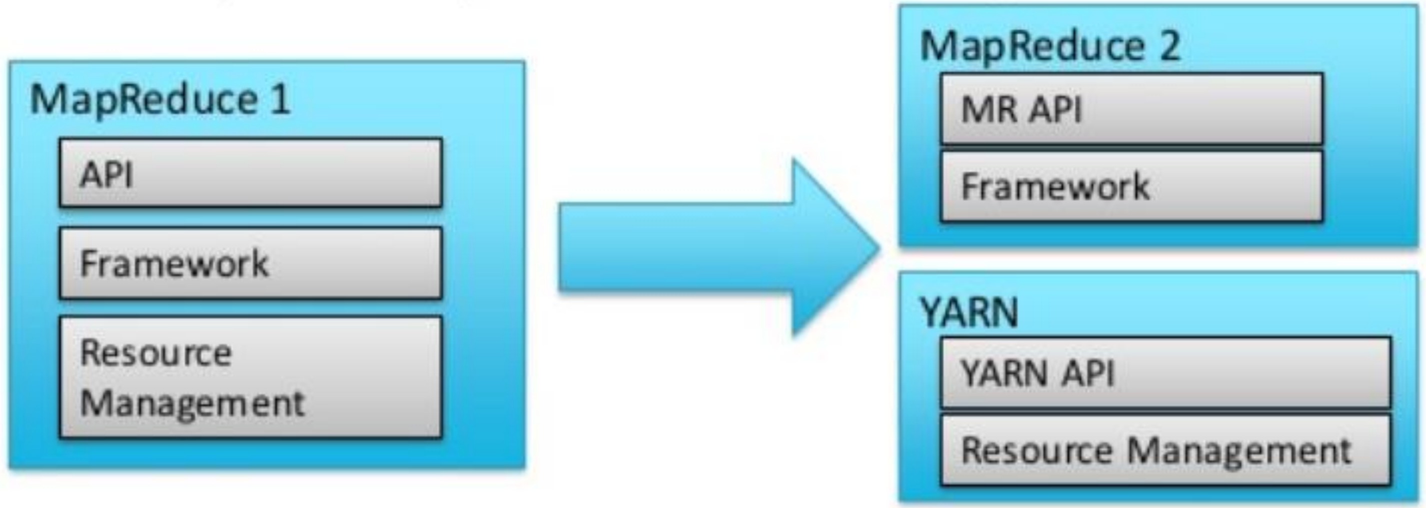
Hadoop MapReduce (Architecture)

Learn to Evolve

Introduction to YARN and MapReduce 2

MRv1 and MRv2

- **MapReduce 1 ("Classic") has three main components**
  - API – for user-level programming of MR applications
  - Framework – runtime services for running Map and Reduce processes, shuffling and sorting, etc.
  - Resource management – infrastructure to monitor nodes, allocate resources, and schedule jobs
- **MapReduce 2 ("NextGen") moves Resource Management into YARN**

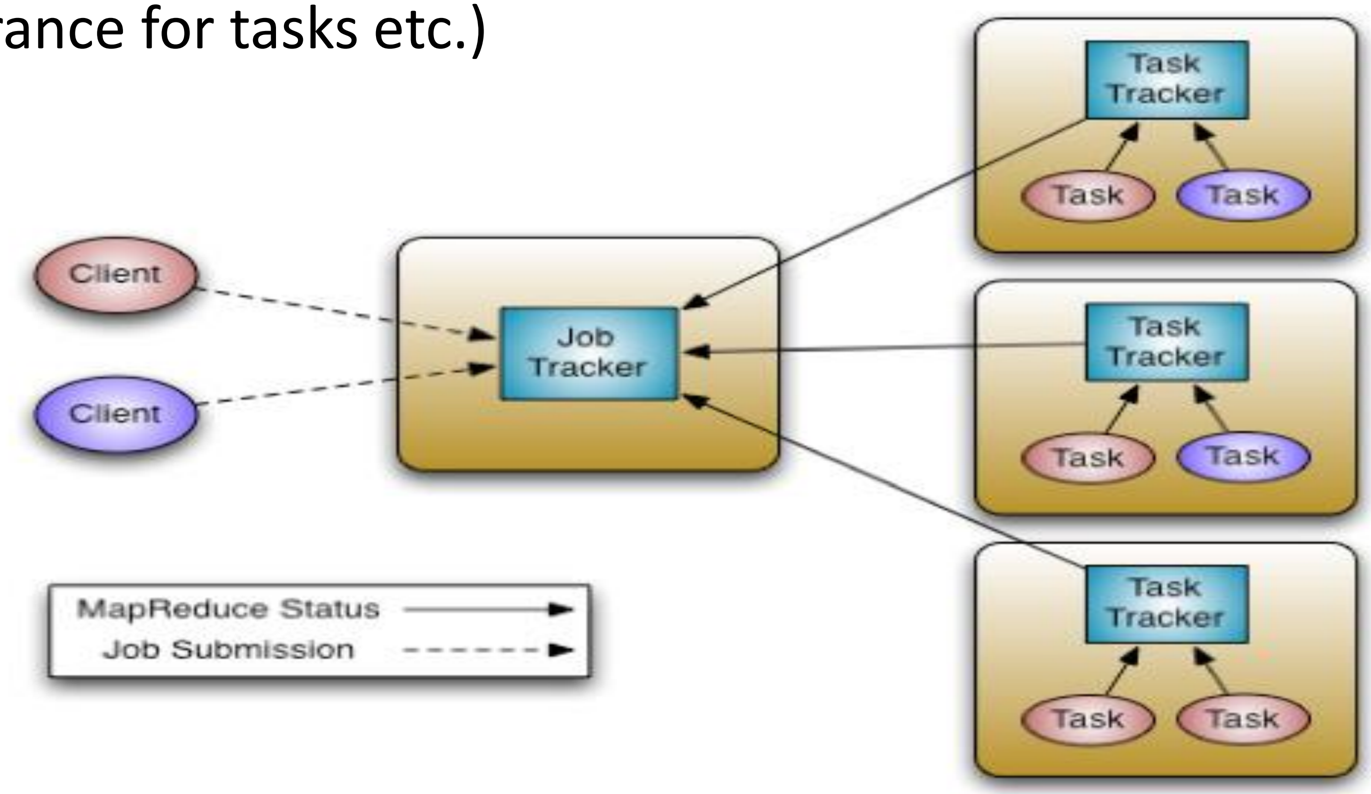


ANALYTIXLABS

MapReduce V1: Job Tracker & Task Tracker

- ✓ The JobTracker is responsible for
  1. *resource management* (managing the worker nodes i.e. TaskTrackers)
  2. *Tracking resource consumption/availability*
  3. *Job life-cycle management* (scheduling individual tasks of the job, tracking progress, providing fault-tolerance for tasks etc.)

- ✓ The TaskTracker has simple responsibilities
  1. Launch/teardown tasks on orders from the JobTracker
  2. Provide task-status information to the JobTracker periodically.



Issues with MapReduce (v1)?

- JobTracker has issues related to
  - scalability
  - cluster utilization
  - ability for customers to control upgrades to the stack
  - supporting workloads other than MapReduce itself
  - resiliency to HDFS issues
- MapReduce is essentially batch-oriented and does not support real-time and near real-time processing such as stream processing
- There are utilization issues because map slots might be ‘full’ while reduce slots are empty (and vice-versa).

YARN (Yet Another Resource Negotiator)

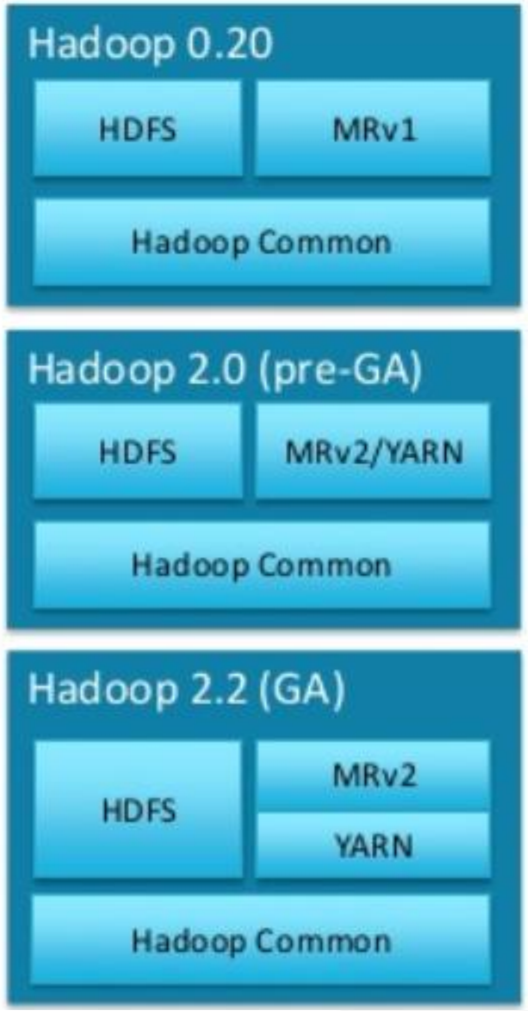
- MapReduce has undergone a complete overhaul and CDH 5 now includes MapReduce 2.0 (MRv2).
- The fundamental idea of MRv2's YARN architecture is to split up the two primary responsibilities of the Job Tracker into separate daemons (Global Resource Manager(RM) and Per-Application Masters(AM)
  - 1. Resource management and
  - 2. job scheduling/monitoring
- Yarn provides its core services via two types of long running daemon: a Resource manager and Node manager

YARN (Yet Another Resource Negotiator)

- **A platform for managing resources in a Hadoop cluster**
- **An Apache Hadoop sub-project**
- **Supports several distributed processing frameworks, including:**
  - MapReduce v2
  - Impala
  - Spark
  - Giraph
- **A full list is at `wiki.apache.org/hadoop/PoweredByYarn`**

MapReduce2 History

- Originally architected at Yahoo in 2008
- "Alpha" in Hadoop 2 pre-GA
  - Included in CDH 4
- YARN promoted to Apache Hadoop sub-project
  - summer 2013
- "Production ready" in Hadoop 2 GA
  - Included in CDH5 (Beta in Oct 2013)

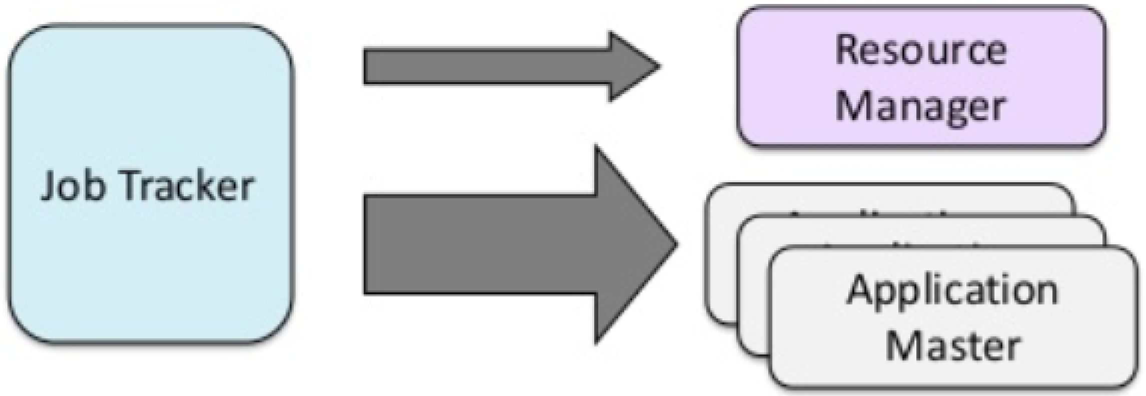


Why is YARN needed? (1)

- **MRv1 resource management issues**
  - Inflexible “slots” configured on nodes – Map or Reduce, not both
    - Underutilization of cluster when more map or reduce tasks are running
  - Can’t share resources with non-MR applications running on Hadoop cluster (e.g. Impala, Giraph)
  - Scalability – one JobTracker per cluster – limit of about 4000 nodes per cluster

Why is YARN needed? (2)

- **YARN solutions**
  - No slots
    - Nodes have "resources" – memory and CPU cores – which are allocated to applications when requested
  - Supports MR and non-MR applications running on the same cluster
  - Most Job Tracker functions moved to Application Master – one cluster can have many Application Masters



YARN Daemons

- **Resource Manager (RM)**
  - Runs on master node
  - Global resource scheduler
  - Arbitrates system resources between competing applications

