Yarn Tutorial

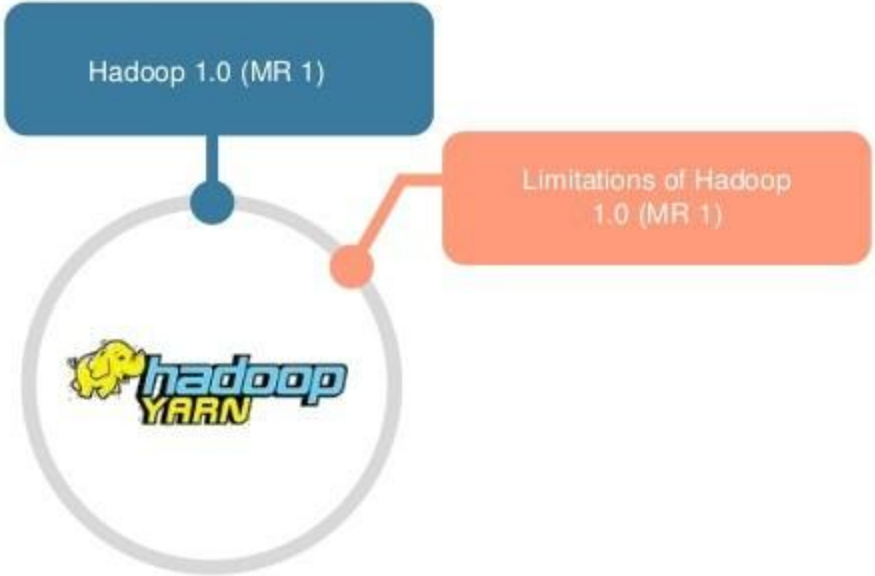
hadoop YARN

simplilearn

What's in it for you?

Hadoop 1.0 (MR 1)

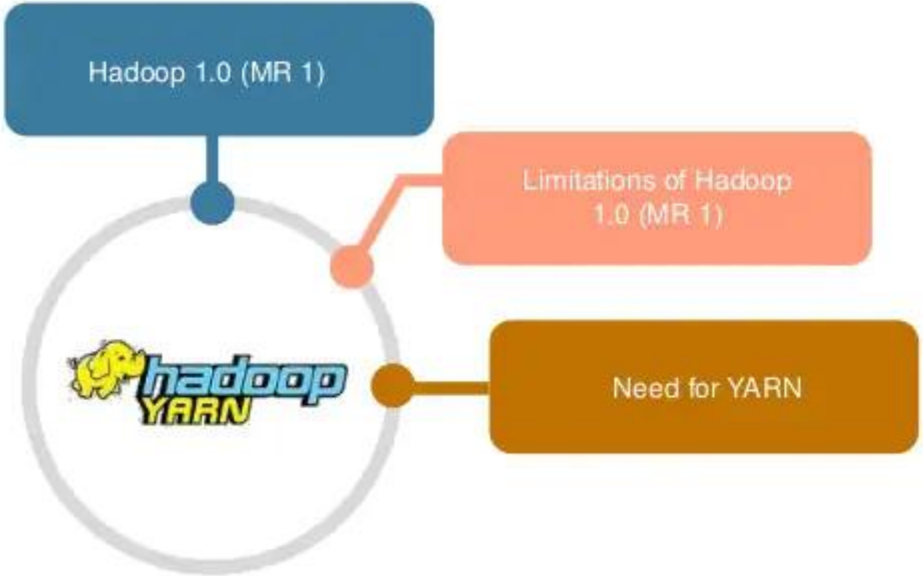
What’s in it for you?



Hadoop 1.0 (MR 1)

Limitations of Hadoop 1.0 (MR 1)

What's in it for you?

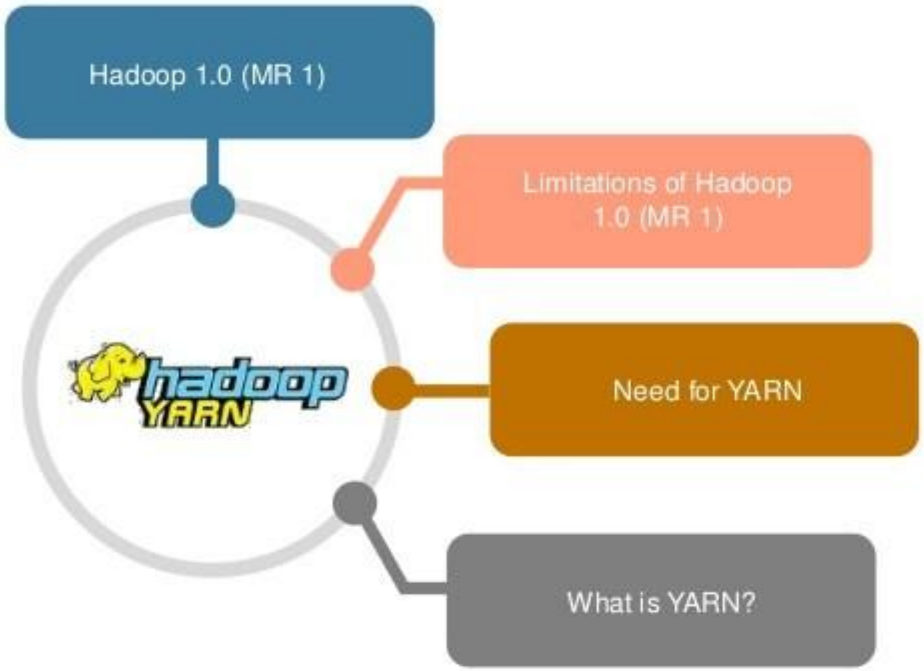


Hadoop 1.0 (MR 1)

Limitations of Hadoop 1.0 (MR 1)

Need for YARN

What’s in it for you?



Hadoop 1.0 (MR 1)

Limitations of Hadoop 1.0 (MR 1)

Need for YARN

What is YARN?

What’s in it for you?

hadoop
YARN

Hadoop 1.0 (MR 1)

Limitations of Hadoop 1.0 (MR 1)

Need for YARN

What is YARN?

Workloads running on YARN

What’s in it for you?

hadoop
YARN

Hadoop 1.0 (MR 1)

Limitations of Hadoop 1.0 (MR 1)

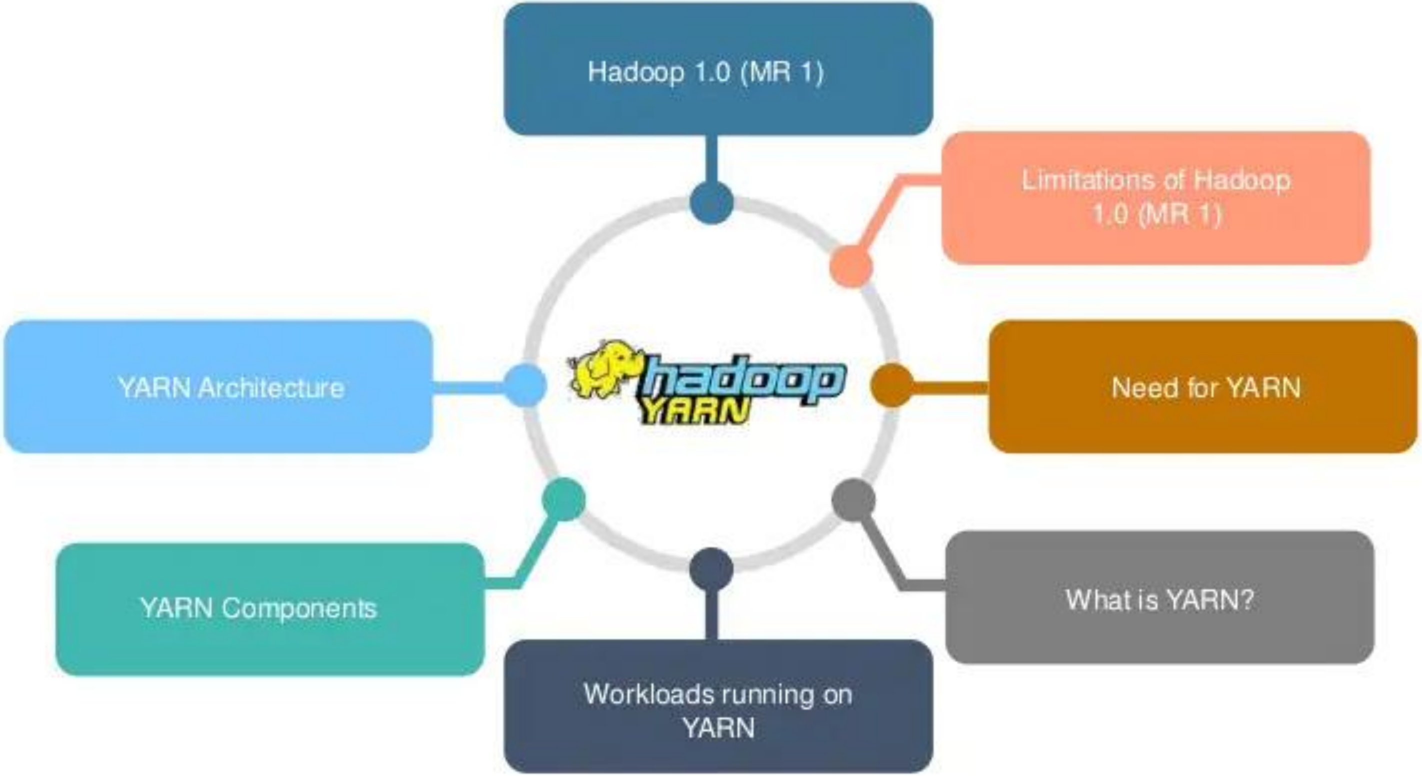
Need for YARN

What is YARN?

Workloads running on YARN

YARN Components

What's in it for you?



Hadoop 1.0 (MR 1)

Limitations of Hadoop 1.0 (MR 1)

Need for YARN

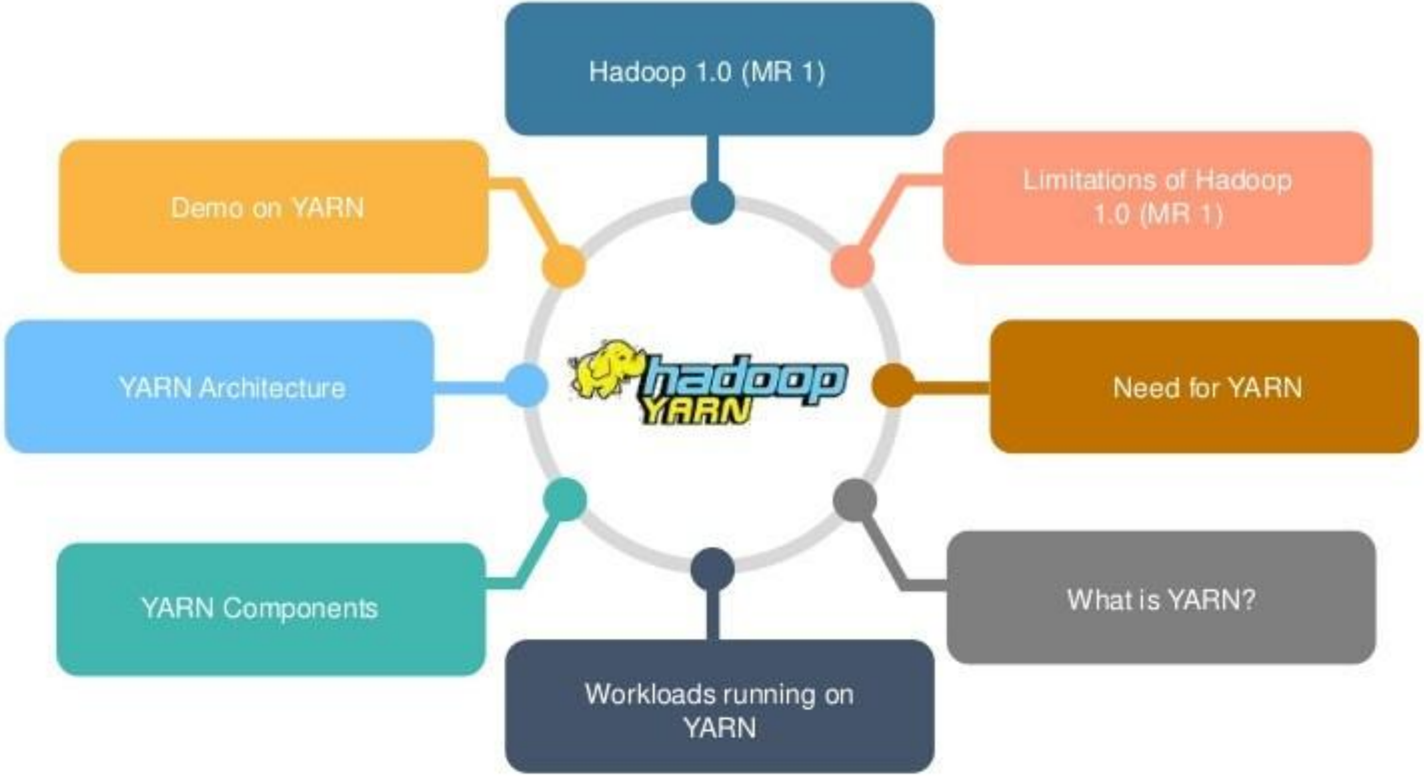
What is YARN?

Workloads running on YARN

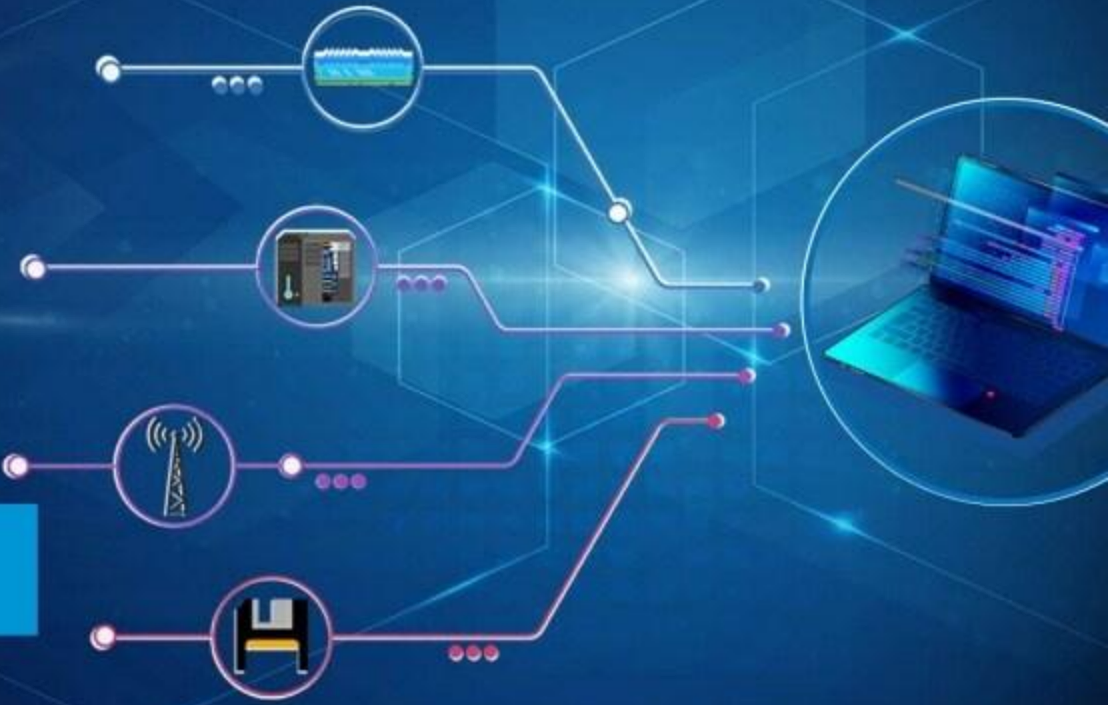
YARN Components

YARN Architecture

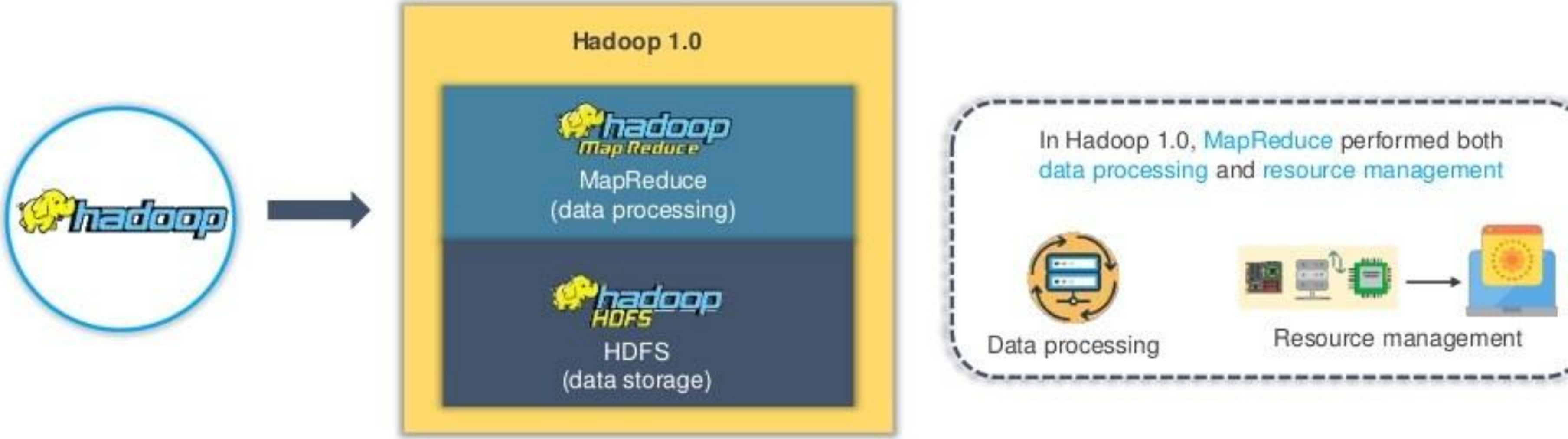
What's in it for you?



