



YARN

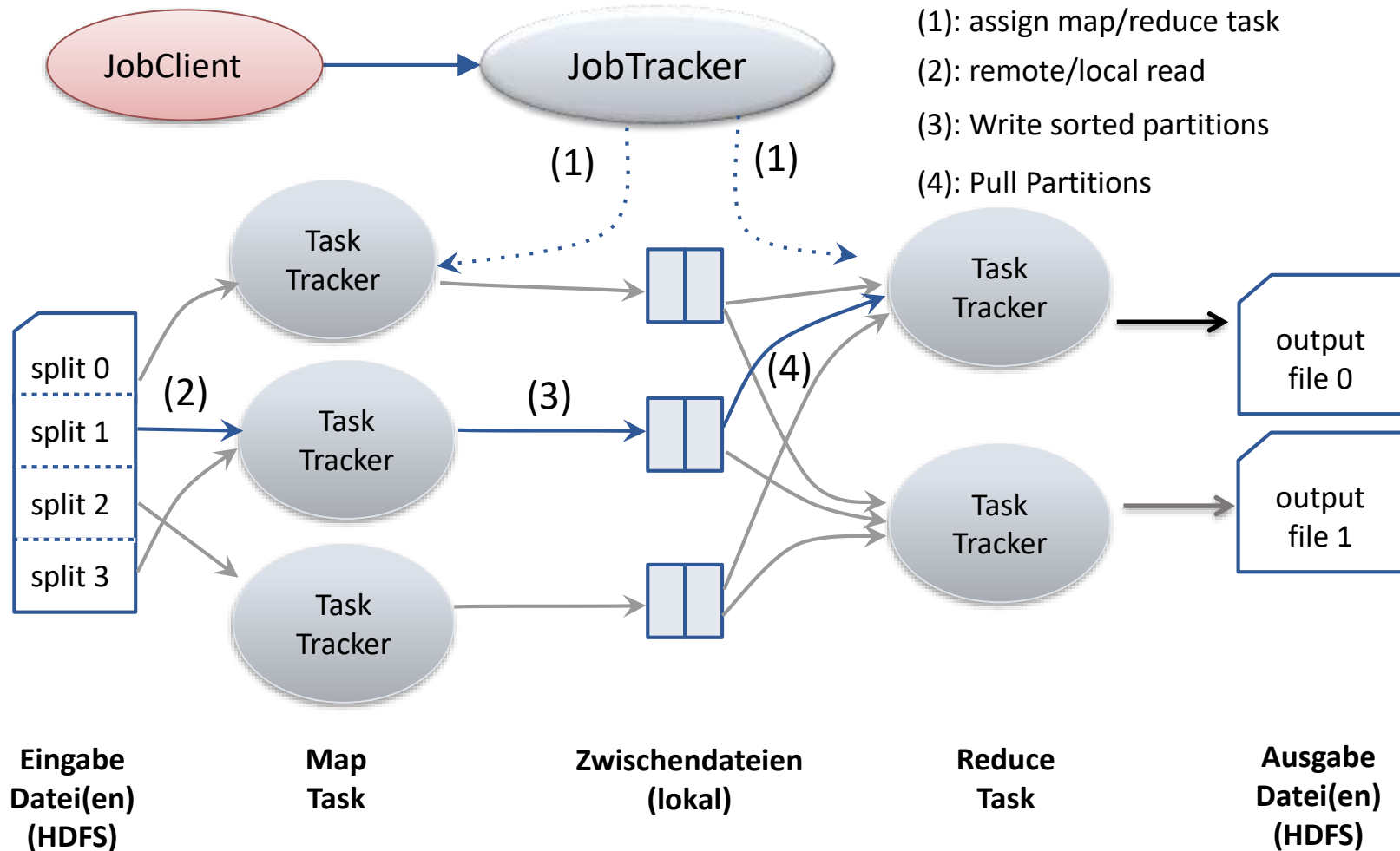
Yet Another Resource Negotiator

Prof. Dr. Stephan Trahasch
Hochschule Offenburg

Outline

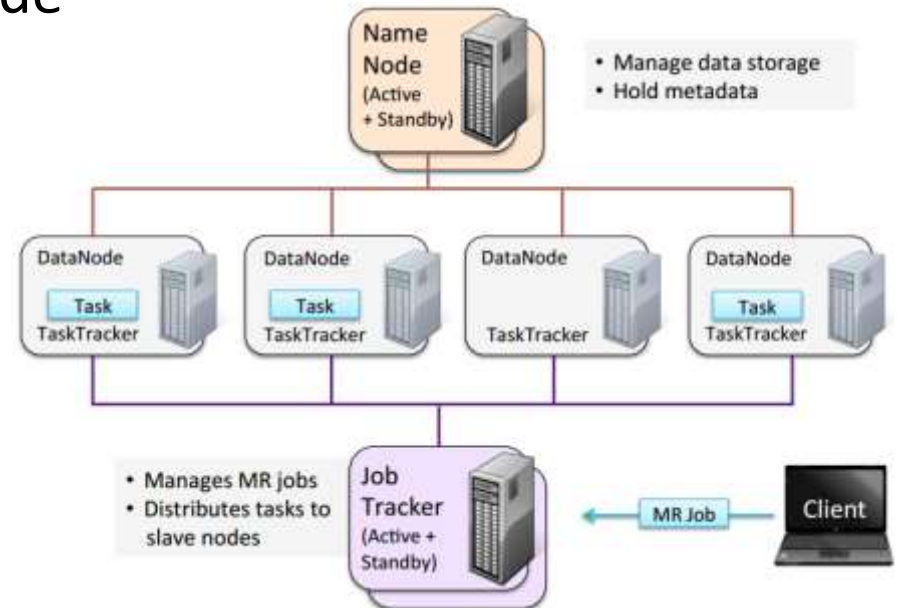
- Introduction to YARN
- Lifecycle of an Application
- Working With YARN

MapReduce: Job execution Hadoop 1.0



Hadoop MapReduce 1.0

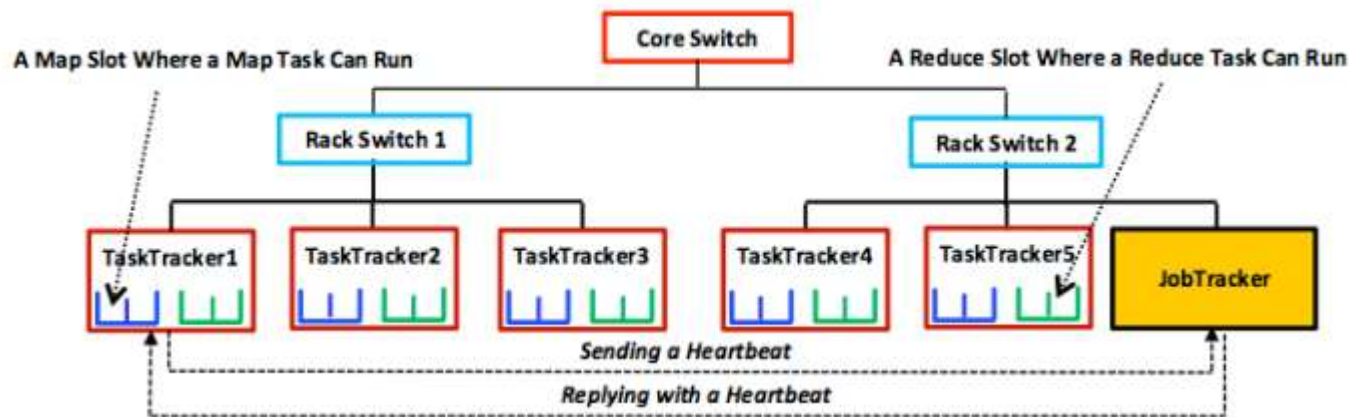
- Designed to run MapReduce jobs only
- JobTracker - Single master for all jobs and one per cluster
 - Resource allocator and job scheduler
 - Manages MapReduce jobs, distributes individual tasks to TaskTrackers
- TaskTracker – one per slave node
 - Starts and monitors individual Map and Reduce tasks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

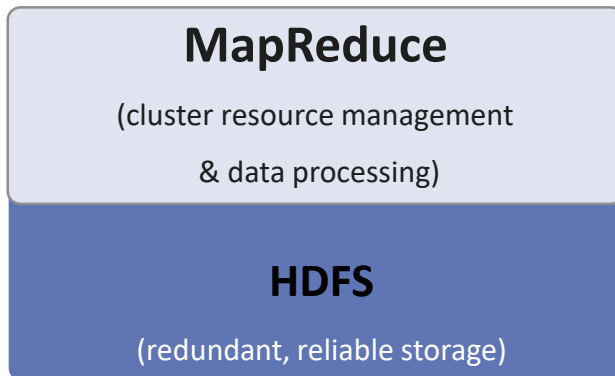


From Hadoop 1.0 to Hadoop 2.0

Single Use System

Batch Apps

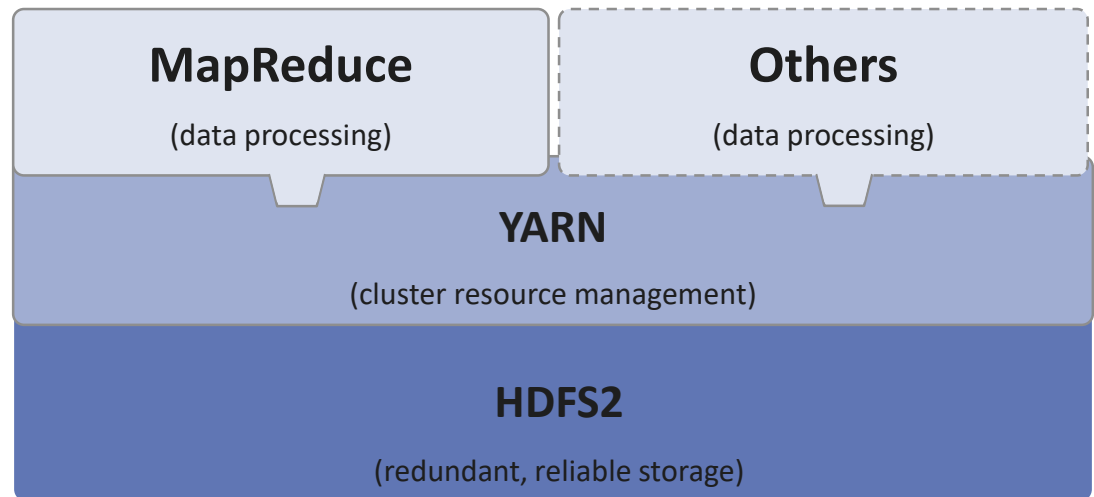
HADOOP 1.0



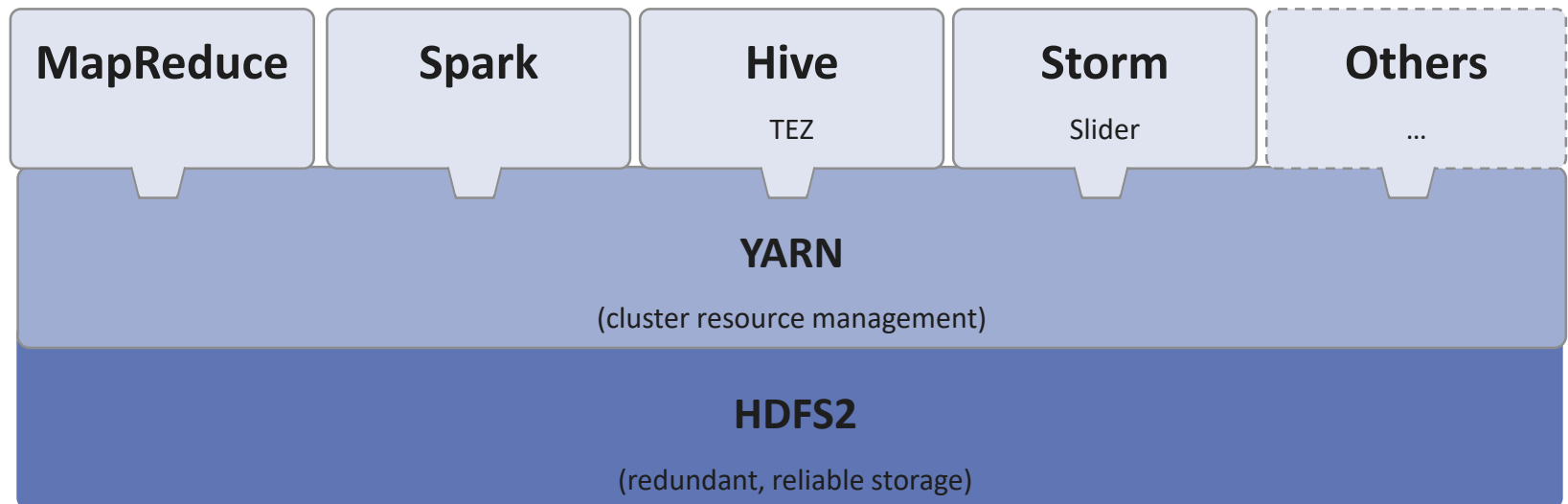
Multi Purpose Platform

Batch, Interactive, Online, Streaming, ...

