(Big) Data Engineering In Depth

From Beginner to Professional

Moustafa Alaa

Senior Data Engineer at Onfido, London, UK

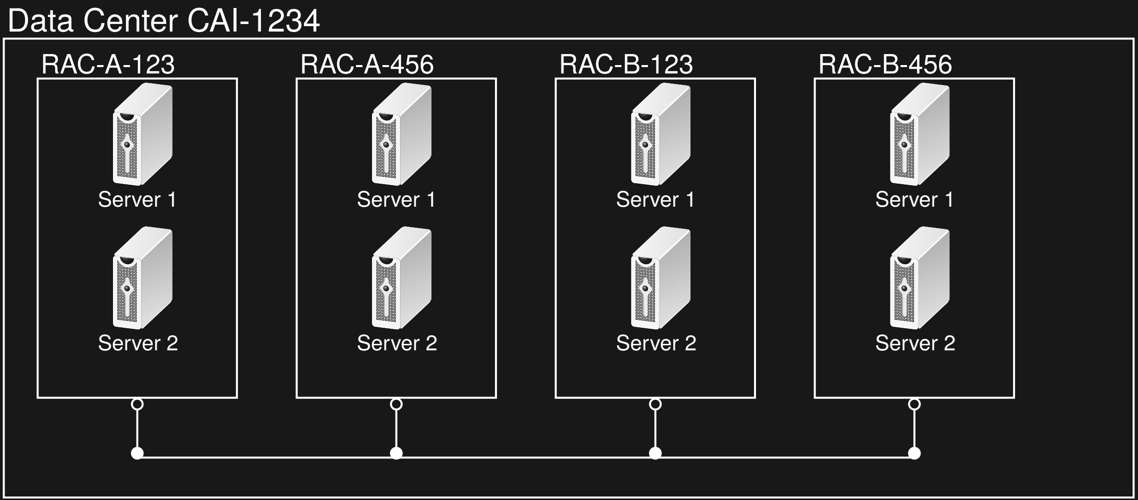
The Definitive Guide to Big Data Engineering Tasks

Previous video recap!

Core Hadoop Concepts

Hadoop Core Components

Data Center Components



Hadoop Core Concepts

- HDFS.

Hadoop Core Concepts

- HDFS.
- Map-Reduce.

Hadoop Core Concepts

- HDFS.
- Map-Reduce.
- YARN.

YARN

- YARN = Yet Another Resource Negotiator.

[1] Apache Hadoop YARN
https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

YARN

- YARN = Yet Another Resource Negotiator.
- YARN is responsible for the data-computation framework in Hadoop.

[1] Apache Hadoop YARN
https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

YARN

- YARN = Yet Another Resource Negotiator.
- YARN is responsible for the data-computation framework in Hadoop.
- The fundamental idea of YARN is to split up the functionalities of **resource management** and **job scheduling/monitoring** into separate daemons.

[1] Apache Hadoop YARN
https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

YARN

- YARN = Yet Another Resource Negotiator.
- YARN is responsible for the data-computation framework in Hadoop.
- 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)**.

[1] Apache Hadoop YARN
https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

YARN

- YARN = Yet Another Resource Negotiator.
- YARN is responsible for the data-computation framework in Hadoop.
- 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.

[1] Apache Hadoop YARN
https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

YARN

YARN allows to run multiple processing engine on the Hadoop cluster .

- Map-Reduce, Hive, and PIG.

YARN

YARN allows to run multiple processing engine on the Hadoop cluster .

- Map-Reduce, Hive, and PIG.
- Spark batch, streaming, ML, and SQL.

YARN

YARN allows to run multiple processing engine on the Hadoop cluster .

- Map-Reduce, Hive, and PIG.
- Spark batch, streaming, ML, and SQL.
- Impala, Mahoot, and other engines.

YARN

– YARN provides APIs for requesting and working with cluster resources.

[1] Hadoop the defenitive guide Ch.4 P.79.
https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[1] Apache Hadoop YARN Ch.4 P.43.
https://www.oreilly.com/library/view/apache-hadooptm-yarn/9780133441925/

YARN

- YARN provides APIs for requesting and working with cluster resources.