Hadoop 1.0 (MR 1)



Hadoop 1.0 (MR 1)



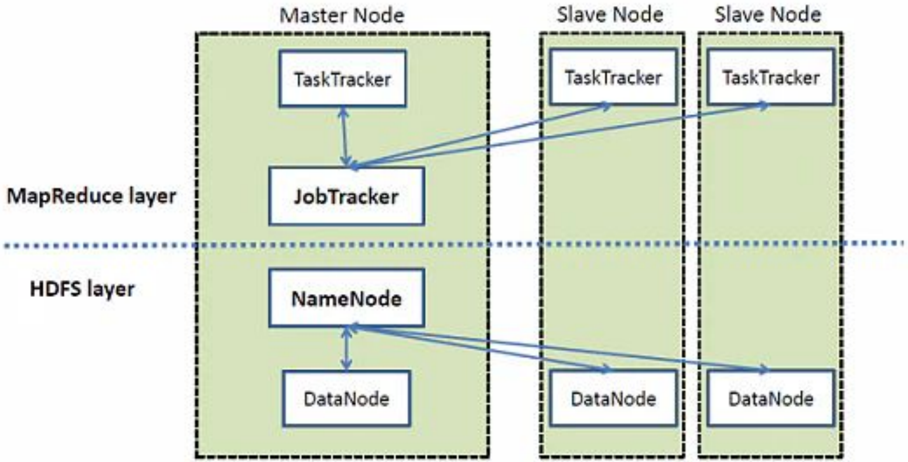
Hadoop 1.0

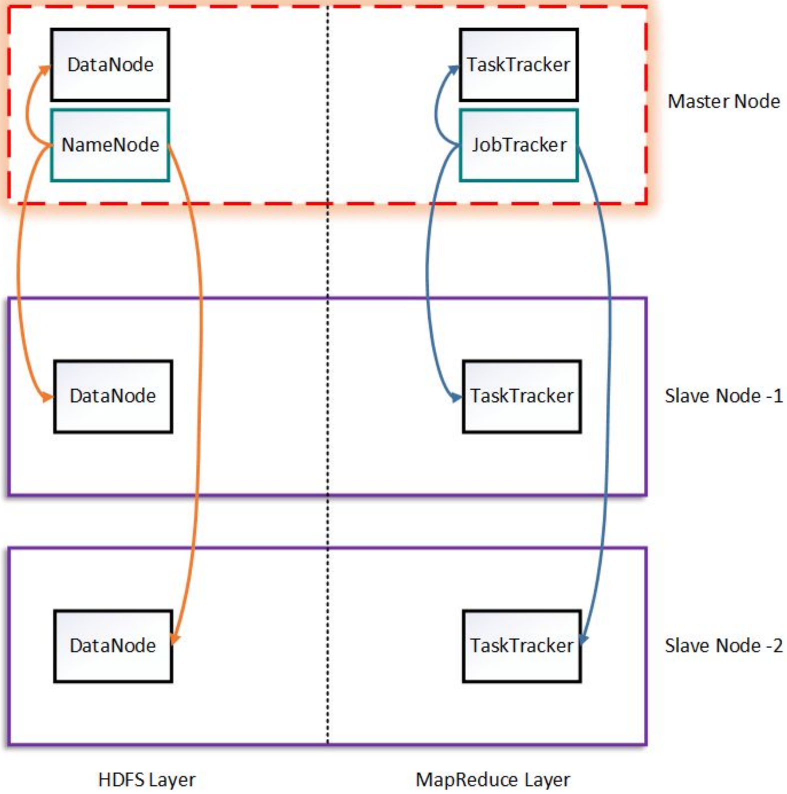
MapReduce (data processing)

HDFS (data storage)

In Hadoop 1.0, MapReduce performed both data processing and resource management

High Level Architecture of Hadoop



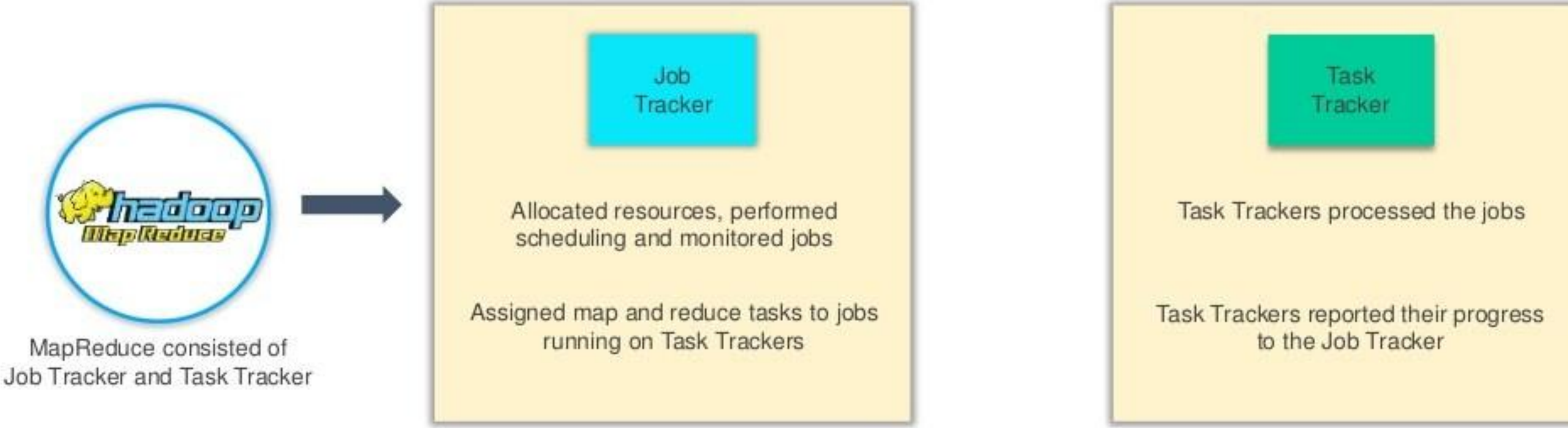


Job Tracker and Task Tracker

Job Tracker and Task Tracker are two essential processes involved in MapReduce execution in MRv1 (or Hadoop version 1).

Both processes are now deprecated in MRv2 (or Hadoop version 2) and replaced by **Resource Manager, Application Master and Node Manager** Daemons.

Hadoop 1.0 (MR 1)



MapReduce consisted of Job Tracker and Task Tracker

Job Tracker

Allocated resources, performed scheduling and monitored jobs

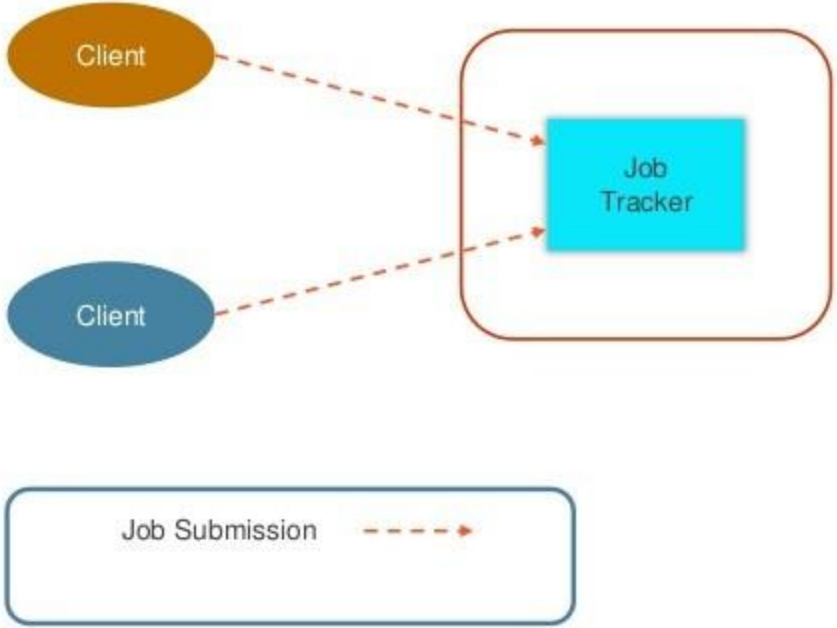
Assigned map and reduce tasks to jobs running on Task Trackers

Task Tracker

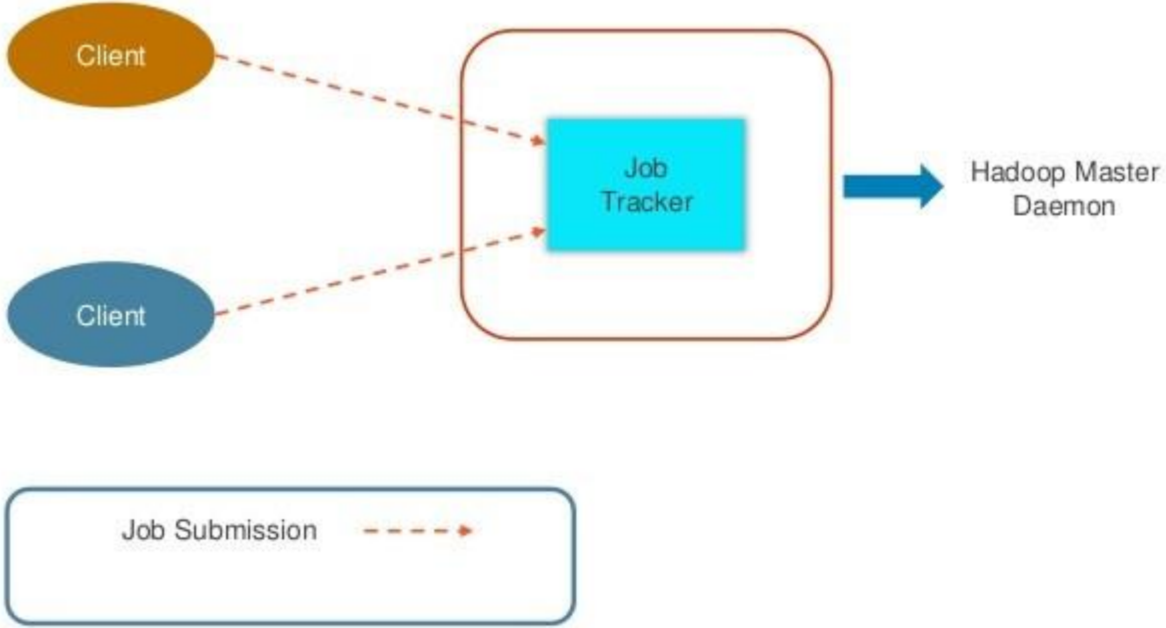
Task Trackers processed the jobs

Task Trackers reported their progress to the Job Tracker

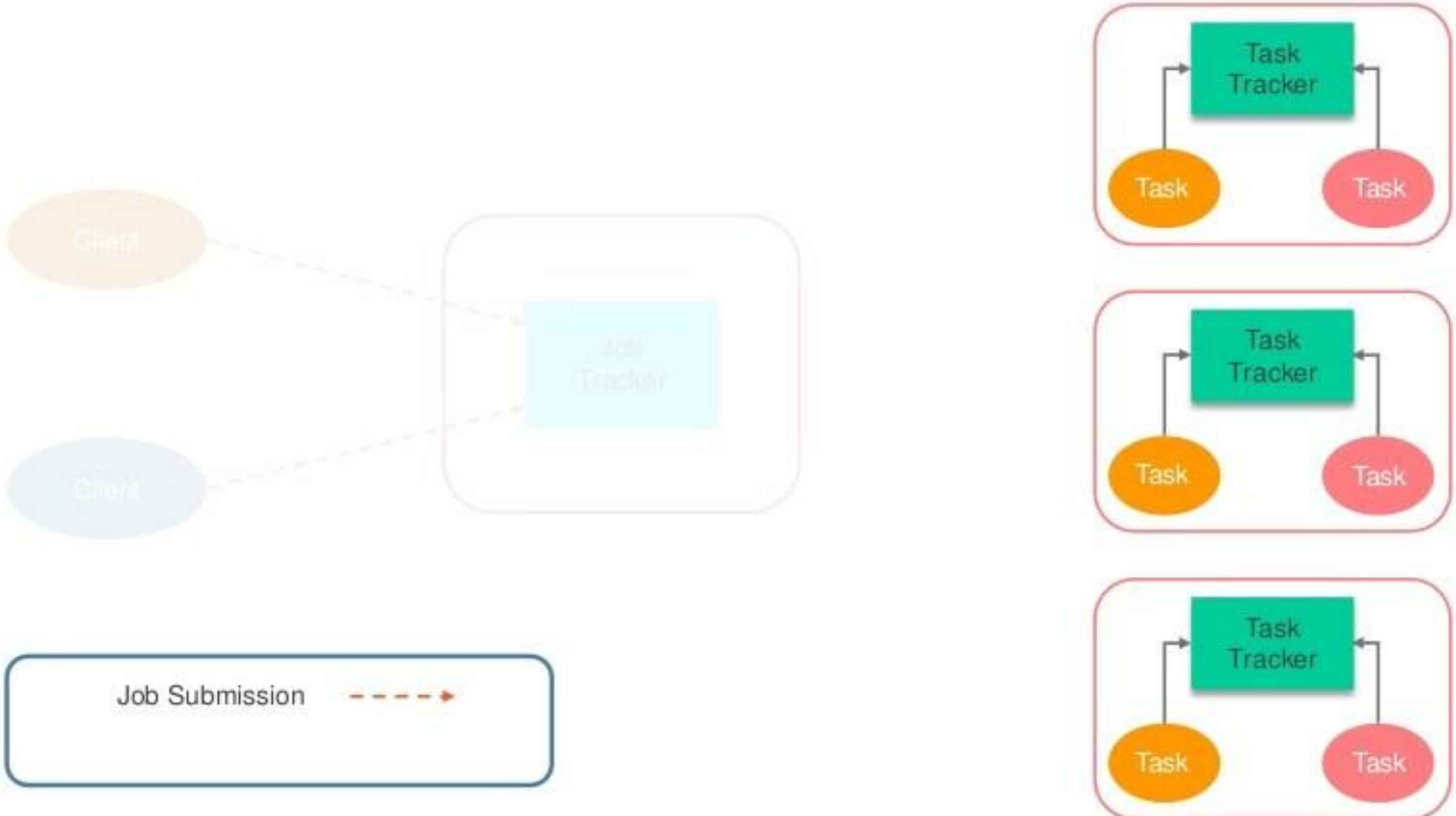
Hadoop 1.0 (MR 1)