HADOOP 2.0



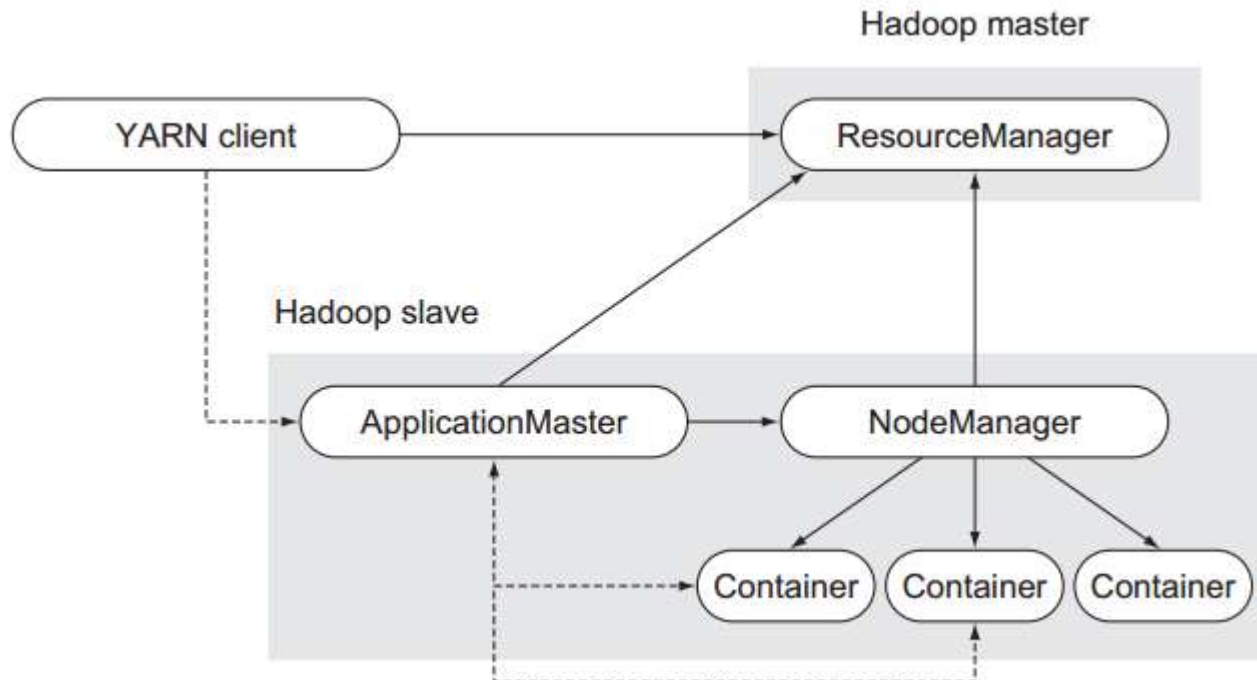
YARN - Yet Another Resource Negotiator



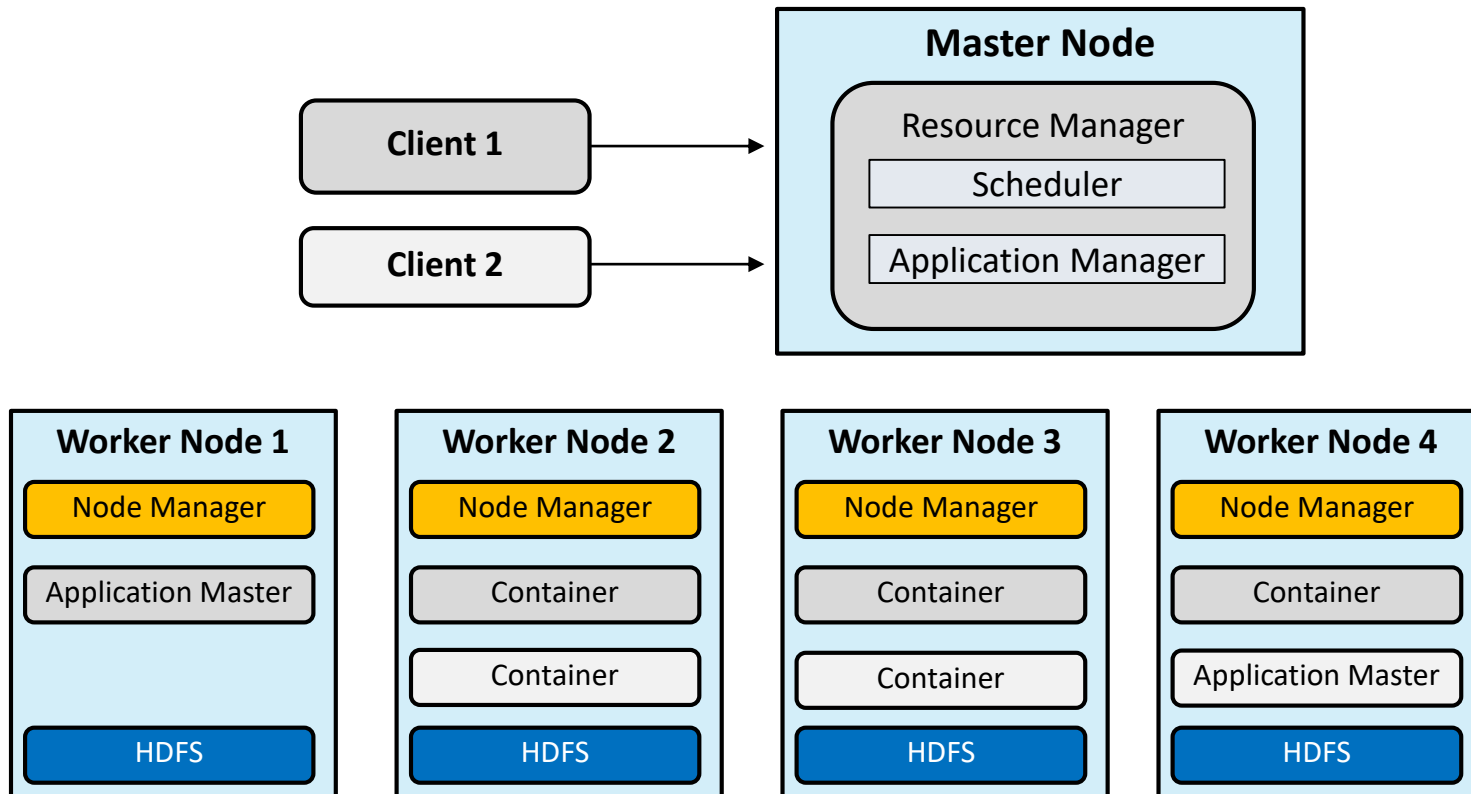
- Cluster Resource Management for distributed data processing
- Dynamic Allocation of resources for tasks.
- allows multiple data processing engines to run on a single Hadoop cluster: MapReduce, Spark, Tez, Storm, Giraph ...

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



Responsibility

1. Resource Manager

- One per Cluster
- Starts Application Masters
- Allocates Resources on Slave Nodes

2. Application Master

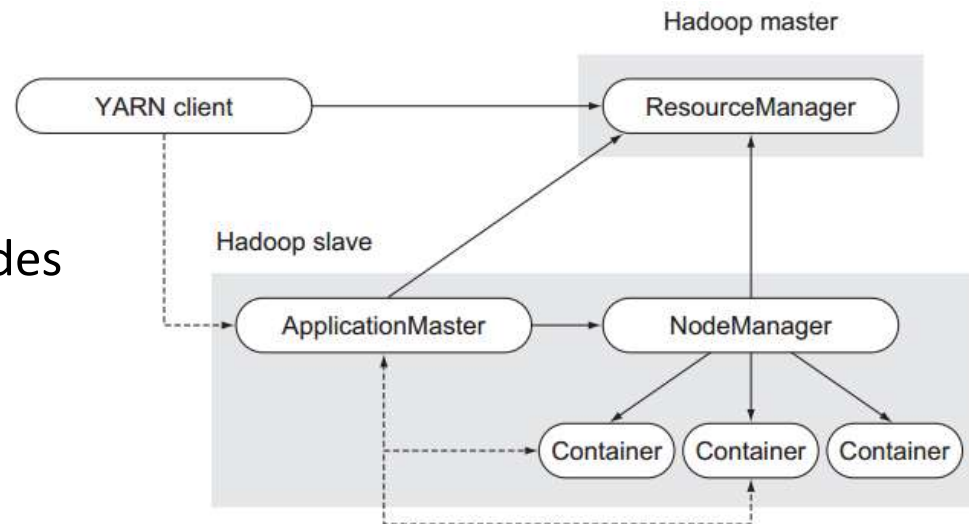
- One per Job (Application)
- Framework/application specific
- Runs in a container
- Requests more containers to run application tasks

