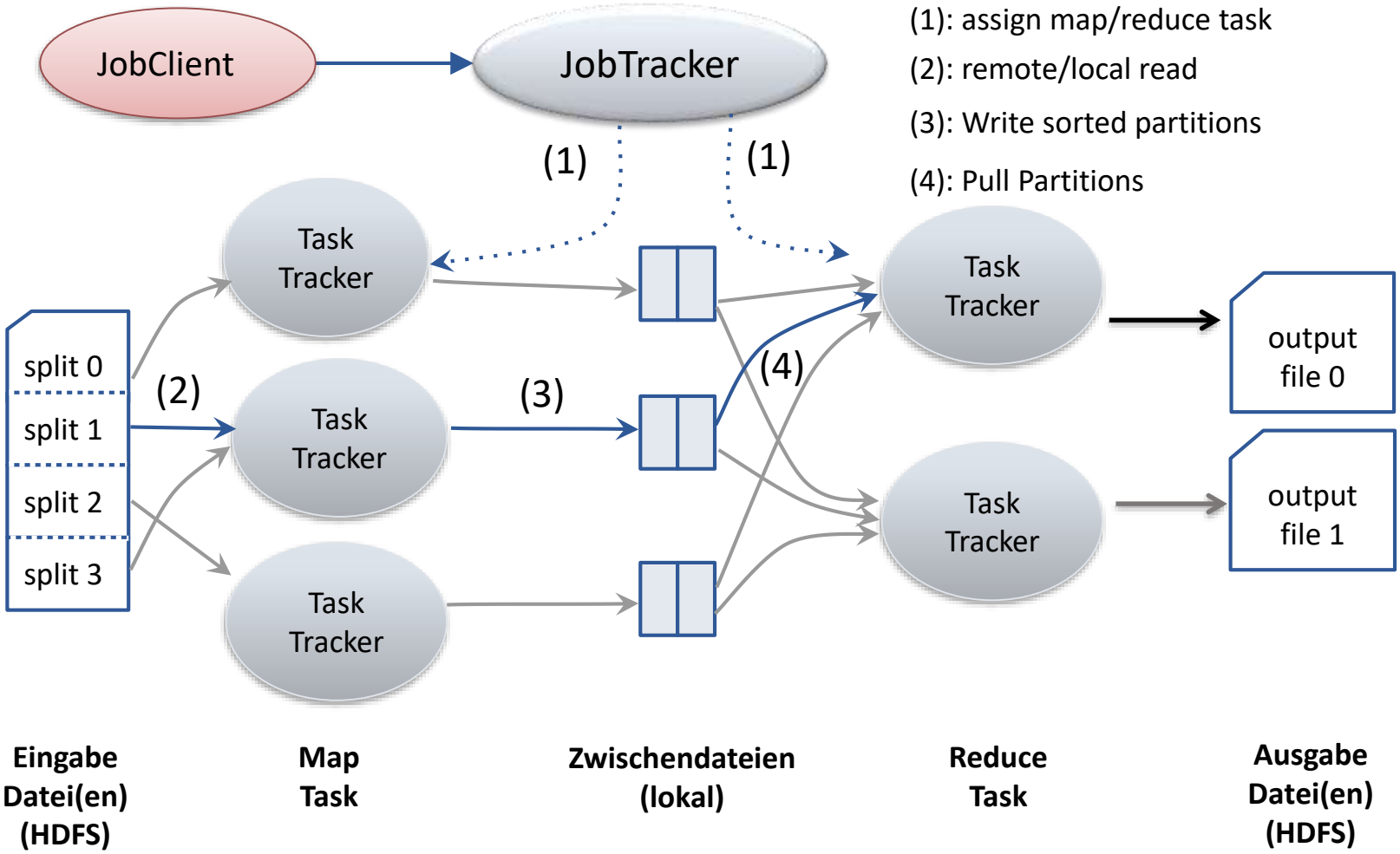
YARN
Yet Another Resource Negotiator

Prof. Dr. Stephan Trahasch
Hochschule Offenburg

Outline

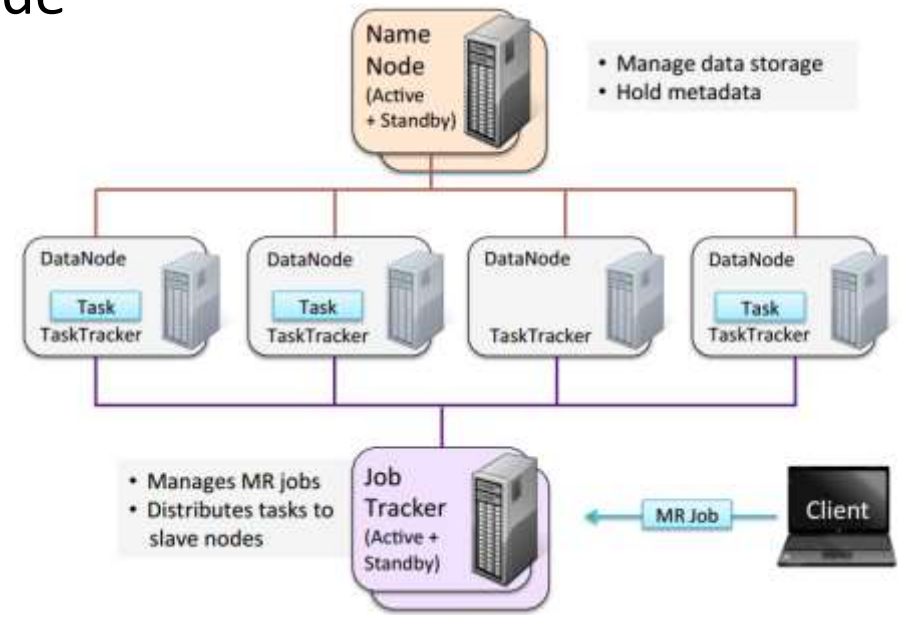
- Introduction to YARN
- Lifecyclye of an Application
- Working With YARN

MapReduce: Job execution Hadoop 1.0



Hadoop MapReduce 1.0

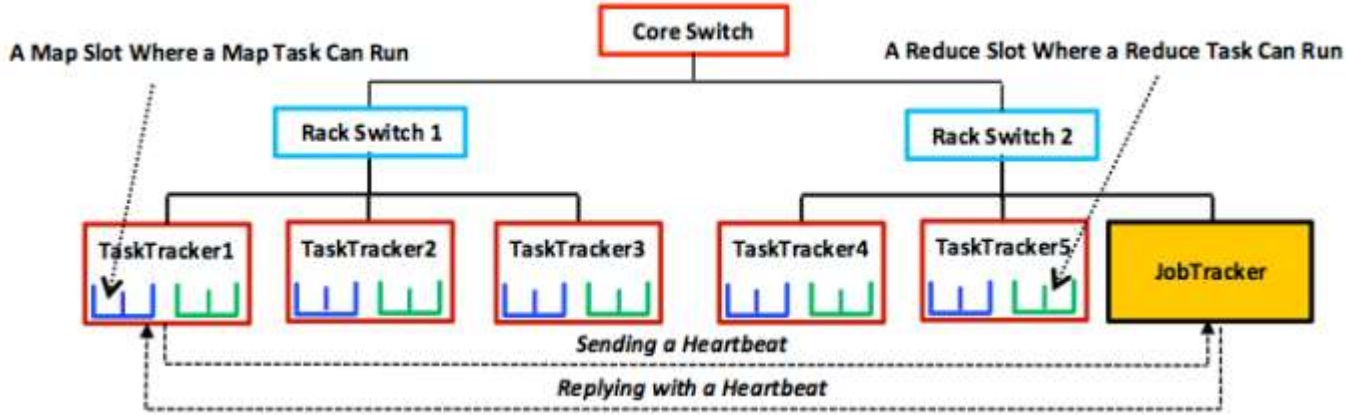
- Designed to run MapReduce jobs only
- JobTracker - Single master for all jobs and one per cluster
  - Resource allocator and job scheduler
  - Manages MapReduce jobs, distributes individual tasks to TaskTrackers
- TaskTracker – one per slave node
  - Starts and monitors individual Map and Reduce tasks
  - Static resource allocation at slaves



Source: Cloudera

Limitations Hadoop MapReduce 1.0

- Single Programming Model: MapReduce
- Centralized handling of jobs
  - SPOF, JobTracker failure kills all running & pending jobs
  - Scalability concerns with a single JobTracker
- Resources (task slots) were specific to either
  - Map tasks
  - Reduce tasks