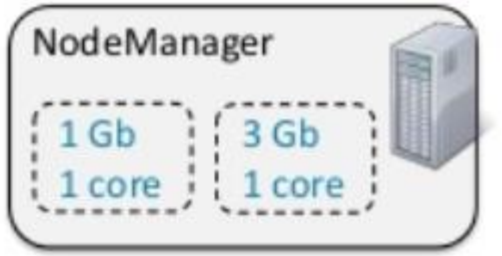
Resource
Manager

- **Node Manager (NM)**
  - Runs on slave nodes
  - Communicates with RM

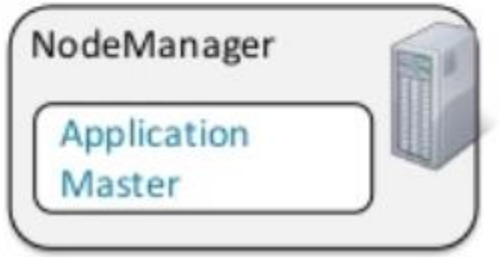
NodeManager
NodeManager

Running an Application in YARN

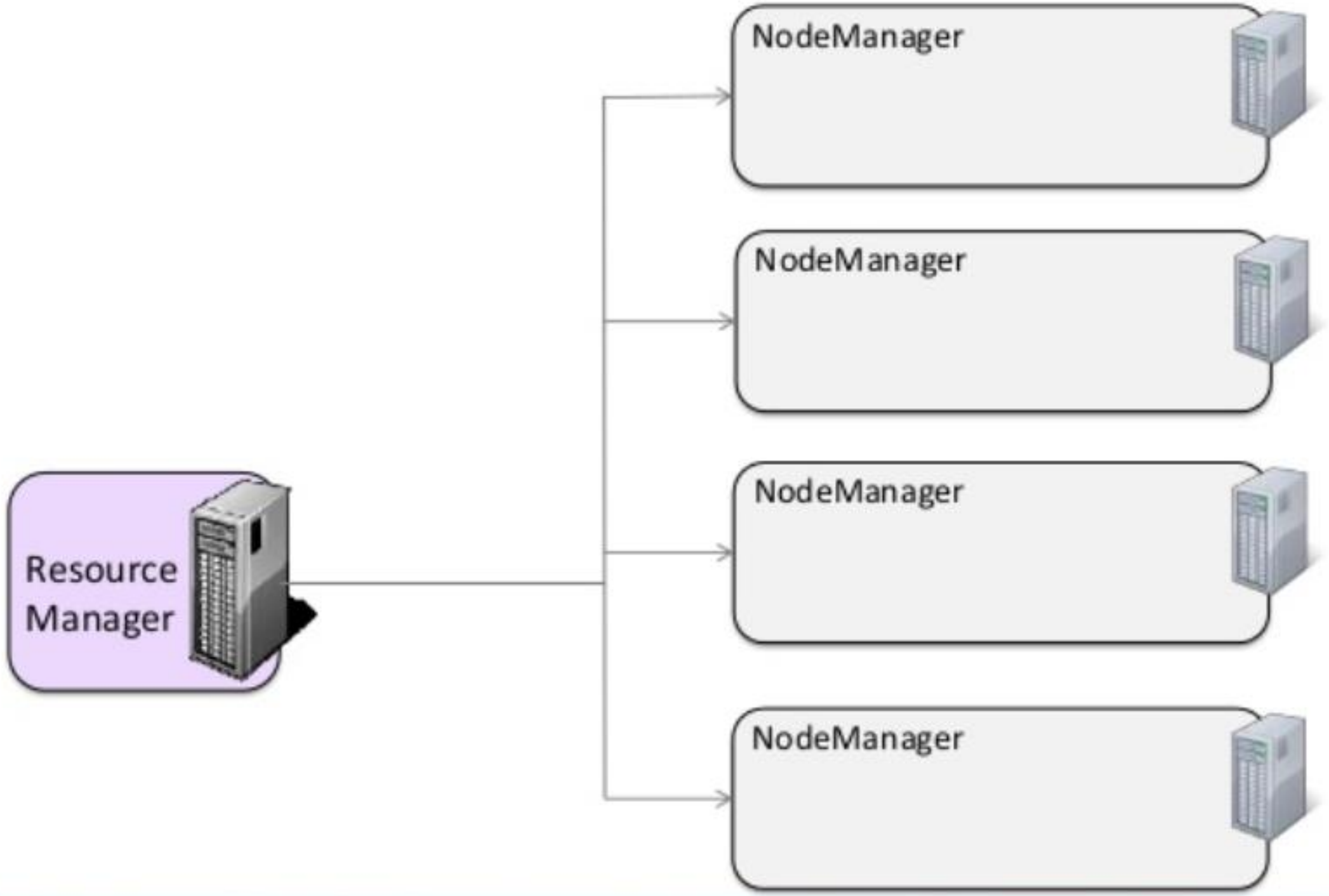
- **Containers**
  - Created by the RM upon request
  - Allocate a certain amount of resources (memory, CPU) on a slave node
  - Applications run in one or more containers



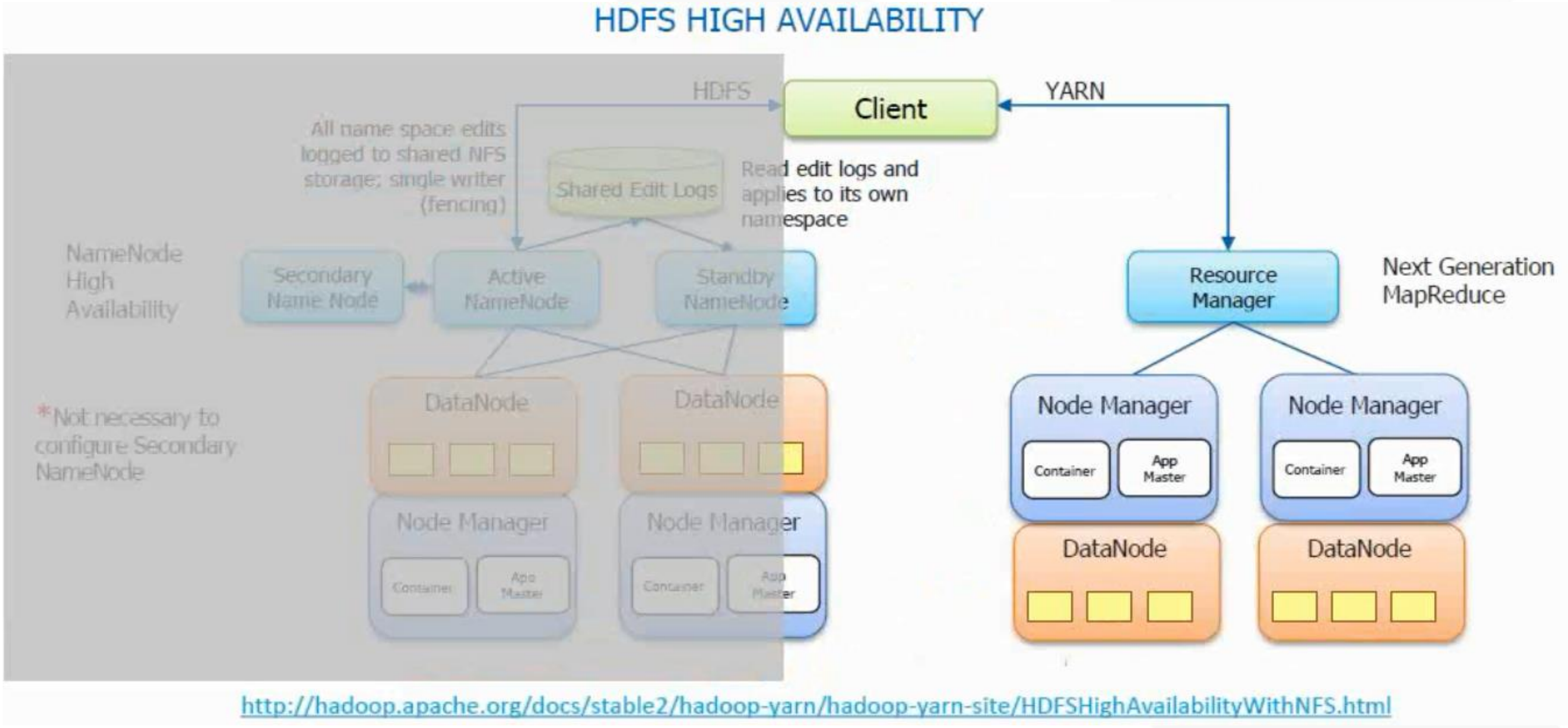
- **Application Master (AM)**
  - One per application
  - Framework/application specific
  - Runs in a container
  - Requests more containers to run application tasks