- These APIs are not typically used directly by user code.

---

[1]Hadoop the defenitive guide Ch.4 P.79.
https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[1]Apache Hadoop YARN Ch.4 P.43.
https://www.oreilly.com/library/view/apache-hadooptm-yarn/9780133441925/

YARN

- YARN provides APIs for requesting and working with cluster resources.
- These APIs are not typically used directly by user code.
- Users write to higher-level APIs provided by distributed computing frameworks, Ex: (Map-reduce or Spark on yarn), which themselves are built on YARN and hide the resource management details from the user..

[1] Hadoop the defenitive guide Ch.4 P.79.
https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
[1] Apache Hadoop YARN Ch.4 P.43.
https://www.oreilly.com/library/view/apache-hadooptm-yarn/9780133441925/

YARN

In YARN, there are at least three actors:

- The clinet : The Job Submitter.

YARN

In YARN, there are at least three actors:

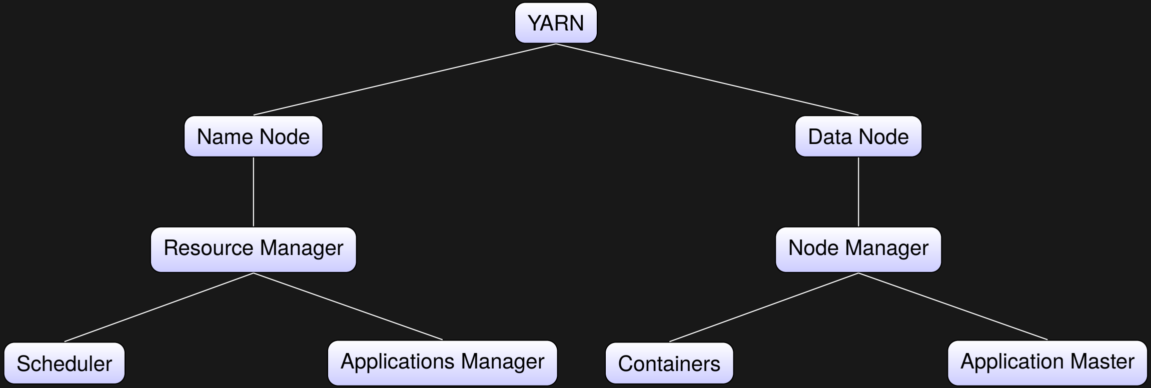
- The clinet : The Job Submitter.
- Node(s) Master: the Resource Manager.

YARN

In YARN, there are at least three actors:

- The clinet : The Job Submitter.
- Node(s) Master: the Resource Manager.
- Data Node(s): the Node Manager.

YARN Components Hierarchy



YARN Daemon

YARN (The data-computation framework) consists of

- Resource Manager (long-running daemon): It is **one per cluster** to manage the use of resources across the cluster.

YARN Daemon

YARN (The data-computation framework) consists of

- Resource Manager (long-running daemon): It is **one per cluster** to manage the use of resources across the cluster.

- Node Manager: It is running on all the nodes in the cluster to launch and monitor **containers**

Resource Manager

The Resource Manager has two main components

- Scheduler.

Resource Manager

The Resource Manager has two main components

- Scheduler.
- Applications Manager.

Resource Manager

- It runs on the master node.

Resource Manager

- It runs on the master node.
- It is the ultimate authority that arbitrates resources among all the applications in the system.

Resource Manager

- It runs on the master node.
- It is the ultimate authority that arbitrates resources among all the applications in the system.
- It is the global resource schedule.

Resource Manager

- It runs on the master node.
- It is the ultimate authority that arbitrates resources among all the applications in the system.
- It is the global resource schedule.
- It is a single point of failure in YARN. We can acheive HA with an active-standby configuration.

Resource Manager - Applications Manager

- It is responsible for accepting job submissions.

Resource Manager - Applications Manager

- It is responsible for accepting job submissions.

- It is responsible for negotiating the first container for executing the application specific **ApplicationMaster**.