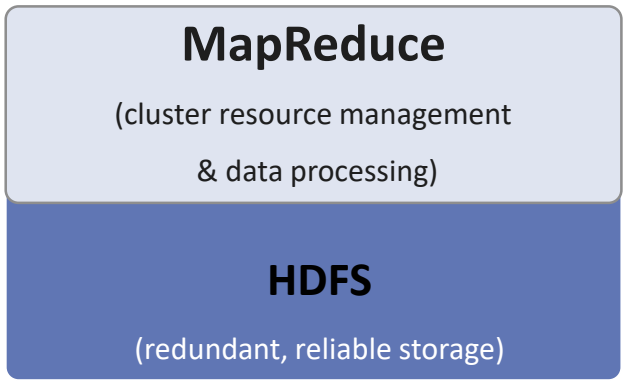


From Hadoop 1.0 to Hadoop 2.0

Single Use System

Batch Apps

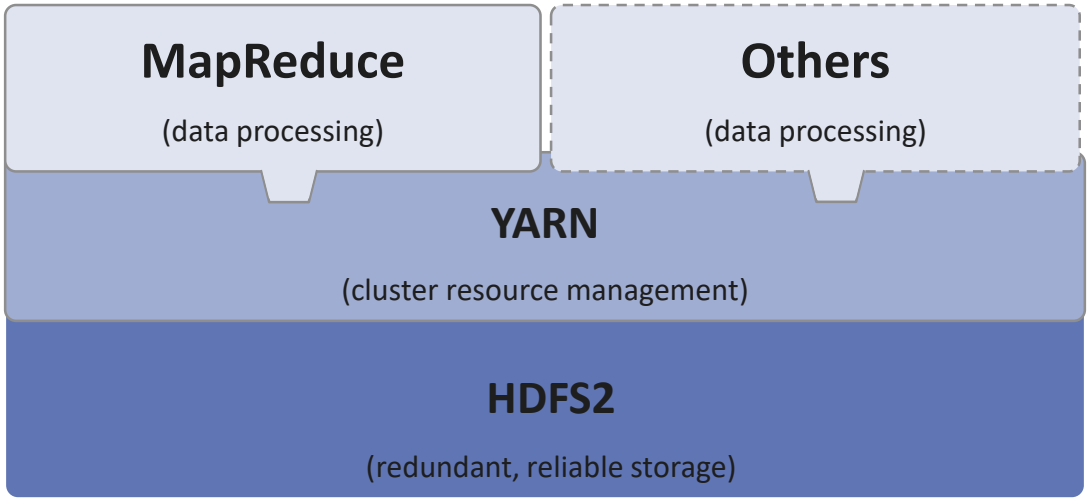
HADOOP 1.0



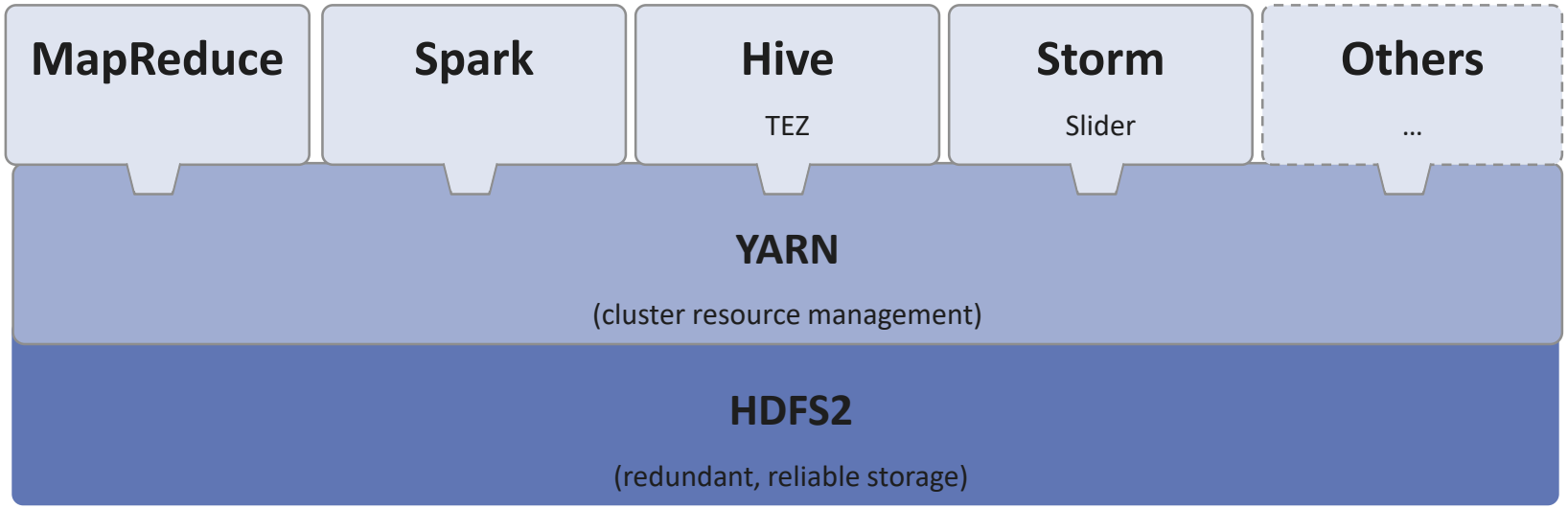
Multi Purpose Platform

Batch, Interactive, Online, Streaming, …

HADOOP 2.0



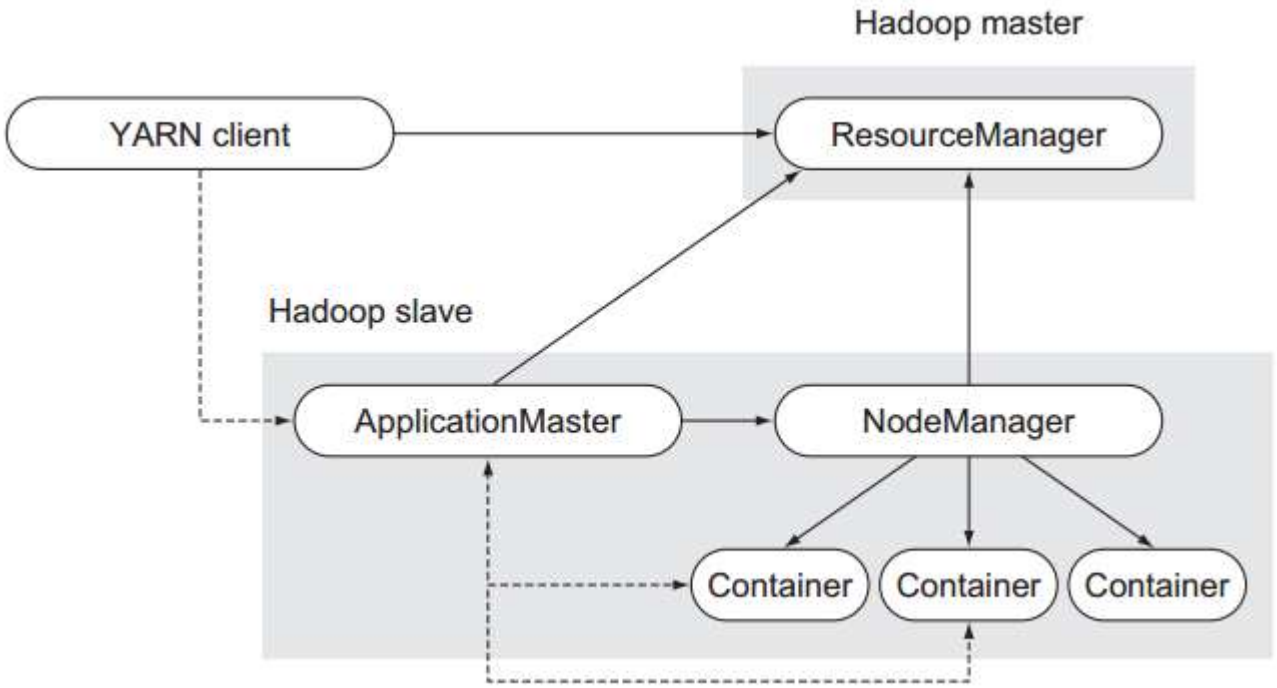
YARN - Yet Another Resource Negotiator



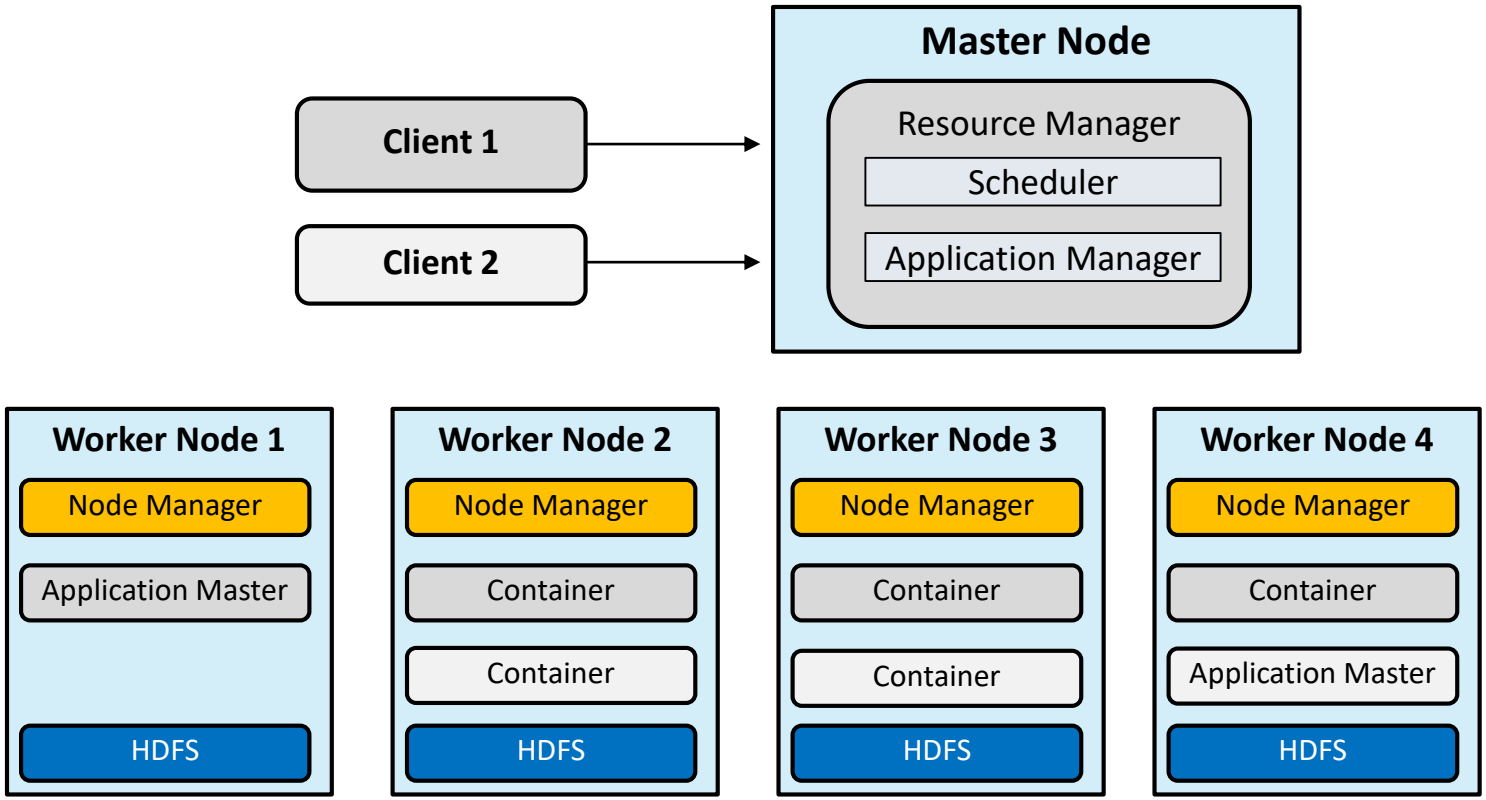
- Cluster Resource Management for distributed data processing
- Dynamic Allocation of resources for tasks.
- allows multiple data processing engines to run on a single Hadoop cluster: MapReduce, Spark, Tez, Storm, Giraph …

YARN - Splits up the two major responsibilities into separate entities

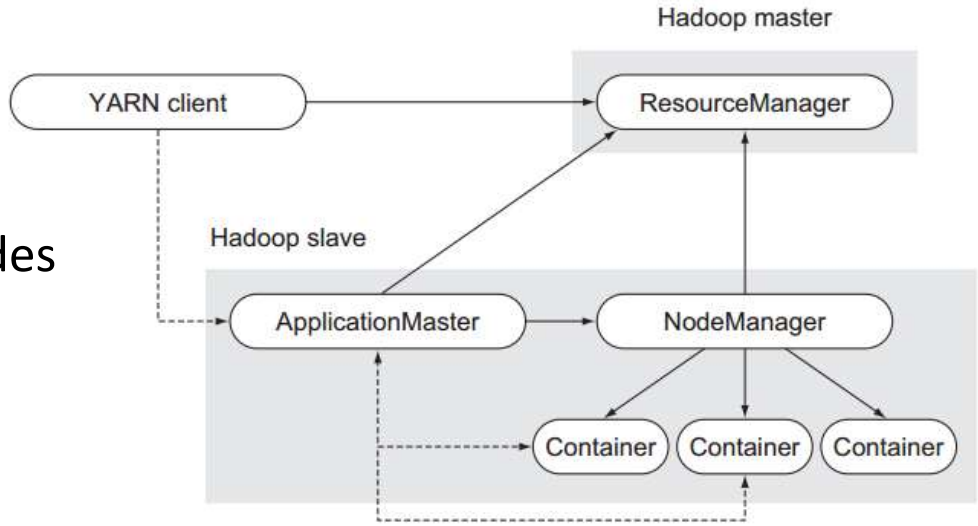
- a global ResourceManager (RM)
- a per-application ApplicationMaster (AM)
- a per-node slave NodeManager (NM)
- a per-application Container running on a NodeManager



Example


Client 1
Client 2
Master Node
Resource Manager
Scheduler
Application Manager
Worker Node 1
Node Manager
Application Master
HDFS
Worker Node 2
Node Manager
Container
Container
HDFS
Worker Node 3
Node Manager
Container
Container
HDFS
Worker Node 4
Node Manager
Container
Application Master
HDFS

Responsibility



1. **Resource Manager**
   - One per Cluster
   - Starts Application Masters
   - Allocates Resources on Slave Nodes
2. **Application Master**
   - One per Job (Application)
   - Framework/application specific
   - Runs in a container
   - Requests more containers to run application tasks
3. **Node Manager**
   - One per Slave
   - Managed Resources on a Slave Node
4. **Container**
   - Created by the RM upon request
   - Allocate a certain amount of resources (memory, CPU) on slave node
   - Applications run in one or more containers

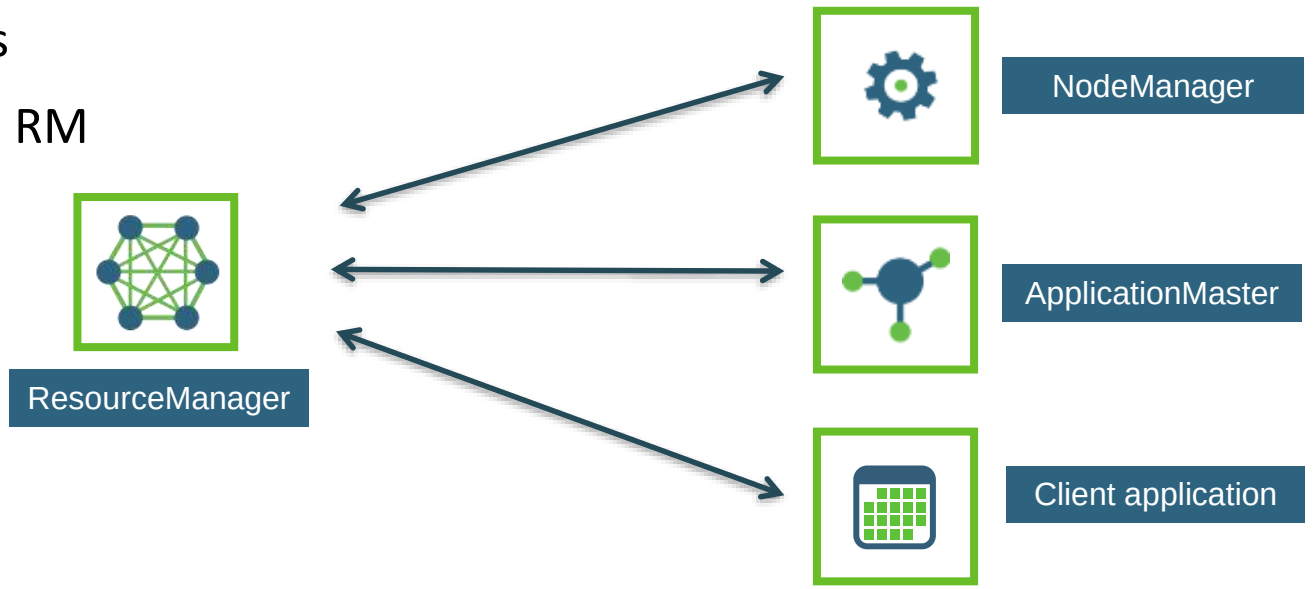
Resource Manager and Node Manager

Resource Manager (RM)

- Runs on master node and is the global resource scheduler
- scheduler supports different algorithms (capacity, fair scheduler, etc.)
- communicates with the NodeManagers, ApplicationMasters, and Client applications