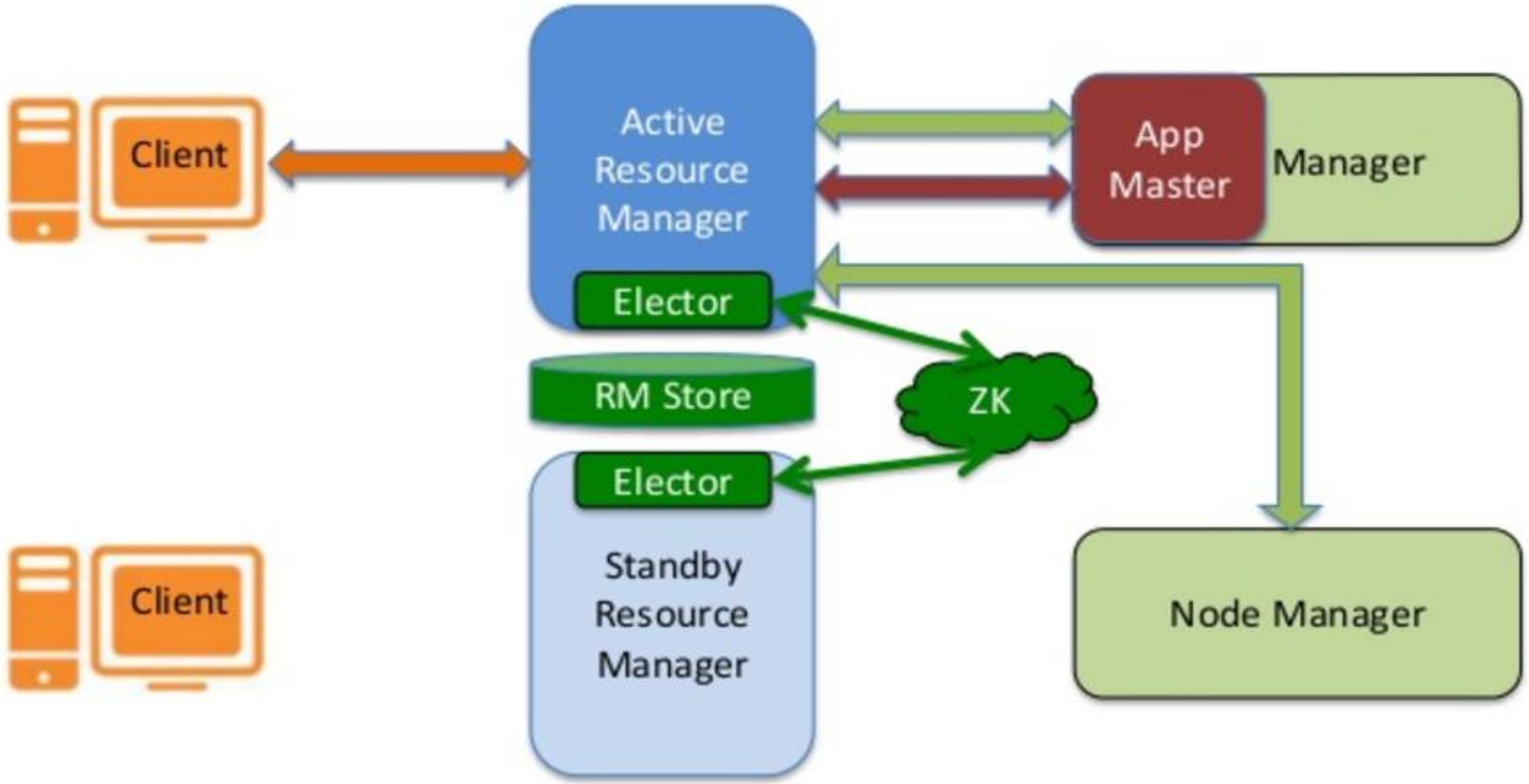
YARN Cluster

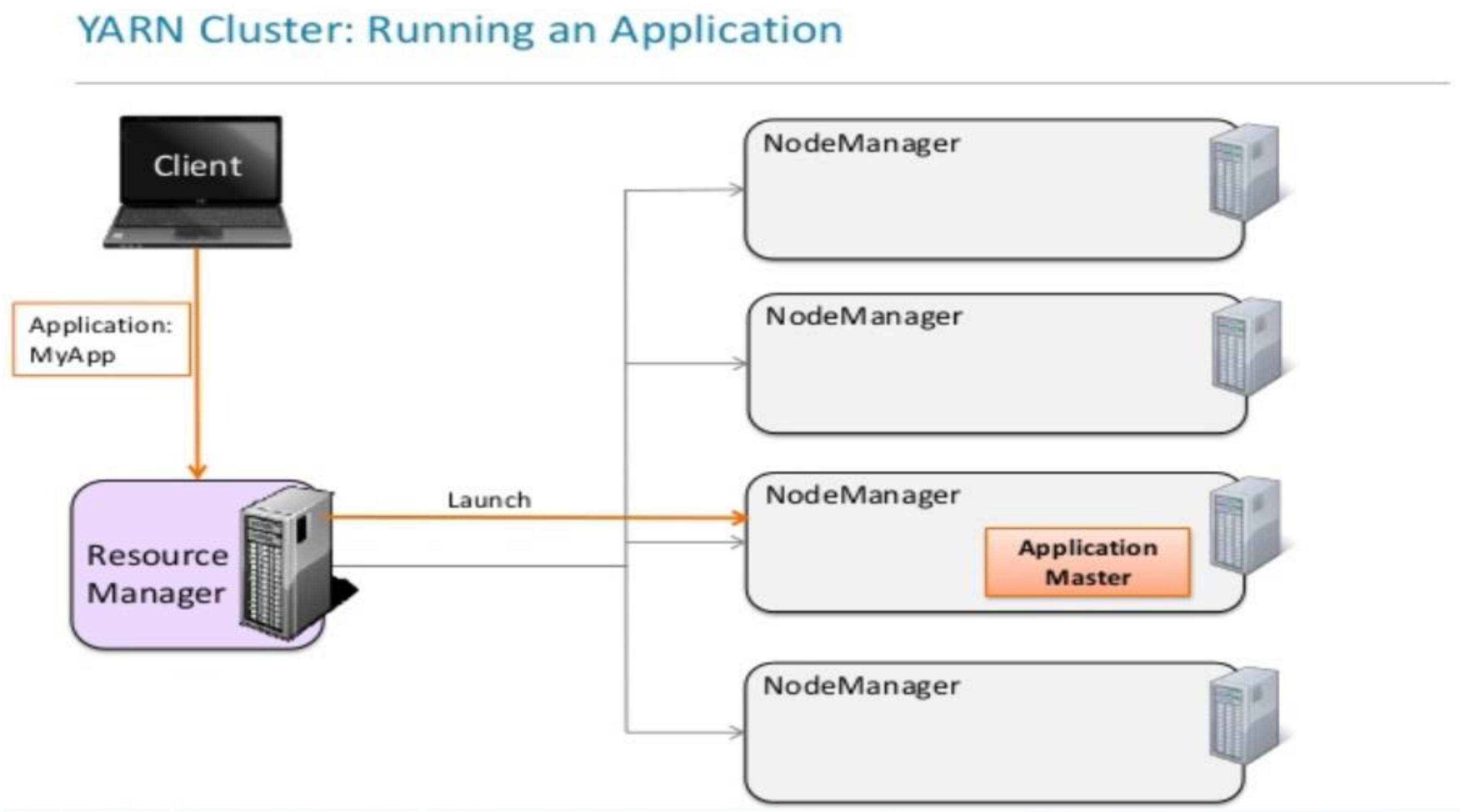


Resource Manager – High Availability

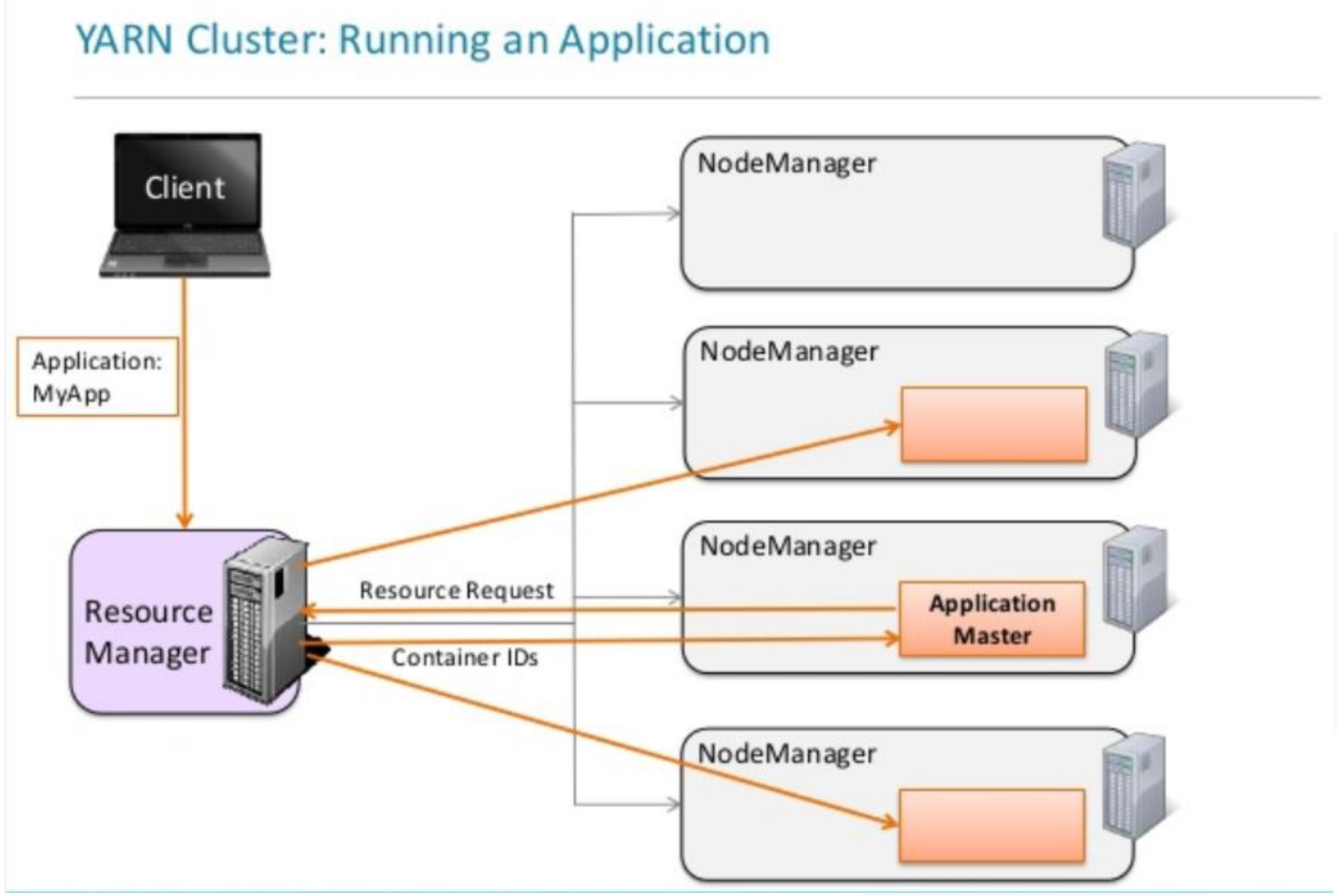

HDFS HIGH AVAILABILITY
HDFS
Client
YARN
All name space edits logged to shared NFS storage; single writer (fencing)
Shared Edit Logs
Read edit logs and applies to its own namespace
NameNode High Availability
Secondary Name Node
Active NameNode
Standby NameNode
Resource Manager
Next Generation MapReduce
DataNode
DataNode
Node Manager
Node Manager
Container
App Master
Container
App Master
*Not necessary to configure Secondary NameNode
Node Manager
Node Manager
Container
App Master
Container
App Master
DataNode
DataNode
http://hadoop.apache.org/docs/stable2/hadoop-yarn/hadoop-yarn-site/HDFSHighAvailabilityWithNFS.html

Resource Manager – High Availability



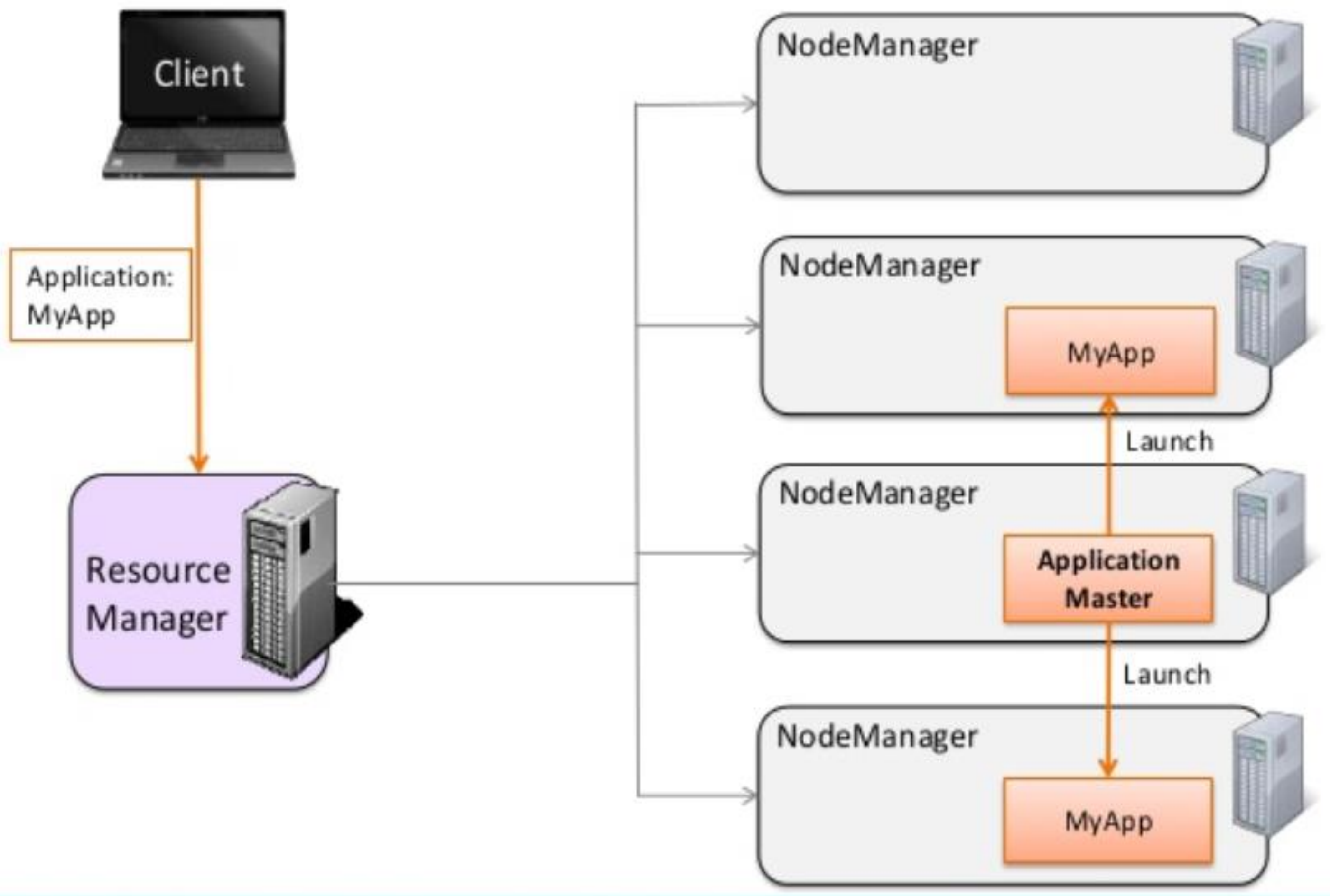


1. A client program submits the application, including the necessary specification to launch the application-specific Application Master
2. The Resource Manager assumes the responsibility to negotiate a specific container in which to start the Application Master and then launches the Application Master



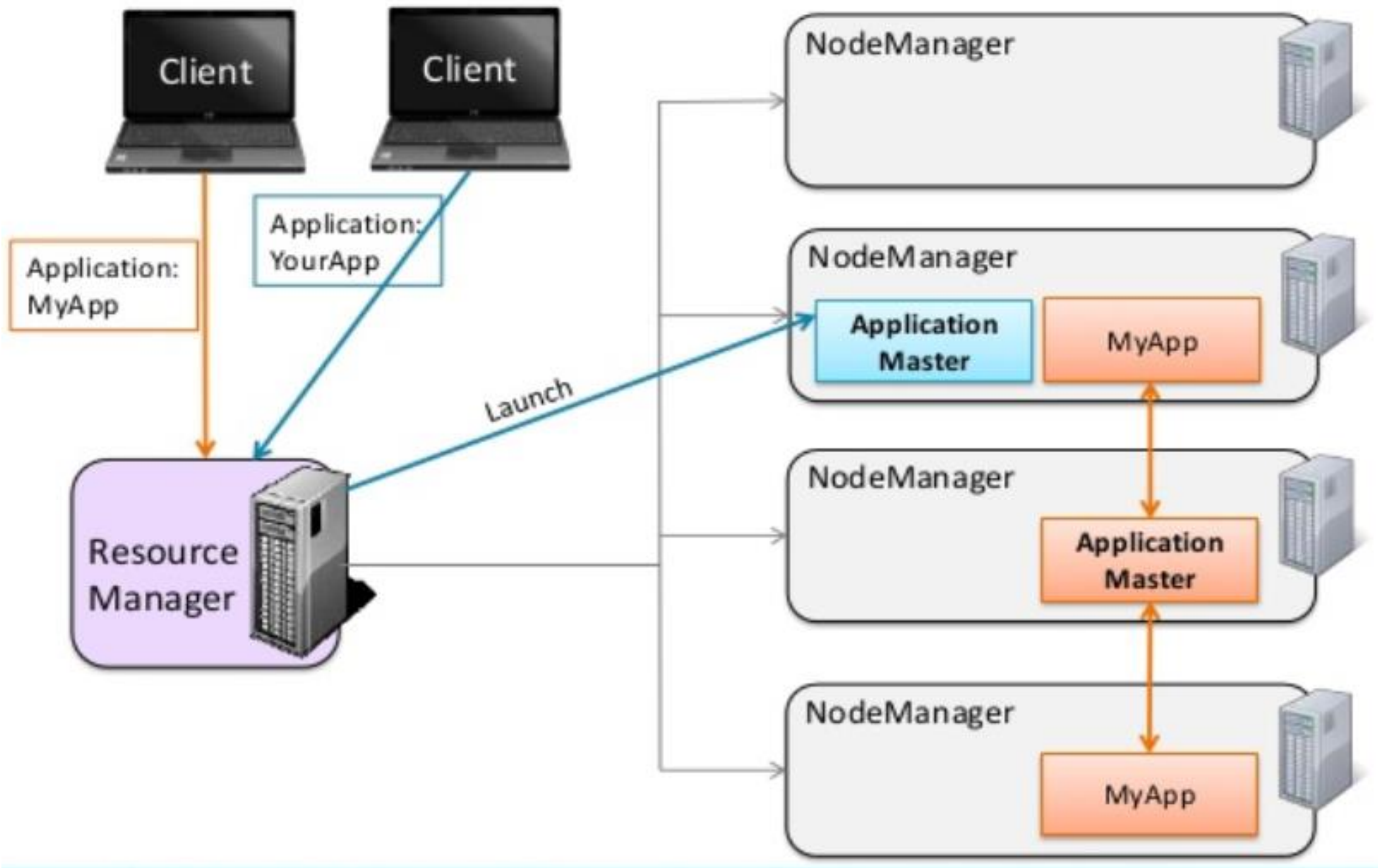
3. The Application Master on boot-up, register with the Resource Manager for details which allow it to directly communicate with its own Application Master
4. During normal operation Application Master negotiates appropriate resource container via the resource-request protocol