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



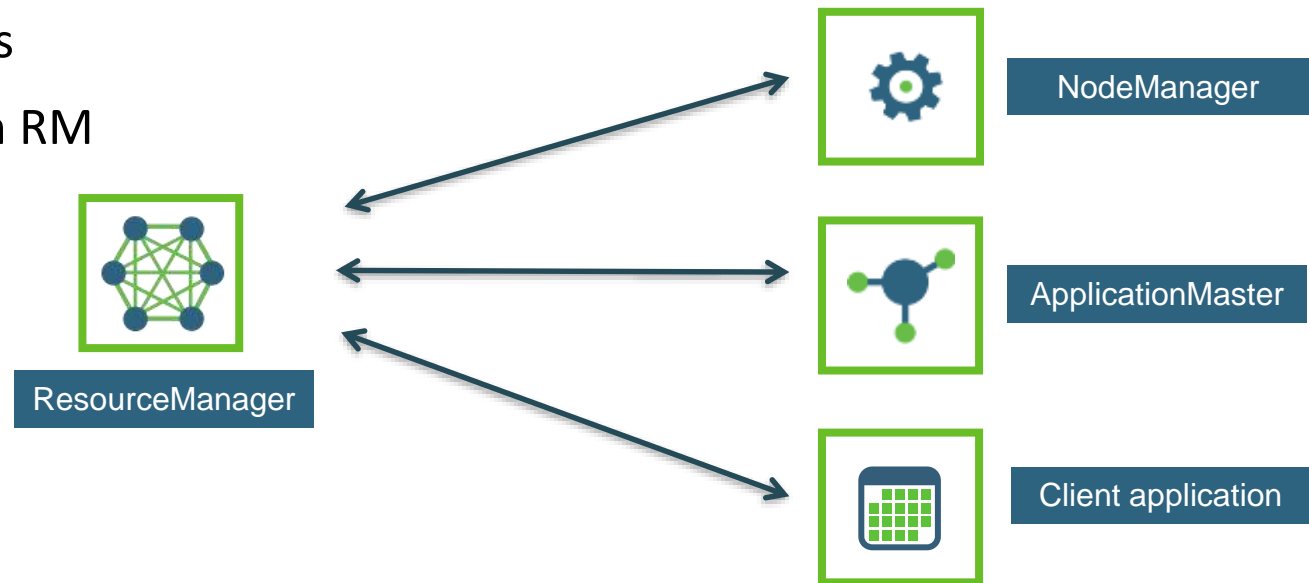
Resource Manager and Node Manager

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

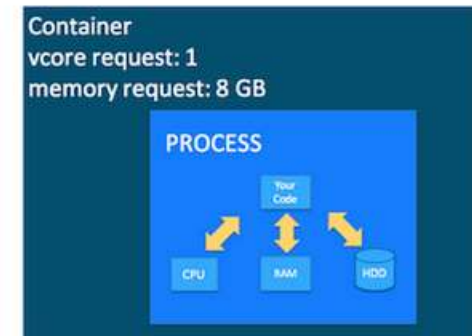
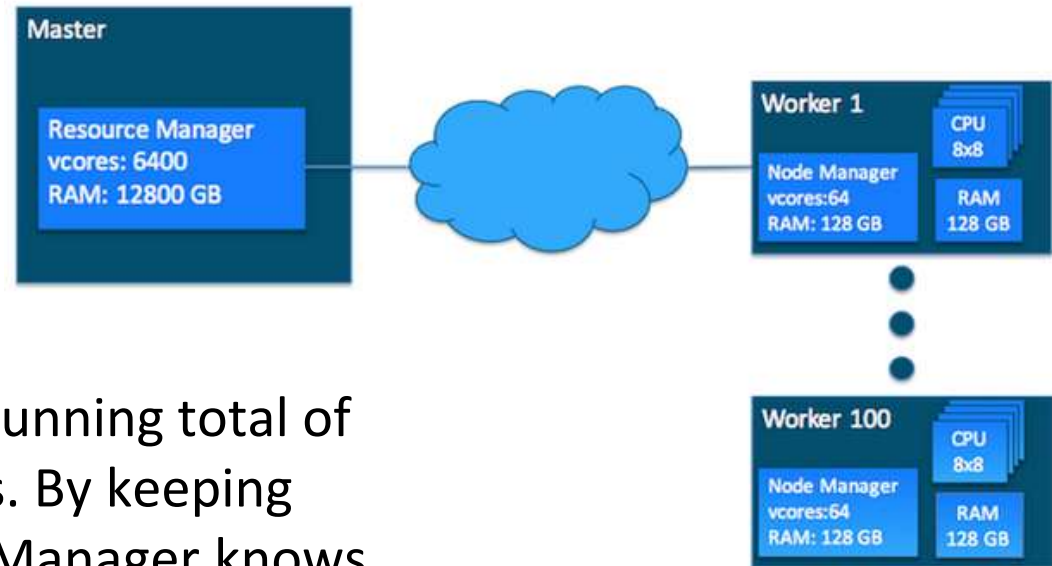
Node Manager (NM)

- Runs on slave nodes
- Communicates with RM

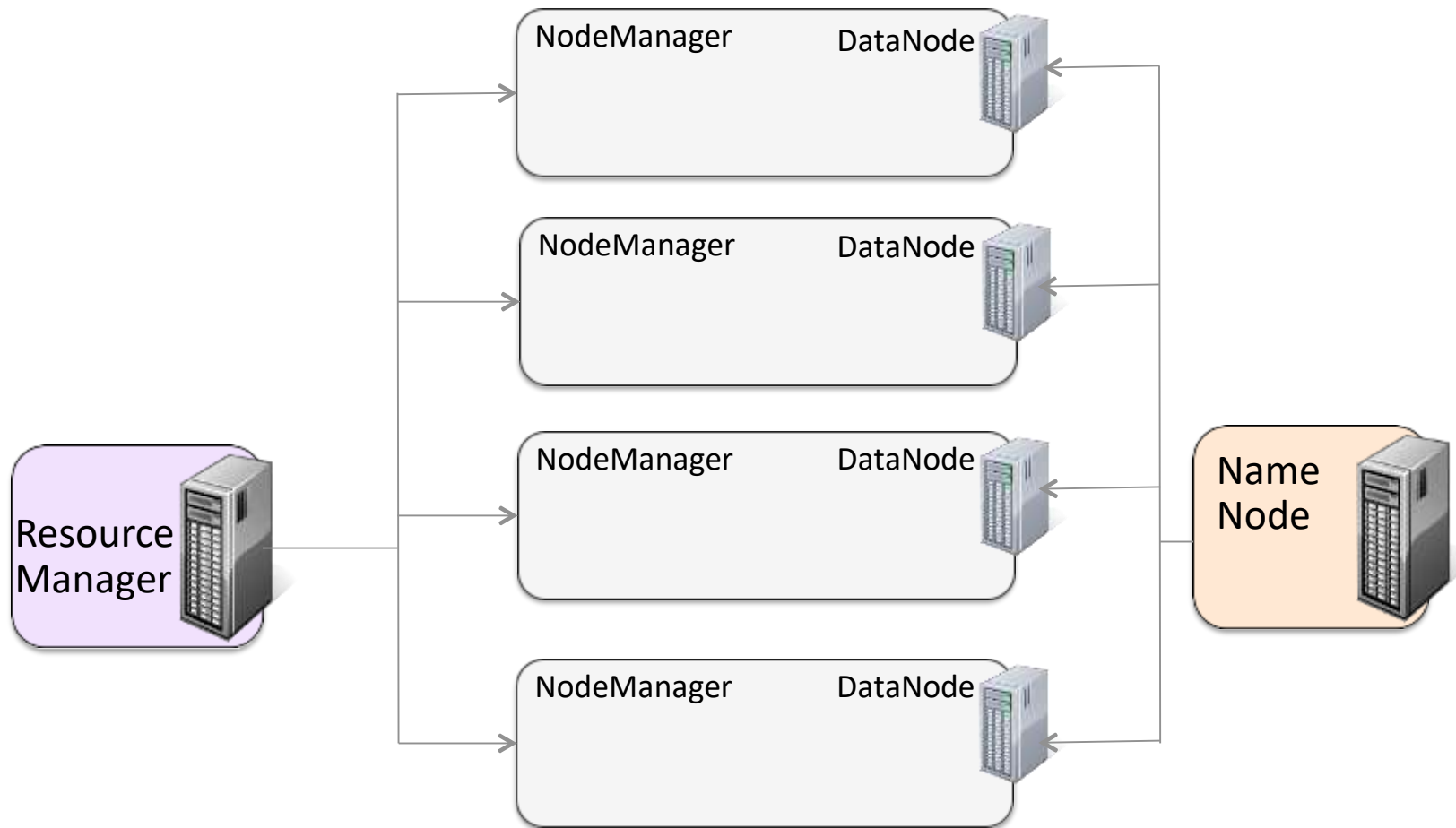


Resources: vcores and memory

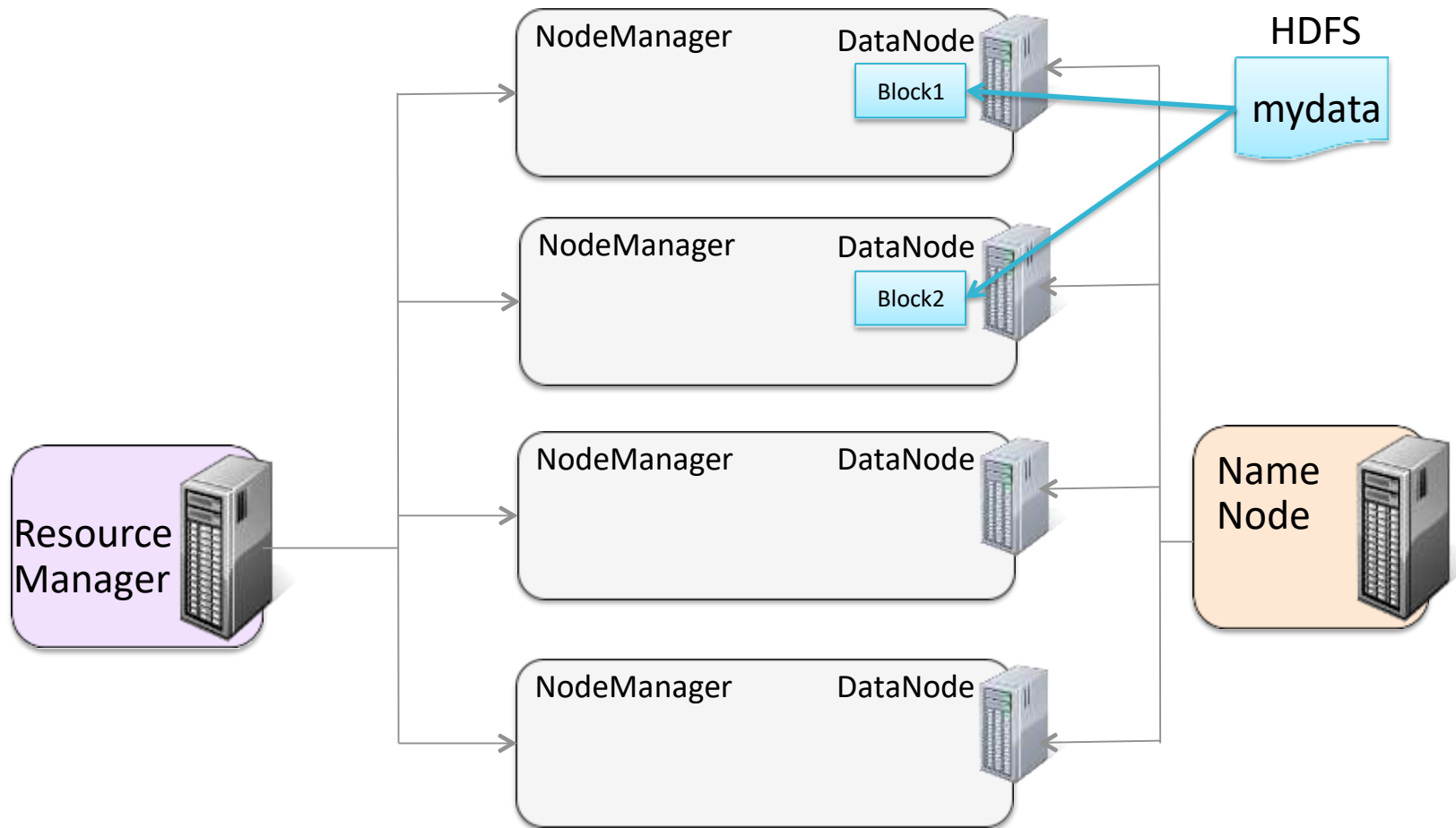
- NodeManager tracks its own local resources and communicates its resource configuration to ResourceManager
- ResourceManager keeps a running total of cluster's available resources. By keeping track of the total, ResourceManager knows how to allocate resources as they are requested.
- A container hold request consists of vcore and memory



Running an Application on YARN (1)