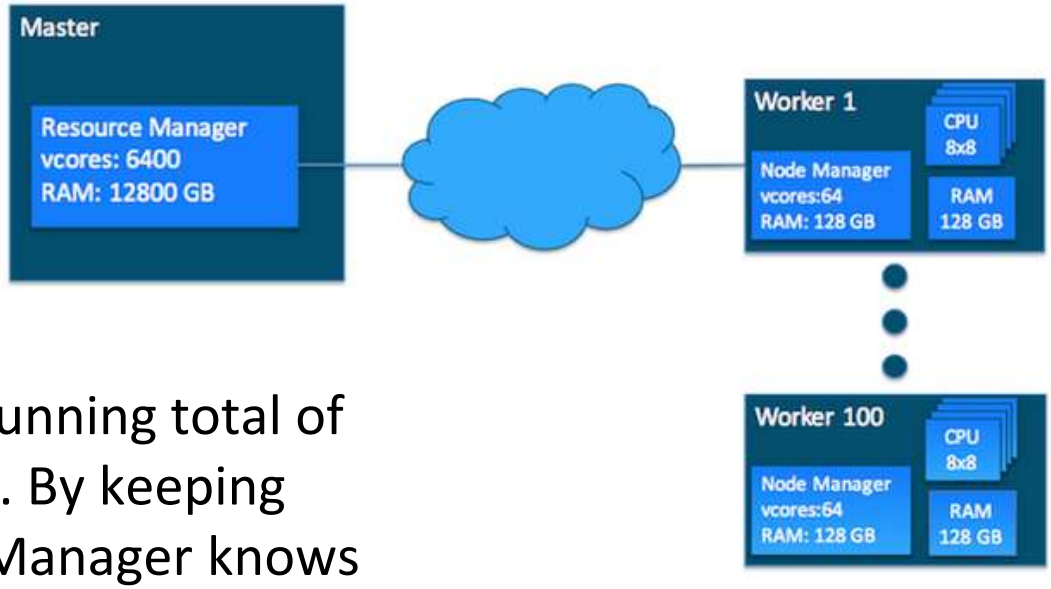
Node Manager (NM)

- Runs on slave nodes
- Communicates with RM



Resources: vcores and memory

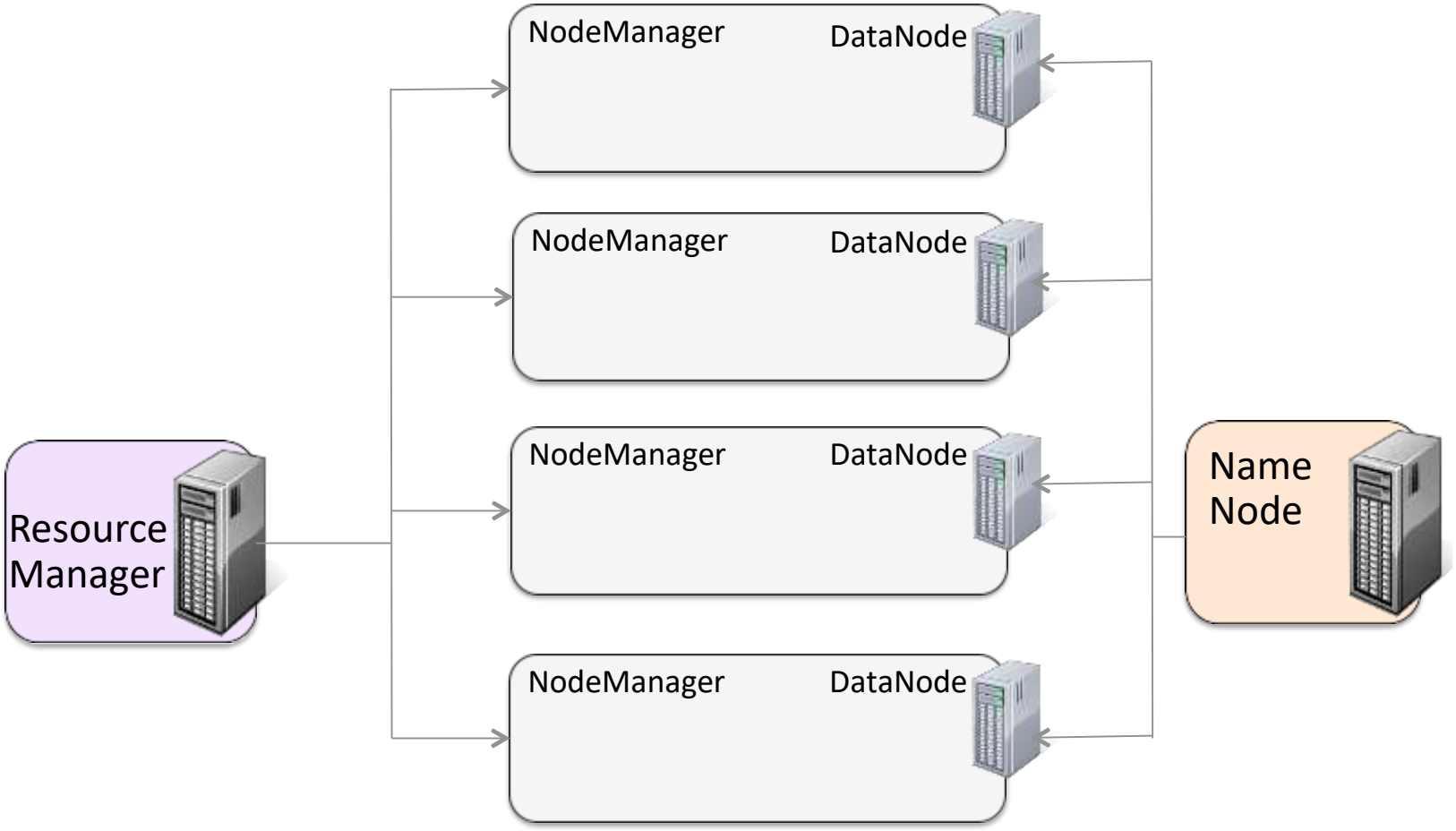
- NodeManager tracks its own local resources and communicates its resource configuration to ResourceManager



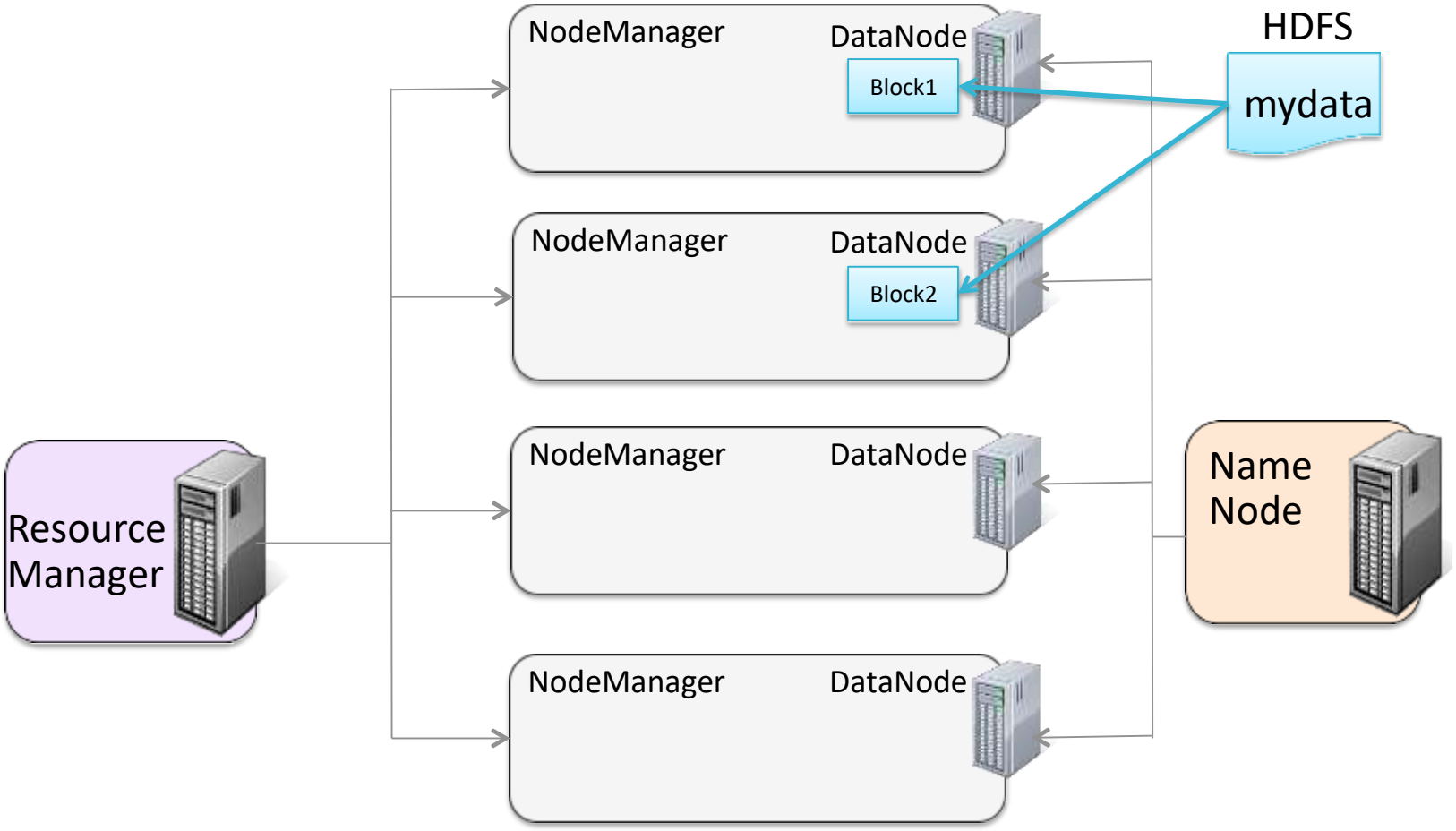
- ResourceManager keeps a running total of cluster's available resources. By keeping track of the total, ResourceManager knows how to allocate resources as they are requested.

- A container hold request consists of vcore and memory

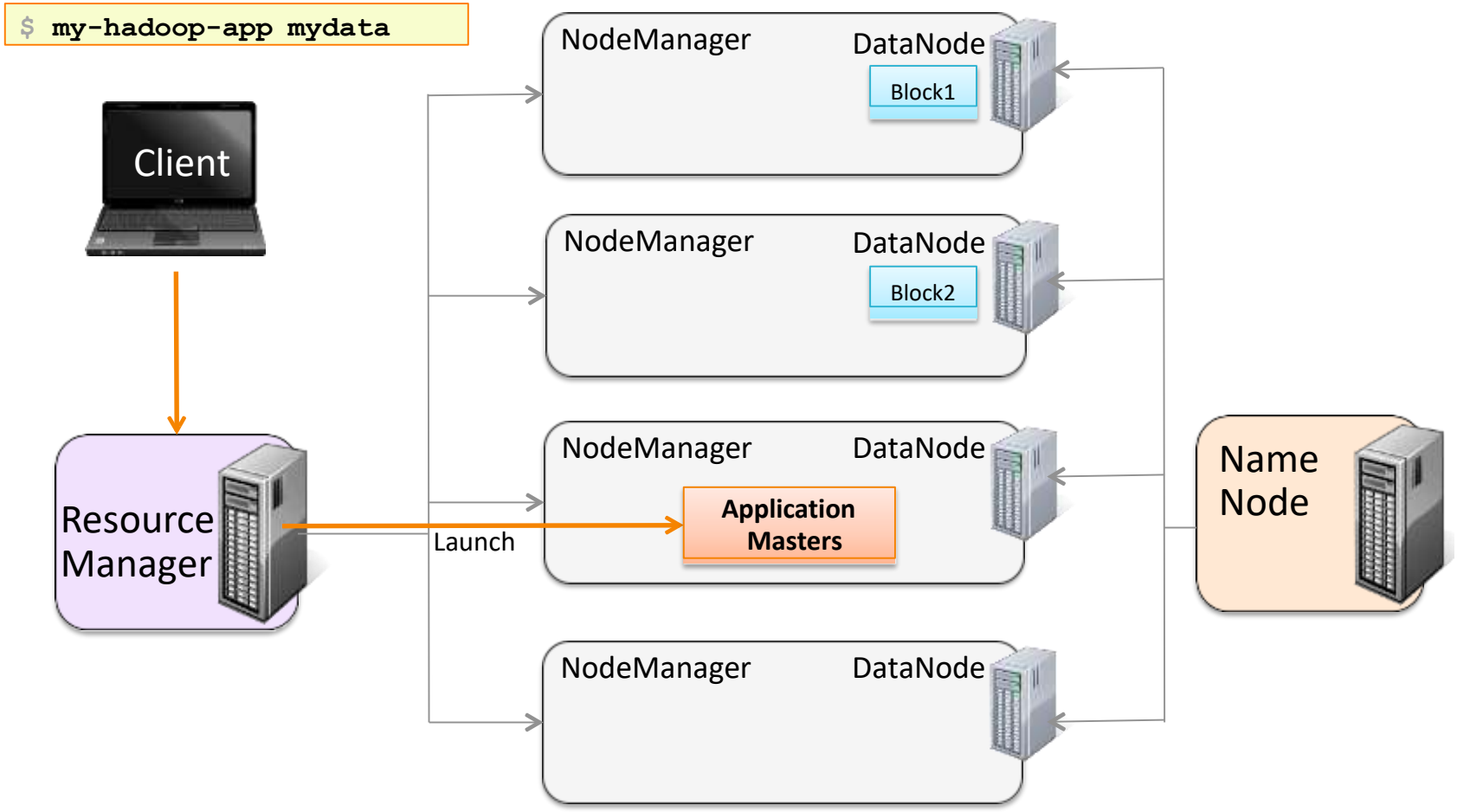
Running an Application on YARN (1)