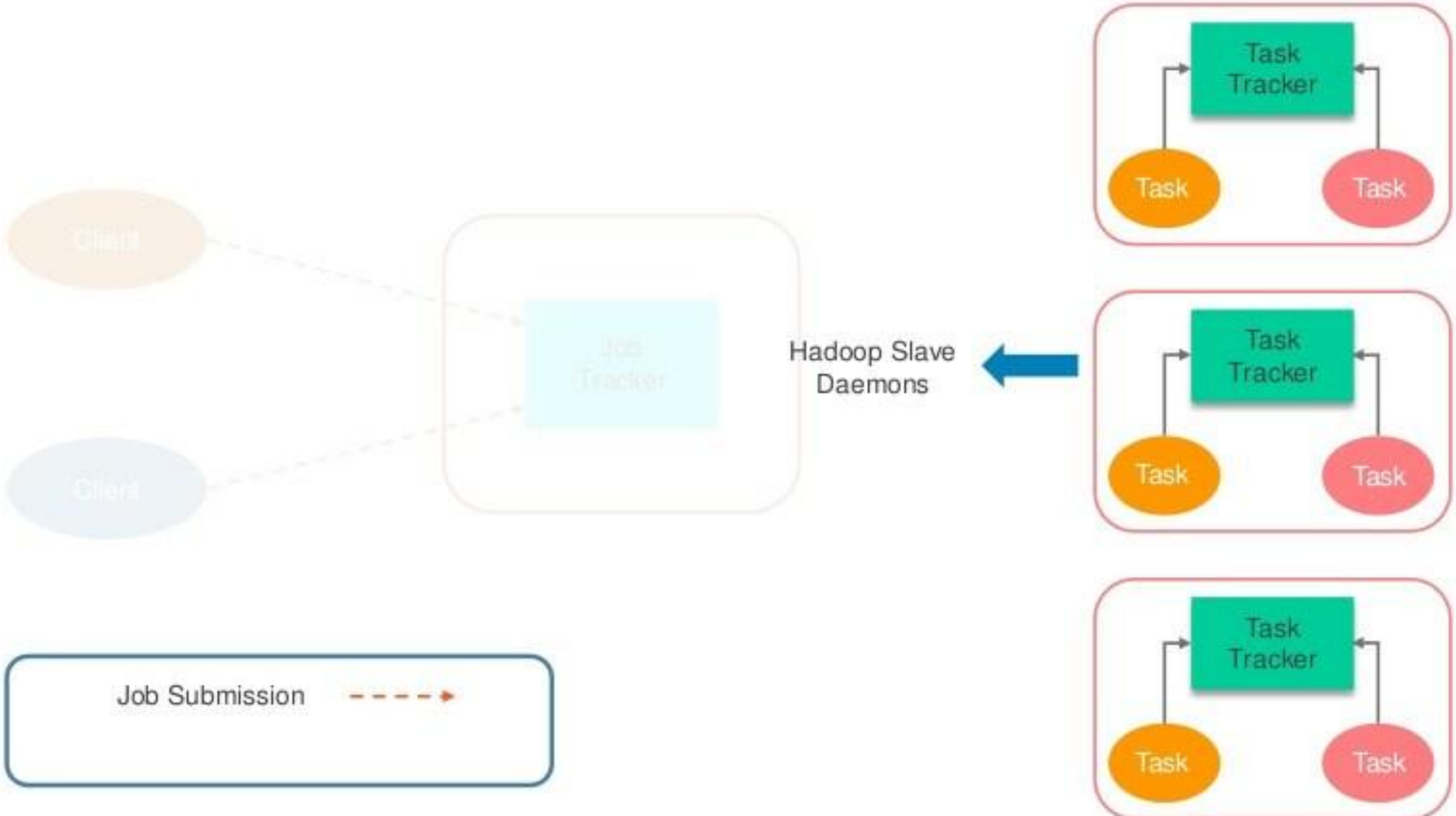
Hadoop 1.0 (MR 1)



Hadoop 1.0 (MR 1)



Hadoop 1.0 (MR 1)



Client

Client

Job Tracker

Hadoop Slave Daemons

Task Tracker

Task

Task

Task Tracker

Task

Task

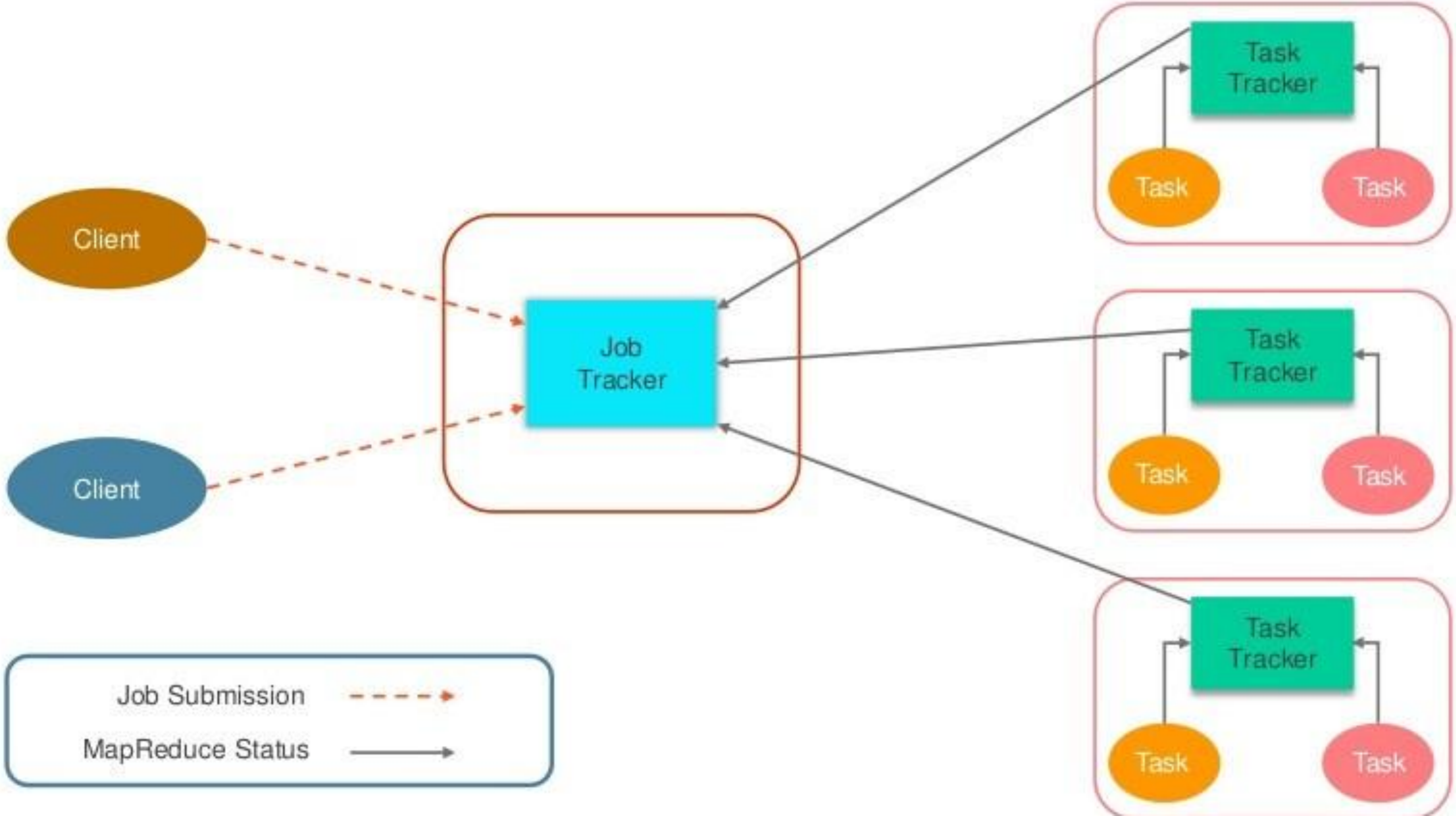
Task Tracker

Task

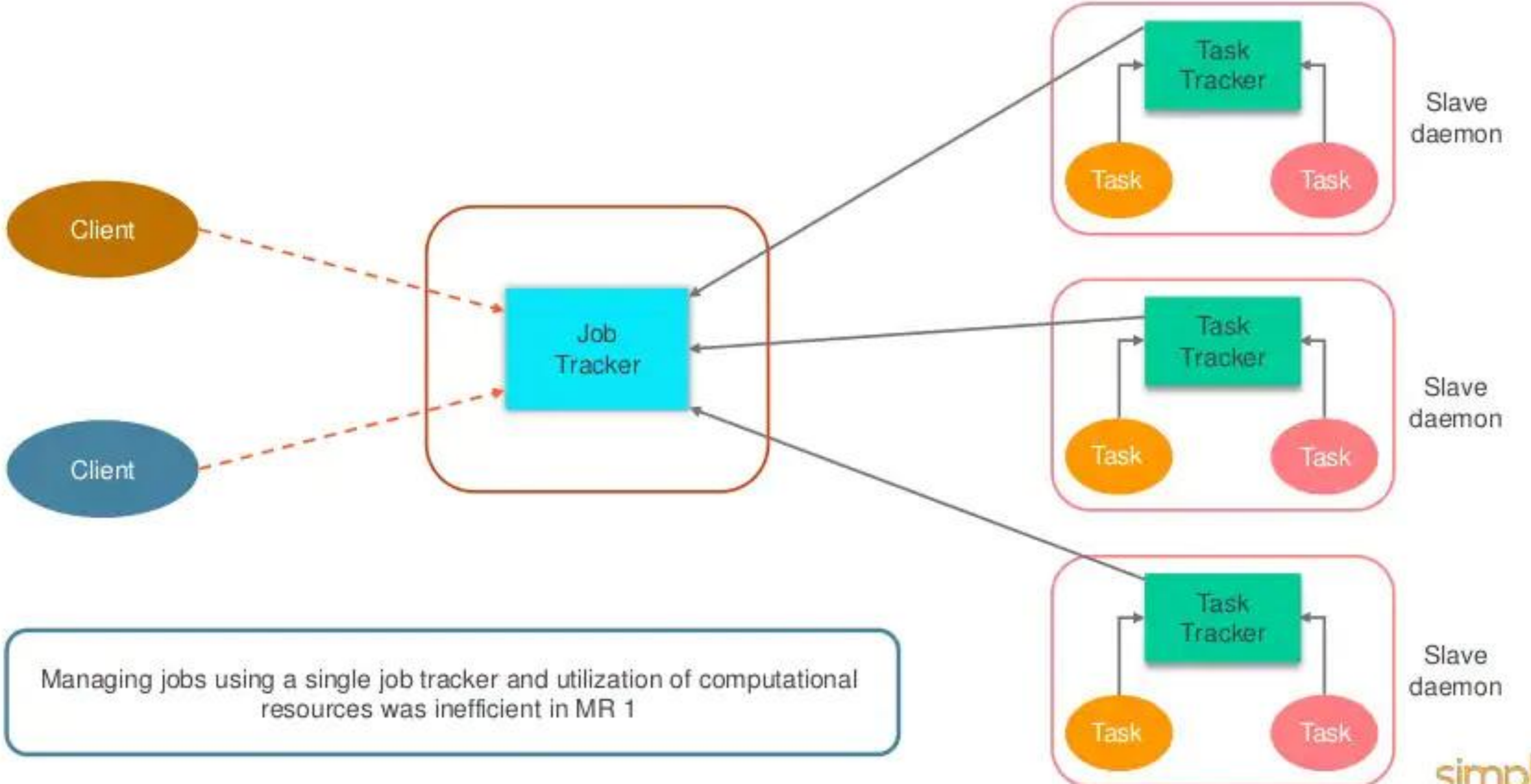
Task

Job Submission - - - - ▸

Hadoop 1.0 (MR 1)



Hadoop 1.0 (MR 1)



Managing jobs using a single job tracker and utilization of computational resources was inefficient in MR 1

Job Tracker

1. Job Tracker process runs on a separate node and not usually on a Data Node.
2. Job Tracker is an essential Daemon for MapReduce execution in MRv1. It is replaced by Resource Manager/Application Master in MRv2.
3. Job Tracker receives the requests for MapReduce execution from the client.
4. Job Tracker talks to the Name Node to determine the location of the data.
5. Job Tracker finds the best Task Tracker nodes to execute tasks based on the **data locality** (proximity of the data) and the available slots to execute a task on a given node.

Job Tracker

6. Job Tracker monitors the individual Task Trackers and submits back the overall status of the job back to the client.
7. Job Tracker process is critical to the Hadoop cluster in terms of MapReduce execution.
8. When the **Job Tracker is down**, HDFS will still be functional but the MapReduce execution can not be started and the existing MapReduce jobs will be halted.

Task Tracker

1. Task Tracker runs on Data Node. Mostly on all Data Nodes.
2. Task Tracker is replaced by Node Manager in MRv2.
3. Mapper and Reducer tasks are executed on Data Nodes **administered by Task Trackers**.
4. Task Trackers will be assigned Mapper and Reducer tasks to execute by Job Tracker.
5. Task Tracker will be in constant communication with the Job Tracker signaling the progress of the task in execution.
6. **Task Tracker failure is not considered fatal**. When a Task Tracker becomes unresponsive, Job Tracker will assign the task executed by the Task Tracker to another node.

Limitations of Hadoop 1.0 (MR 1)

1 Scalability

Due to a single JobTracker, scalability became a bottleneck.

Cannot have a cluster size of more than 4000 nodes and cannot run more than 40000 concurrent tasks

Limitations of Hadoop 1.0 (MR 1)

1 Scalability

Due to a single JobTracker, scalability became a bottleneck.