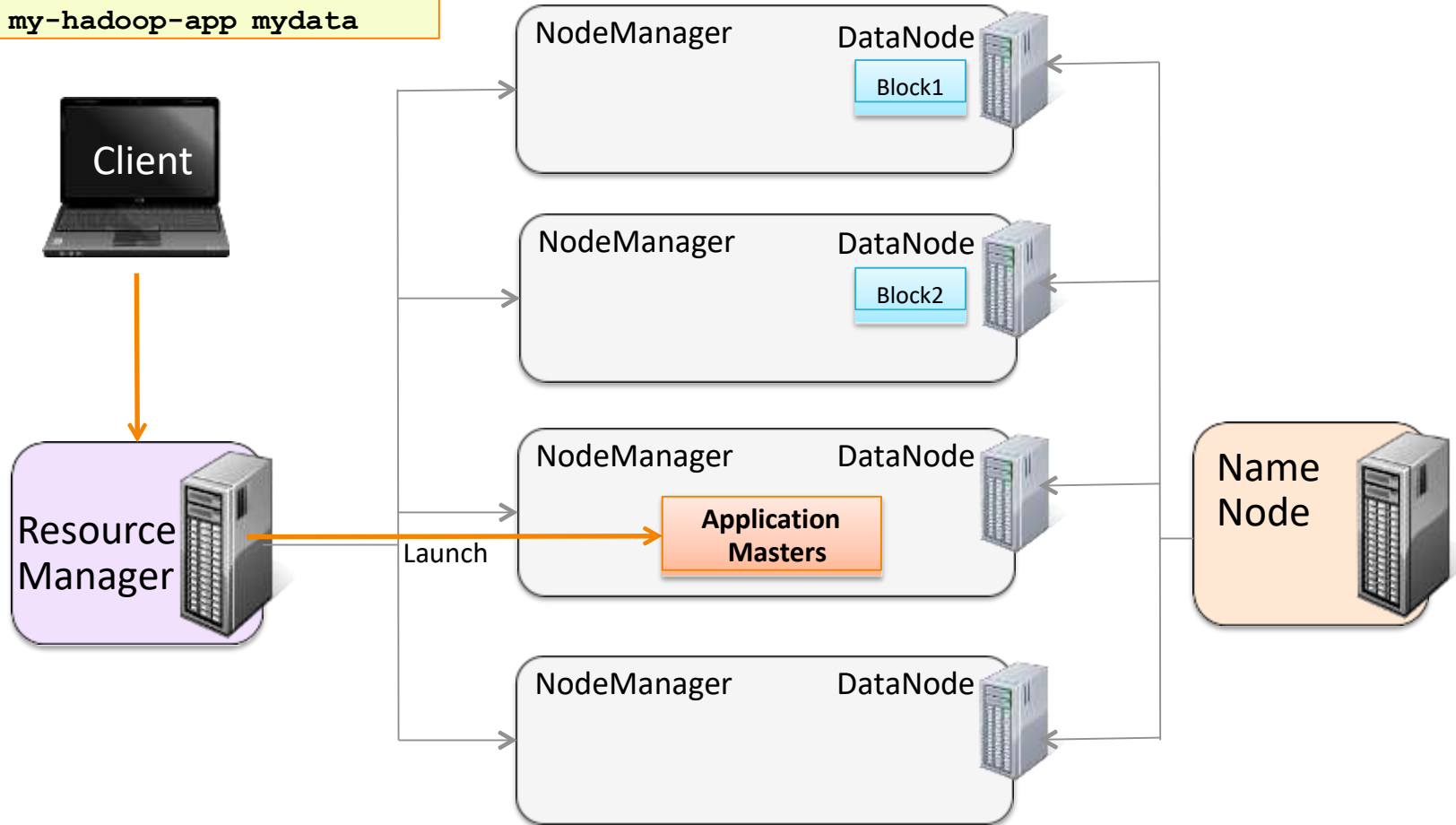


Running an Application on YARN (2)



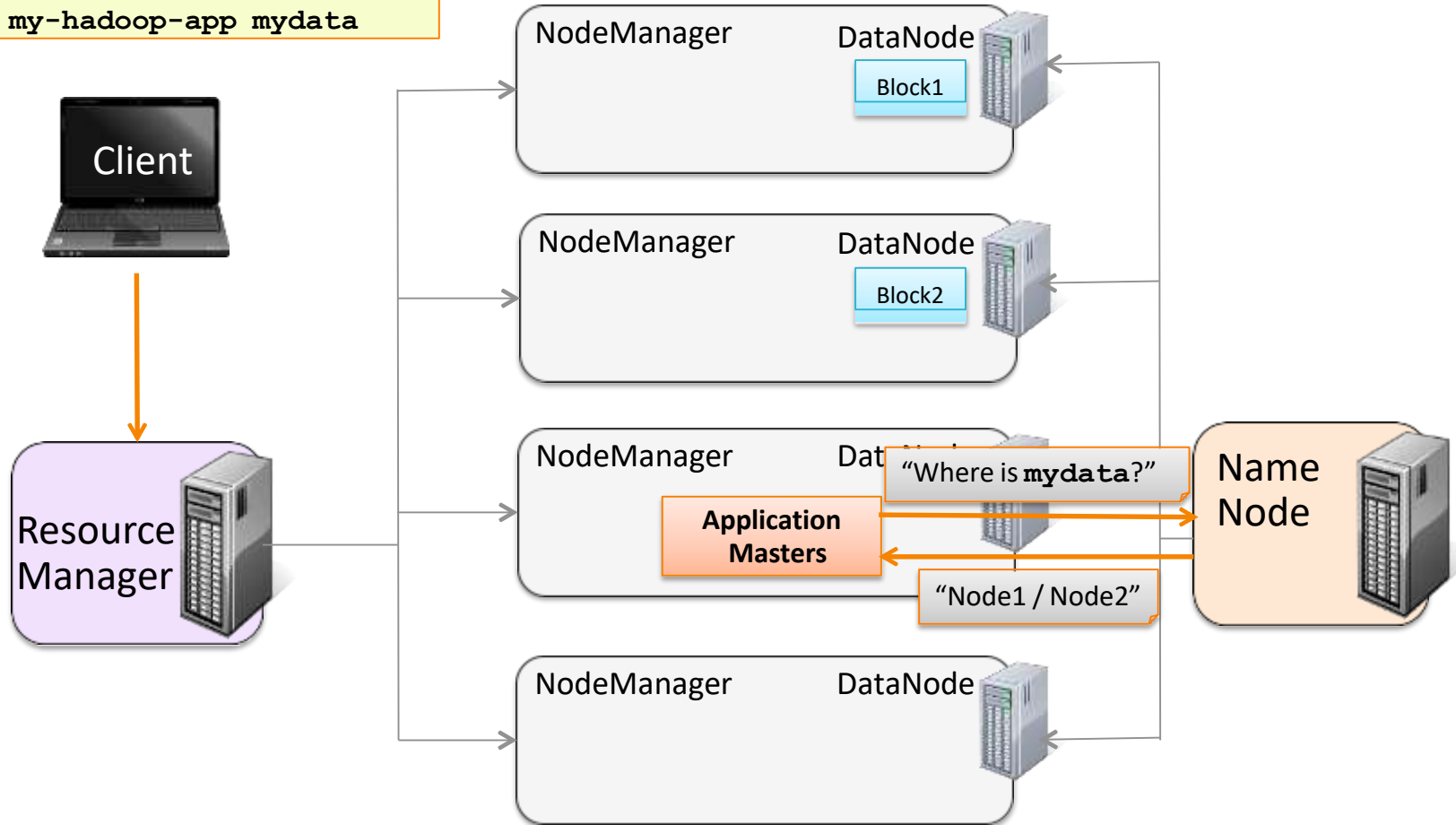
Running an Application on YARN (3)

```
$ my-hadoop-app mydata
```



Running an Application on YARN (4)