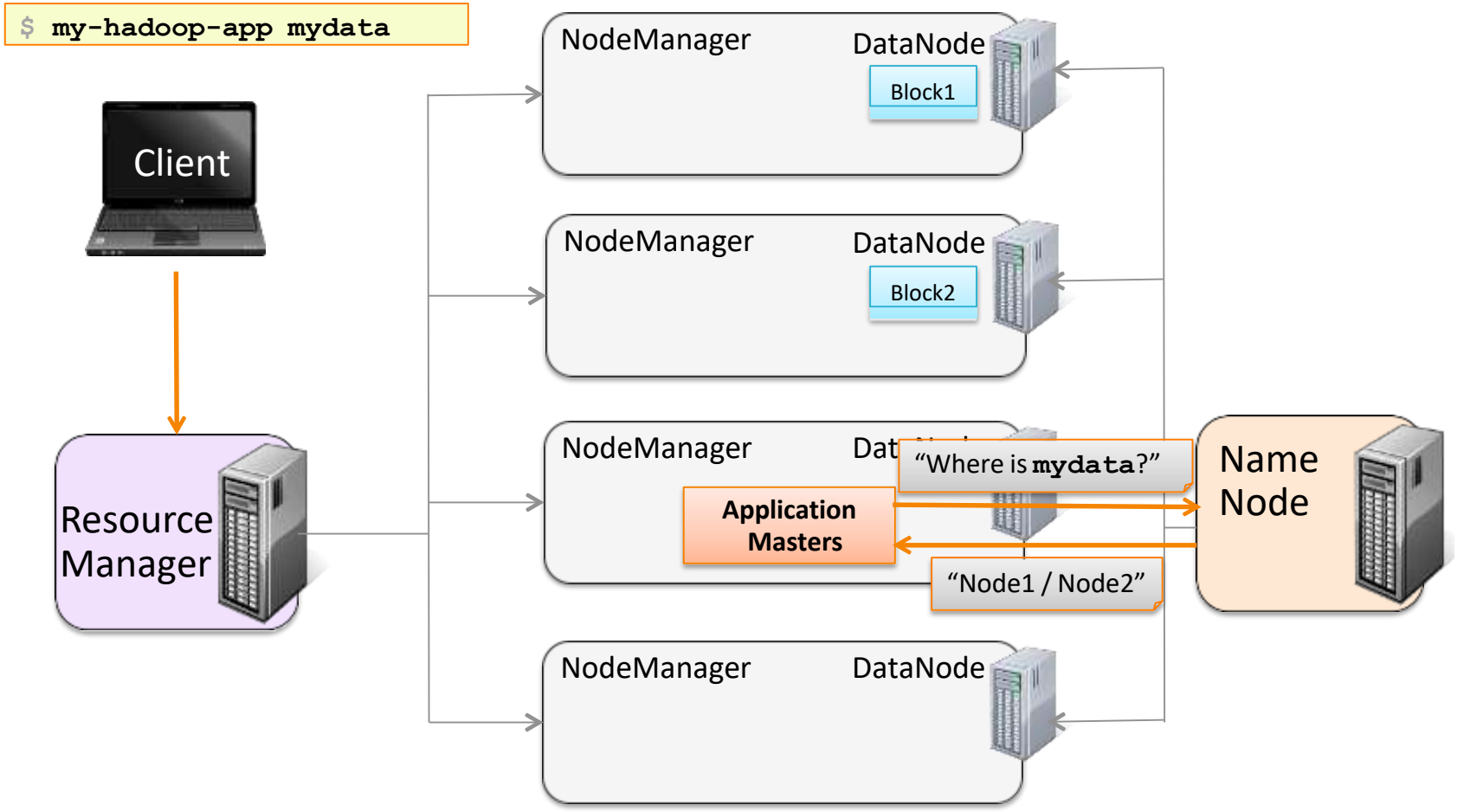
Running an Application on YARN (2)



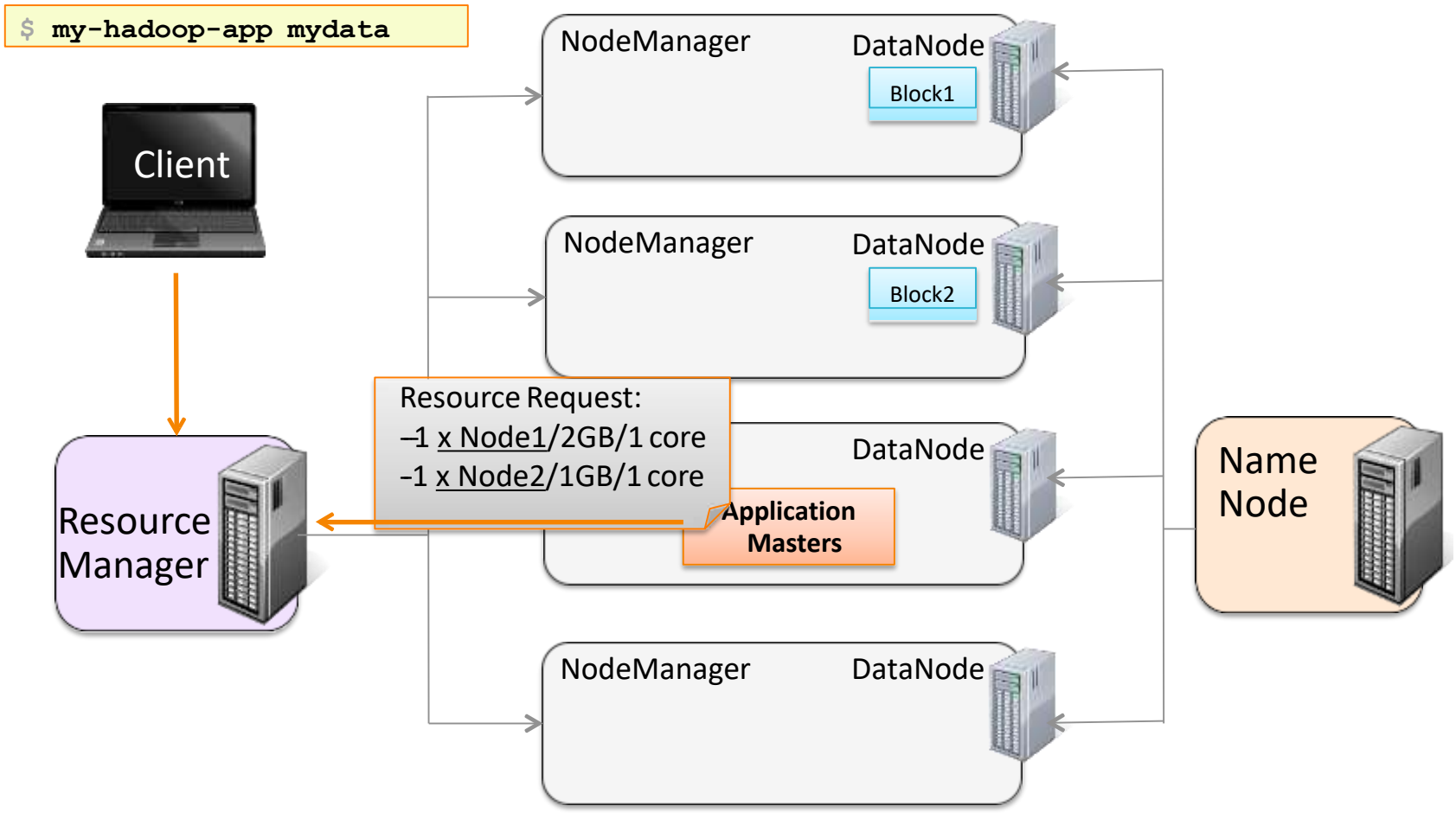
Running an Application on YARN (3)



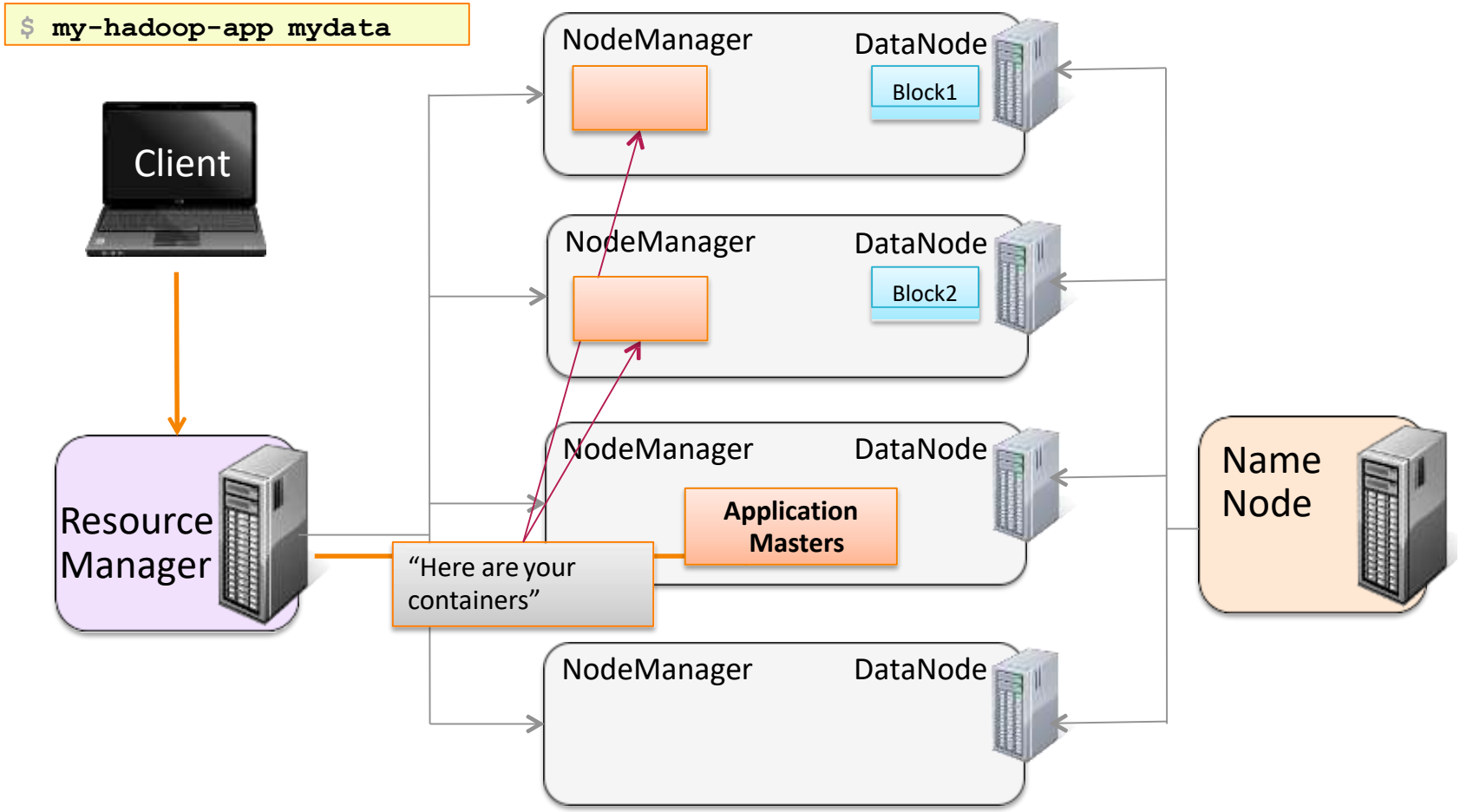
Running an Application on YARN (4)



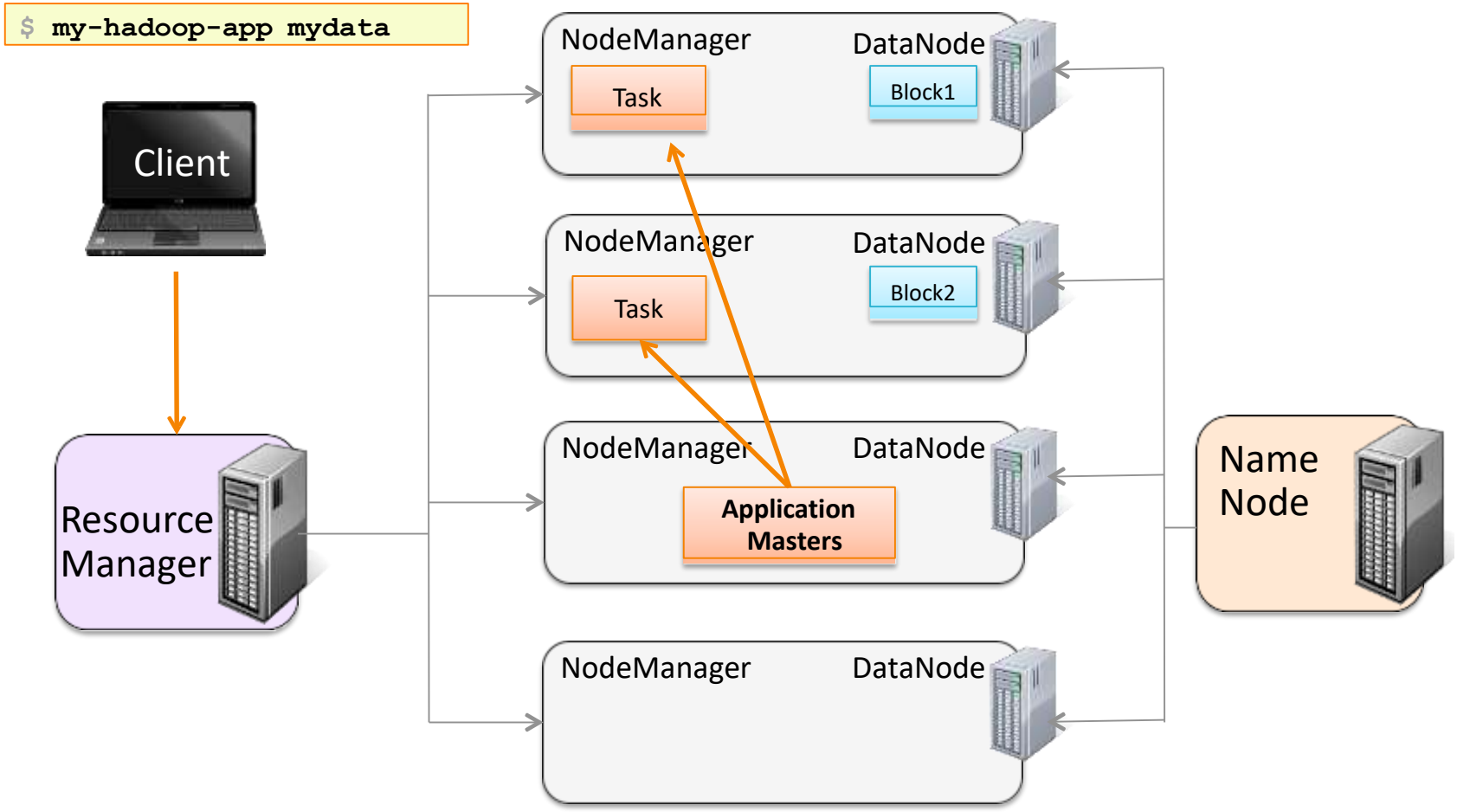
Running an Application on YARN (5)