YARN Cluster: Running an Application



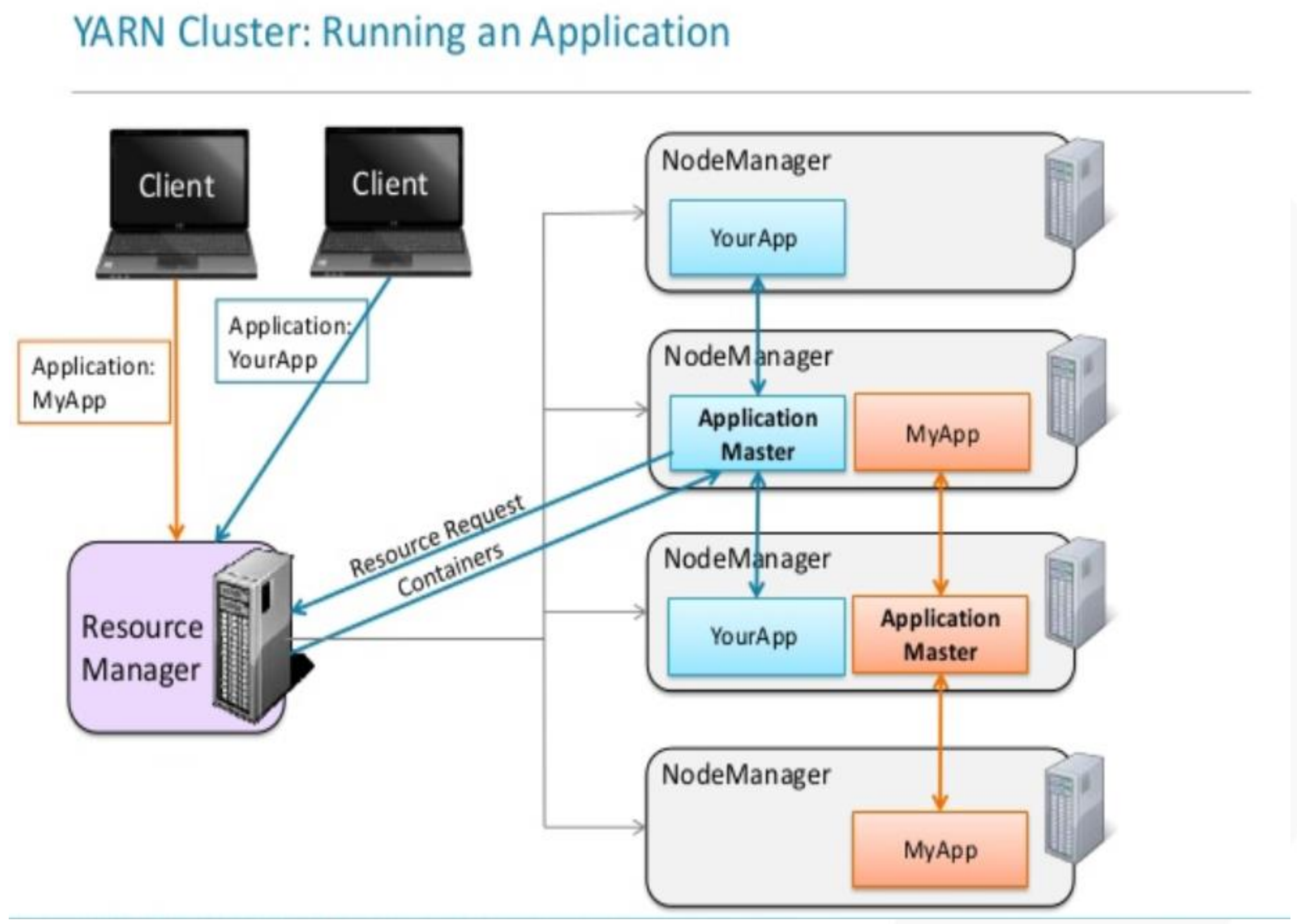
5. On successful container allocation, the Application Master launches the container by providing the container launch specification to the Node Manager. The launch specification, typically includes the necessary information to allow the container to communicate with the Application Master itself

YARN Cluster: Running an Application



6. In case of new application submitted, Resource Manager launches new application specific Application Master

ANALYTIXLABS



7. The application code executing within the container then provides the necessary information(progress, status) to its Application Master via application specific protocol
8. During the application execution, the client that submitted the program communicates directly with the Application Master to get the status, progress update etc. via an application specific protocol
9. Once the application is complete, and all necessary work has been finished, the application master deregisters with the Resource Manager and shuts down, allowing its own container to be repurposed.

YARN Schedulers (1)

- **Pluggable in Resource Manager**
- **YARN includes** three **schedulers**
  - CapacityScheduler
  - FairScheduler

- FIFO Scheduler **- Preemptive Scheduling***

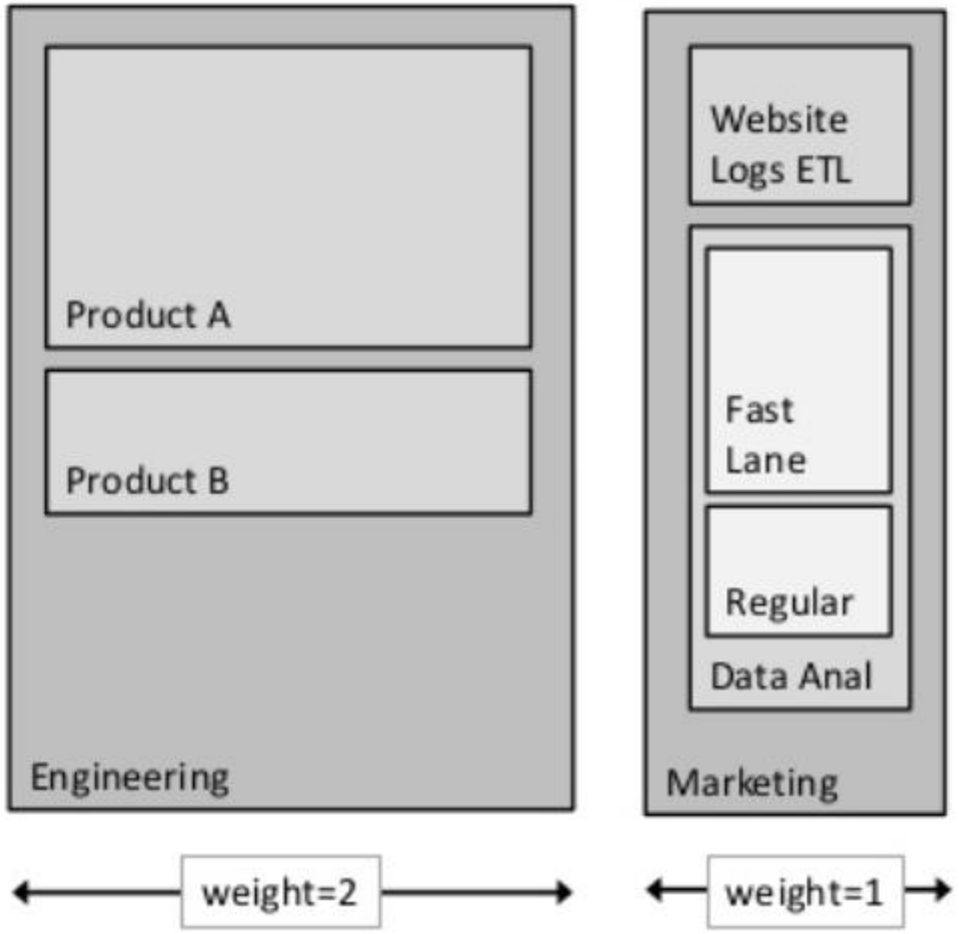
- Dominant Resource Fairness* **- Delay Scheduling**

- Speculative Execution*

- **How are these different than MRv1 schedulers?**
  - Support any YARN application, not just MR
  - No more "slots" – tasks are allocated based on resources (memory and CPU for now)
  - FairScheduler: *pools* are now called *queues*

YARN Schedulers (2)

- **Hierarchical queues**
  - Queues can contain sub-queues
  - Sub-queues share the resources assigned to queues



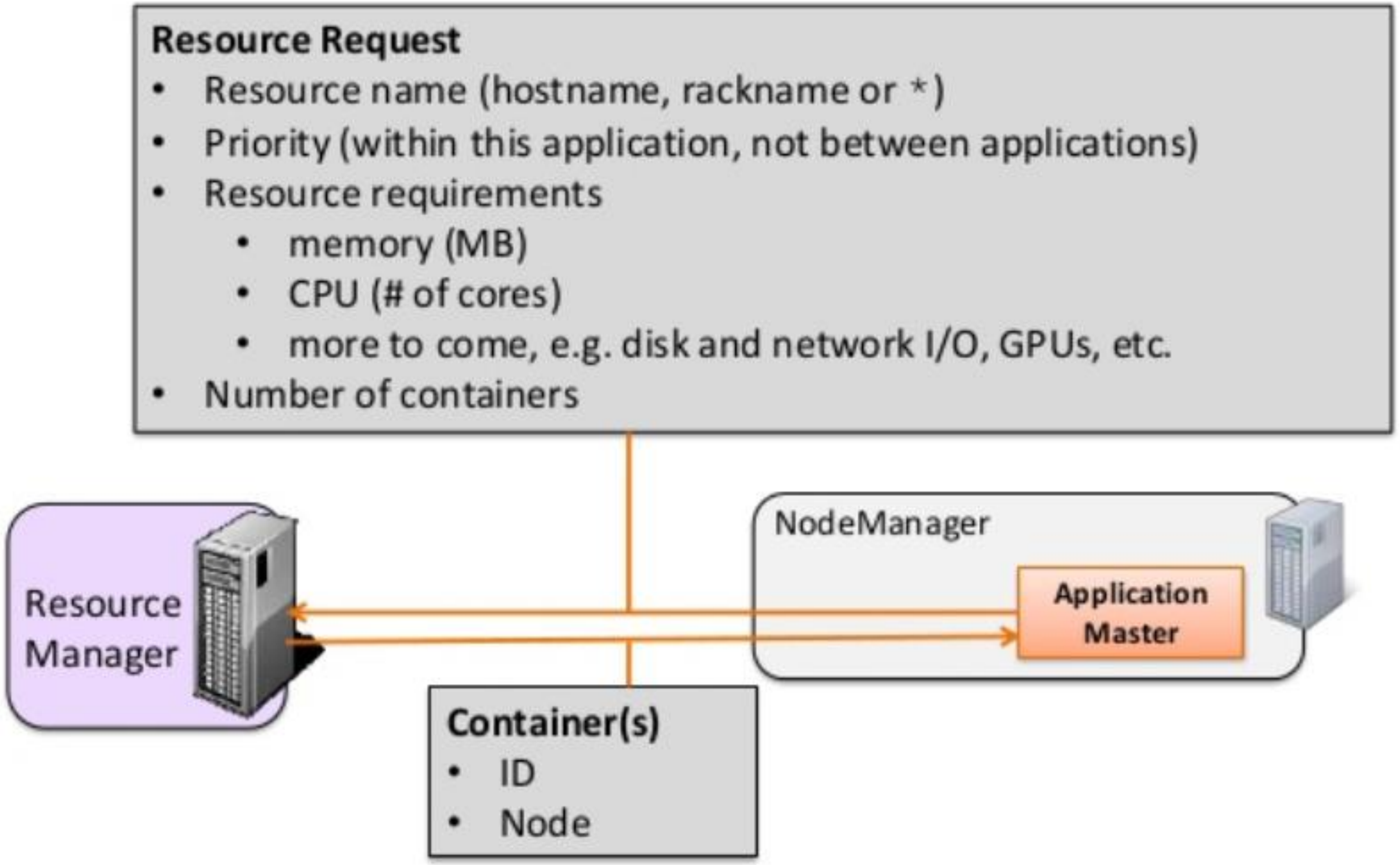
Resource Manager Things to Know

- **What it does**
  - Manages nodes
    - Tracks heartbeats from NodeManagers
  - Manages containers
    - Handles AM requests for resources
    - De-allocates containers when they expire or the application completes
  - Manages ApplicationMasters
    - Creates a container for AMs and tracks heartbeats
  - Manages security
    - Supports Kerberos

Node Manager Things to Know

- **What it does**
  - Communicates with the RM
    - Registers and provides info on node resources
    - Sends heartbeats and container status
  - Manages processes in containers
    - Launches AMs on request from the RM
    - Launches application processes on request from AM
    - Monitors resource usage by containers; kills run-away processes
  - Provides logging services to applications
    - Aggregates logs for an application and saves them to HDFS
  - Runs auxiliary services
  - Maintains node level security via ACLs