Resource Manager - Applications Manager

- It is responsible for accepting job submissions.
- It is responsible for negotiating the first container for executing the application specific **ApplicationMaster**.
- After application submission

Resource Manager - Applications Manager

- It is responsible for accepting job submissions.
- It is responsible for negotiating the first container for executing the application specific **ApplicationMaster**.
- After application submission
    - It first validates the application’s specifications.

Resource Manager - Applications Manager

- It is responsible for accepting job submissions.

- It is responsible for negotiating the first container for executing the application specific **ApplicationMaster**.

- After application submission
  - It first validates the application's specifications.
  - It rejects any application that requests unsatisfiable resources for its ApplicationMaster (i.e., no node in the cluster has enough resources to run the ApplicationMaster itself).

Resource Manager - Applications Manager

- It is responsible for accepting job submissions.
- It is responsible for negotiating the first container for executing the application specific **ApplicationMaster**.
- After application submission
  - It first validates the application's specifications.
  - It rejects any application that requests unsatisfiable resources for its ApplicationMaster (i.e., no node in the cluster has enough resources to run the ApplicationMaster itself).
  - It then ensures that no other application was already submitted with the same application ID.

Resource Manager - Applications Manager

- It is responsible for accepting job submissions.
- It is responsible for negotiating the first container for executing the application specific **ApplicationMaster**.
- After application submission
  - It first validates the application's specifications.
  - It rejects any application that requests unsatisfiable resources for its ApplicationMaster (i.e., no node in the cluster has enough resources to run the ApplicationMaster itself).
  - It then ensures that no other application was already submitted with the same application ID.
  - It forwards the admitted application to the scheduler.

Resource Manager - Applications Manager

- It is responsible for recording and managing finished applications for a while before being completely evacuated from the ResourceManager’s memory.

Resource Manager - Applications Manager

- It is responsible for recording and managing finished applications for a while before being completely evacuated from the ResourceManager’s memory.

- It places an ApplicationSummary in the daemon’s log file after the application finishes.

Resource Manager - Applications Manager

- It is responsible for recording and managing finished applications for a while before being completely evacuated from the ResourceManager’s memory.
- It places an ApplicationSummary in the daemon’s log file after the application finishes.
- Finally, the ApplicationsManager keeps a cache of completed applications long after applications finish to support users’ requests for application data

Resource Manager - Scheduler

- It is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues etc.

Resource Manager - Scheduler

- It is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues etc.

- It performs its scheduling function based on the resource requirements of the applications.

Resource Manager - Scheduler

- It is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues etc.

- It performs its scheduling function based on the resource requirements of the applications.

- It does so based on the abstract notion of a resource **containers** which incorporates elements such as memory, cpu, disk, network etc.

Resource Manager - Scheduler

- It is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues etc.

- It performs its scheduling function based on the resource requirements of the applications.

- It does so based on the abstract notion of a resource **containers** which incorporates elements such as memory, cpu, disk, network etc.

- The current schedulers such as the **CapacityScheduler** and the **FairScheduler** would be some examples of plug-ins.

Node Manager

The Node Manager has two main components

- Containers

Node Manager

The Node Manager has two main components

- Containers
- Application Master (AM)

Node Manager

– It runs on the data node.

Node Manager

- It runs on the data node.
- It is YARN’s per-node “worker” agent, taking care of the individual compute nodes in a Hadoop cluster.

Node Manager

- It runs on the data node.
- It is YARN’s per-node “worker” agent, taking care of the individual compute nodes in a Hadoop cluster.
- On start-up, the NodeManager registers with the ResourceManager; it then sends heartbeats with its status and waits for instructions.

Node Manager

- It runs on the data node.
- It is YARN’s per-node “worker” agent, taking care of the individual compute nodes in a Hadoop cluster.
- On start-up, the NodeManager registers with the ResourceManager; it then sends heartbeats with its status and waits for instructions.
- Its primary goal is to manage application containers assigned to it by the ResourceManager.

Node Manager

Node Manager is responsilble for

- Node Status Updater: Keeping up-to-date with the ResourceManage.

