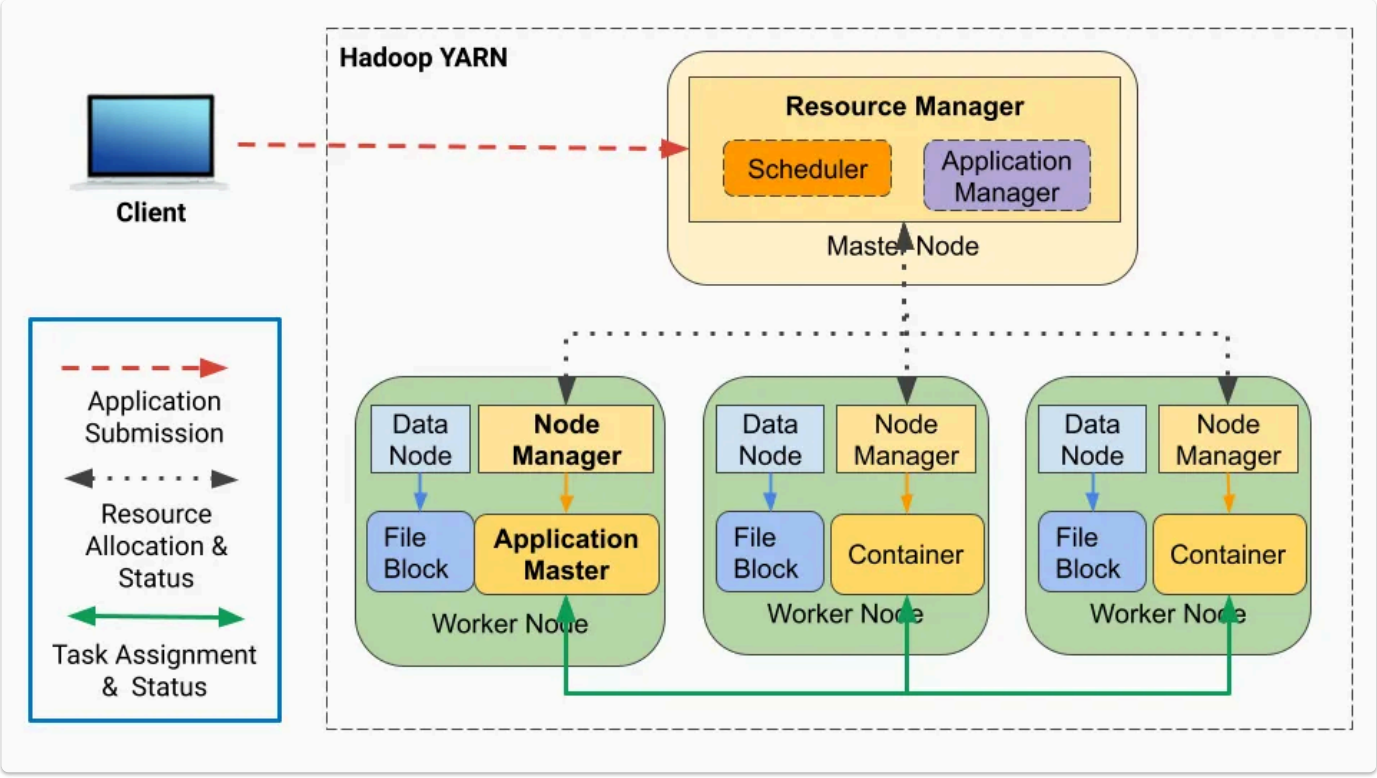
Hadoop 1 : MR (resource manager and processing) was tightly coupled with HDFS, limited to Run MR jobs only.

Benefits of YARN

1. Centralized Resource Management:
2. Flexibility
3. Better Cluster Utilization
4. Cost-Effective
5. Reduced Data Movement

YARN Architecture and its Components

1. Resource Manager (RM)
2. Application Master (AM)
3. Node Managers (NM)



1. Resource Manager (RM):

A Java process running on the master node.
Responsible for managing and allocating resources such as CPU, memory, and disk across the Hadoop cluster based on the needs of various jobs.

- **Cluster resource tracking:** It maintains a global view of the cluster and tracks the available resources on each node.
- **Cluster health monitoring:** It monitors the health of nodes in the cluster and manages the failover of resources in case of node failures.
- **Cluster resource allocation:** It receives resource requests from application masters and allocates the necessary resources to run the application.

Scheduler:

Scheduling and mediating available resources in the cluster among submitted applications, in accordance with a defined policy.

1. FIFO Scheduler

For smaller clusters and simple workloads.

2.Capacity Scheduler

It is a scheduler that divides cluster resources into multiple queues, each with its own reserved resources while able to dynamically utilize unused resources from other queues. It is suitable for large-scale, multi-user environments.

3. Fair Scheduler

Application Manager:

An interface that maintains a list of applications that are submitted, running, or completed. It is responsible for:

- **Handling job submission:** Accepting job submissions to YARN,
- **Negotiating resources for the ApplicationMaster:** Negotiating the first container for executing the application-specific application master, and
- **Managing failover of the ApplicationMaster:** Restarting the application master container on failure.