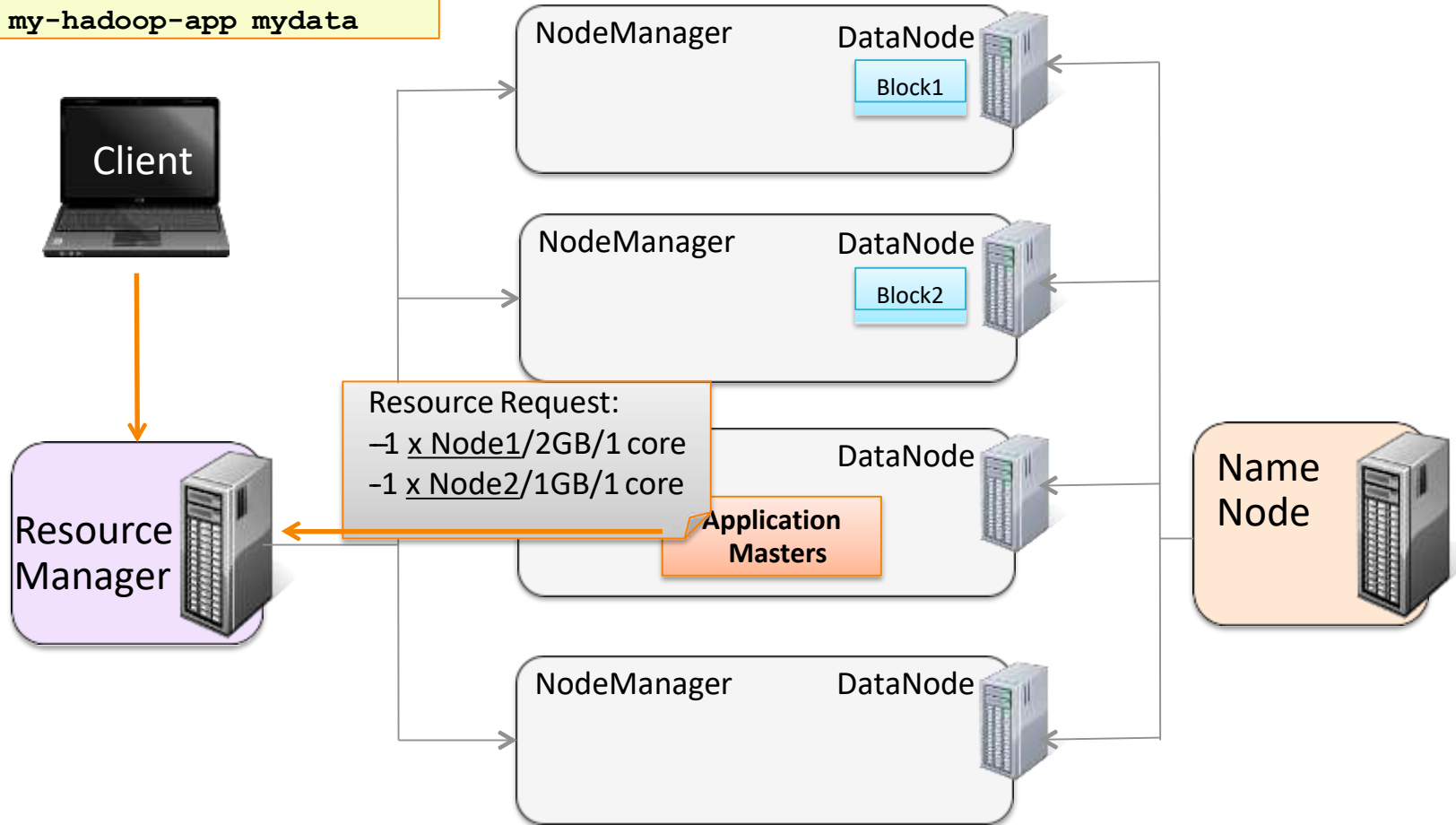
```
$ my-hadoop-app mydata
```



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

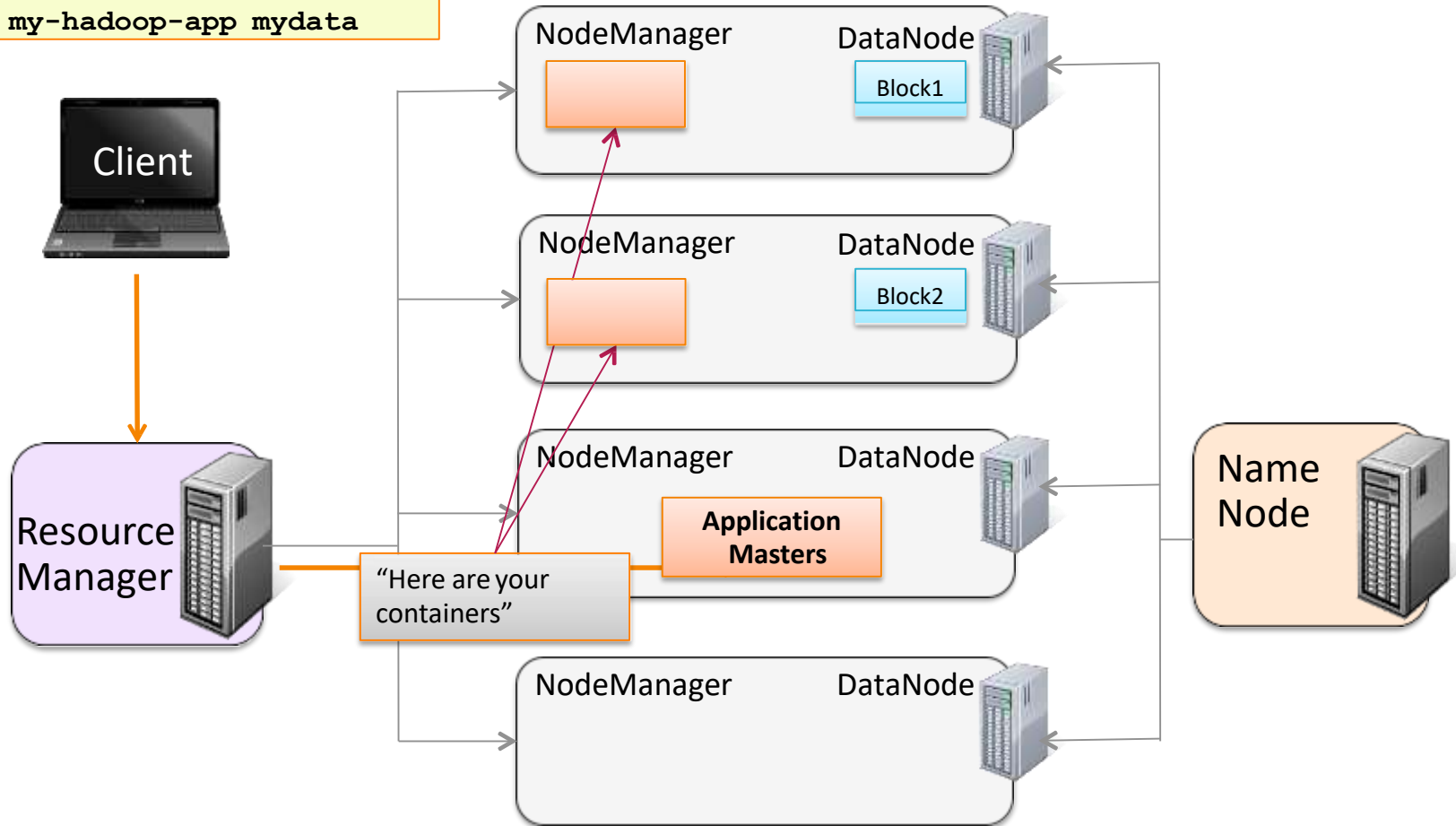
Running an Application on YARN (5)

```
$ my-hadoop-app mydata
```



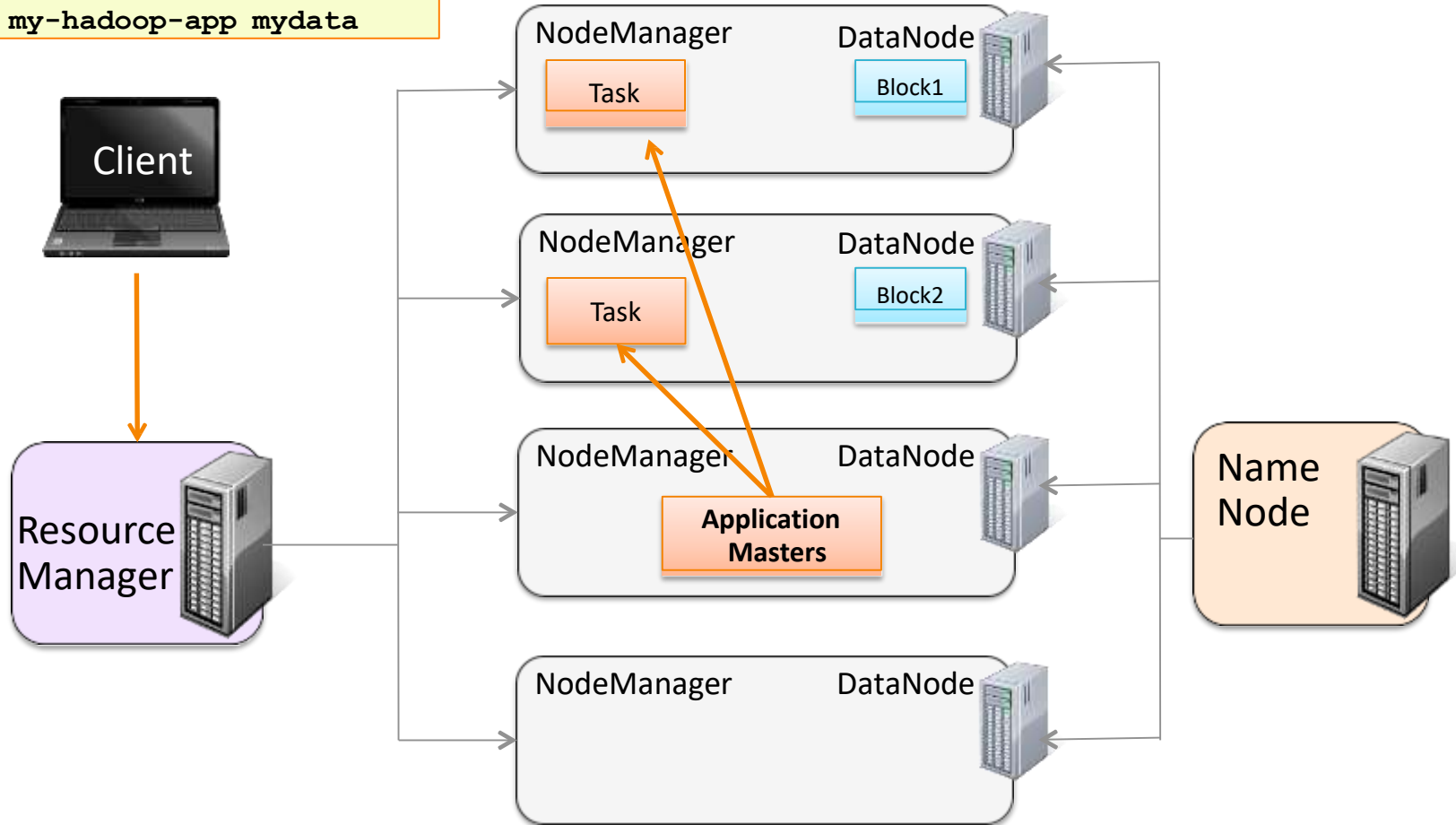
Running an Application on YARN (6)

```
$ my-hadoop-app mydata
```



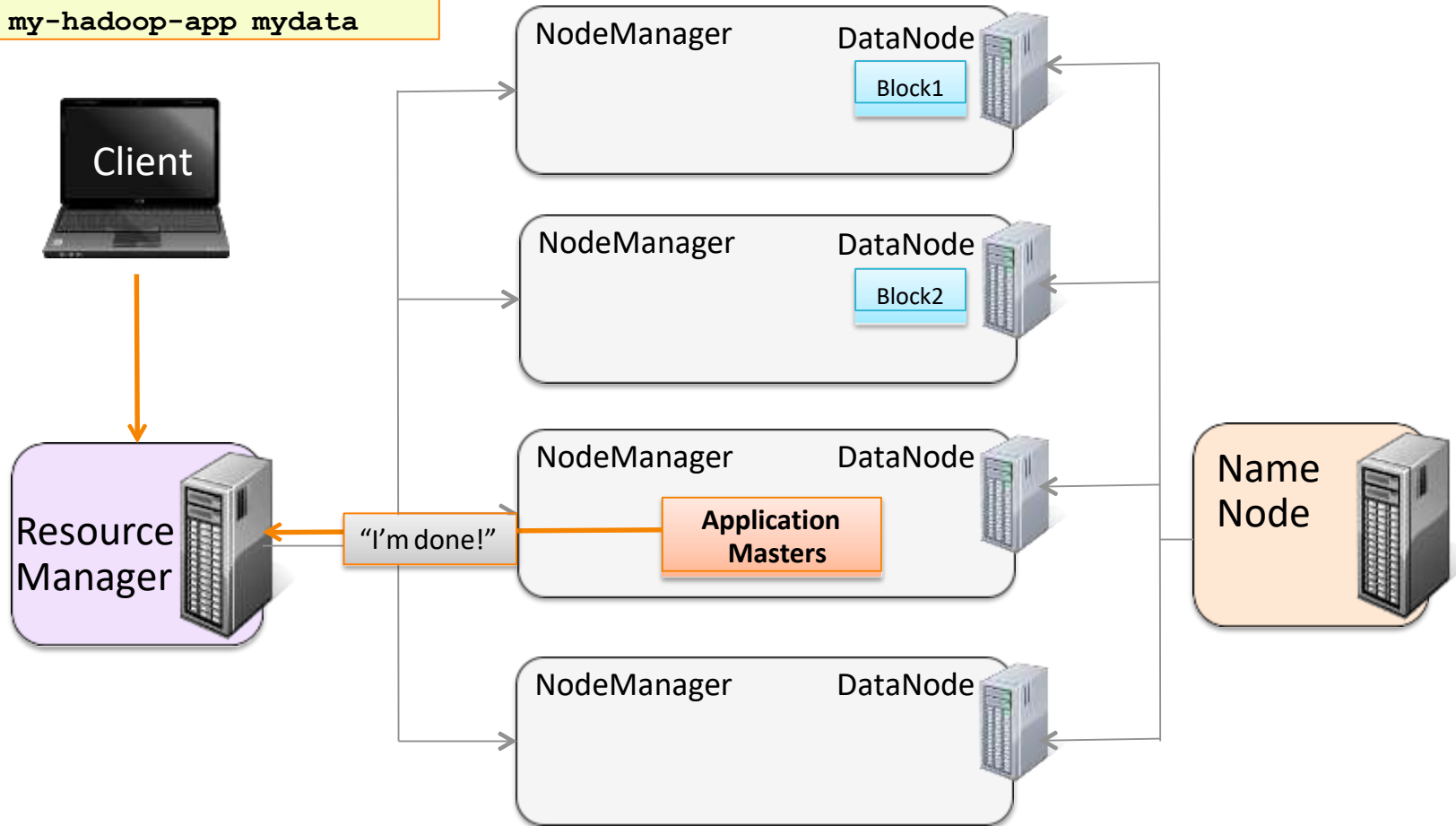
Running an Application on YARN (7)

```
$ my-hadoop-app mydata
```