Node Manager

Node Manager is responsilble for

- Node Status Updater: Keeping up-to-date with the ResourceManage.

- Container Manager: Overseeing application containers' life-cycle management, and monitoring resource usage (memory, CPU) of individual containers.

Node Manager

Node Manager is responsilble for

- Node Status Updater: Keeping up-to-date with the ResourceManage.

- Container Manager: Overseeing application containers' life-cycle management, and monitoring resource usage (memory, CPU) of individual containers.

- Node Health Checker Service: Tracking node health.

Node Manager

Node Manager is responsilble for

- Node Status Updater: Keeping up-to-date with the ResourceManage.
- Container Manager: Overseeing application containers' life-cycle management, and monitoring resource usage (memory, CPU) of individual containers.
- Node Health Checker Service: Tracking node health.
- Log Handler: keeping the containers' logs, and uploading them onto a file-system.

Node Manager - Containers

- A container is a collection of physical resources such as RAM, CPU cores, and disks on a single node.

Node Manager - Containers

- A container is a collection of physical resources such as RAM, CPU cores, and disks on a single node.

- There can be multiple containers on a single node.

Node Manager - Containers

- A container is a collection of physical resources such as RAM, CPU cores, and disks on a single node.

- There can be multiple containers on a single node.

- Every node in the system is considered to be composed of multiple containers of minimum size of memory (e.g., 512 MB or 1 GB) and CPU.

Node Manager - Containers

- A container is a collection of physical resources such as RAM, CPU cores, and disks on a single node.

- There can be multiple containers on a single node.

- Every node in the system is considered to be composed of multiple containers of minimum size of memory (e.g., 512 MB or 1 GB) and CPU.

- A container executes an application-specific process with a constrained set of resources(memory, CPU, and so on).

Node Manager - Containers

- A container is a collection of physical resources such as RAM, CPU cores, and disks on a single node.
- There can be multiple containers on a single node.
- Every node in the system is considered to be composed of multiple containers of minimum size of memory (e.g., 512 MB or 1 GB) and CPU.
- A container executes an application-specific process with a constrained set of resources(memory, CPU, and so on).
- RM is creating containers based on the application requirements.

Node Manager - Containers

- A container is a collection of physical resources such as RAM, CPU cores, and disks on a single node.
- There can be multiple containers on a single node.
- Every node in the system is considered to be composed of multiple containers of minimum size of memory (e.g., 512 MB or 1 GB) and CPU.
- A container executes an application-specific process with a constrained set of resources(memory, CPU, and so on).
- RM is creating containers based on the application requirements.
- Applicalons run in one or more containers.

Node Manager - Containers

- A container is a collection of physical resources such as RAM, CPU cores, and disks on a single node.
- There can be multiple containers on a single node.
- Every node in the system is considered to be composed of multiple containers of minimum size of memory (e.g., 512 MB or 1 GB) and CPU.
- A container executes an application-specific process with a constrained set of resources(memory, CPU, and so on).
- RM is creating containers based on the application requirements.
- Applicalons run in one or more containers.

Node Manager - Containers

- Each application starts out as an ApplicationMaster, which is itself a container (often referred to as container 0).

Node Manager - Containers

- Each application starts out as an ApplicationMaster, which is itself a container (often referred to as container 0).
- Once started, the ApplicationMaster must negotiate with the ResourceManager for more containers.

Node Manager - Containers

- Each application starts out as an ApplicationMaster, which is itself a container (often referred to as container 0).

- Once started, the ApplicationMaster must negotiate with the ResourceManager for more containers.

- Container requests (and releases) can take place in a dynamic fashion at run time. For instance, a MapReduce job may request a certain amount of mapper containers; as they finish their tasks, it may release them and request more reducer containers to be started.

Node Manager - Application Master

- The ApplicationMaster is the process that coordinates an application’s execution in the cluster (It runs in container 0).

Node Manager - Application Master

- The ApplicationMaster is the process that coordinates an application’s execution in the cluster (It runs in container 0).