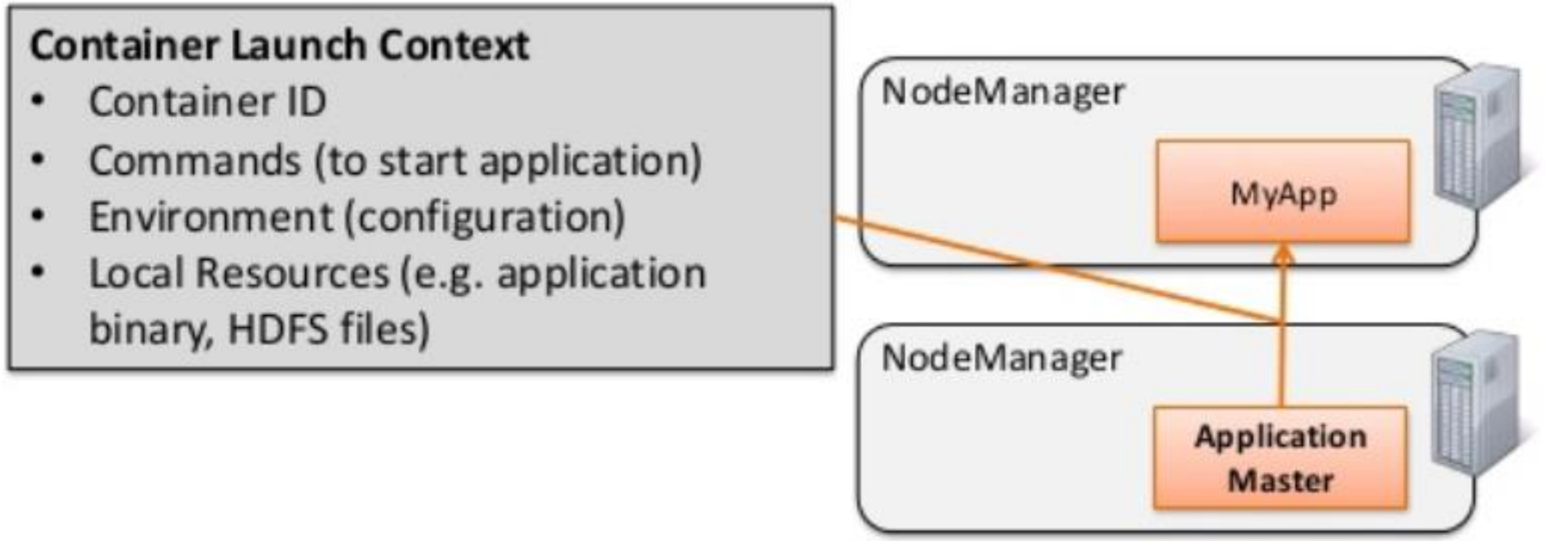
Resource Requests



Resource Request
- Resource name (hostname, rackname or *)
- Priority (within this application, not between applications)
- Resource requirements
  - memory (MB)
  - CPU (# of cores)
  - more to come, e.g. disk and network I/O, GPUs, etc.
- Number of containers

Container(s)
- ID
- Node

Launch Container

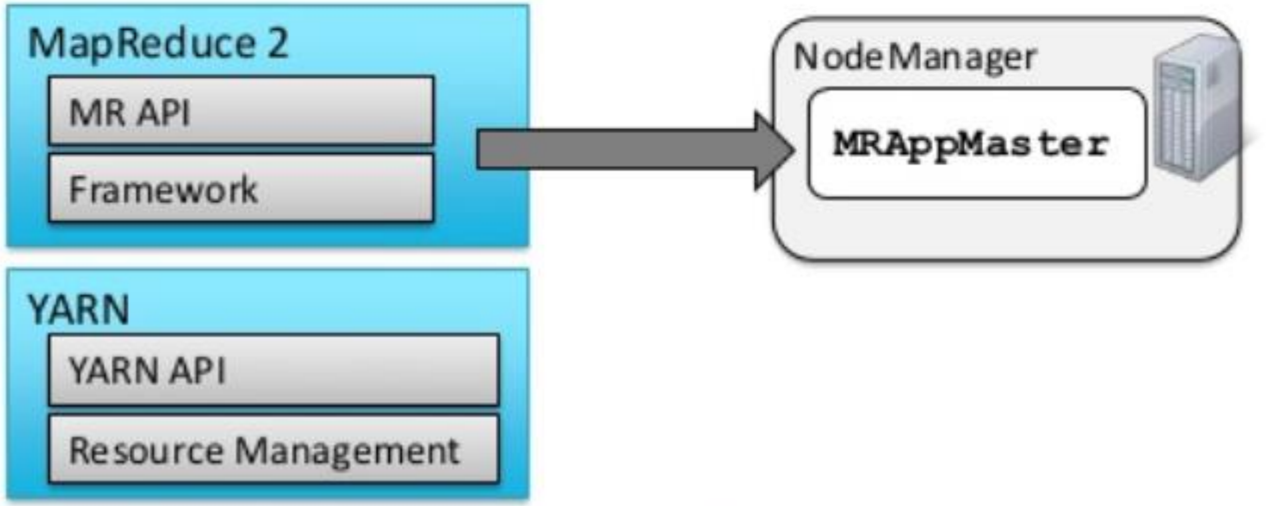

Container Launch Context
• Container ID
• Commands (to start application)
• Environment (configuration)
• Local Resources (e.g. application binary, HDFS files)
NodeManager
MyApp
NodeManager
Application Master

Non-MR2 YARN Applications

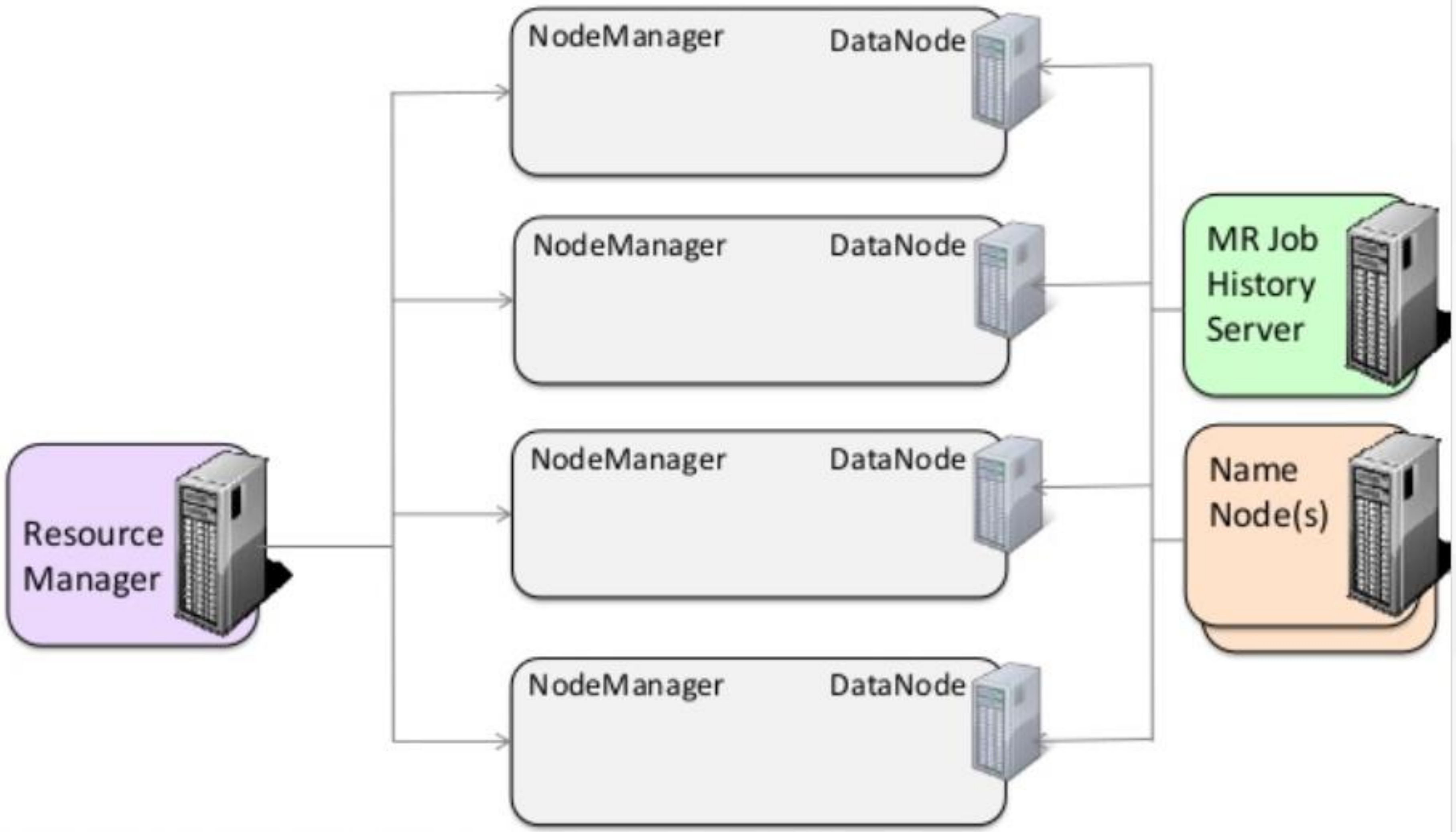
- **Distributed Shell**
- **Impala**
- **Apache Giraph**
- **Spark**
- **Others**
  - http://wiki.apache.org/hadoop/PoweredByYarn

YARN and MapReduce

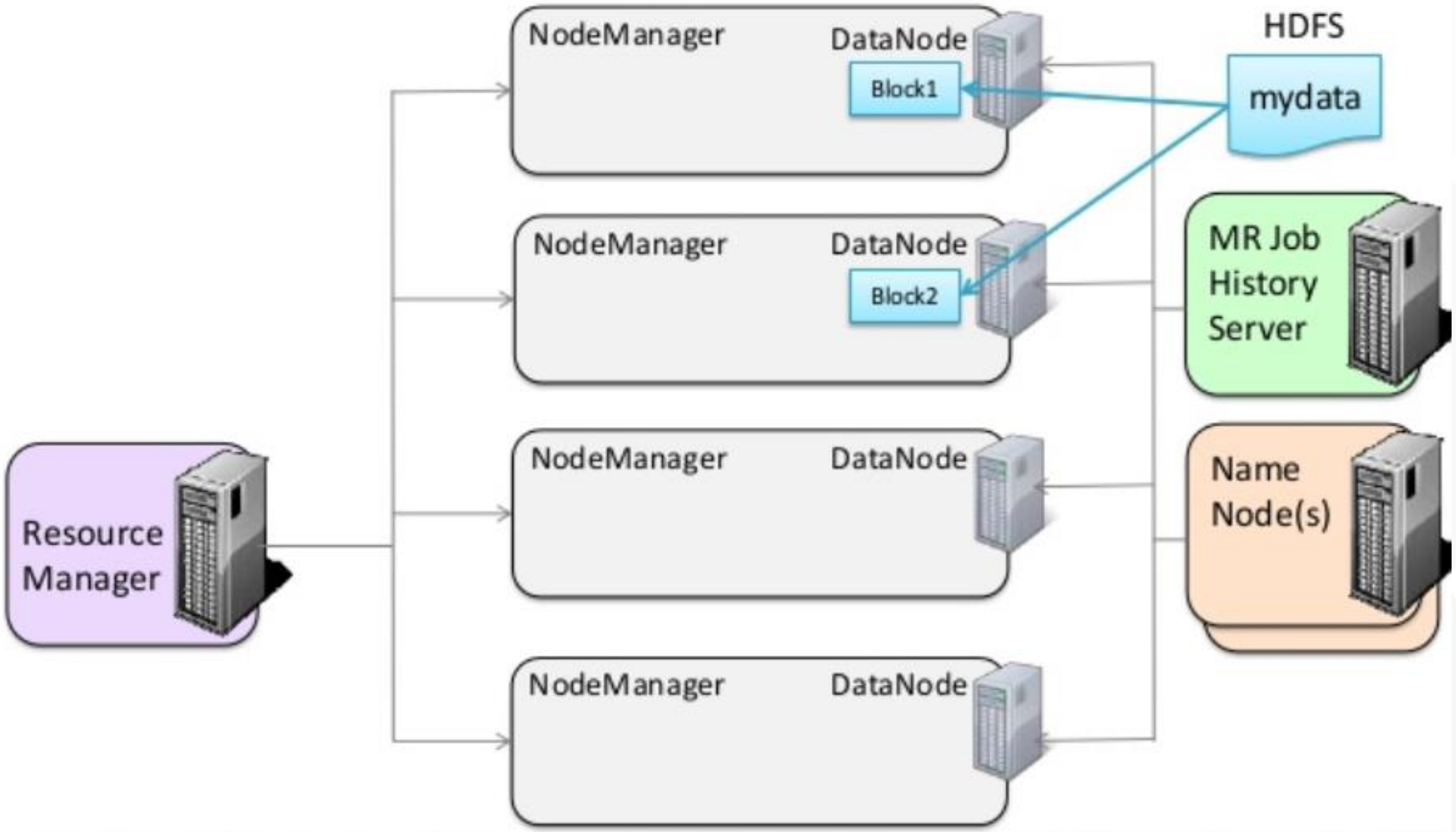
- **YARN does not know or care what kind of application it is running**
  - Could be MR or something else (e.g. Impala)
- **MR2 uses YARN**
  - Hadoop includes a MapReduce ApplicationMaster (MRAppMaster) to manage MR jobs
  - Each MapReduce job is an a new instance of an application