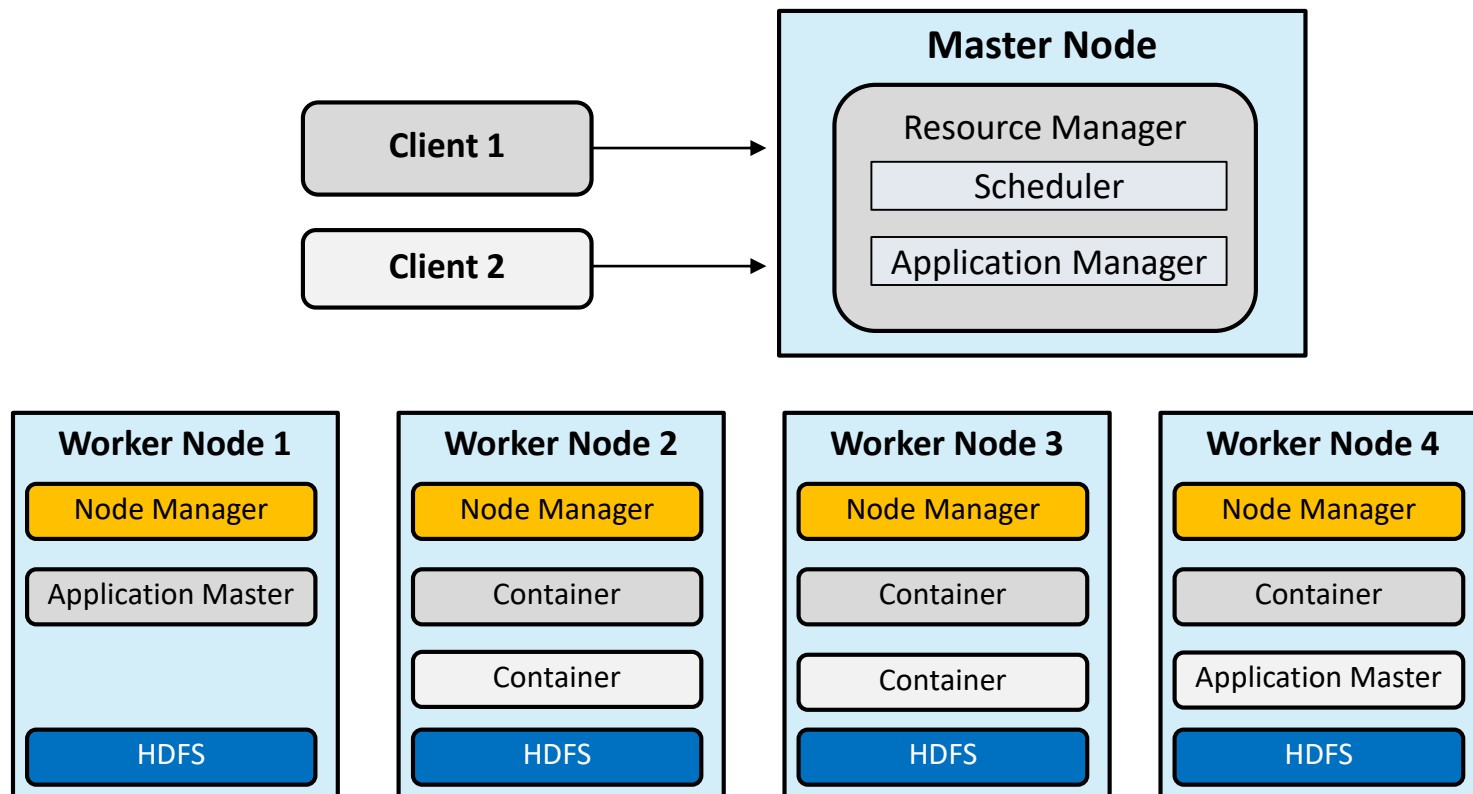
Running an Application on YARN (8)

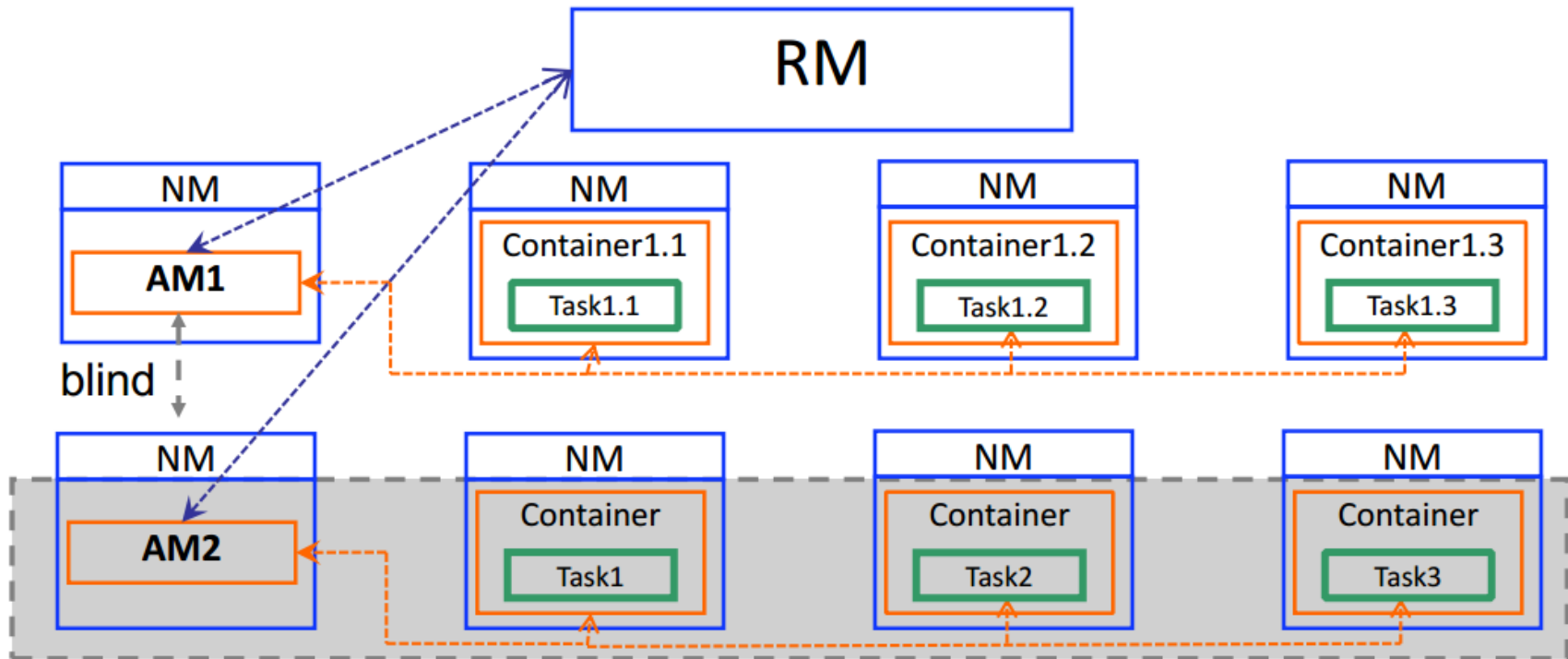
```
$ my-hadoop-app mydata
```



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

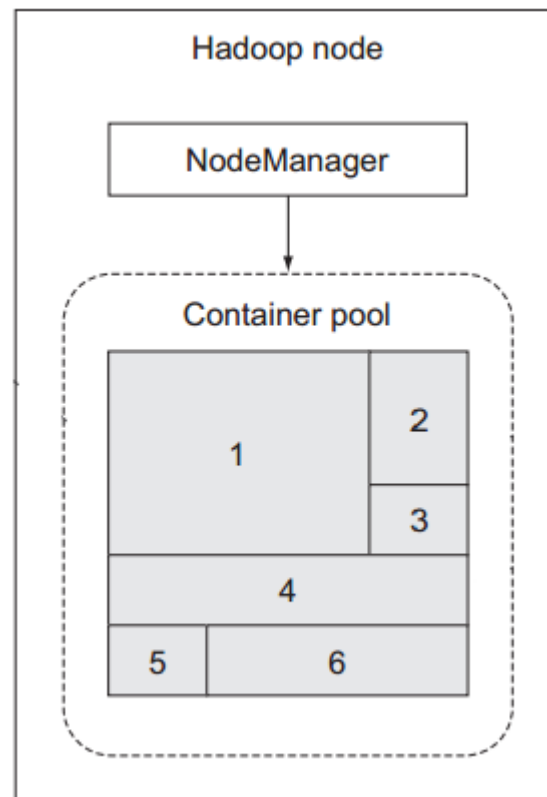


[illegible]

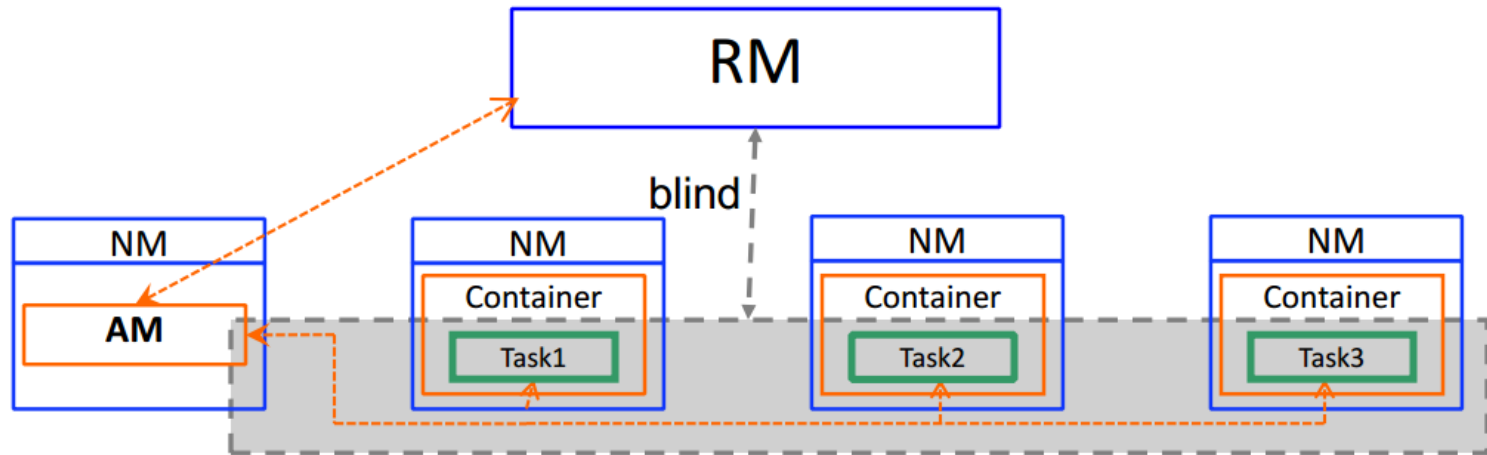
- AM is like the job tracker in Hadoop 1.0
 - Creates and manages task lifecycle
 - Monitors task status
- AM has no view of other running applications.

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



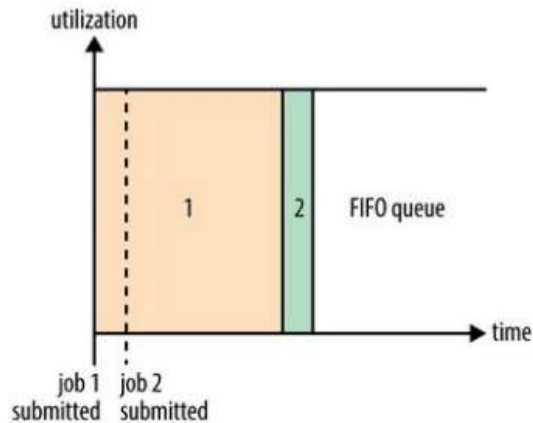
- When a job is submitted, RM assigns a jobID to it and allocates a container to run the corresponding AM.
- AM then asks for resources to run its job. After it gets the lease, the AM starts tasks and assigns tasks to containers.
- RM is blind to the tasks running within an application.