Maximum cluster size – 4000 nodes
Maximum concurrent tasks – 40000

2 Availability issue

JobTracker is single point of failure. Any failure kills all queued and running jobs. Jobs need to be resubmitted by users

Limitations of Hadoop 1.0 (MR 1)

3 Resource Utilization

Due to predefined number of map and reduce slots for each TaskTracker, resource utilization issues occur

Limitations of Hadoop 1.0 (MR 1)

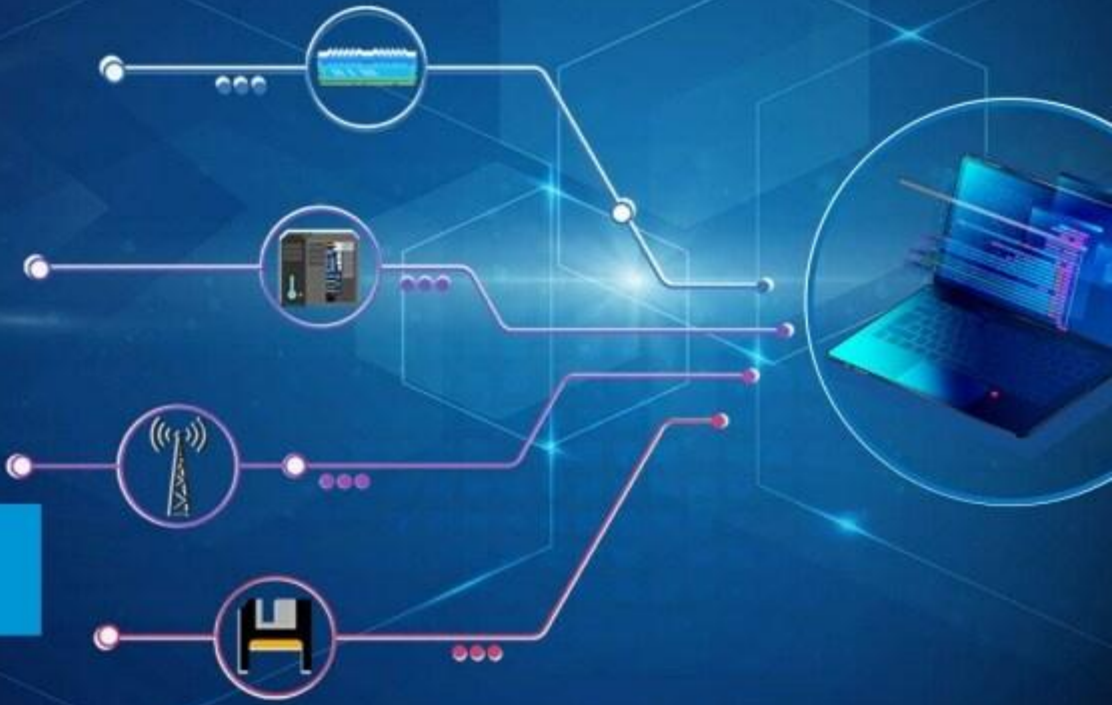
3 Resource Utilization

Due to predefined number of map and reduce slots for each TaskTracker, resource utilization issues occur

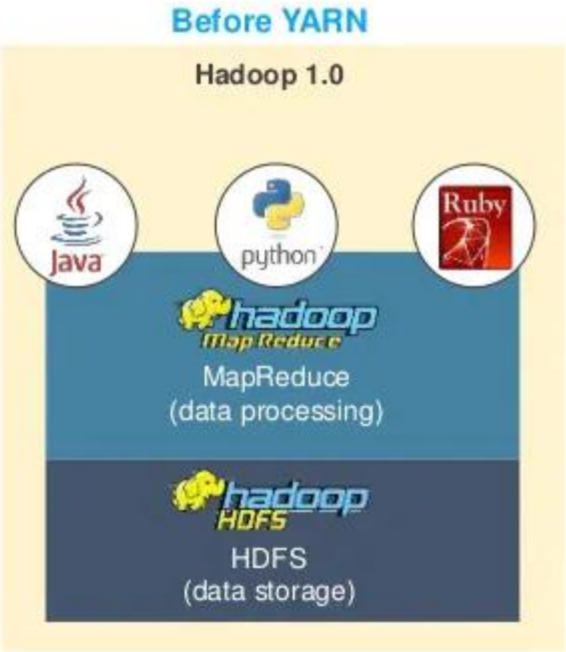
4 Limitations in running non-MapReduce applications

Problem in performing real-time analysis and running Ad-hoc query as MapReduce is batch driven

Need for YARN



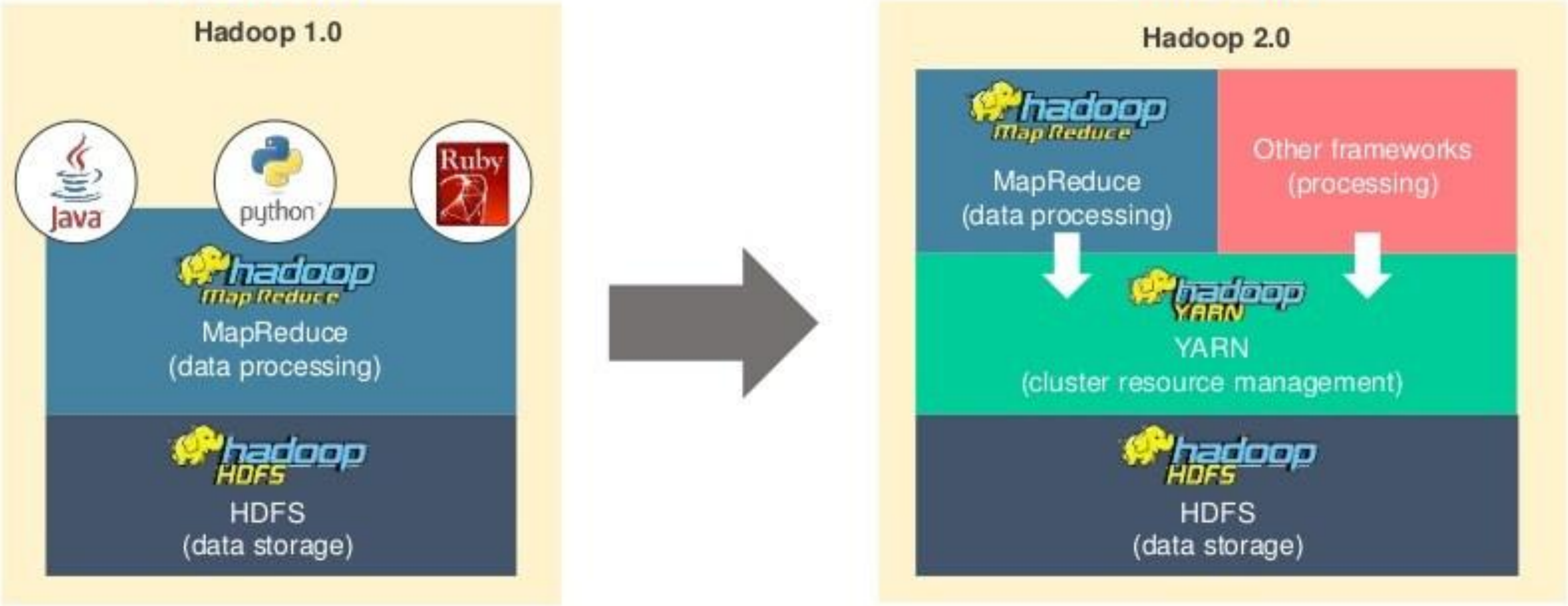
Need for YARN



Designed to run MapReduce jobs only and had issues in scalability, resource utilization, etc.

Need for YARN

Before YARN **After YARN**



Hadoop 1.0

MapReduce
(data processing)

HDFS
(data storage)

Hadoop 2.0

MapReduce
(data processing)

Other frameworks
(processing)

YARN
(cluster resource management)

HDFS
(data storage)

Designed to run MapReduce jobs only and had issues in scalability, resource utilization, etc.

YARN solved those issues and users could work on multiple processing models along with MapReduce

Hadoop 2.0 (YARN)

Solution - Hadoop 2.0 (YARN)

Scalability

Can have a cluster size of more than 10,000 nodes and can run more than 1,00,000 concurrent tasks

Solution - Hadoop 2.0 (YARN)

Scalability

Can have a cluster size of more than 10,000 nodes and can run more than 1,00,000 concurrent tasks

Compatibility

Applications developed for Hadoop 1 runs on YARN without any disruption or availability issues

Solution - Hadoop 2.0 (YARN)

hadoop YARN

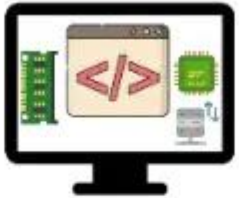
Scalability

Can have a cluster size of more than 10,000 nodes and can run
more than 1,00,000 concurrent tasks

Compatibility

Applications developed for Hadoop 1 runs on YARN without any disruption or availability issues

Resource utilization



Allows dynamic allocation of cluster resources to improve resource utilization

Solution - Hadoop 2.0 (YARN)

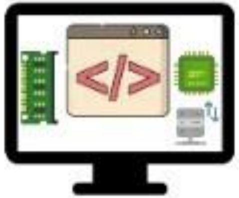
Scalability

Can have a cluster size of more than 10,000 nodes and can run more than 1,00,000 concurrent tasks

Compatibility

Applications developed for Hadoop 1 runs on YARN without any disruption or availability issues

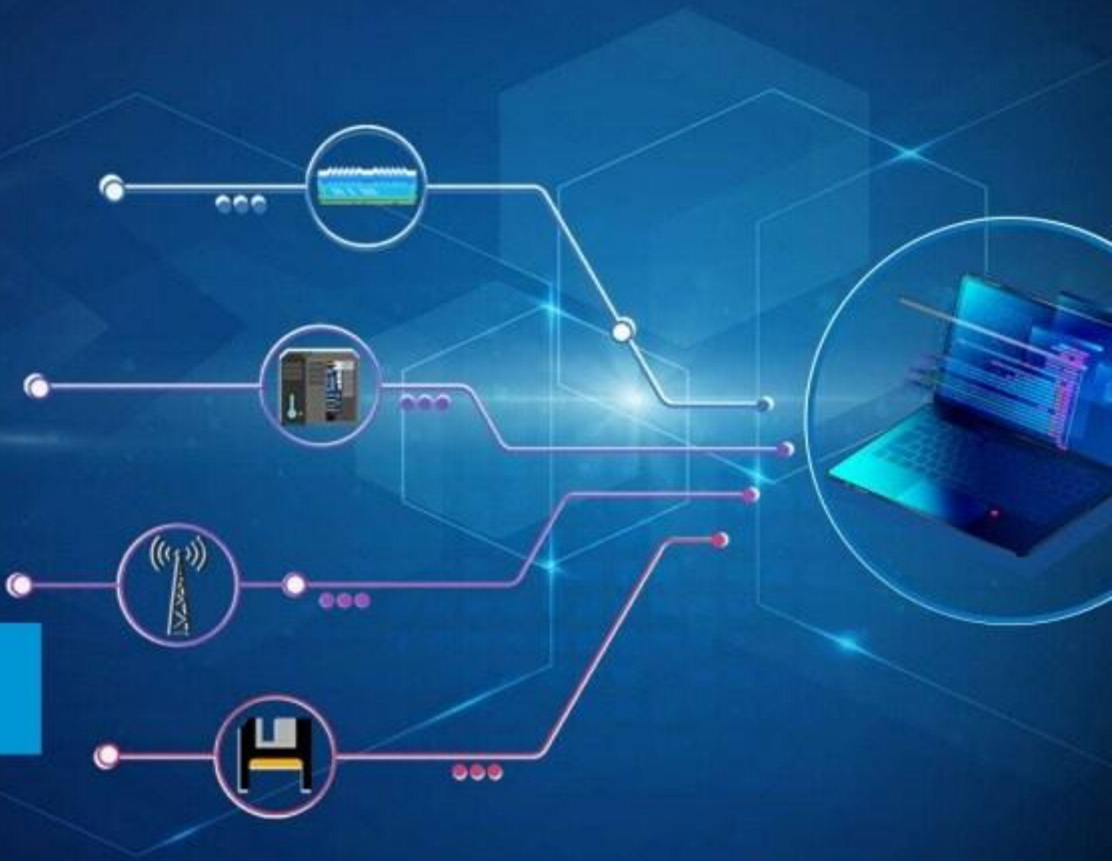
Resource utilization



Allows dynamic allocation of cluster resources to improve resource utilization

Multitenancy

Can use open-source and propriety data access engines and perform real-time analysis and running ad-hoc query



What is YARN?

What is YARN?

YARN – Yet Another Resource Negotiator

YARN is the cluster resource management layer of the Apache Hadoop Ecosystem, which schedules jobs and assigns resources

What is YARN?

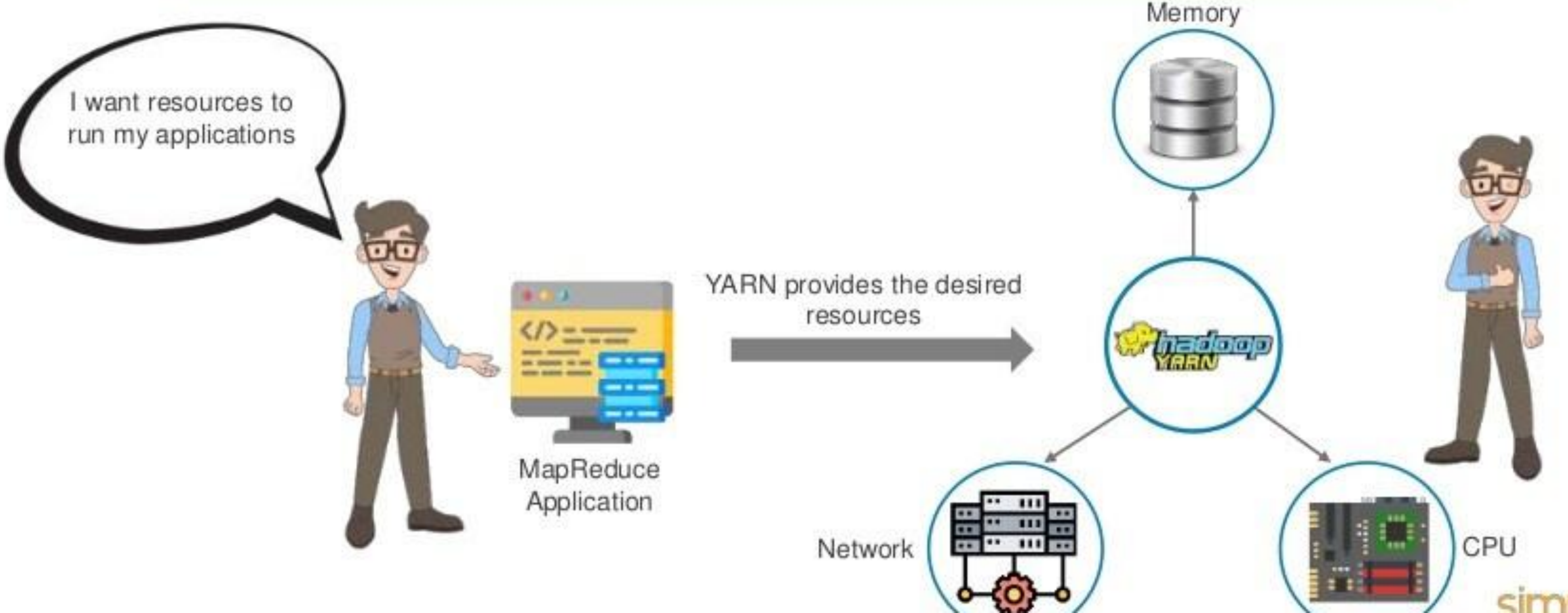
YARN – Yet Another Resource Negotiator

YARN is the cluster resource management layer of the Apache Hadoop Ecosystem, which schedules jobs and assigns resources

What is YARN?