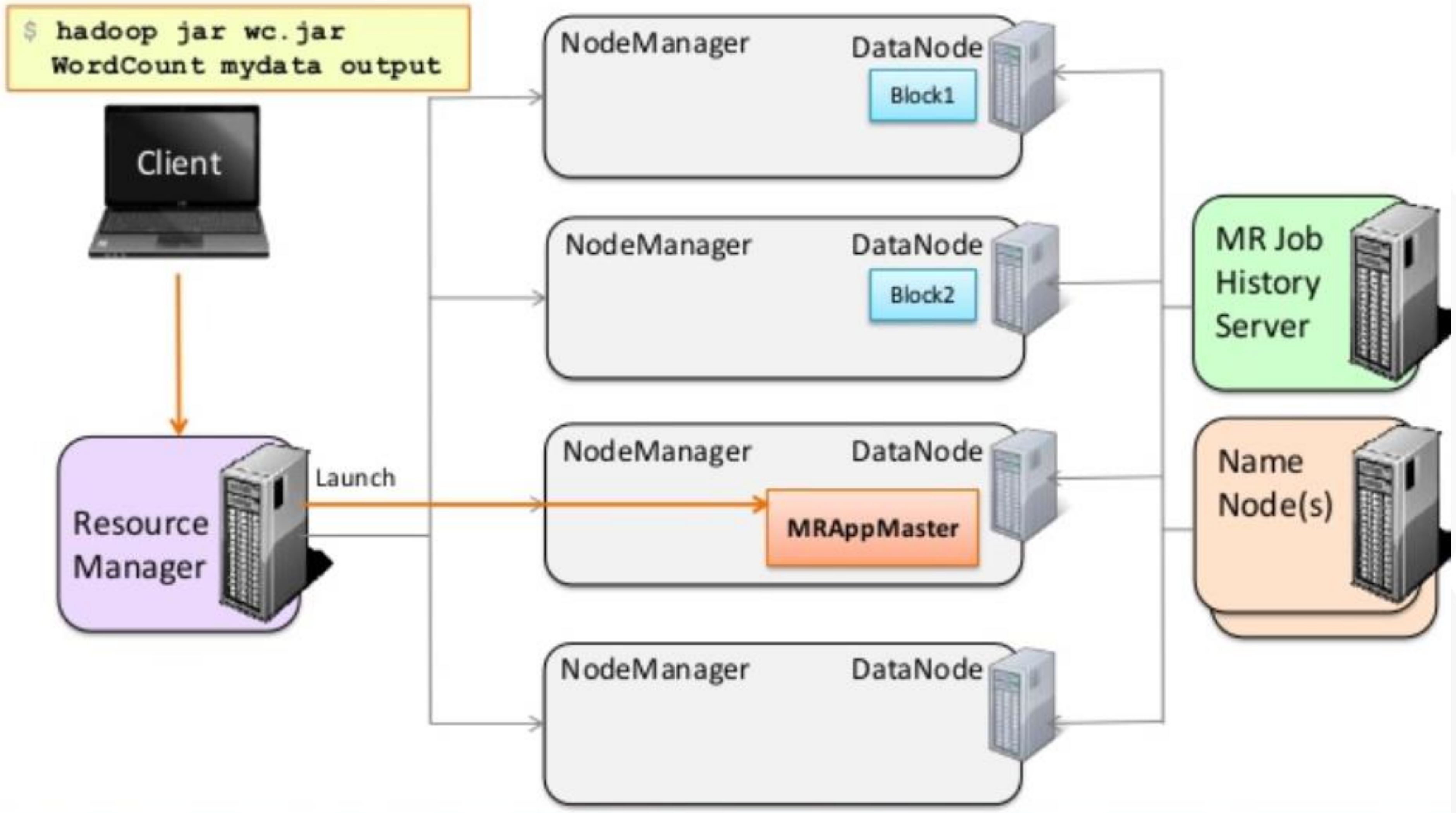
Running a MapReduce Application in MRv2



Running a MapReduce Application in MRv2

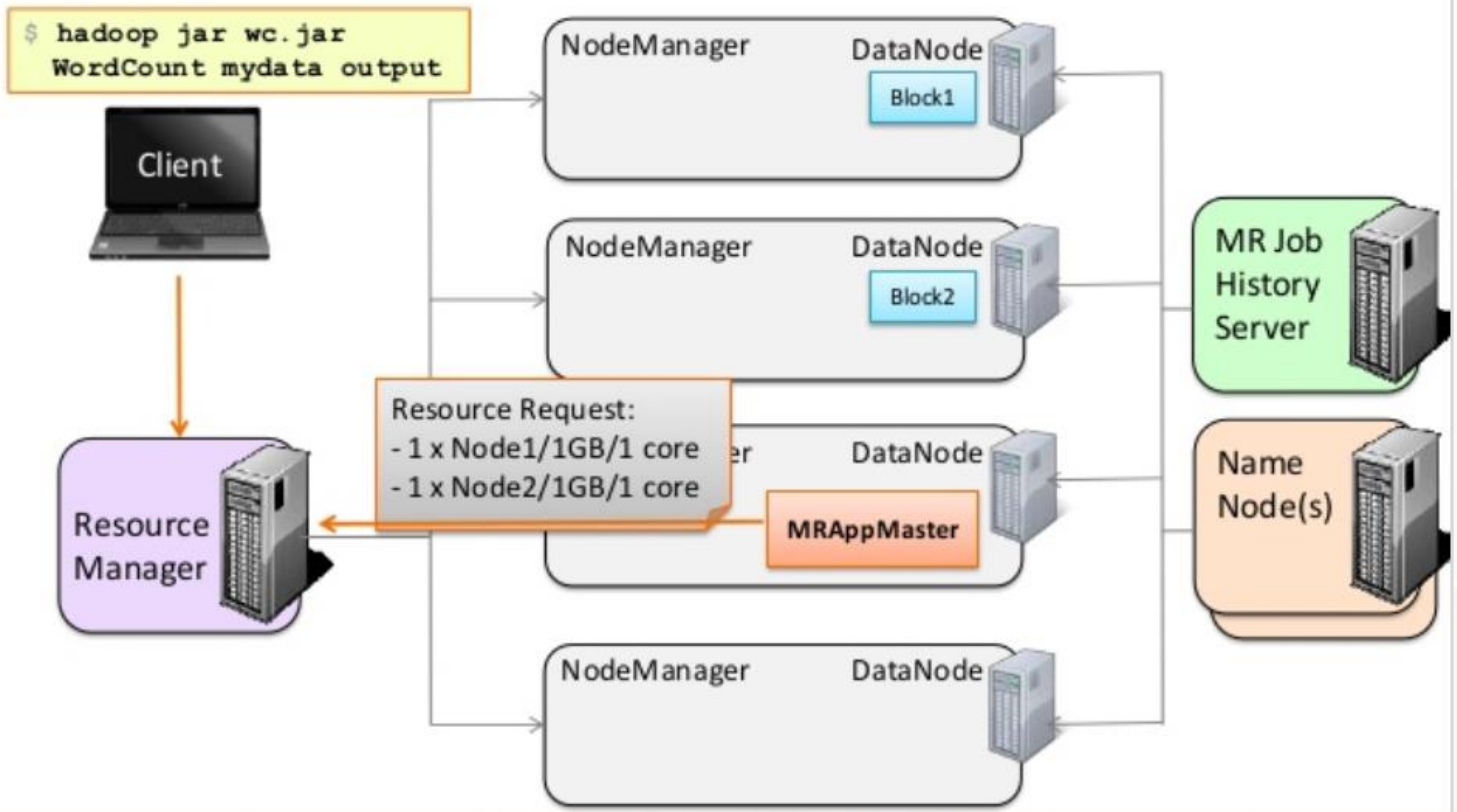


Running a MapReduce Application in MRv2



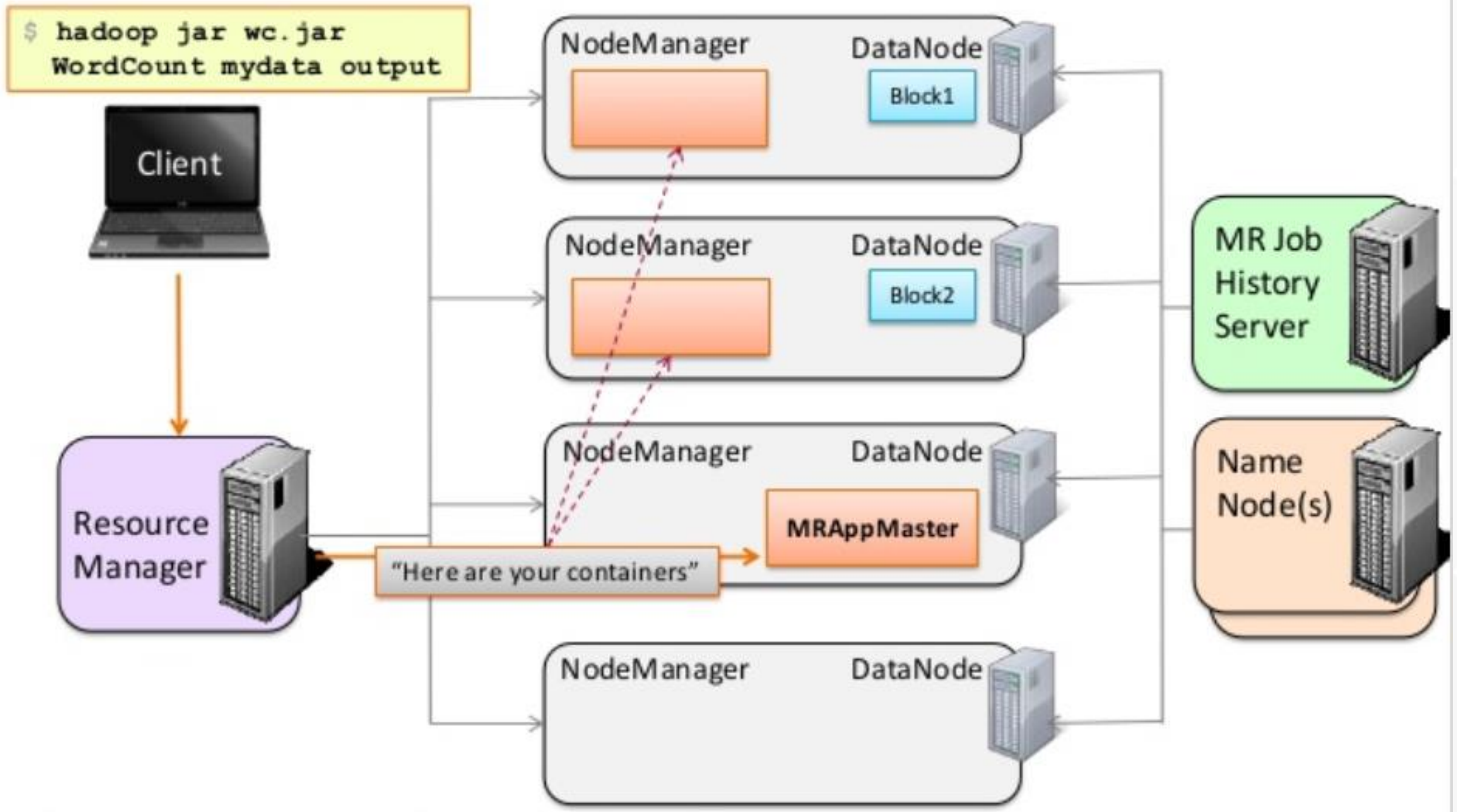
```
$ hadoop jar wc.jar
  WordCount mydata output
```

Running a MapReduce Application in MRv2

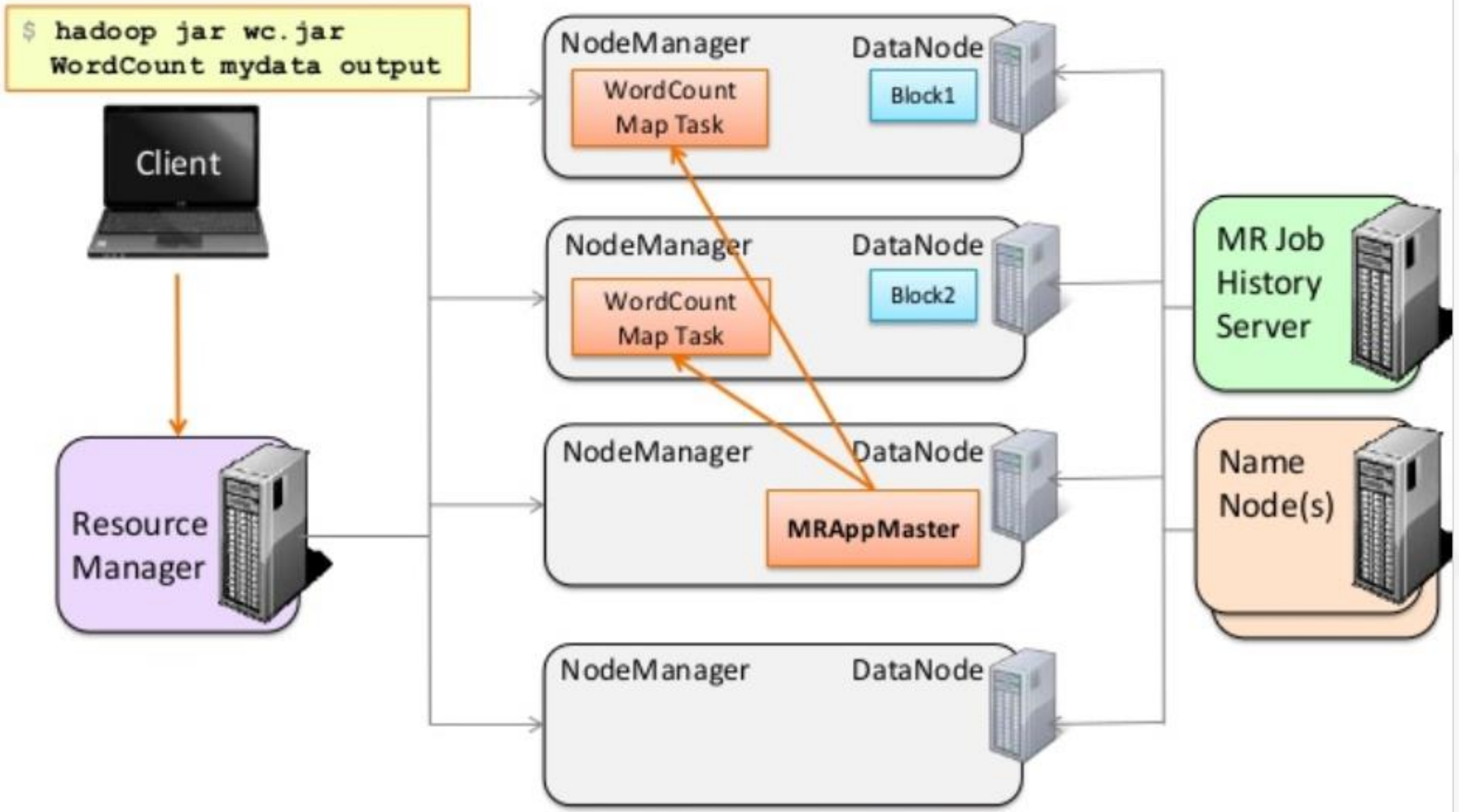


```
$ hadoop jar wc.jar
  WordCount mydata output
```