Scheduling in YARN, part of Resource Manager

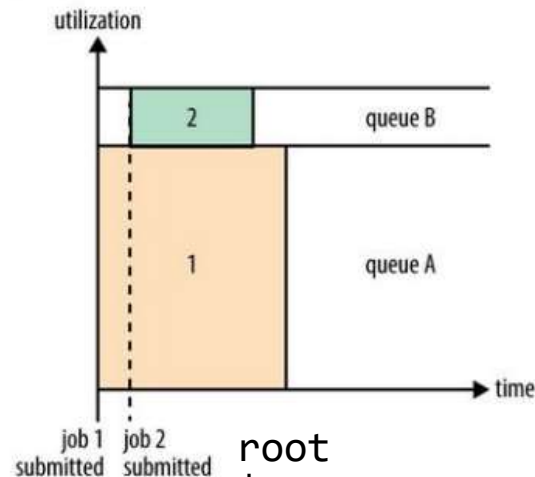
Scheduler is responsible for allocating resources to the various running applications.

FIFO, Capacity, Fair Schedulers:

i. FIFO Scheduler



ii. Capacity Scheduler



root

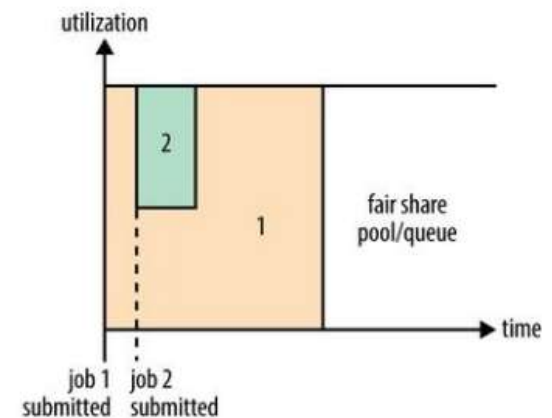
└─ prod

└─ dev

└─ eng

└─ science

iii. Fair Scheduler

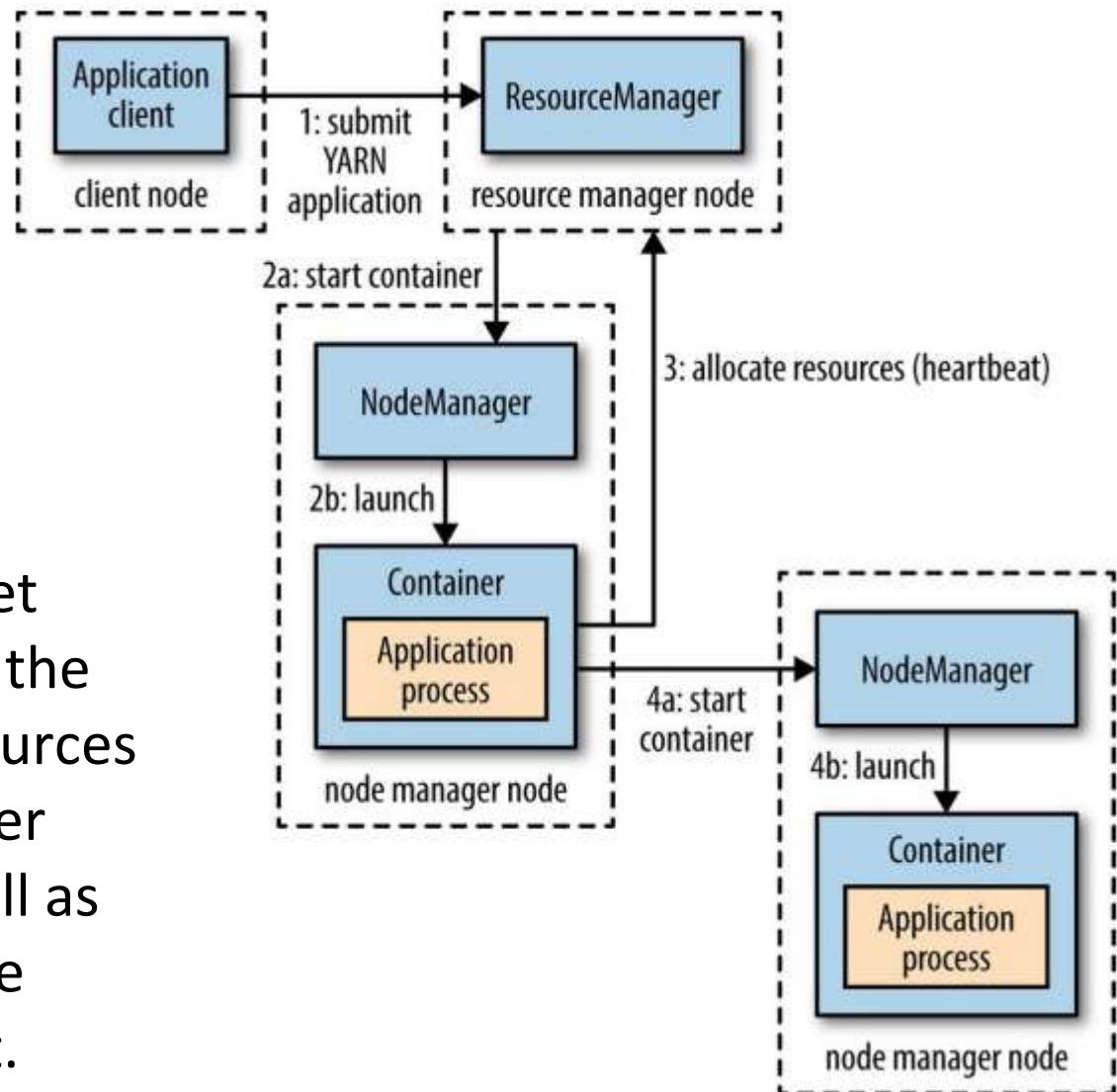


Outline

- Introduction to YARN
- Lifecycle 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.



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 from 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.

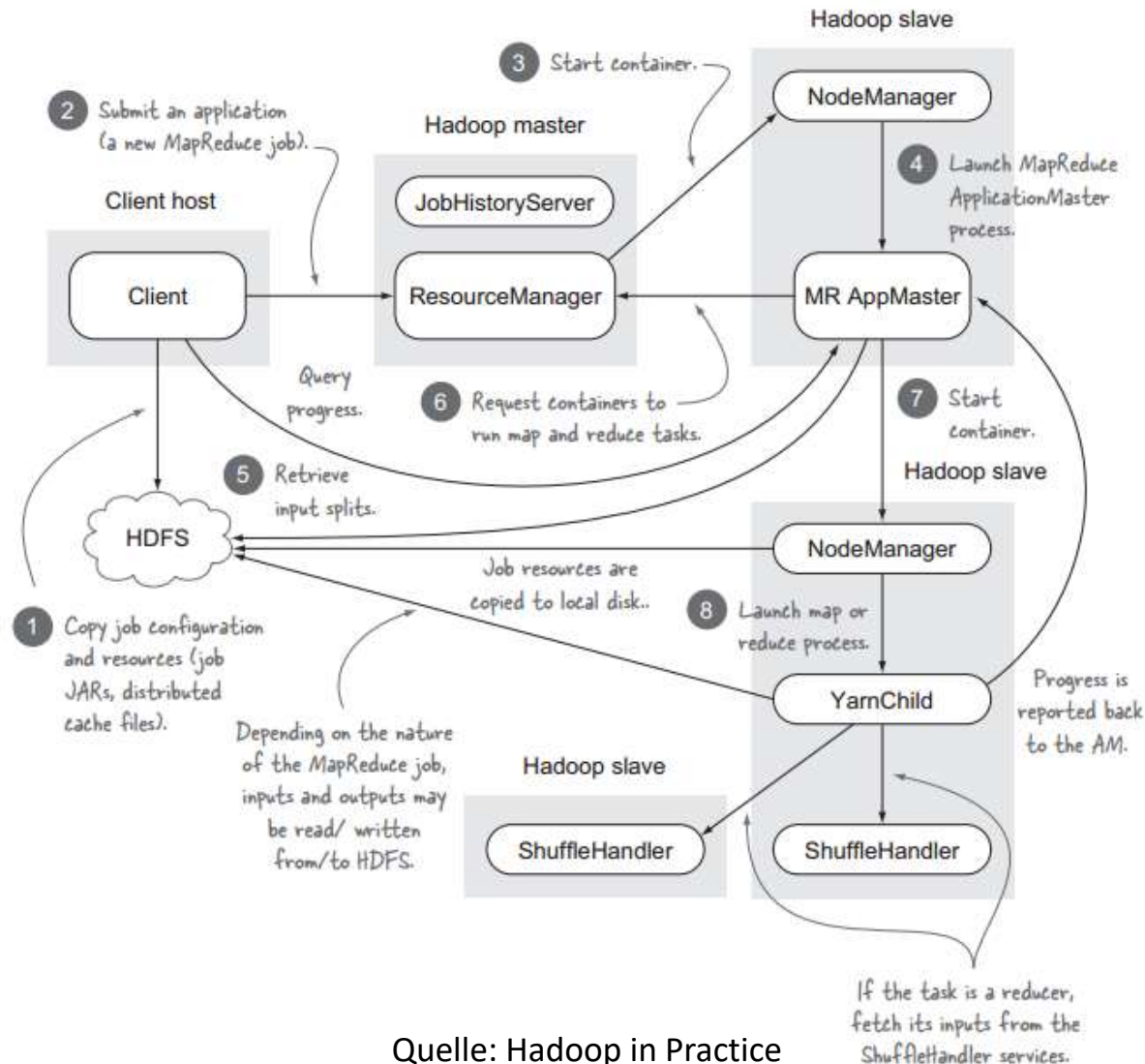
Lifecycle: Task assignment

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

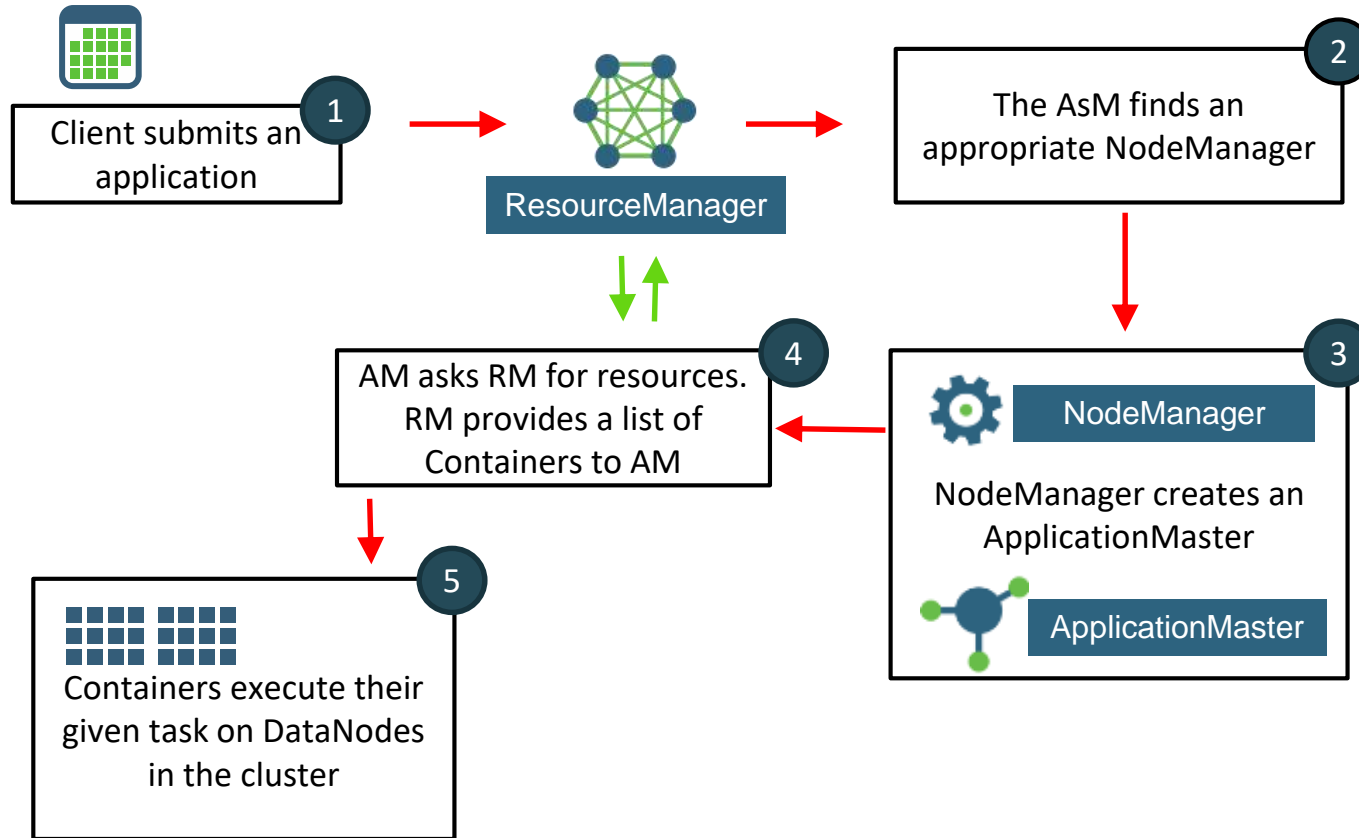
- The MapReduce AM also runs appropriate task cleanup code of completed tasks
- Once the entire map and reduce tasks are complete, the MapReduce AM runs the requisite job commit
- The MapReduce AM informs the RM then exits since the job is complete