- Each application has its own unique ApplicationMaster (one per applicaIon), which is tasked with negotiating resources (containers) from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks.

Node Manager - Application Master

- The ApplicationMaster is the process that coordinates an application’s execution in the cluster (It runs in container 0).

- Each application has its own unique ApplicationMaster (one per applicaIon), which is tasked with negotiating resources (containers) from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks.

- It will periodically send heartbeats to the ResourceManager to affirm its health and to update the record of its resource demands.

Node Manager - Application Master

- The ApplicationMaster is the process that coordinates an application’s execution in the cluster (It runs in container 0).

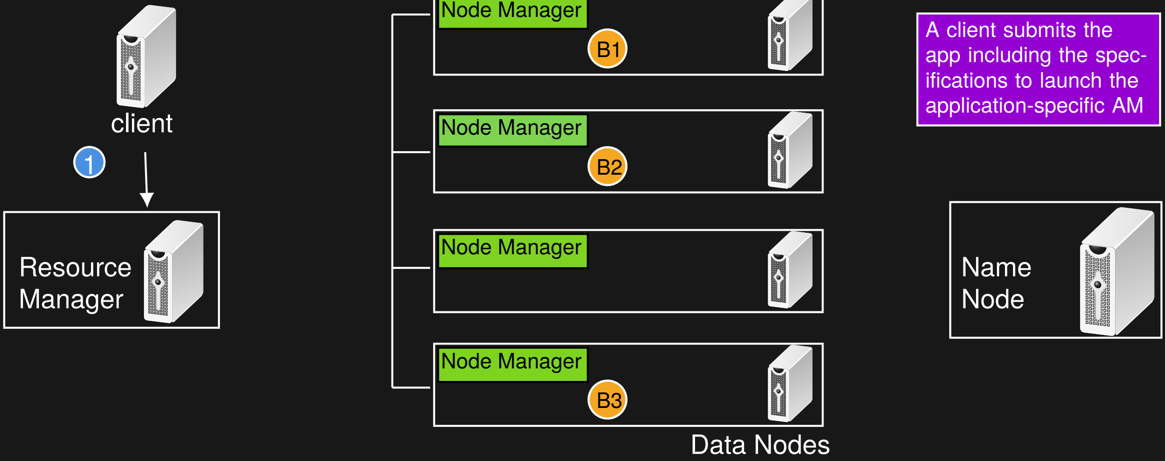
- Each application has its own unique ApplicationMaster (one per applicaIon), which is tasked with negotiating resources (containers) from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks.

- It will periodically send heartbeats to the ResourceManager to affirm its health and to update the record of its resource demands.

- It is framework/applicaIon specific.

Running an Application on YARN

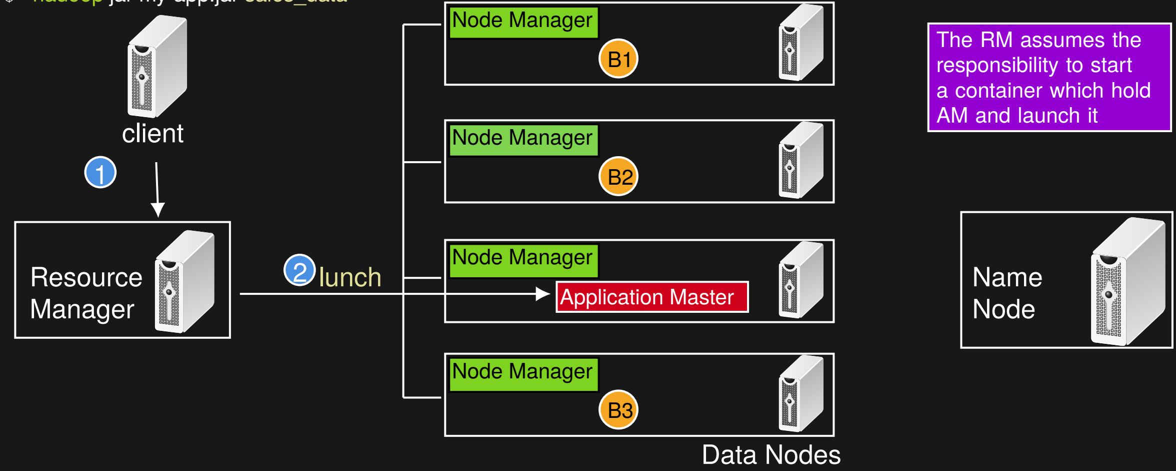
```
$  hadoop jar my-app.jar sales_data
```