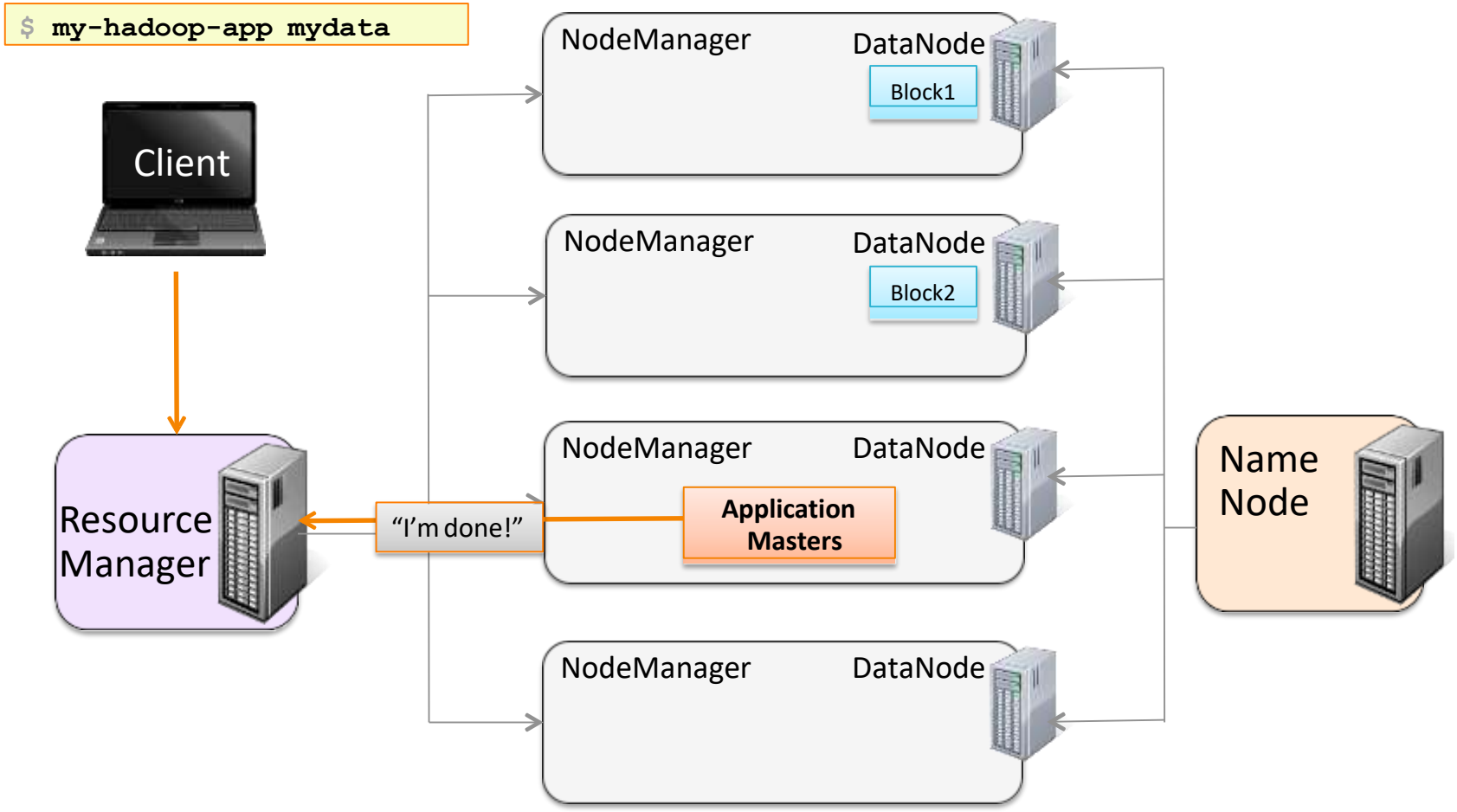
Running an Application on YARN (6)



Running an Application on YARN (7)

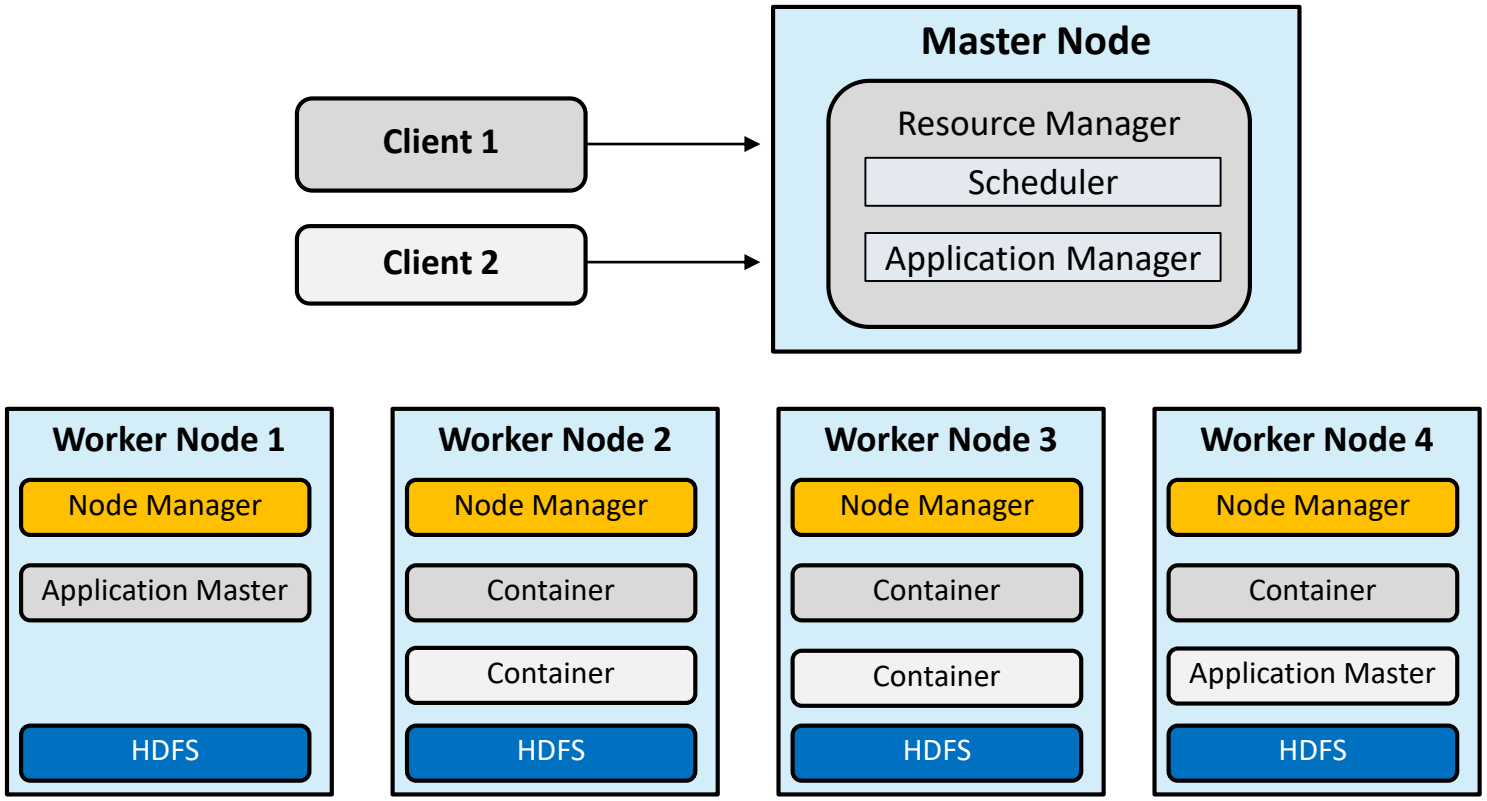


Running an Application on YARN (8)

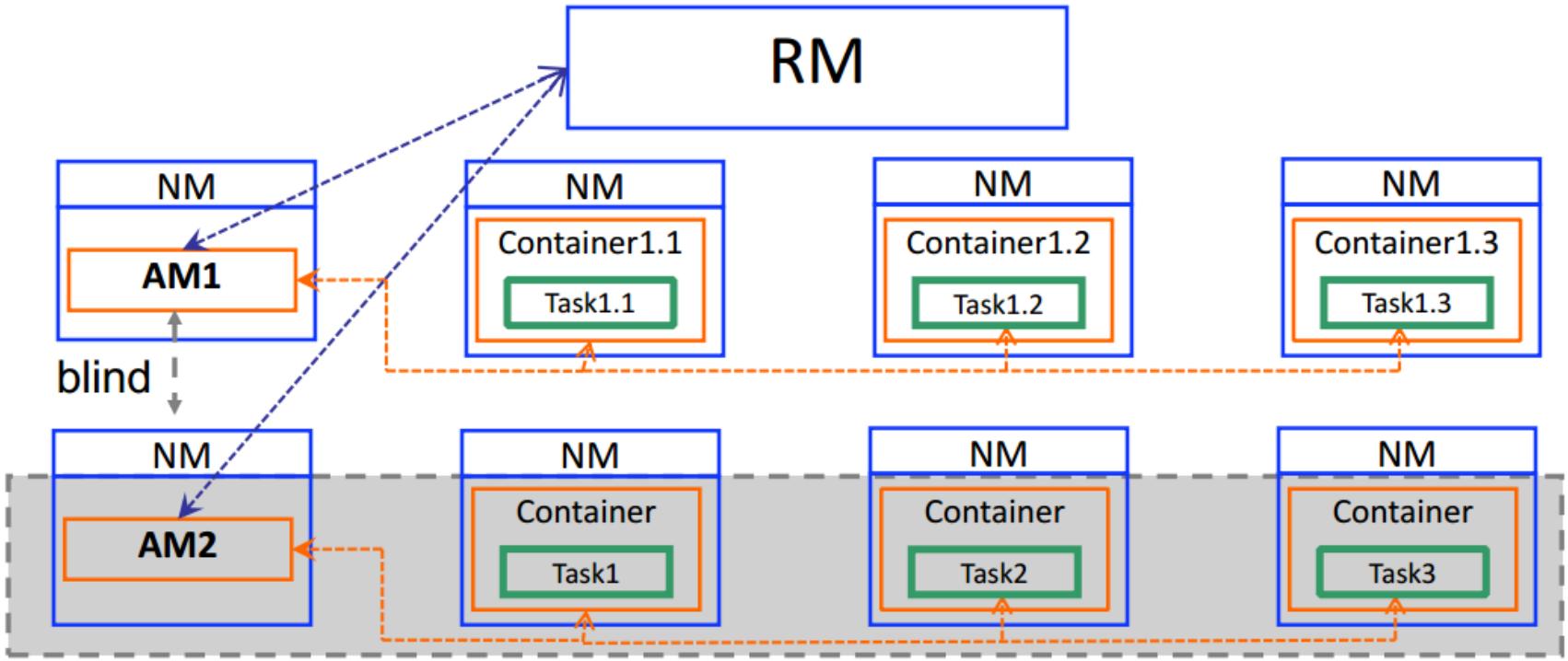


Resource Manager consist of Scheduler and Application Master

- **Scheduler**: Manages and enforces the resource scheduling policy
- **Application Manager:** Manages running the Application Master Starting, Monitoring, Restarting Failed AM



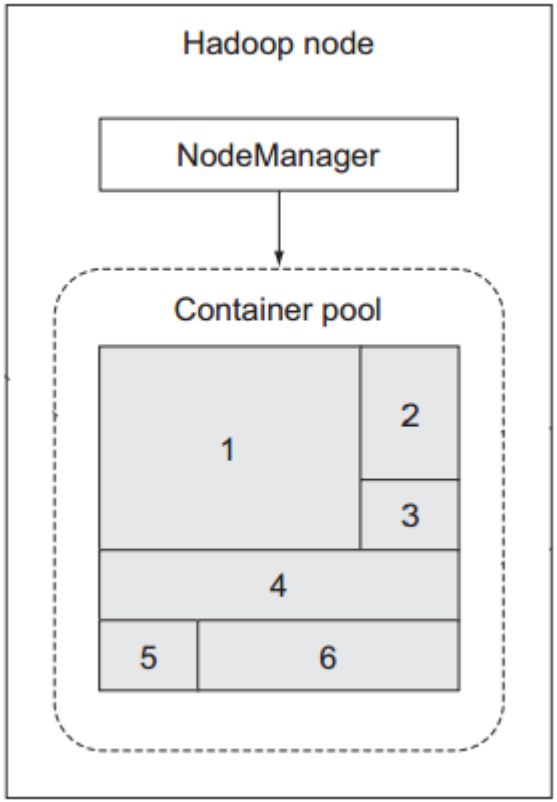
Visibility of Application Masters (AM)