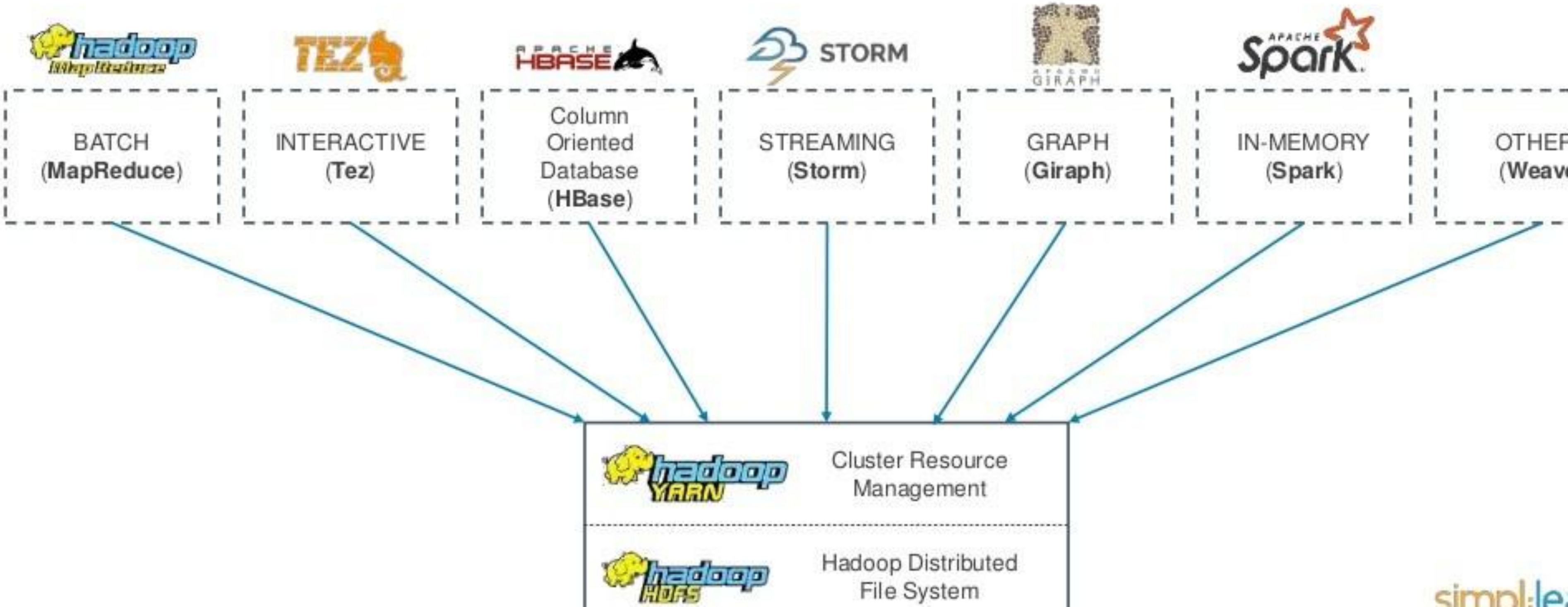
YARN – Yet Another Resource Negotiator

YARN is the cluster resource management layer of the Apache Hadoop Ecosystem, which schedules jobs and assigns resources


I want resources to run my applications
MapReduce Application
YARN provides the desired resources
hadoop YARN
Memory
Network
CPU

Workloads running on YARN

List of frameworks that runs on top of YARN:



BATCH (**MapReduce**)

INTERACTIVE (**Tez**)

Column Oriented Database (**HBase**)

STREAMING (**Storm**)

GRAPH (**Giraph**)

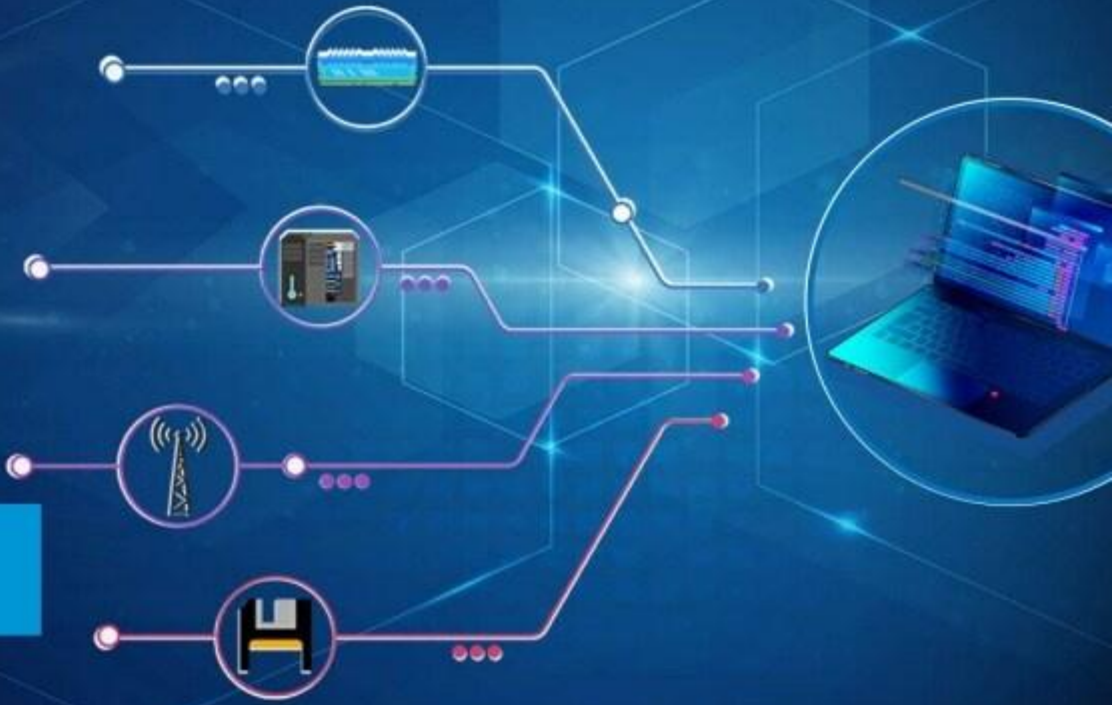
IN-MEMORY (**Spark**)

OTHER (**Weav**

Cluster Resource Management

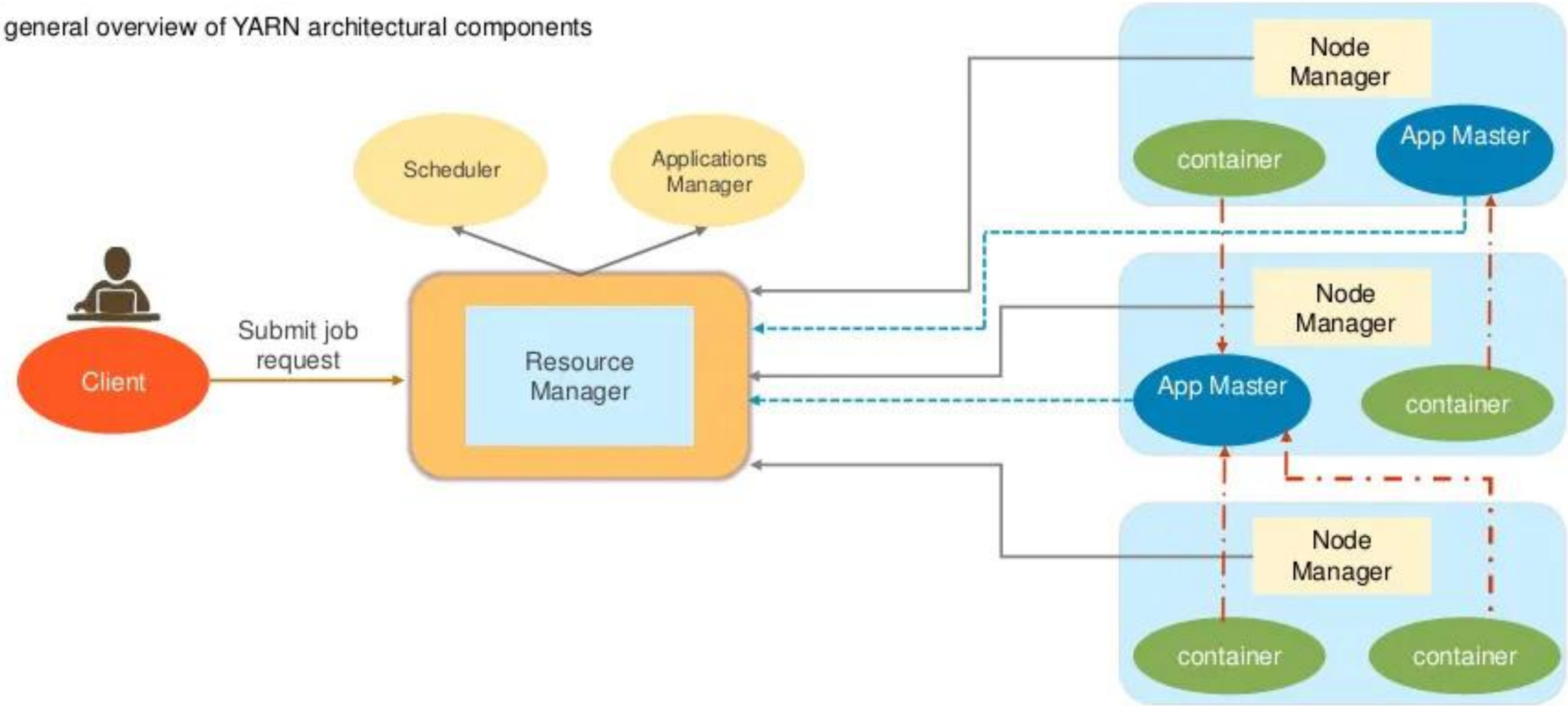
Hadoop Distributed File System

YARN Components



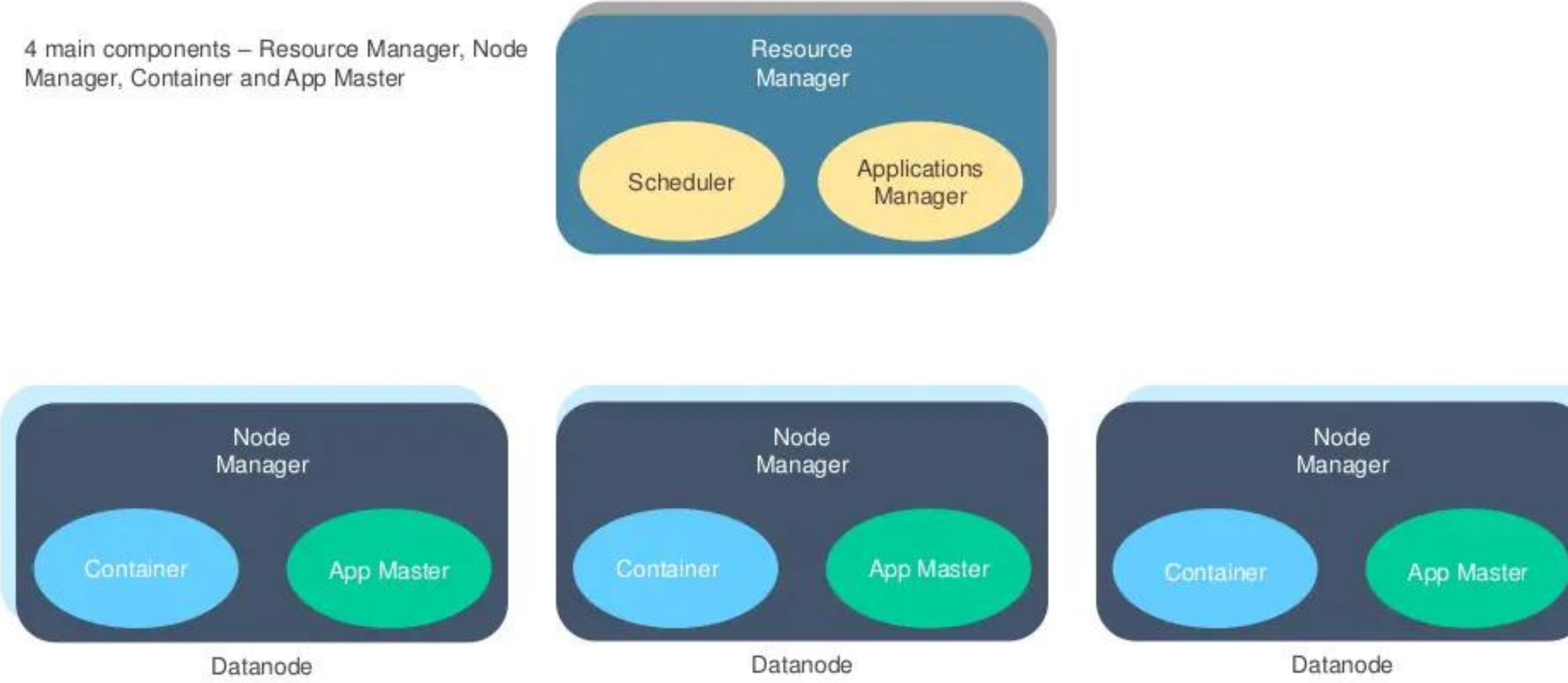
YARN Components

A general overview of YARN architectural components

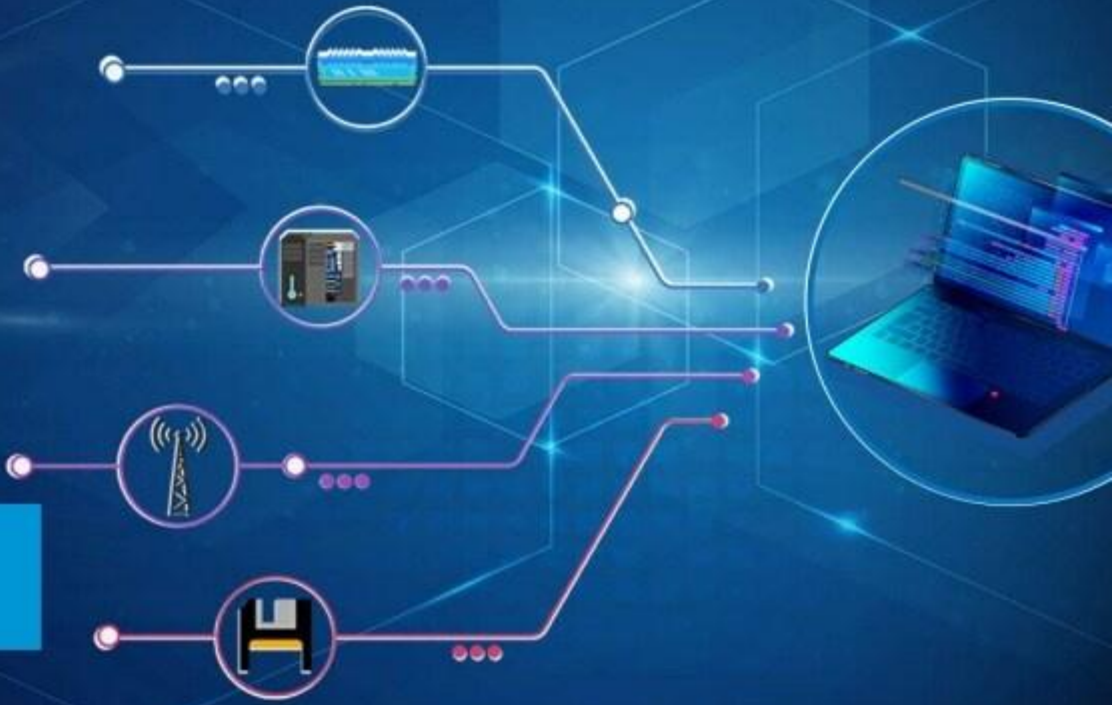


YARN Components

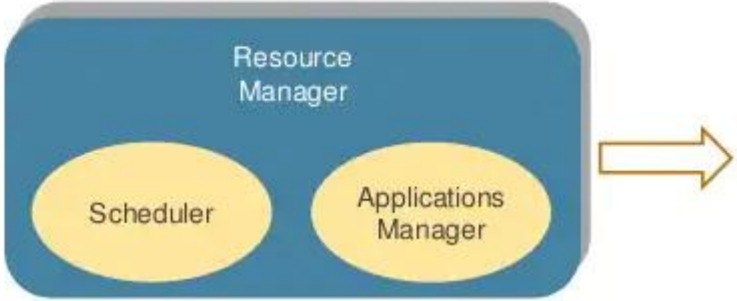
4 main components – Resource Manager, Node Manager, Container and App Master



YARN Components – Resource Manager

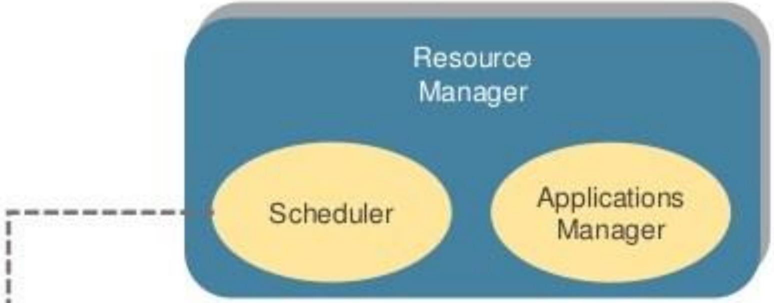


YARN Components – Resource Manager



Ultimate authority that decides the allocation of resources among all the applications in the system