A client submits the app including the specifications to launch the application-specific AM

Running an Application on Hadoop-YARN

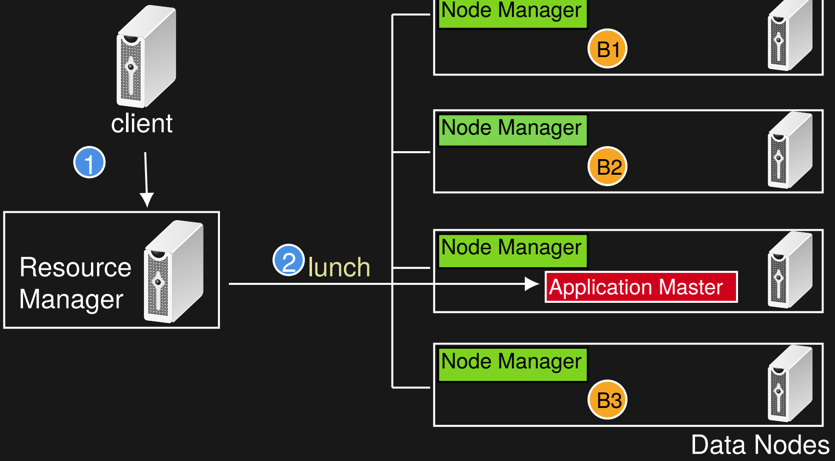
```
$  hadoop jar my-app.jar sales_data
```



The RM assumes the responsibility to start a container which hold AM and launch it

Running an Application on Hadoop-YARN

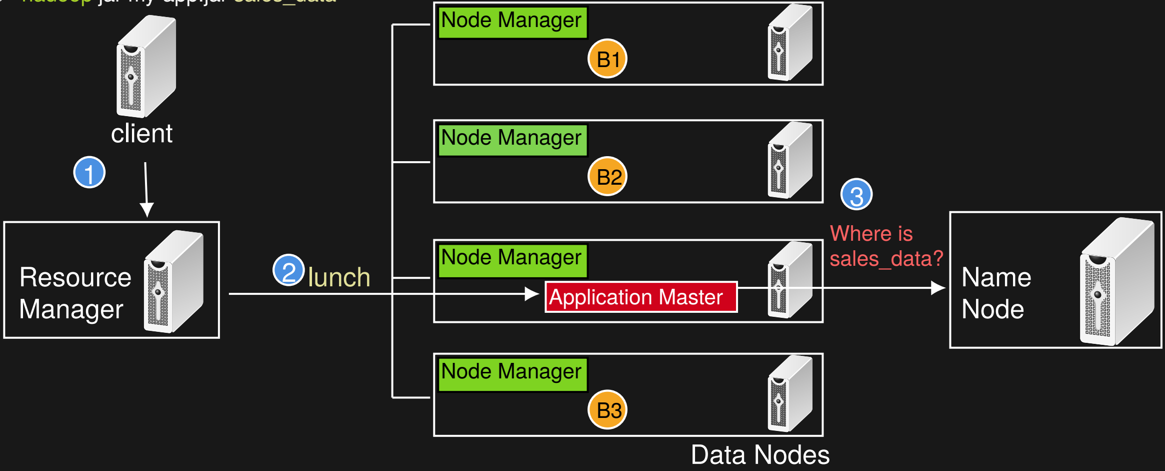
```
$   hadoop jar my-app.jar sales_data
```



On The AM boot up, it registers with the RM. This registration allows the client program to query the RM for details