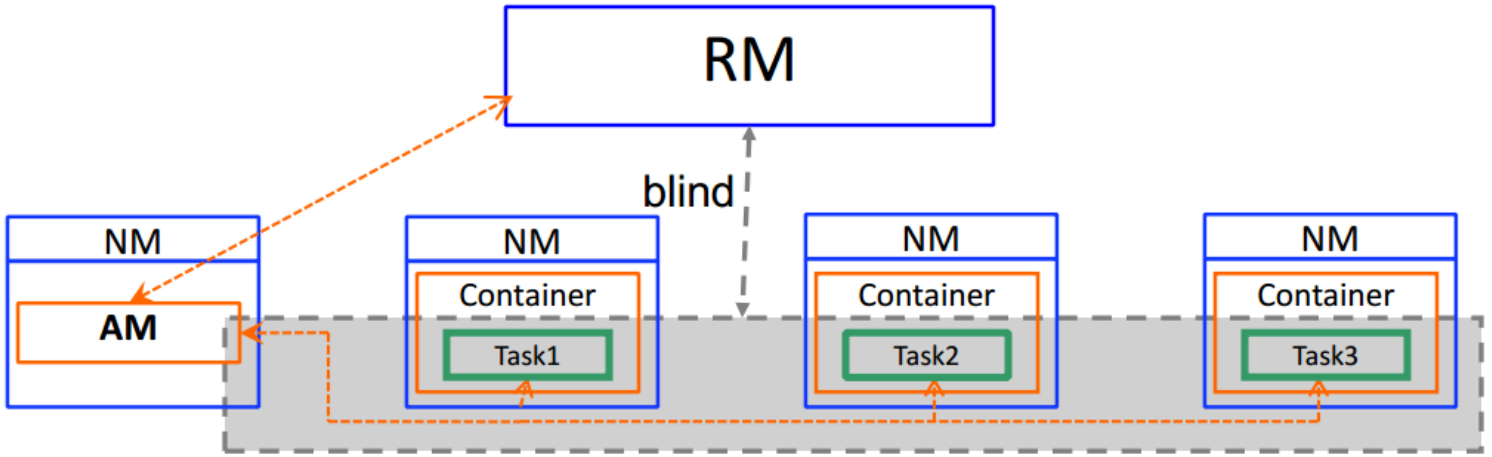
- AM is like the job tracker in Hadoop 1.0
  - Creates and manages task lifecycle
  - Monitors task status
- AM has no view of other running applications.

Container are managed by Node Manager

A Container is an arbitrary application-specific process that's created by a Node Manager on behalf of an Application Master.



Visibility of Tasks running inside of a Container

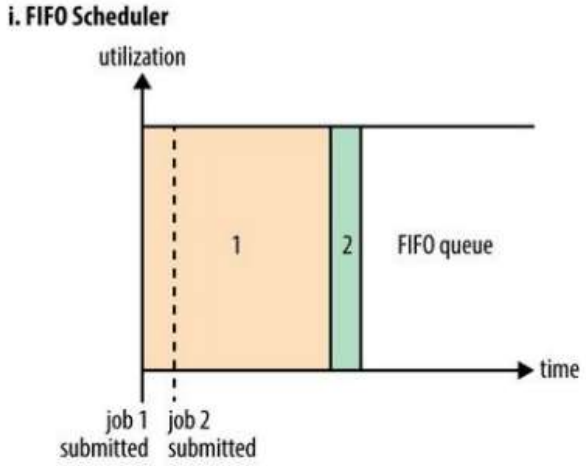


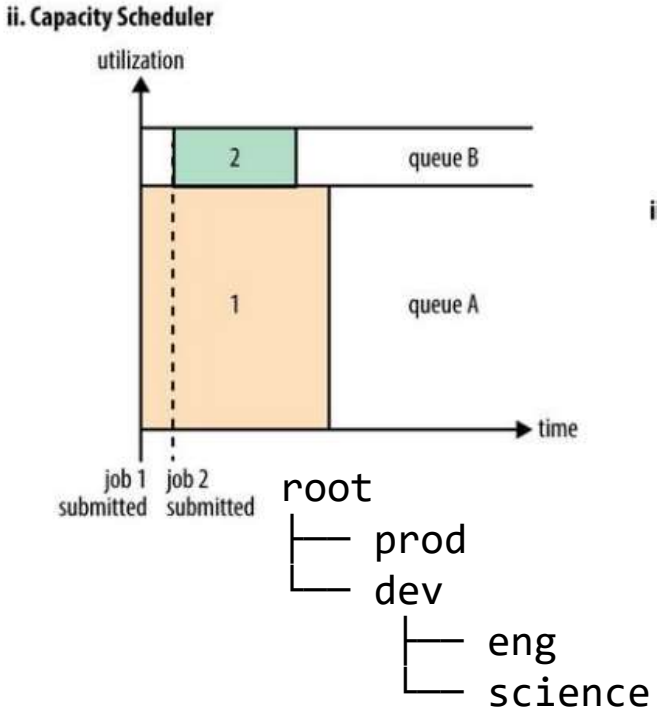
- When a job is submitted, RM assigns a jobID to it and allocates a container to run the corresponding AM.
- AM then asks for resources to run its job. After it gets the lease, the AM starts tasks and assigns tasks to containers.
- RM is blind to the tasks running within an application.

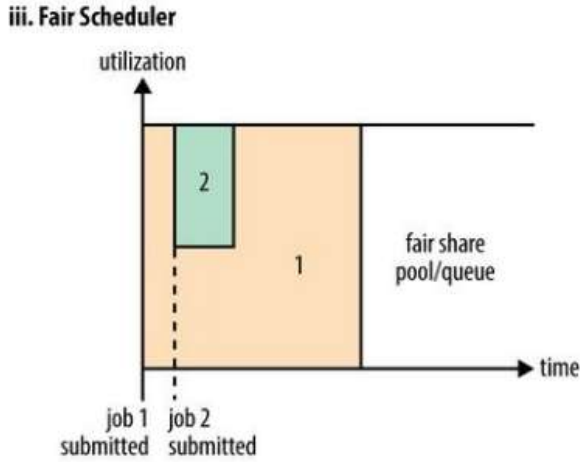
Scheduling in YARN, part of Resource Manager

Scheduler is responsible for allocating resources to the various running applications.

FIFO, Capacity, Fair Schedulers:





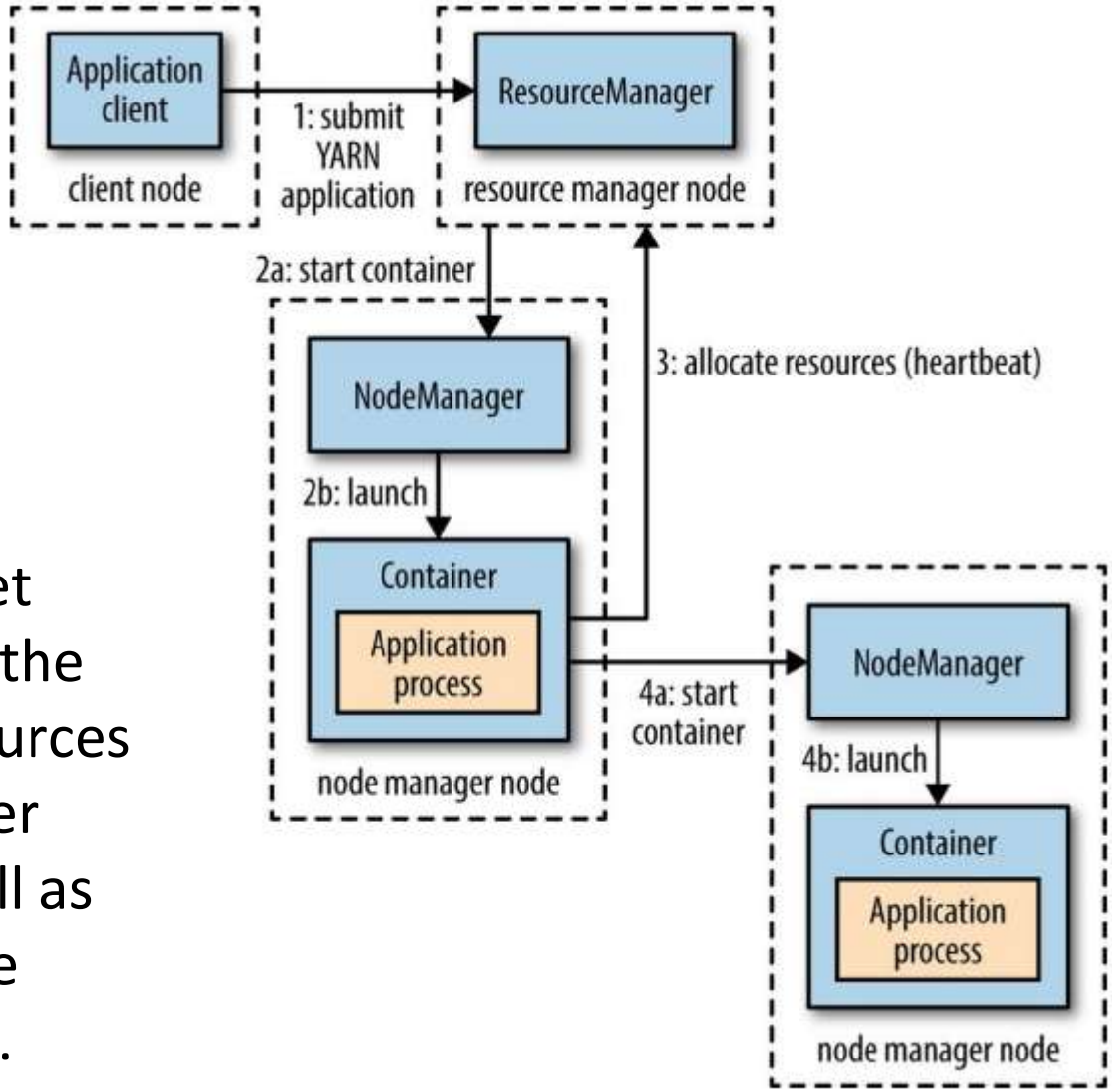


Source: Hadoop – The Definitive Guide

Outline

- Introduction to YARN
- Lifecyclye of an Application
- Working With YARN

Application Life Cycle with YARN



Resource requests for a set of containers can express the amount of computer resources required for each container (memory and CPU), as well as locality constraints for the containers in that request.

Source: Hadoop – The Definitive Guide

Lifecycle: Job submission

1. MapReduce client uses the same API to submit a job to YARN.
2. The new job ID is retrieved from the RM. However, sometimes a jobID in YARN is also called applicationID.
3. Necessary job resources, such as the job JAR, configuration files, and split information are copied to a shared file system in preparation to run the job.
4. The job client calls submitApplication() on the RM to submit the job.

Lifecycle: Job initialization

1. RM passes the job request to its Scheduler. The Scheduler allocates resources to run a container where the Application Master (AM) will reside. Then the RM sends the resource lease to some Node Manager (NM).
2. The NM receives a message form RM and launches a container for the AM.
3. The AM takes the responsibility of initializing the job. Several bookkeeping objects are created to monitor the job. Afterwards, while the job is running, the AM will keep receive updates with the progress of its tasks.
4. AM interacts with the shared file system (e.g. HDFS) to get its input splits and other information which were copied to the shared file system in Step 3 Job Initialization.