Running a MapReduce Application in MRv2

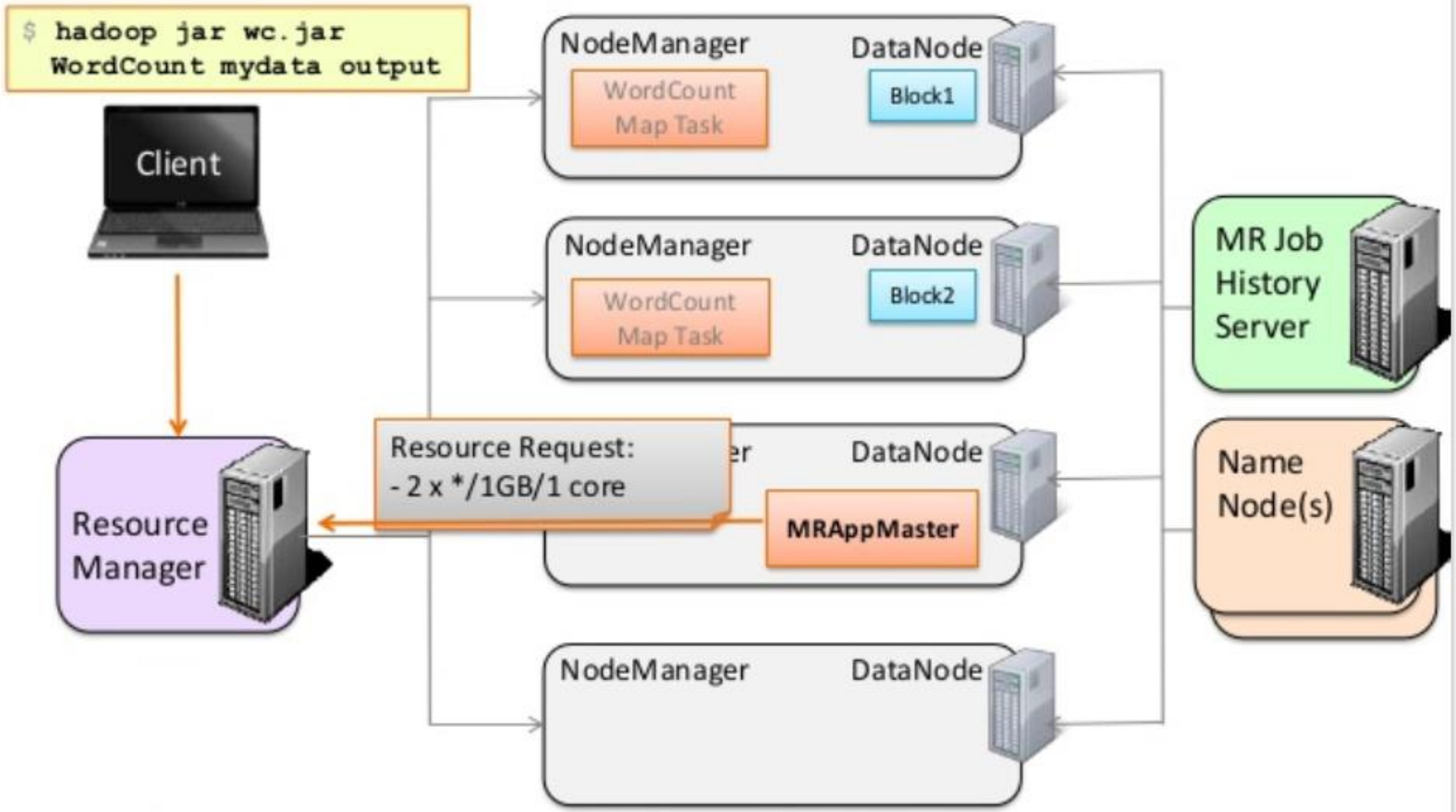

```
$ hadoop jar wc.jar
  WordCount mydata output
```

Running a MapReduce Application in MRv2

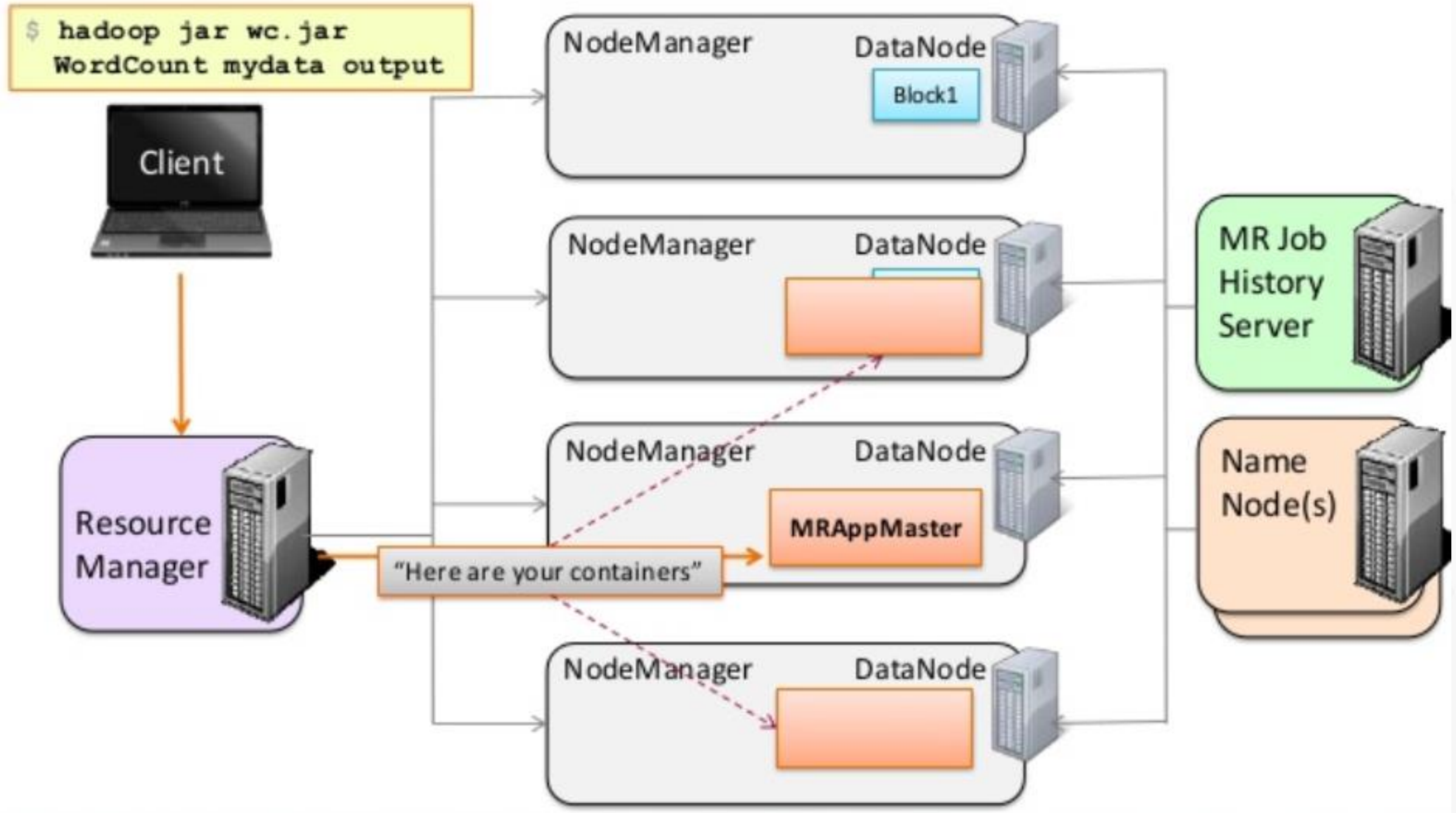


```
$ hadoop jar wc.jar
  WordCount mydata output
```

Running a MapReduce Application in MRv2

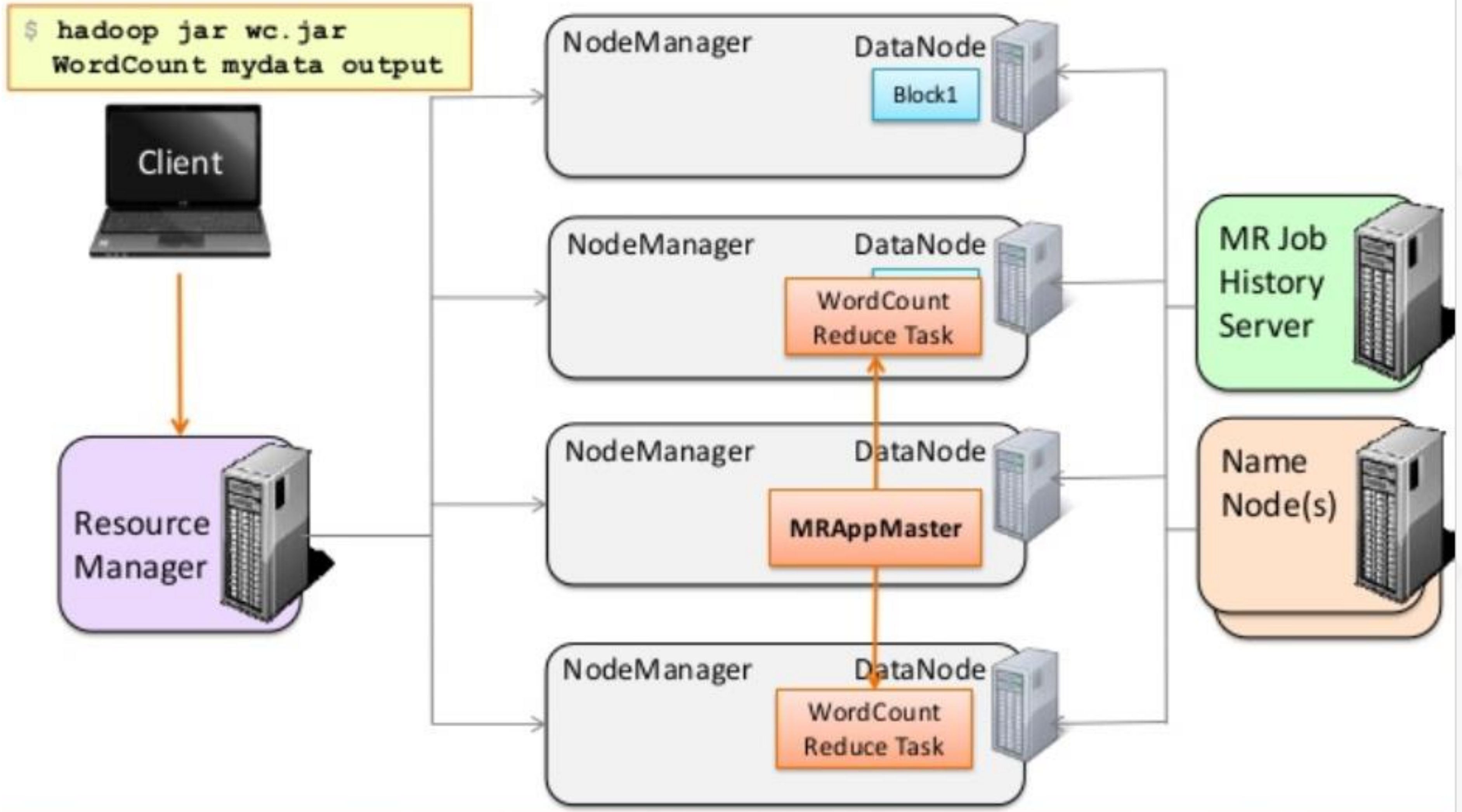

```
$ hadoop jar wc.jar
  WordCount mydata output
```