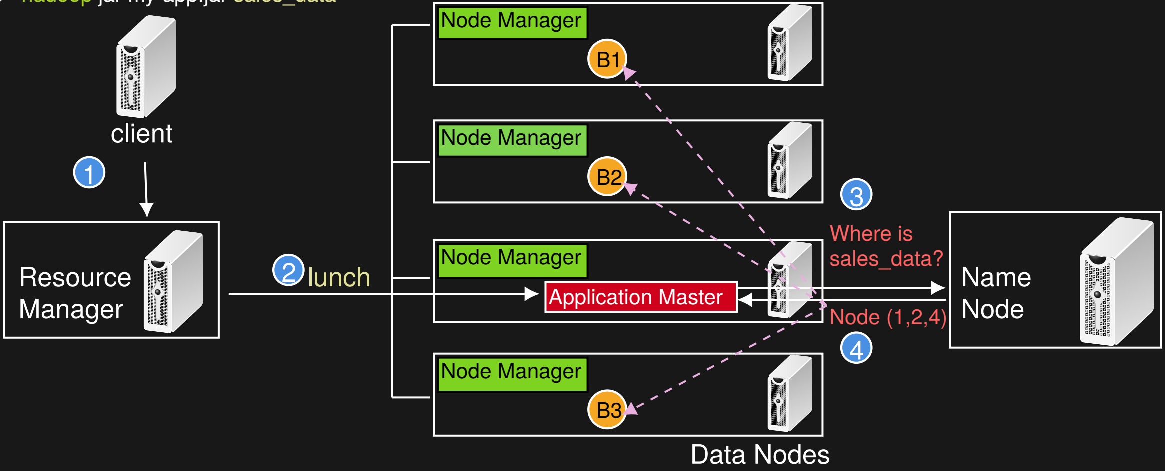
Running an Application on Hadoop-YARN

```
$ hadoop jar my-app.jar sales_data
```



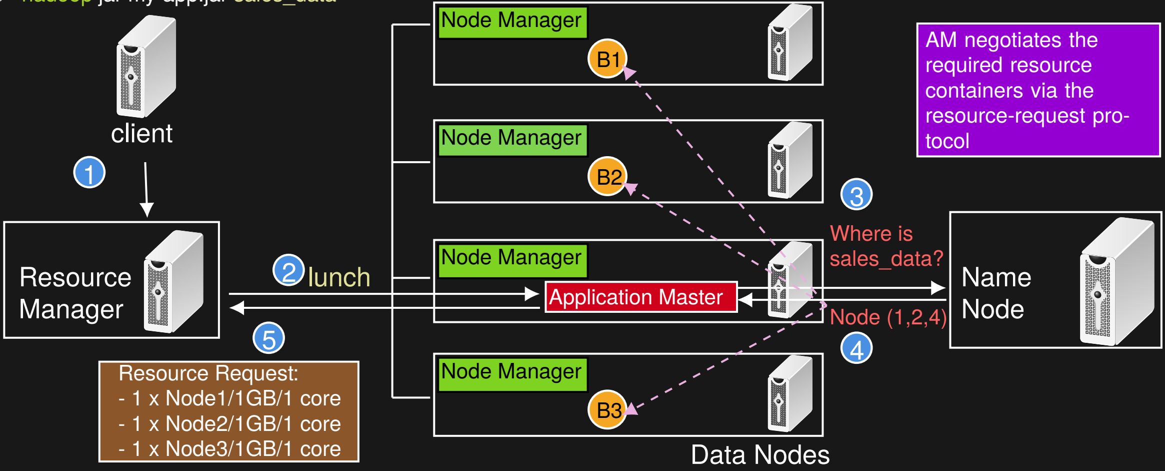
Running an Application on Hadoop-YARN

```
$  hadoop jar my-app.jar sales_data
```



Running an Application on Hadoop-YARN

```
$  hadoop jar my-app.jar sales_data
```