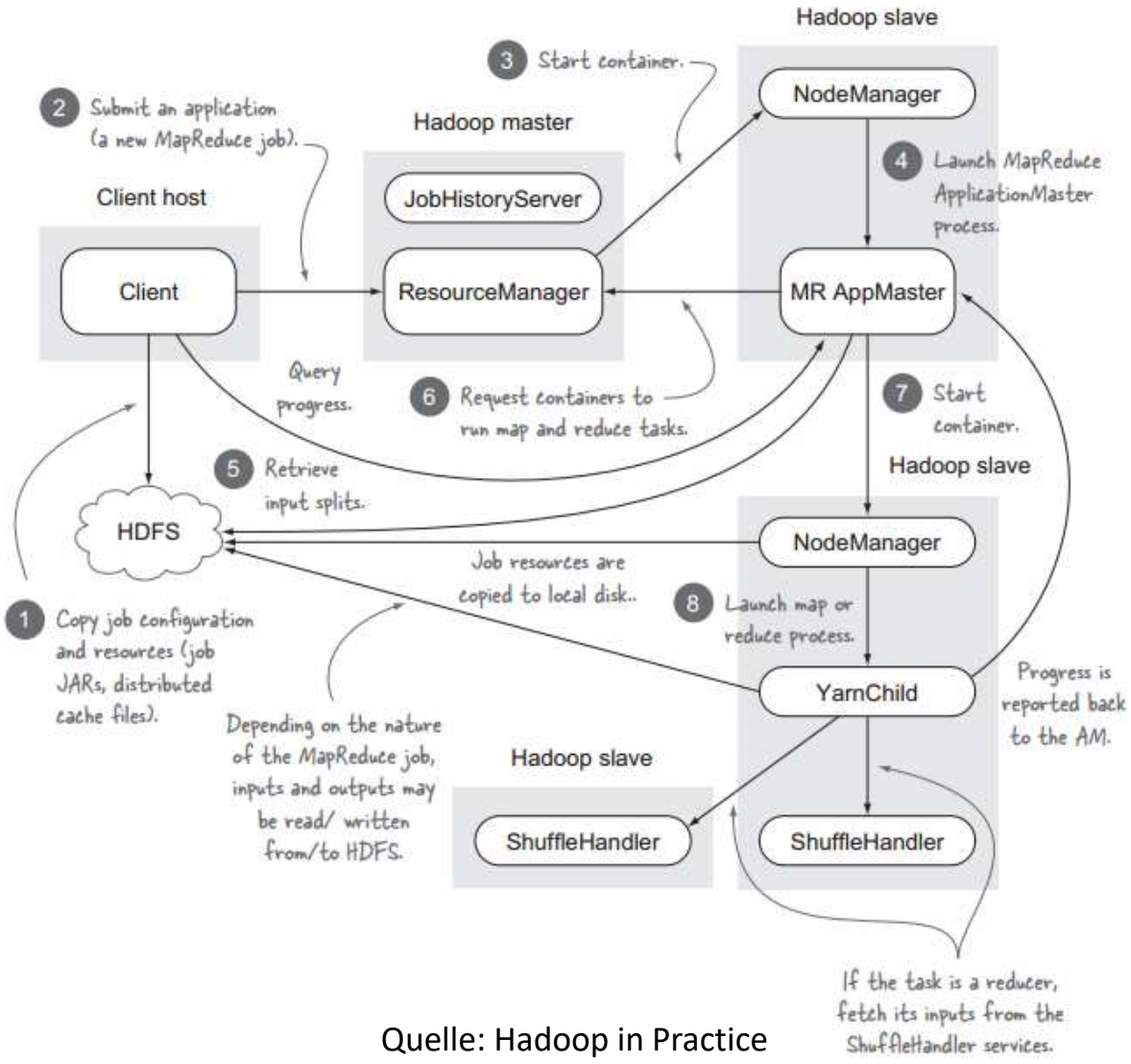
Lifecycle: Task assignment

1. AM computes the number of map tasks (based on the number of input splits) and the number of reduce tasks (configurable). AM submits the resource request for the map and reduce tasks along with its heartbeat to the RM. A request includes preferences in terms of data locality (for map tasks), the amount of memory and the number of CPUs in each container .
2. After the RM responds with container leases, the AM communicates with the NMs.
3. The NMs start the containers.
4. AM assigns a task to this container based on its knowledge of locality.
5. The task runs in the container. The MapReduce AM monitors the individual tasks to completion, requests alternate resource if any of the tasks fail or stop responding.

Lifecycle: Job Completion

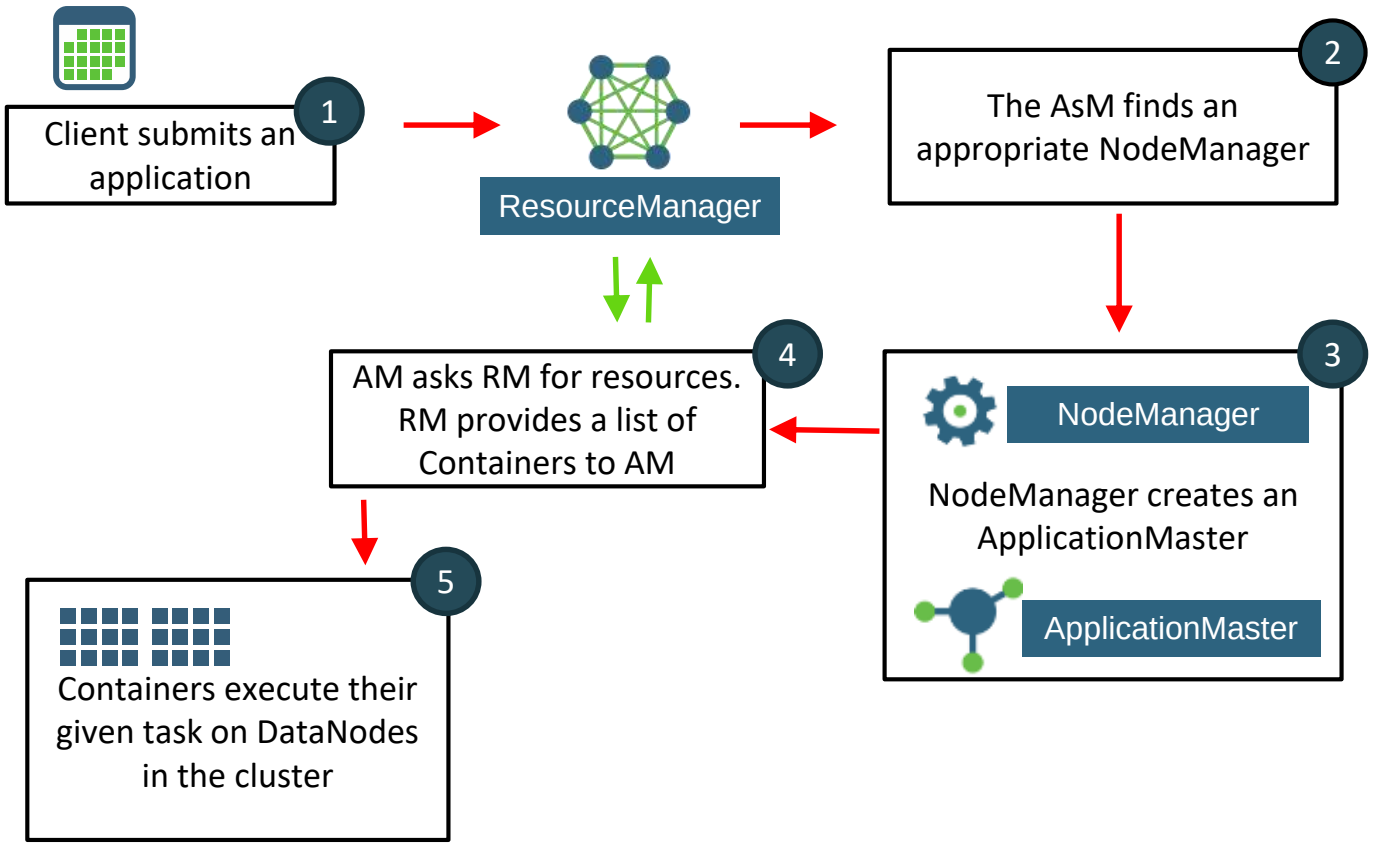
- The MapReduce AM also runs appropriate task cleanup code of completed tasks
- Once the entire map and reduce tasks are complete, the MapReduce AM runs the requisite job commit
- The MapReduce AM informs the RM then exits since the job is complete

Application Life Cycle im Detail



Quelle: Hadoop in Practice

Lifecycle of a YARN Application



Outline

- Introduction to YARN
- Lifecyclye of an Application
- Working With YARN

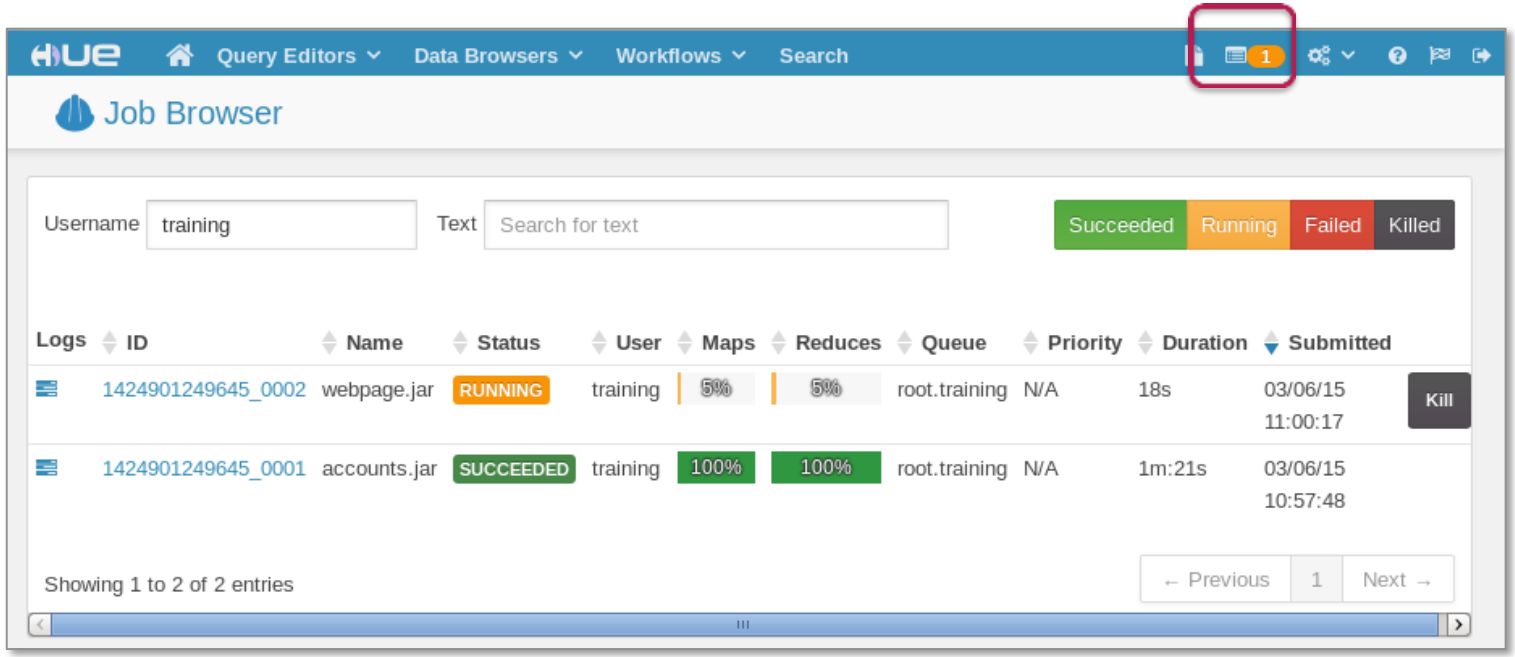
Hadoop includes three major YARN tools for developers

- Hue Job Browser
- YARN Web UI
- YARN command line

There are also distribution specific tools like Cloudera Manager

The Hue Job Browser

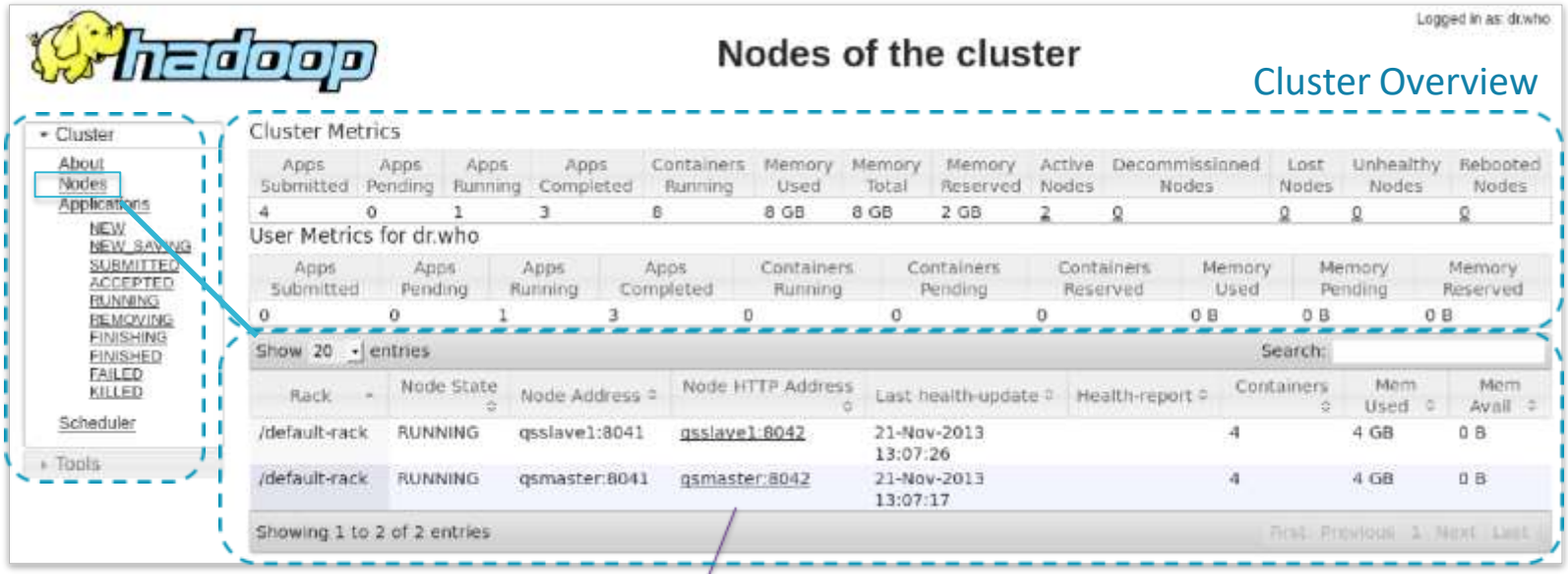
- Monitor the status of a job
- View the logs
- Kill a running job