YARN Components – Resource Manager

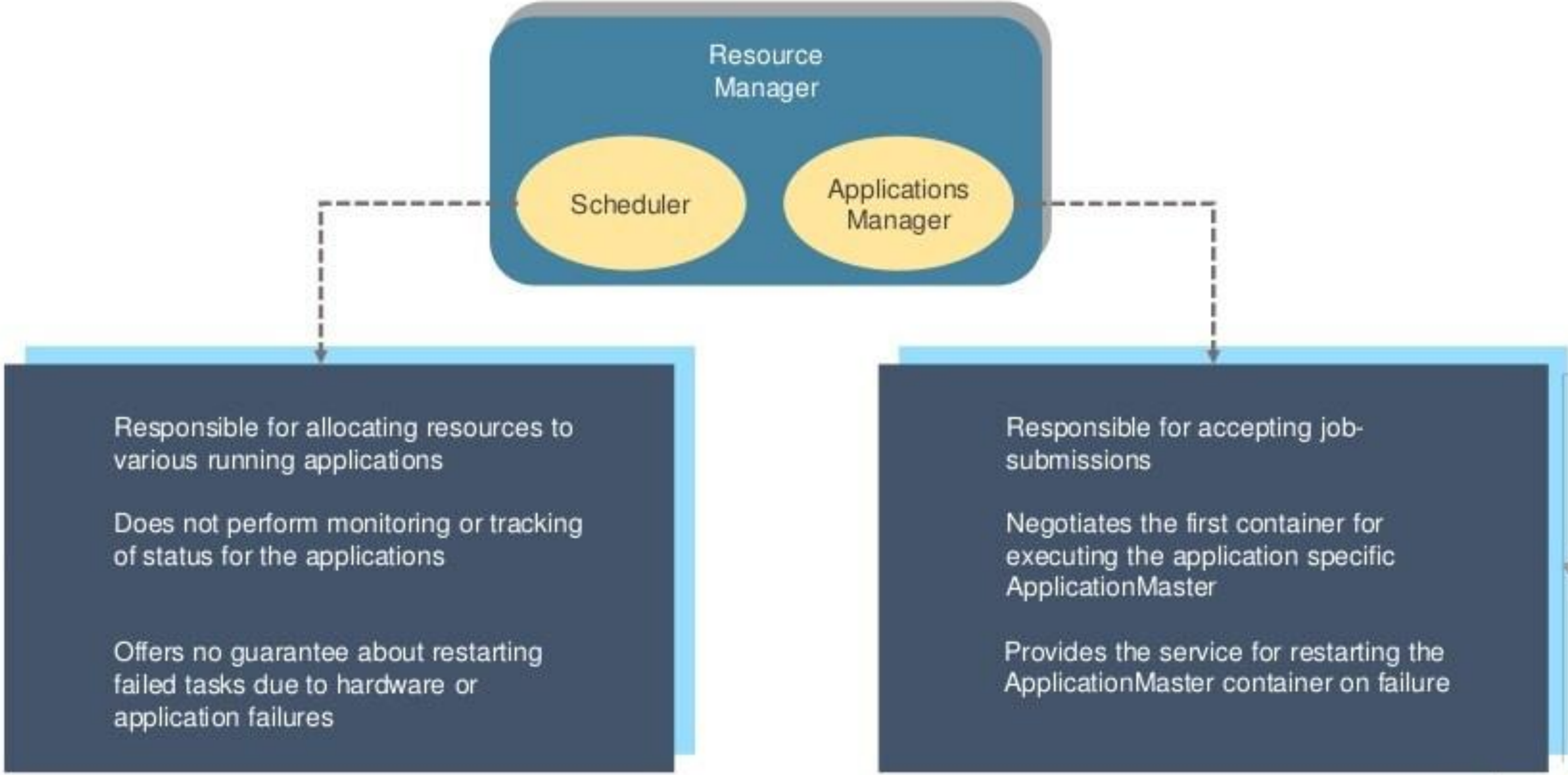


Responsible for allocating resources to various running applications

Does not perform monitoring or tracking of status for the applications

Offers no guarantee about restarting failed tasks due to hardware or application failures

YARN Components – Resource Manager



Responsible for allocating resources to various running applications

Does not perform monitoring or tracking of status for the applications

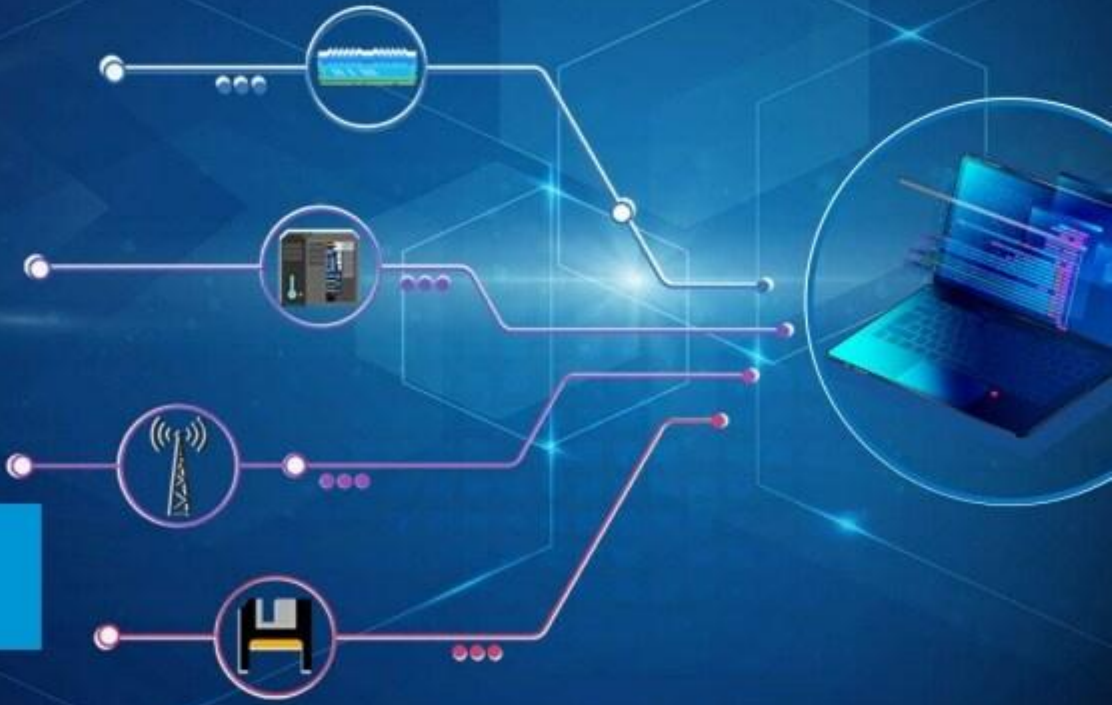
Offers no guarantee about restarting failed tasks due to hardware or application failures

Responsible for accepting job-submissions

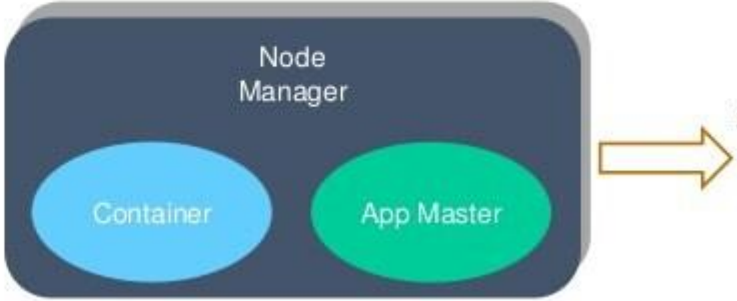
Negotiates the first container for executing the application specific ApplicationMaster

Provides the service for restarting the ApplicationMaster container on failure

YARN Components – Node Manager

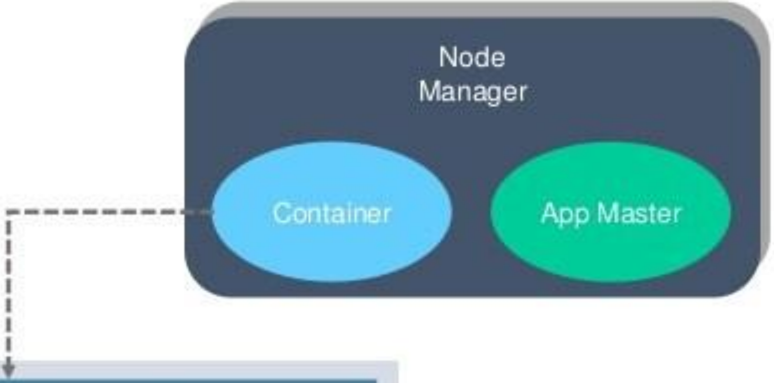


YARN Components – Node Manager



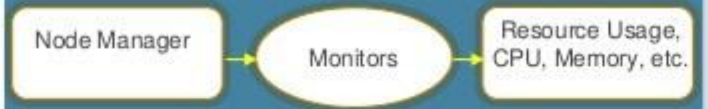
Slaves track processes and running jobs and monitor each container's resource utilization

YARN Components – Node Manager

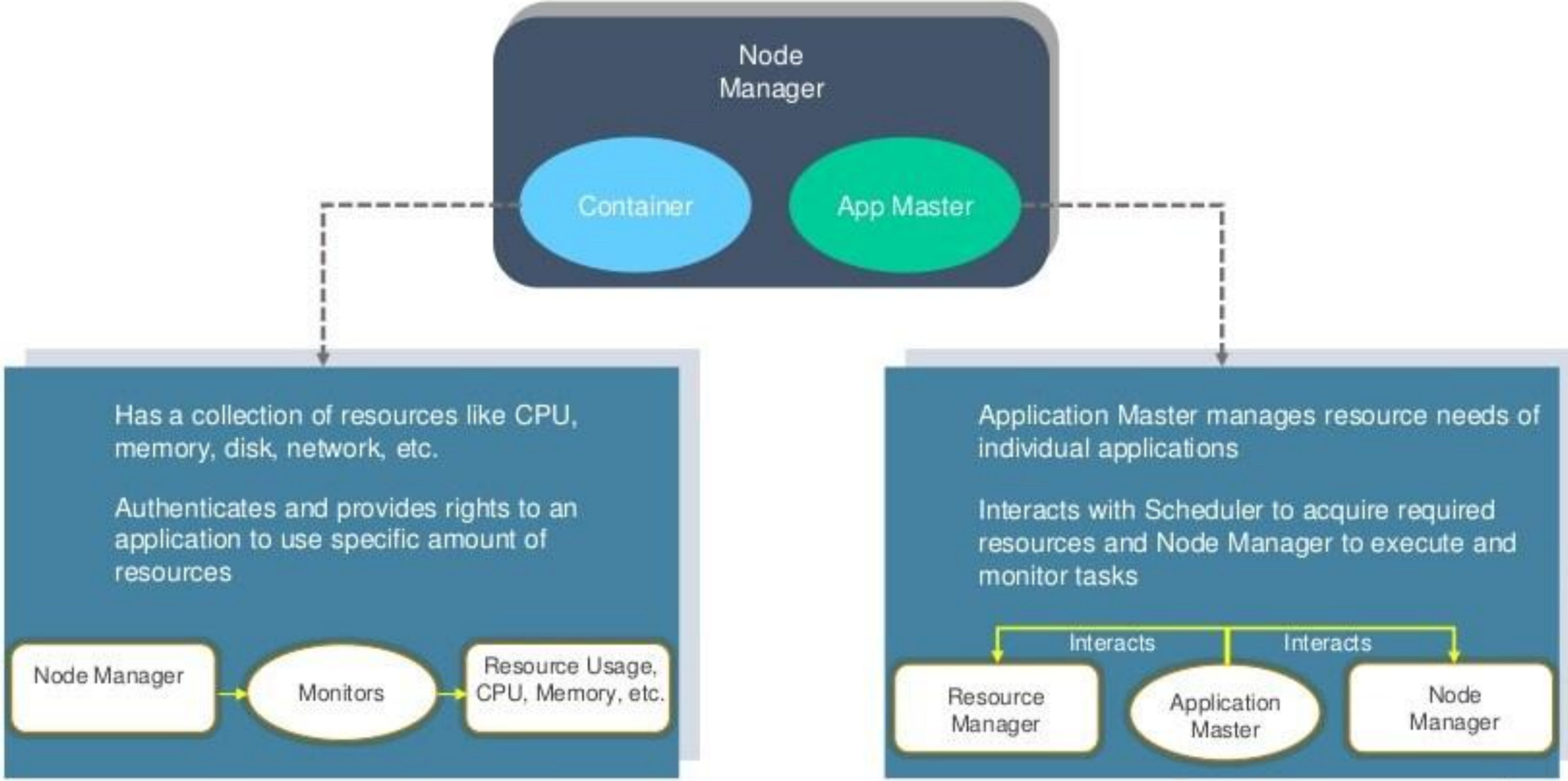


Has a collection of resources like CPU, memory, disk, network, etc.

Authenticates and provides rights to an application to use specific amount of resources



YARN Components – Node Manager



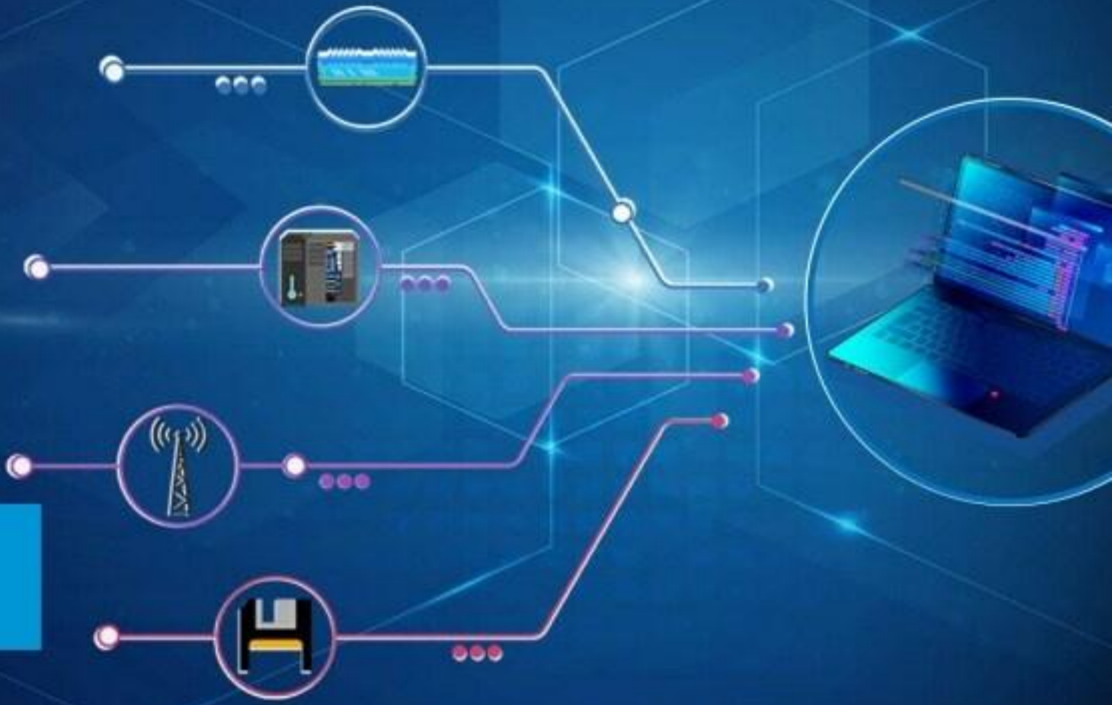
Has a collection of resources like CPU, memory, disk, network, etc.