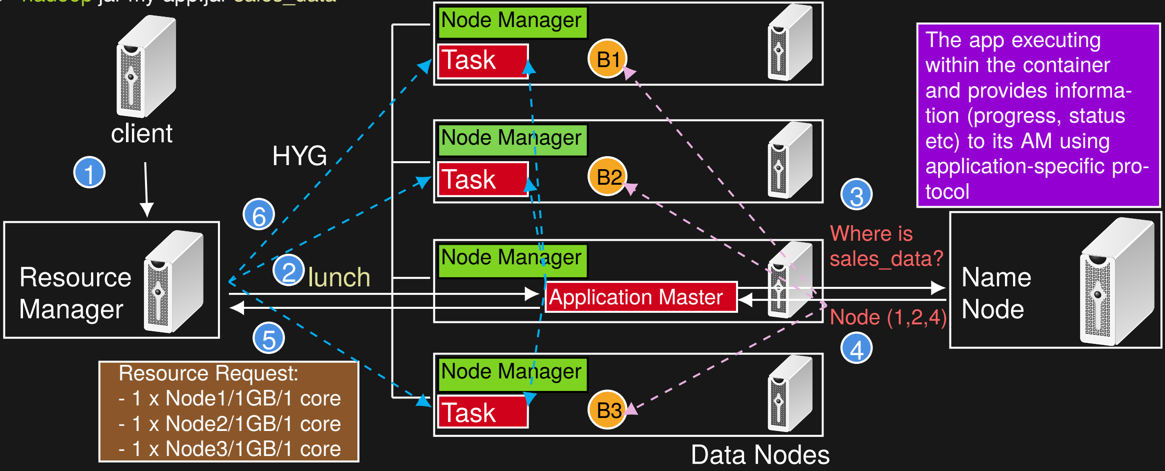


AM negotiates the required resource containers via the resource-request protocol

Resource Request:
- 1 x Node1/1GB/1 core
- 1 x Node2/1GB/1 core
- 1 x Node3/1GB/1 core

Running an Application on Hadoop-YARN

```
$   hadoop jar my-app.jar sales_data
```

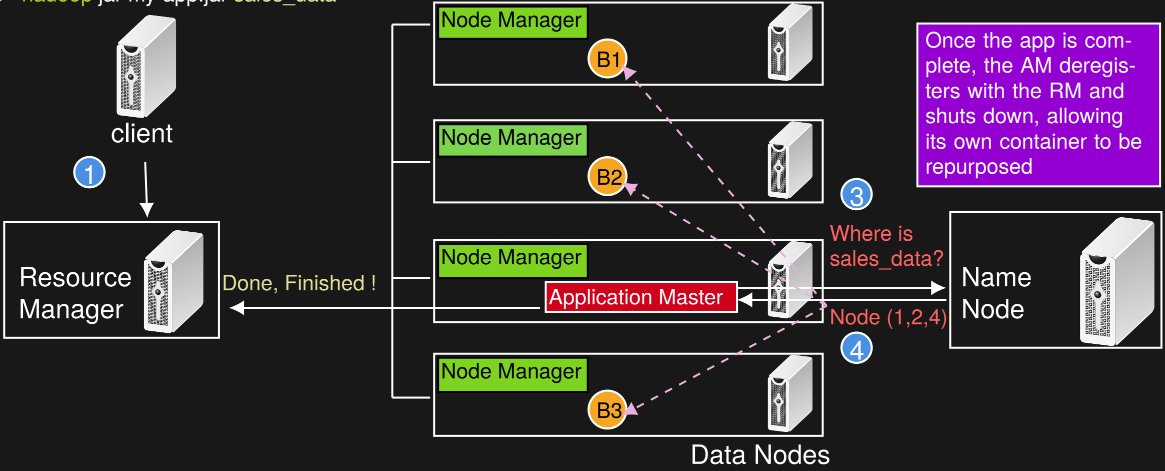


The app executing within the container and provides information (progress, status etc) to its AM using application-specific protocol

Resource Request:
- 1 x Node1/1GB/1 core
- 1 x Node2/1GB/1 core
- 1 x Node3/1GB/1 core

Running an Application on Hadoop-YARN

```
$  hadoop jar my-app.jar sales_data
```



Once the app is complete, the AM deregisters with the RM and shuts down, allowing its own container to be repurposed

Thank you for watching!

See you in the next video ☺