2. Application Master (AM)

ApplicationMaster is a process that runs the main function/entry point of an application, such as the Spark driver. It has several responsibilities, including:

- **Requesting resources:** Negotiating with the Resource Manager to obtain resources for launching containers to execute tasks.
- **Running the Master/Driver program:** It runs the Master/Driver program, such as Spark Driver, which devises the job execution plan, assigns tasks in the allocated containers, tracks task execution status, monitors progress, and handles task failures.

3. Node Manager (NM)

NodeManagers are Java processes that run on slave/worker nodes.
Responsibilities:

- **Reporting node health:** Each Node Manager announces itself to the ResourceManager and periodically sends a heartbeat to provide node status and information, including memory and virtual cores. In case of failure, the Node

Manager reports any issues to the Resource Manager, diverting resource allocations to healthy nodes.

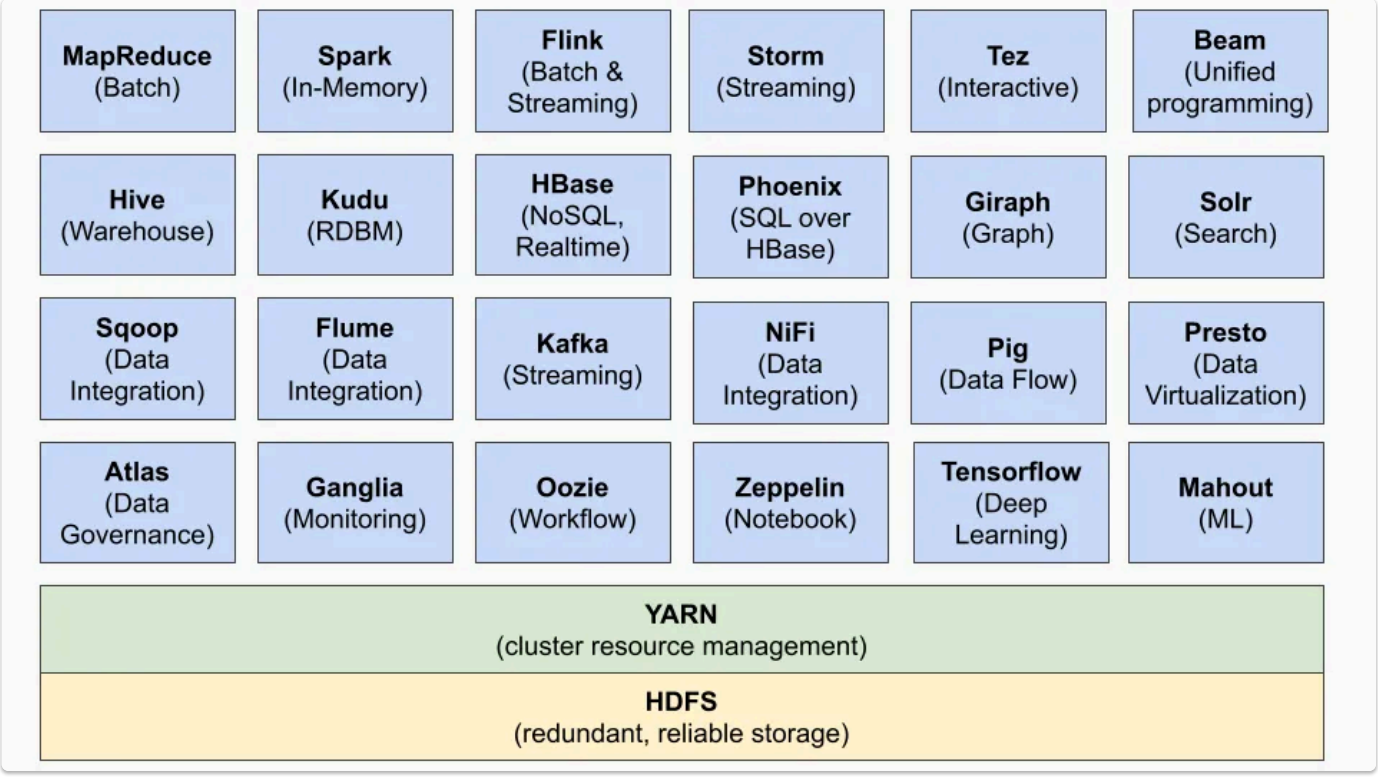
- **Launching Containers:** Node Managers take instructions from the ResourceManager, launch containers on their nodes, and set up the container environment with the specified resource constraints.
- **Container Management:** Node Managers manage container life cycle, dependencies, leases, resource usage, and log management.