Authenticates and provides rights to an application to use specific amount of resources

Application Master manages resource needs of individual applications

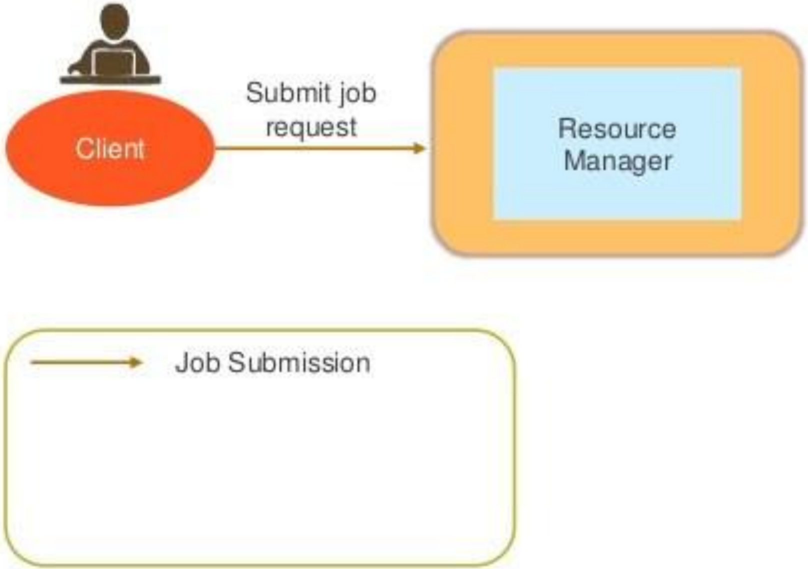
Interacts with Scheduler to acquire required resources and Node Manager to execute and monitor tasks