Application Life Cycle im Detail



Quelle: Hadoop in Practice

Lifecycle of a YARN Application



Outline

- Introduction to YARN
- Lifecycle 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

Username: Text:

Succeeded Running Failed Killed

Logs	ID	Name	Status	User	Maps	Reduces	Queue	Priority	Duration	Submitted	
	1424901249645_0002	webpage.jar	RUNNING	training	5%	5%	root.training	N/A	18s	03/06/15 11:00:17	Kill
	1424901249645_0001	accounts.jar	SUCCEEDED	training	100%	100%	root.training	N/A	1m:21s	03/06/15 10:57:48	

Showing 1 to 2 of 2 entries

← Previous 1 Next →

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

hadoop

Nodes of the cluster

Cluster Overview

Logged in as: dr.who

Cluster

About

Nodes

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

REMOVING

FINISHING

FINISHED

FAILED

KILLED

Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
4	0	1	3	8	8 GB	8 GB	2 GB	2	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved
0	0	1	3	0	0	0	0 B	0 B	0 B

Show 20 entries

Search:

Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail
/default-rack	RUNNING	qsslave1:8041	qsslave1:8042	21-Nov-2013 13:07:26		4	4 GB	0 B
/default-rack	RUNNING	qsmaster:8041	qsmaster:8042	21-Nov-2013 13:07:17		4	4 GB	0 B

Showing 1 to 2 of 2 entries

First Previous 1 Next Last

link to Node Manager UI

List of each node in cluster

Resource Manager UI: Applications

hadoop

Logged in as: dr.who

All Applications

Cluster Overview

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
8	0	1	7	5	6 GB	8 GB	0 B	1	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved
0	0	1	7	0	0	0	0 B	0 B	0 B

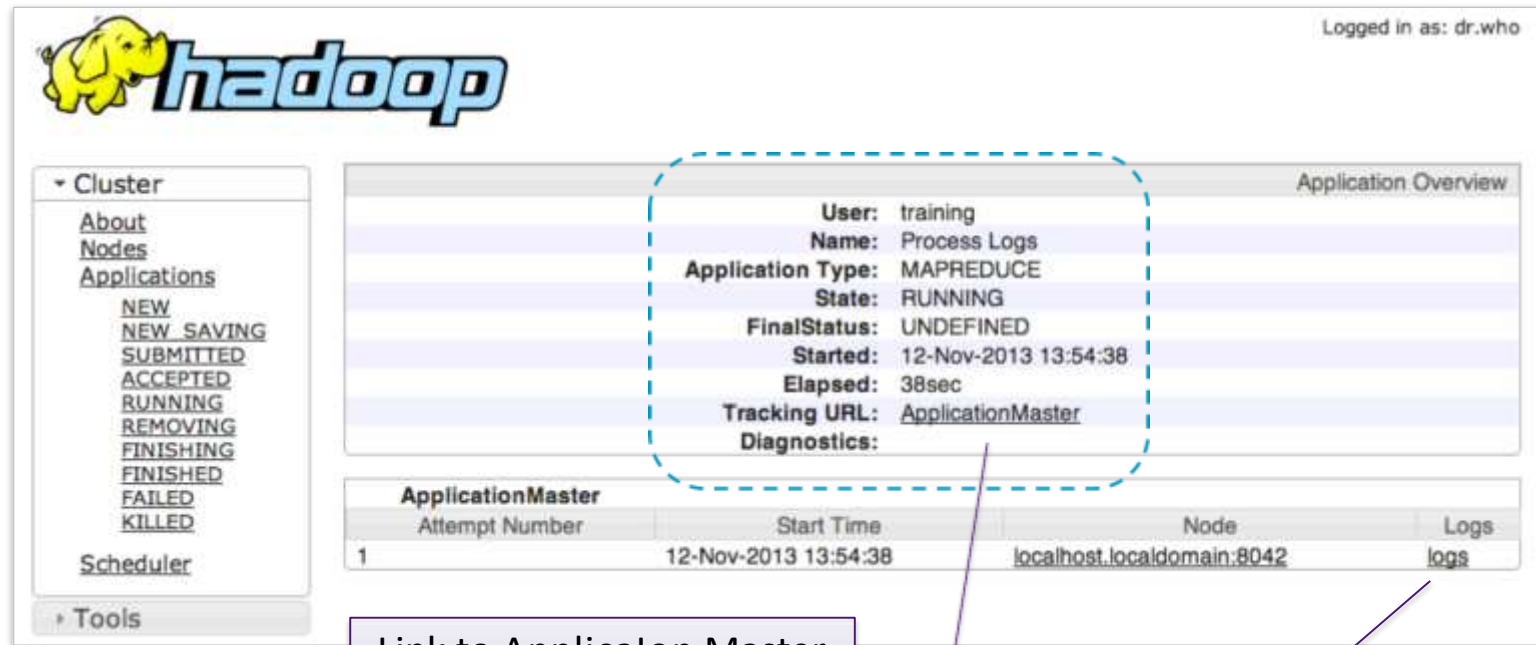
Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1384200217415_0006	training	Process Logs	MAPREDUCE	root.training	Tue, 12 Nov 2013 18:54:38 GMT	N/A	RUNNING	UNDEFINED	<div></div>	ApplicationMaster
application_1384200217415_0008	training	Average Word Length	MAPREDUCE	root.training	Mon, 11 Nov 2013 21:55:21 GMT	Mon, 11 Nov 2013 21:57:30 GMT	FINISHED	SUCCEEDED	<div></div>	History
application_1384200217415_0007	training	Process Logs	MAPREDUCE	root.training	Mon, 11 Nov 2013 21:36:39 GMT	Mon, 11 Nov 2013 21:44:19 GMT	FINISHED	SUCCEEDED	<div></div>	History
application_1384200217415_0006	training	Process Logs	MAPREDUCE	root.training	Mon, 11 Nov 2013	Mon, 11 Nov 2013	FINISHED	SUCCEEDED	<div></div>	History

Link to Application Details... (next slide)

List of running and recent applications

Resource Manager UI: Application Detail



The screenshot displays the Hadoop Resource Manager UI. The top left features the Hadoop logo and a navigation menu under 'Cluster' including 'About', 'Nodes', 'Applications' (with sub-links: NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING, REMOVING, FINISHING, FINISHED, FAILED, KILLED), 'Scheduler', and 'Tools'. The top right shows the user 'dr.who' is logged in. The main area is titled 'Application Overview' and contains a summary box for the application 'Process Logs' (User: training, Application Type: MAPREDUCE, State: RUNNING, FinalStatus: UNDEFINED, Started: 12-Nov-2013 13:54:38, Elapsed: 38sec, Tracking URL: ApplicationMaster, Diagnostics:). Below this is a table with columns 'ApplicationMaster', 'Attempt Number', 'Start Time', 'Node', and 'Logs'. The table shows one attempt (1) starting at 12-Nov-2013 13:54:38 on node localhost.localdomain:8042, with a link to 'logs'.

Link to Application Master
(UI depends on specific framework)

View aggregated log files (optional)

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

History
Server



hadoop **JobHistory** Logged in as: drowho

Application: About Jobs Tools

Retired Jobs

Show 20 entries Search:

Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed
2013.11.21 13:07:38 PST	2013.11.21 13:08:27 PST	job_1385066116114_0004	Process Logs	cloudera	default	SUCCEEDED	4	4	12	12
2013.11.21 13:03:53 PST	2013.11.21 13:04:42 PST	job_1385066116114_0003	Process Logs	cloudera	default	SUCCEEDED	4	4	12	12
2013.11.21 13:01:35 PST	2013.11.21 13:02:28 PST	job_1385066116114_0002	Process Logs	cloudera	default	SUCCEEDED	4	4	12	12
2013.11.21 12:48:00 PST	2013.11.21 12:50:43 PST	job_1385066116114_0001	Word Count	cloudera	default	SUCCEEDED	4	4	1	1
2013.11.21 09:24:45 PST	2013.11.21 09:28:19 PST	job_1385049040288_0003	Word Count	cloudera	default	SUCCEEDED	4	4	1	1

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

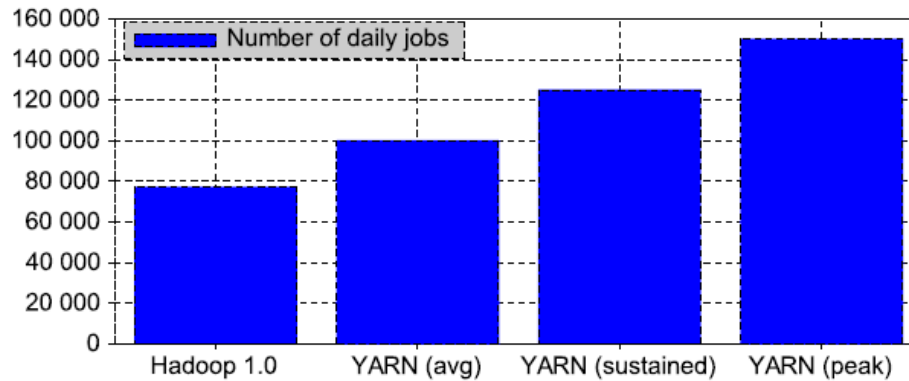
yarn application -kill <id> kills the application

yarn node -list Lists all running nodes.

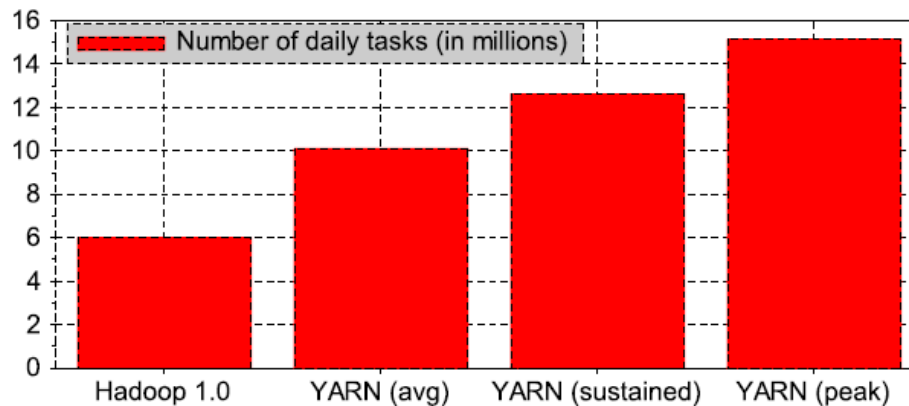
Essential Points

- 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]