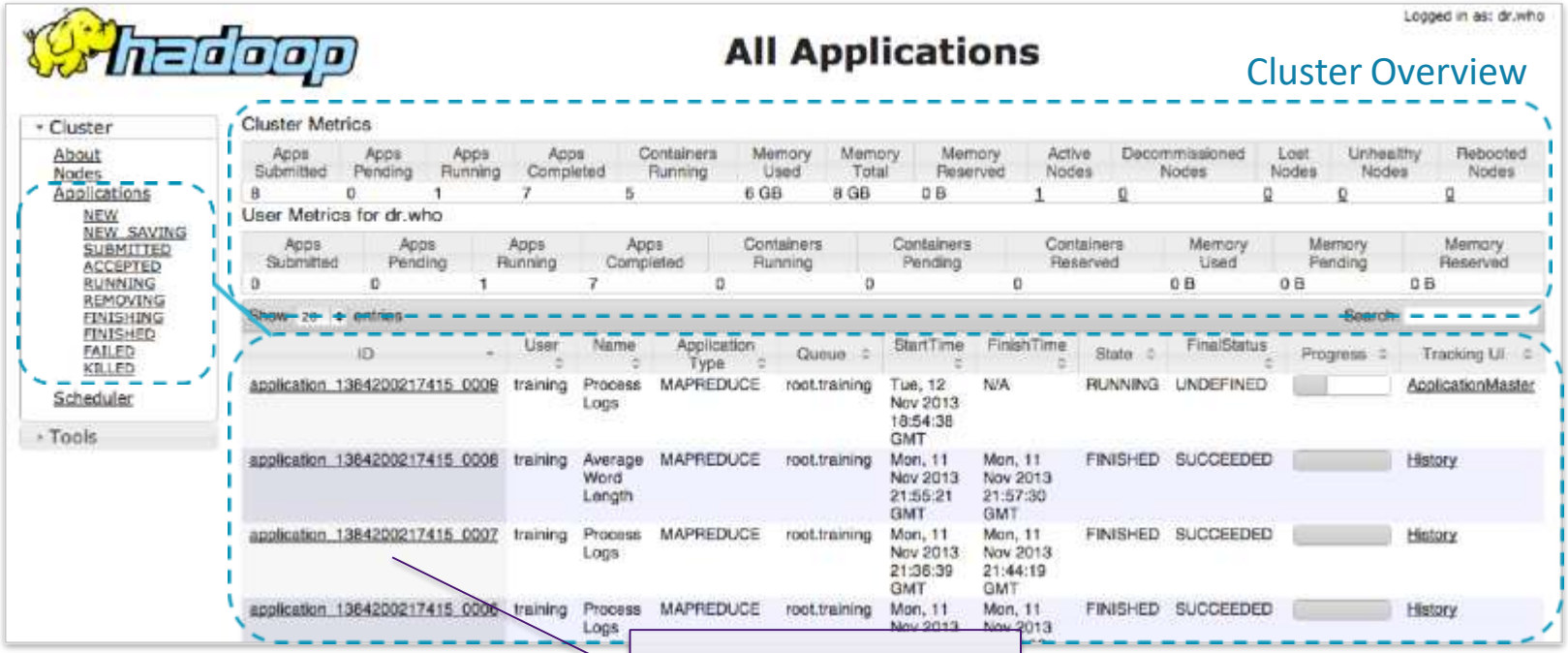
The YARN Web UI

- Resource Manager UI is the main entry point
- Runs on the RM host on port 8080 by default
- Provides more detailed view than Hue
- Does not provide any control or configuration

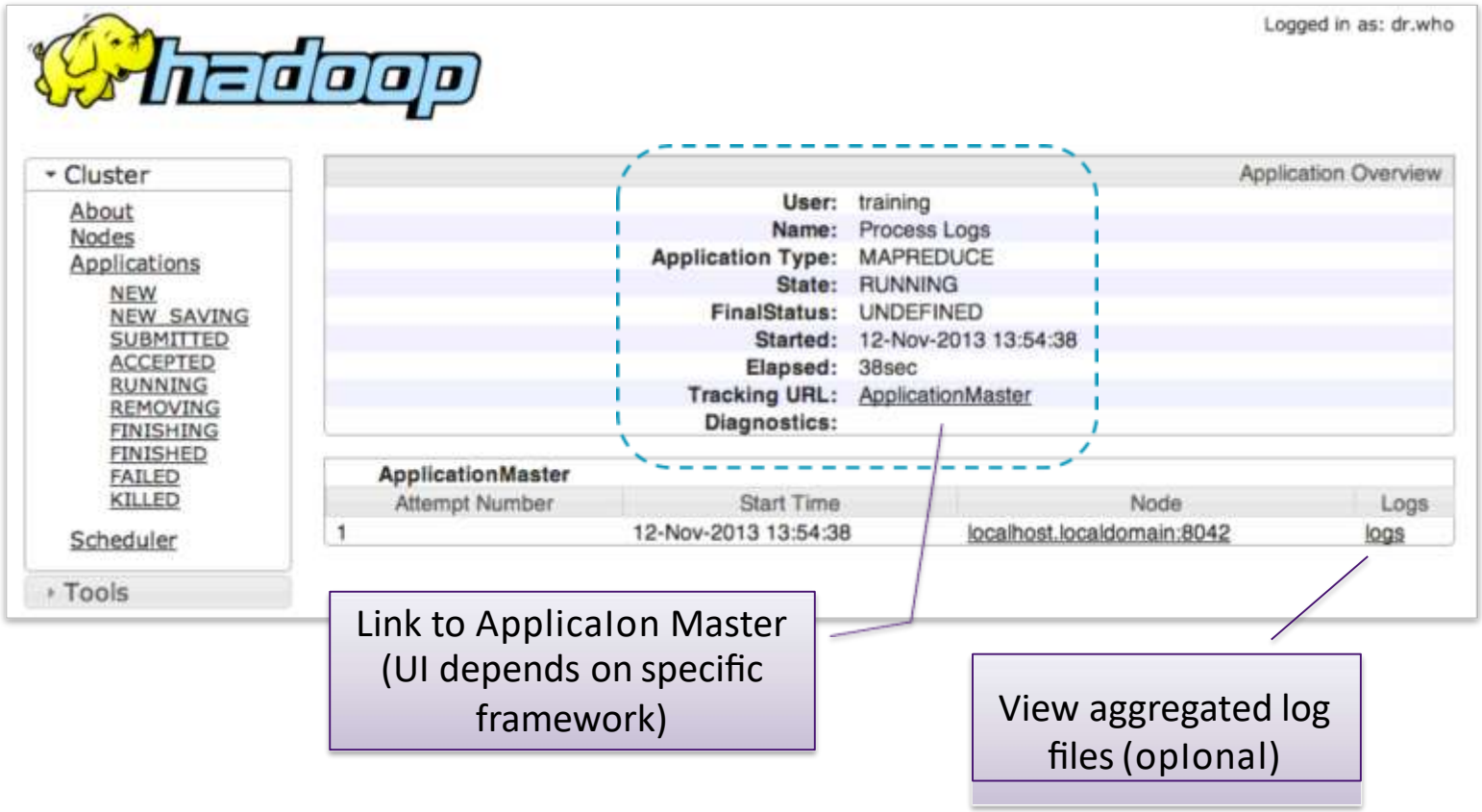
Resource Manager UI: Nodes



Resource Manager UI: Applications

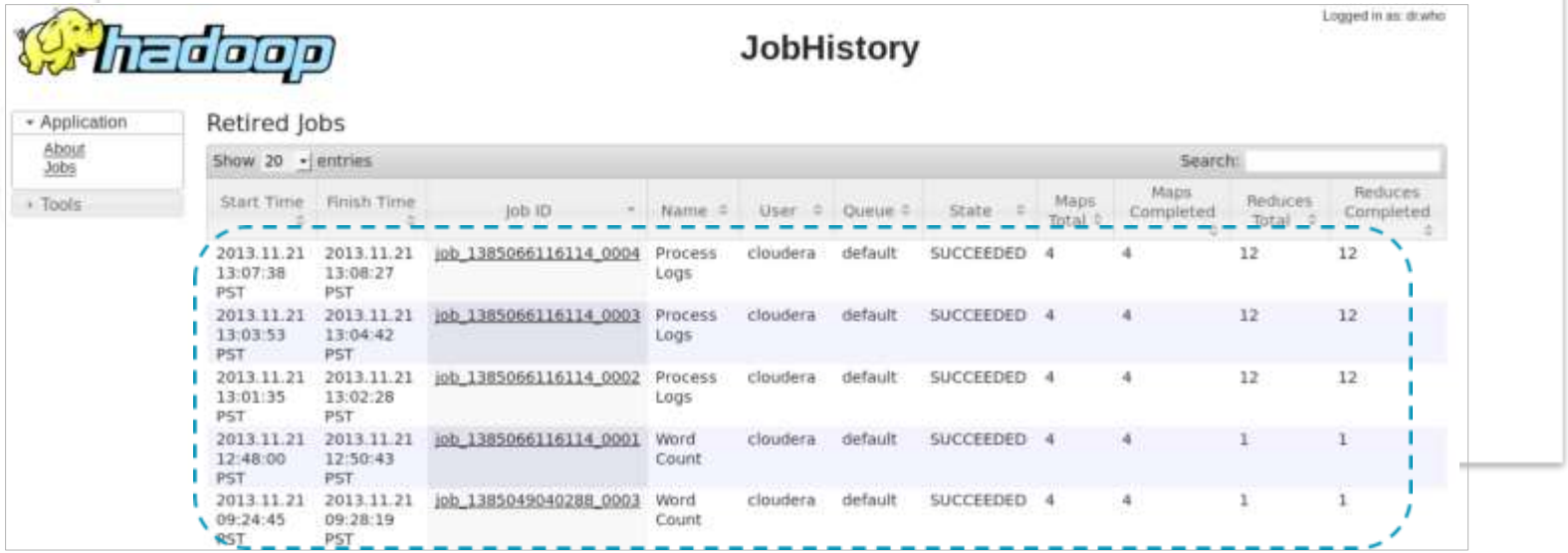


Resource Manager UI: Application Detail



Job History Server

- YARN does not keep track of job history
- Spark and MapReduce each provide a Job History Server
- Archives job's metrics and metadata
- Can be accessed through Job History UI or Hue



YARN Command Line

- Most YARN command line tools are for administrators
- Command to configure and view information about the YARN cluster:
  `yarn [--config confdir] COMMAND [--loglevel loglevel] [GENERIC_OPTIONS] [COMMAND_OPTIONS]`

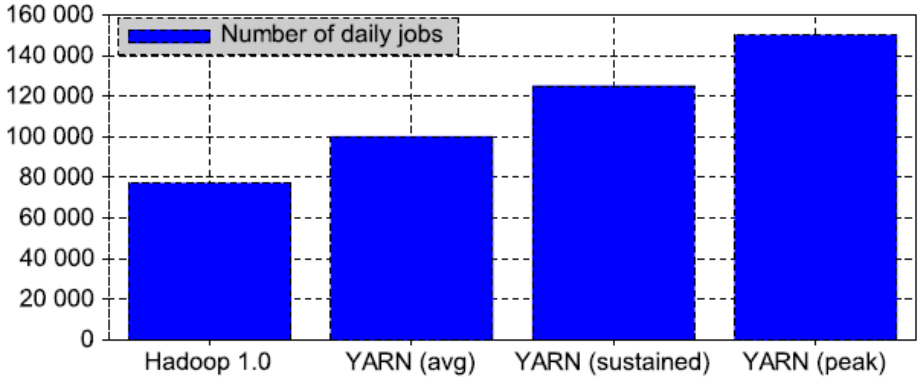
Some helpful commands for developers yarn application

`yarn application –list`	Lists applications from the RM
`yarn application –kill <id>`	kills the application
`yarn node –list`	Lists all running nodes.

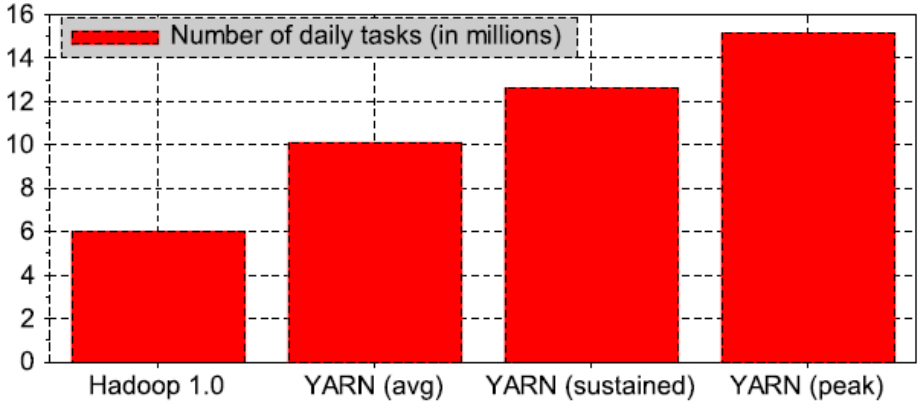
Essential Points

- YARN manages resources in a Hadoop cluster and schedules jobs
- YARN works with HDFS to run tasks where the data is stored
- Slave nodes run NodeManager daemons, managed by a ResourceManager on a master node
- Scheduler: FIFO, Capacity, Fair Schedulers
- Monitor jobs using Hue, the YARN Web UI or the yarn command

Hadoop 2.x at Yahoo!



(a) Daily jobs



(b) Daily tasks

Figure 2: YARN vs Hadoop 1.0 running on a 2500 nodes production grid at Yahoo!.

Quelle: [1]