YARN Architecture

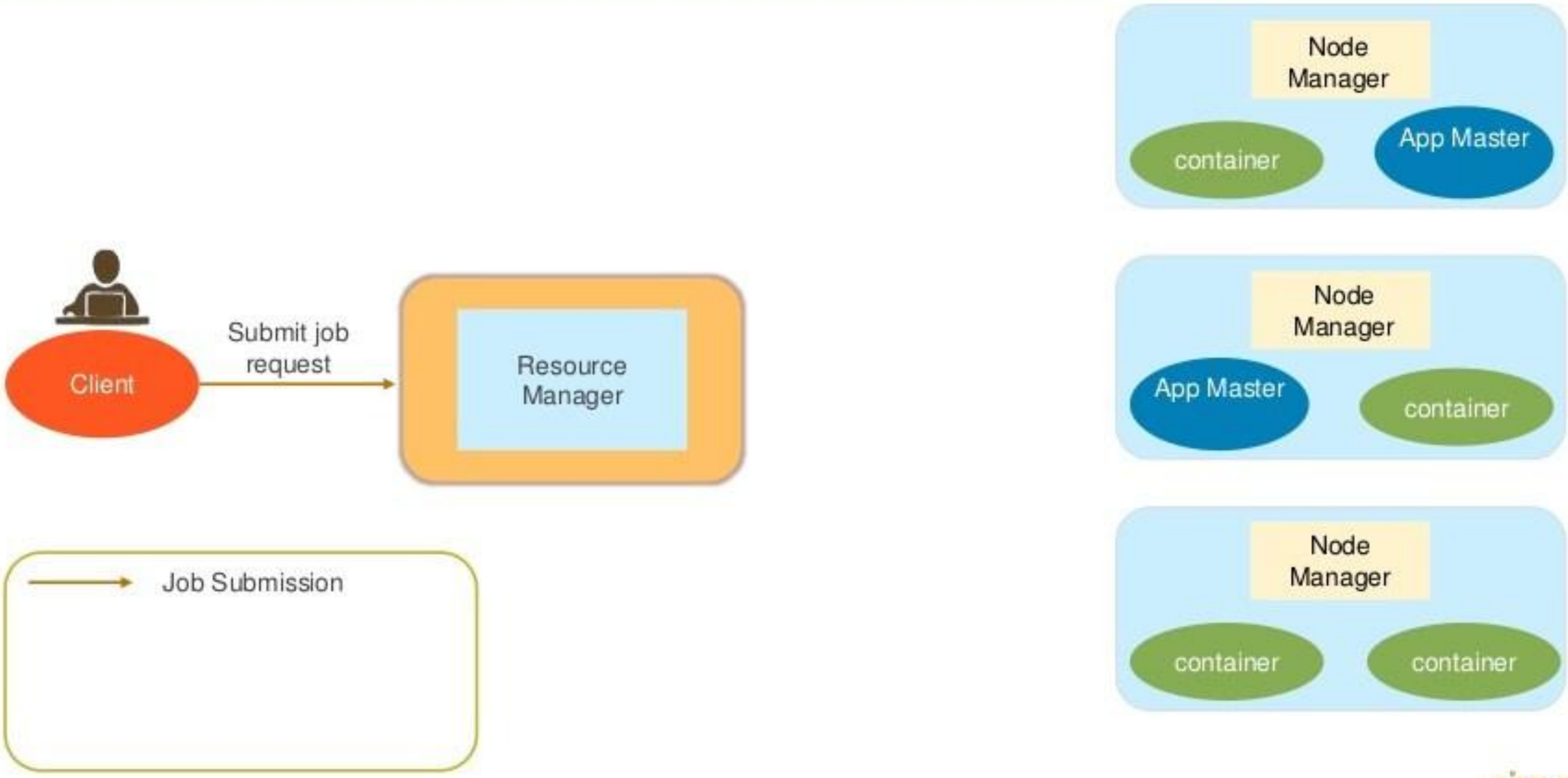


YARN Architecture

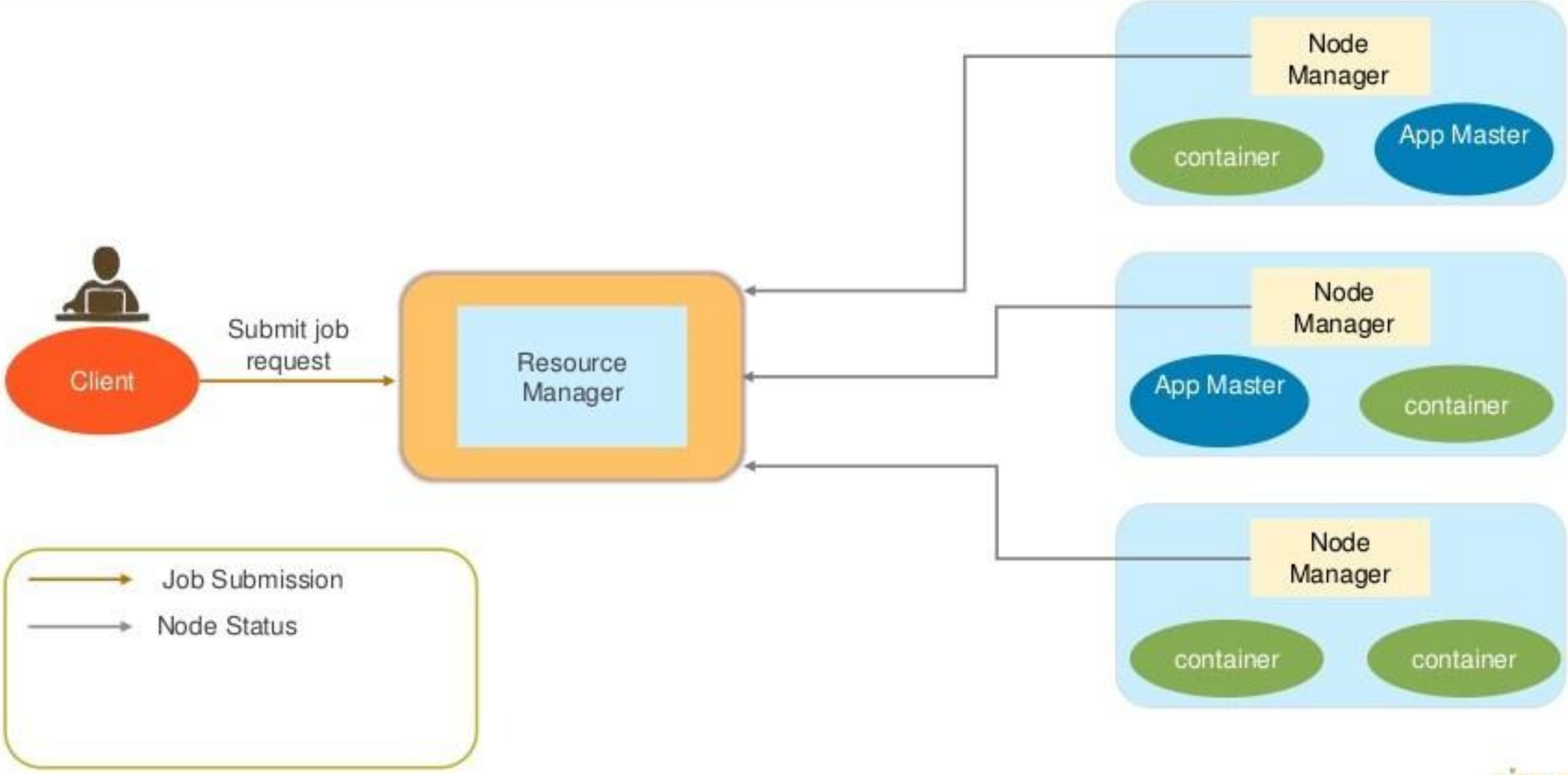
YARN Architecture



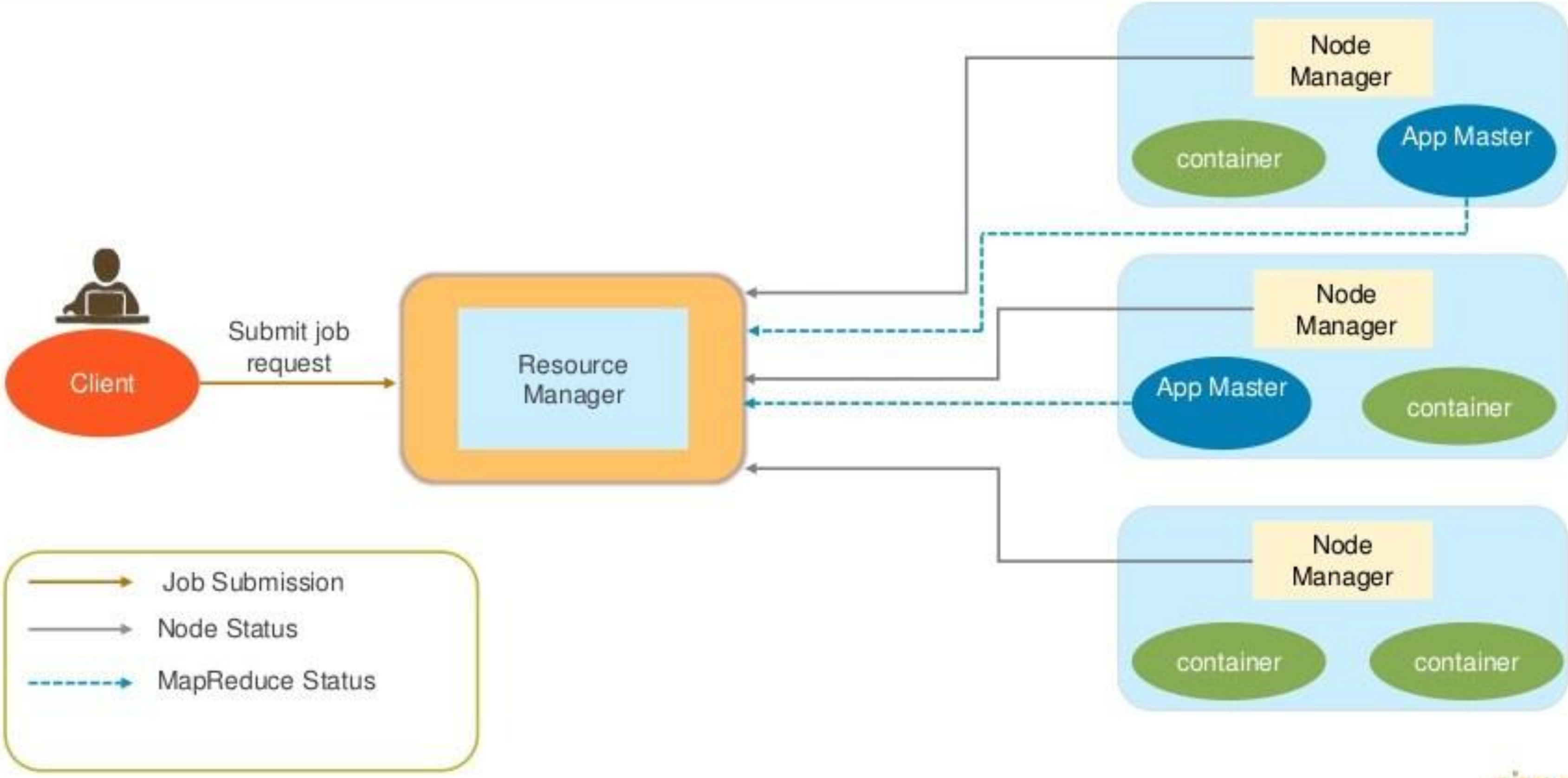
YARN Architecture



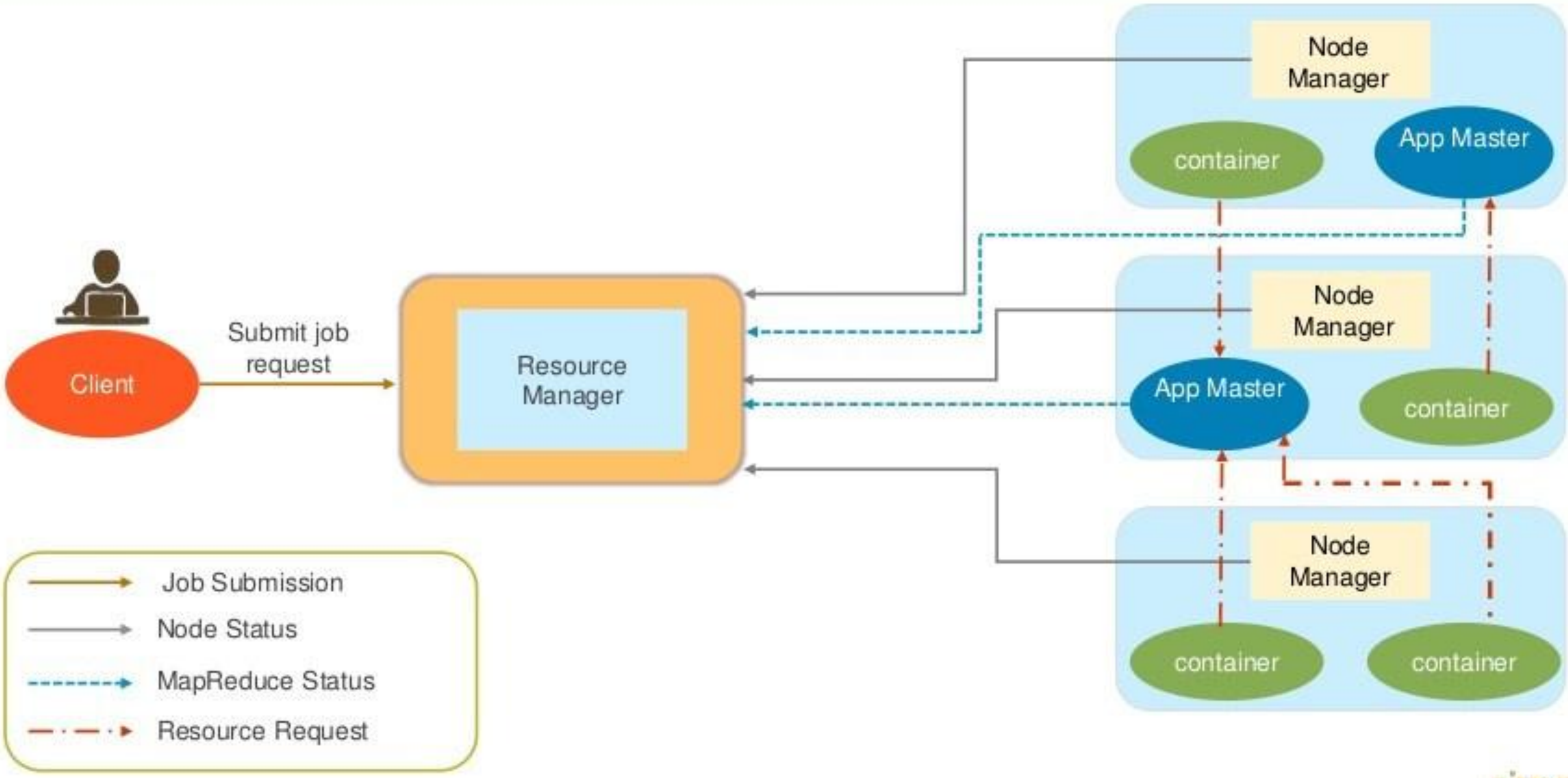
YARN Architecture

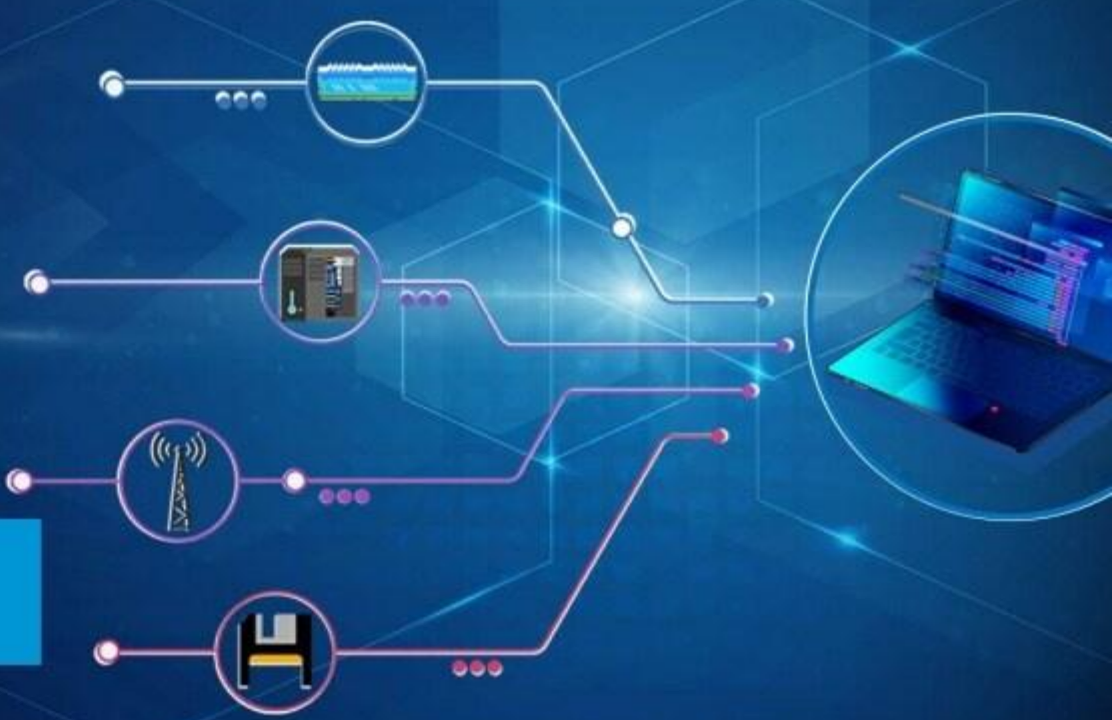


YARN Architecture



YARN Architecture





Running an application in YARN

Running an application in YARN

Client submits an application to the ResourceManager

Running an application in YARN

1
Client
2
Resource Manager

Client submits an application to the ResourceManager

ResourceManager allocates a container

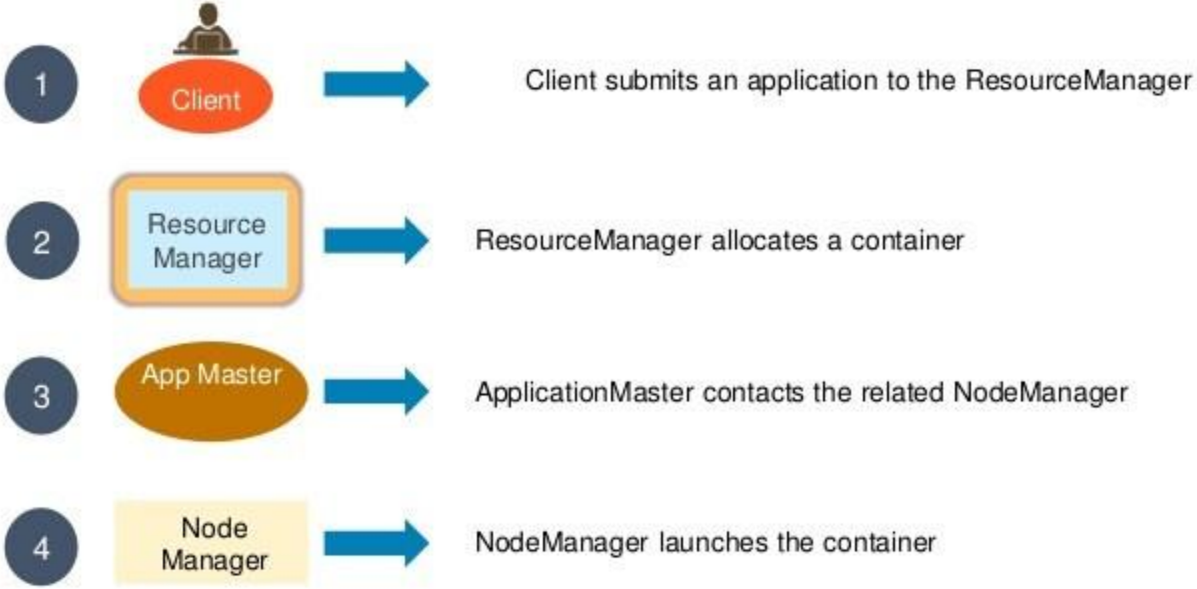
Running an application in YARN

1 Client: Client submits an application to the ResourceManager

2 Resource Manager: ResourceManager allocates a container

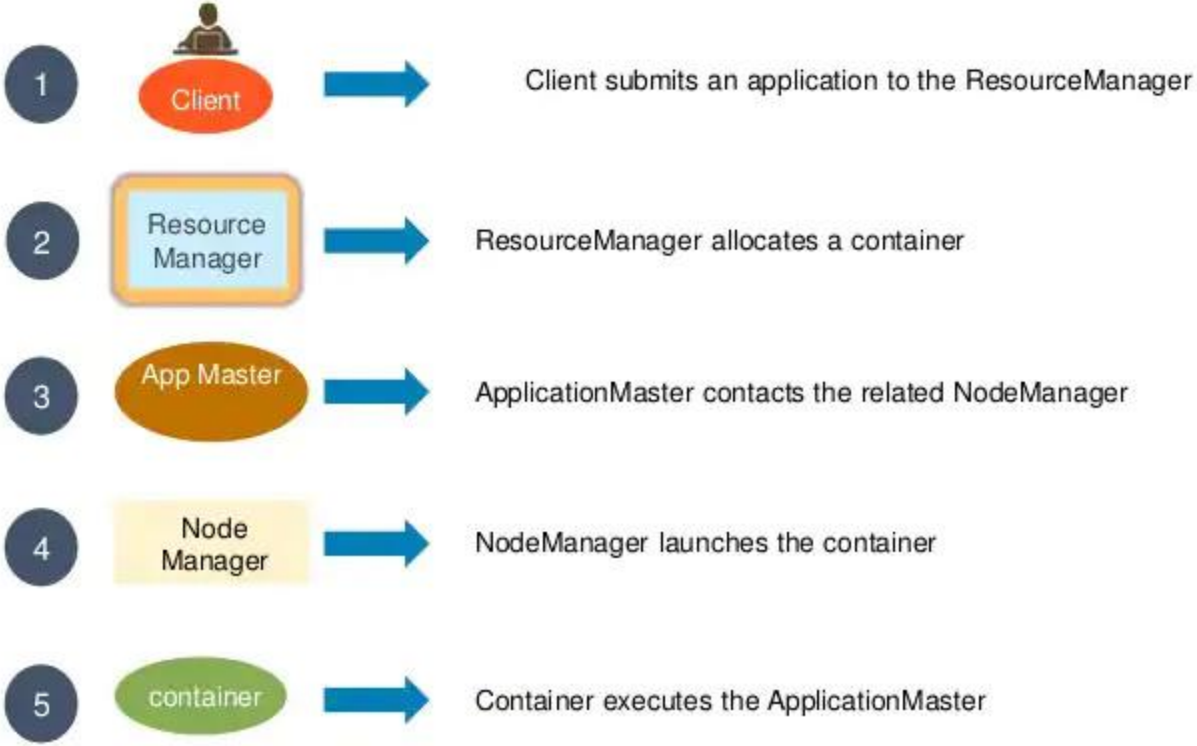
3 App Master: ApplicationMaster contacts the related NodeManager

Running an application in YARN



1. Client → Client submits an application to the ResourceManager
2. Resource Manager → ResourceManager allocates a container
3. App Master → ApplicationMaster contacts the related NodeManager
4. Node Manager → NodeManager launches the container

Running an application in YARN



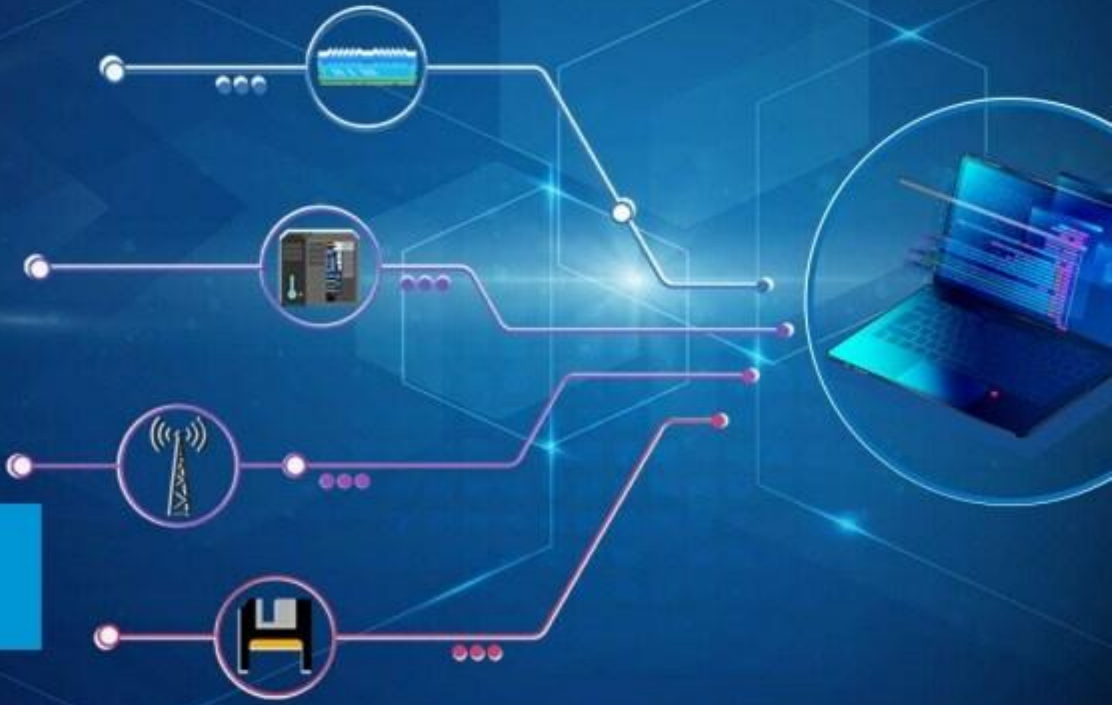
1 Client → Client submits an application to the ResourceManager

2 Resource Manager → ResourceManager allocates a container

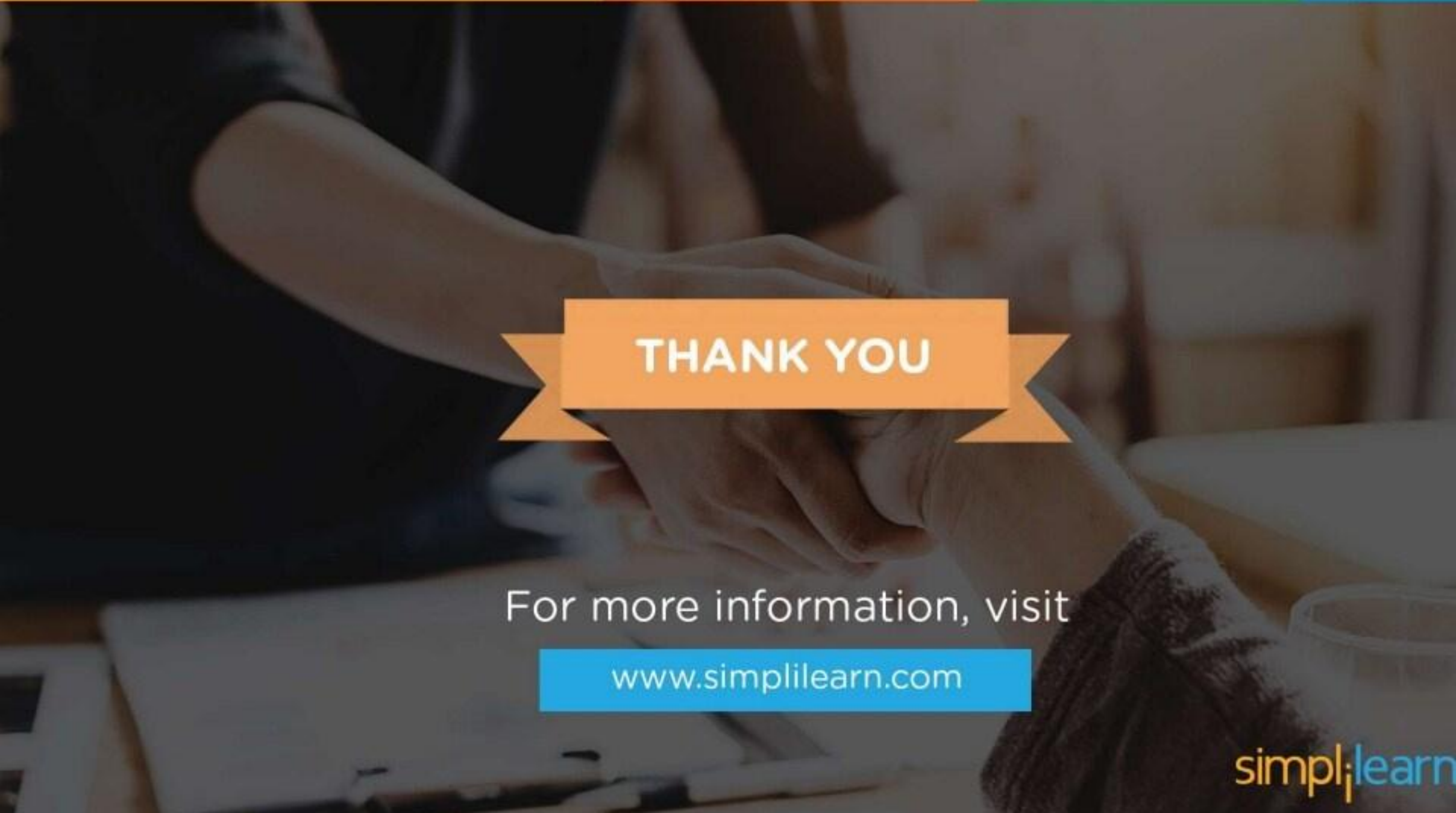
3 App Master → ApplicationMaster contacts the related NodeManager

4 Node Manager → NodeManager launches the container

5 container → Container executes the ApplicationMaster



Demo on YARN



THANK YOU

For more information, visit

www.simplilearn.com