Container

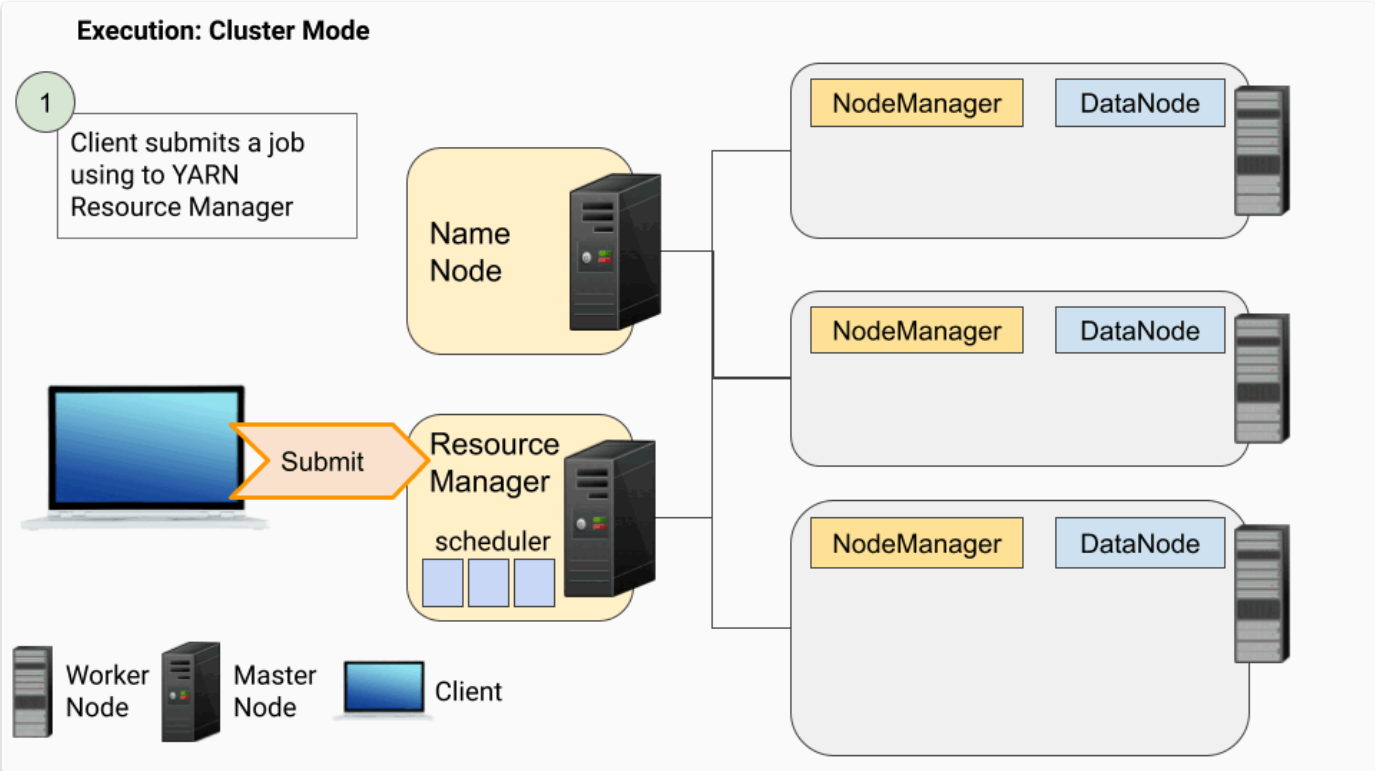
A container is a unit of resource allocation, an abstraction representing on a specific worker node in a Hadoop cluster.
Containers execute tasks of MapReduce or Spark.
Each container has a specific amount of resources allocated to it, such as CPU, memory, and disk space, allowing the task to run in a controlled and isolated environment.

- **ResourceManager:** The ResourceManager in YARN is responsible for *allocating containers to Application Masters* based on their resource requests. It *provides the container launch context (CLC)* that includes environment variables, dependencies, security tokens, and commands to create the application's launch process.
- **NodeManager:** The NodeManager in YARN is responsible for *launching the containers* with specified resource constraints (CLC).
- **ApplicationMasters:** The Application Masters *manage the execution of tasks within these containers*, monitor their progress, and handle any task failures or reassignments.

Applications that run on YARN



How Applications run on YARN



- **Step 1:** A client submits a job using "spark submit" to the YARN Resource Manager.
- **Step 2:** The job enters a scheduler queue in the ResourceManager, waiting to be executed.

- **Step 3:** When it is time for the job to be executed, the ResourceManager finds a NodeManager capable of launching a container to run the ApplicationMaster.
- **Step 4:** The ApplicationMaster launches the Driver Program (the entry point of the program that creates the SparkSession/SparkContext).
- **Step 5:** The ApplicationMaster/Spark calculates the required resources (CPU, RAM, number of executors) for the job and sends a request to the Resource Manager to launch the executors.

> *The ApplicationMaster communicates with the NameNode to determine the file (block) locations within the cluster using the HDFS protocol.*

- **Step 6:** The Driver Program assigns tasks to the executor containers and keeps track of the task status.
- **Step 7:** The executor containers execute the tasks and return the results to the Driver Program. The Driver Program aggregates the results and produces the final output.