Running a MapReduce Application in MRv2



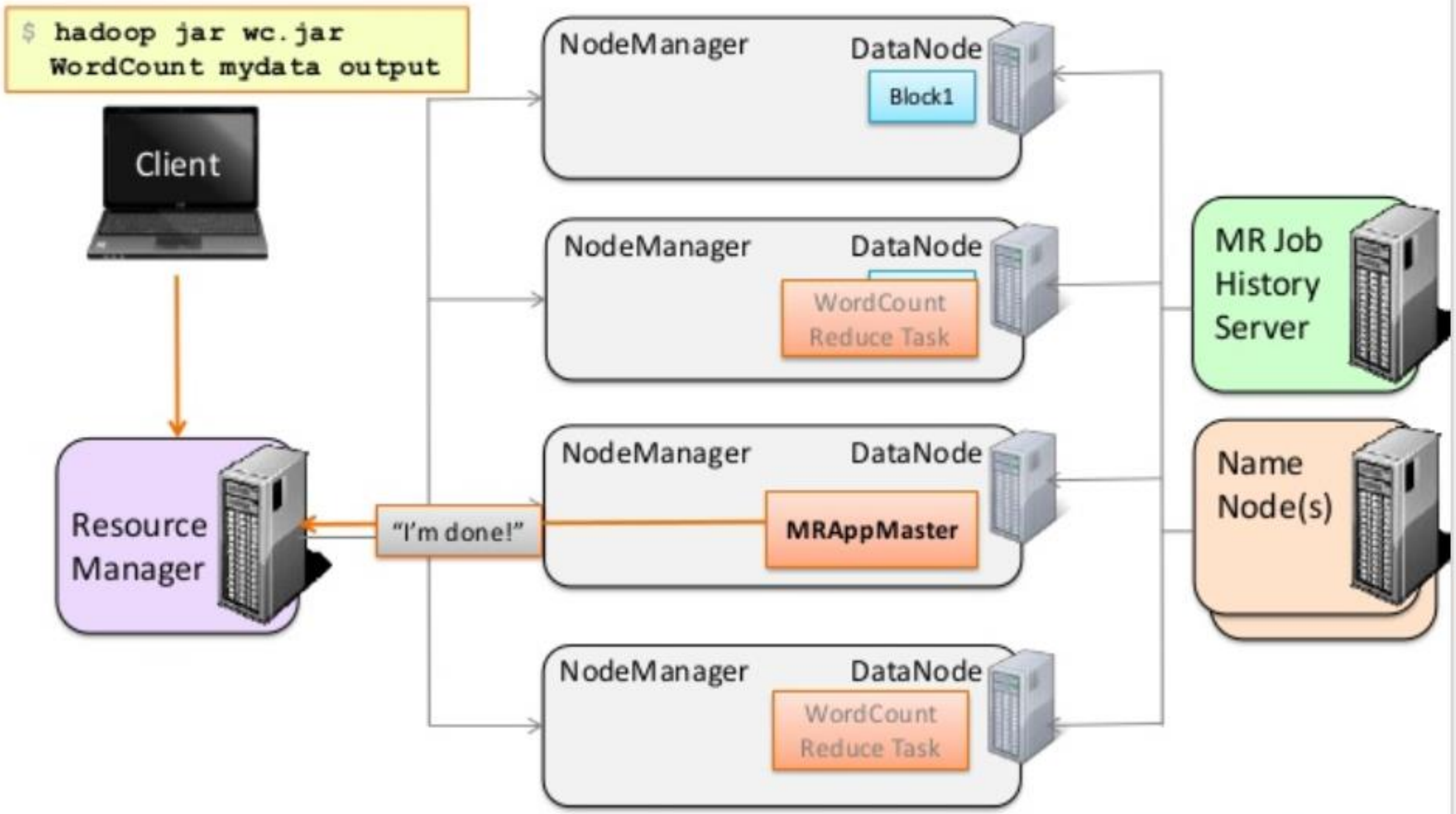
```
$ hadoop jar wc.jar
  WordCount mydata output
```

Running a MapReduce Application in MRv2



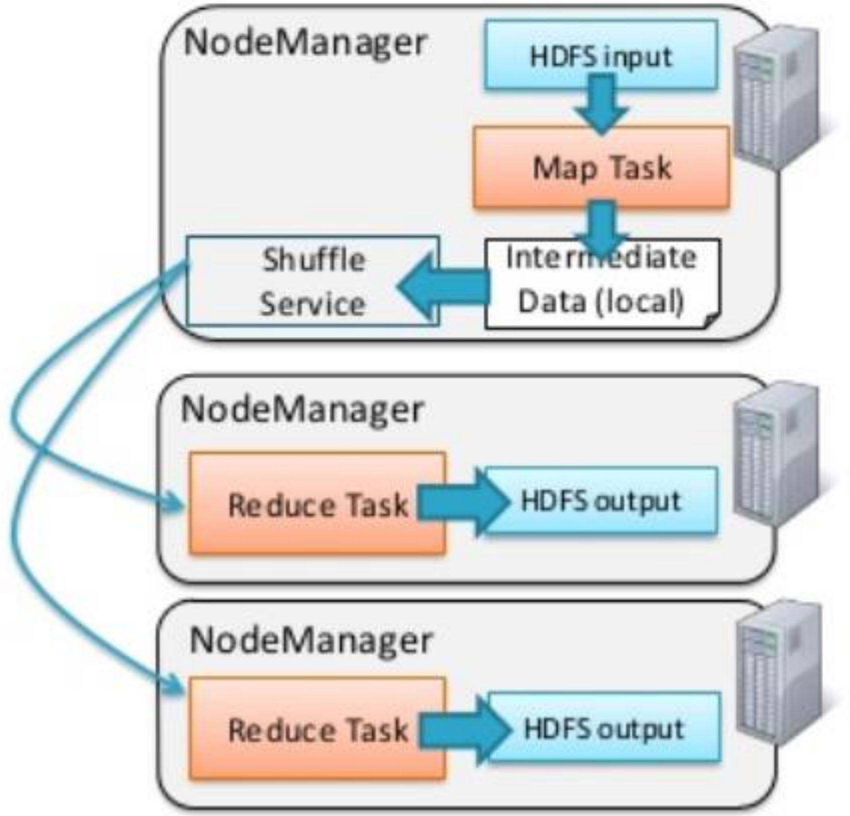
```
$ hadoop jar wc.jar
  WordCount mydata output
```

Running a MapReduce Application in MRv2


```
$ hadoop jar wc.jar
  WordCount mydata output
```

The MapReduce Framework on YARN

- **In YARN, Shuffle is run as an auxiliary service**
  - Runs in the NodeManager JVM as a persistent service



Fault Tolerance

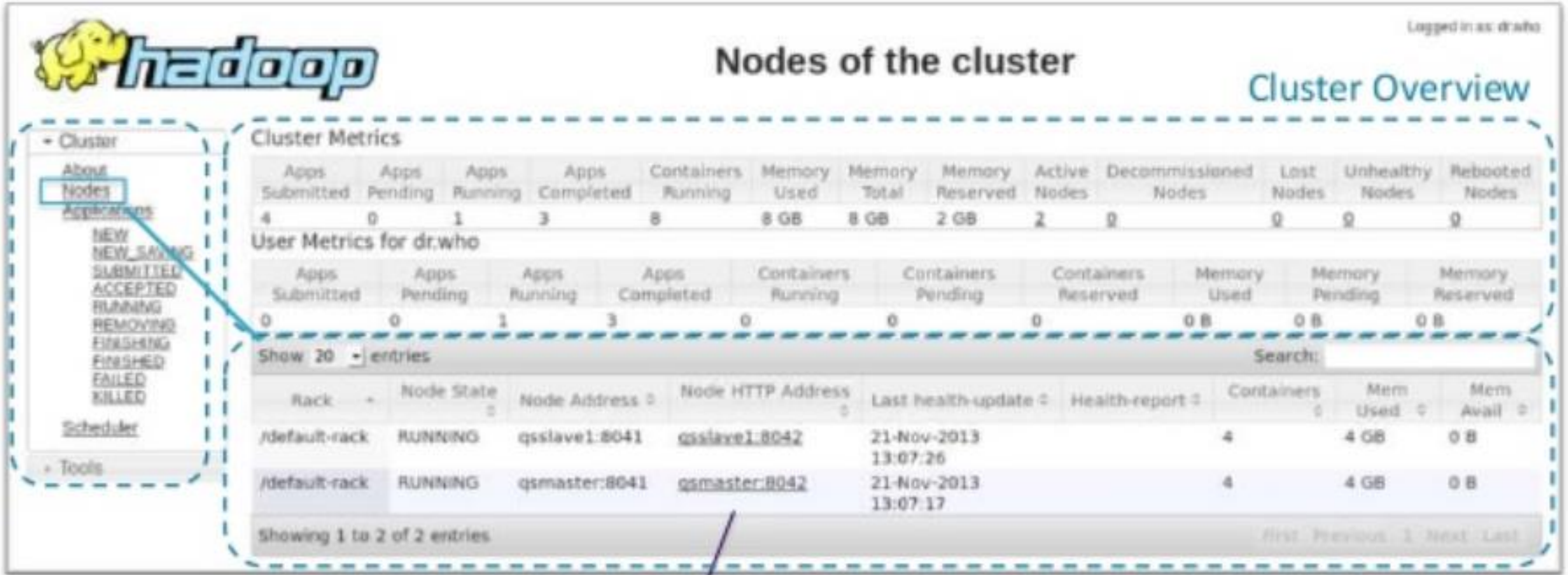
- **Any of the following can fail**
  - Task (Container) – Handled just like in MRv1
    - MRAppMaster will re-attempt tasks that complete with exceptions or stop responding (4 times by default)
    - Applications with too many failed tasks are considered failed

Fault Tolerance

- **Any of the following can fail**
  - Task (Container) – Handled just like in MRv1
    - MRAppMaster will re-attempt tasks that complete with exceptions or stop responding (4 times by default)
    - Applications with too many failed tasks are considered failed
  - Application Master
    - If application fails or if AM stops sending heartbeats, RM will re-attempt the whole application (2 times by default)
    - MRAppMaster optional setting: Job recovery
      - if false, all tasks will re-run
      - if true, MRAppMaster retrieves state of tasks when it restarts; only incomplete tasks will be re-run

Resource Manager UI: Nodes

`http://rmhost:8088/cluster/nodes`



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

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

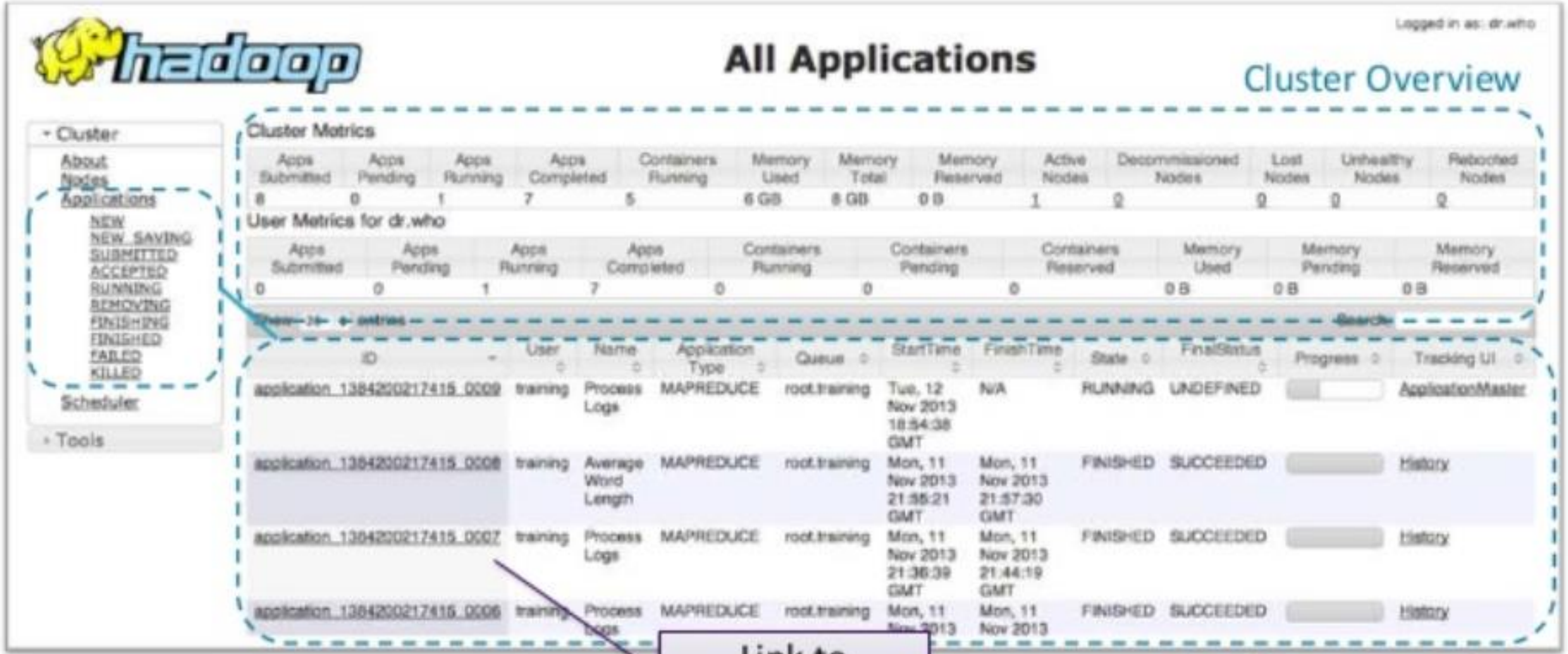
List of each node in cluster

link to Node Manager UI

ANALYTIXLABS

Resource Manager UI: Applications

`http://rmhost:8088/cluster/apps`



All Applications

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

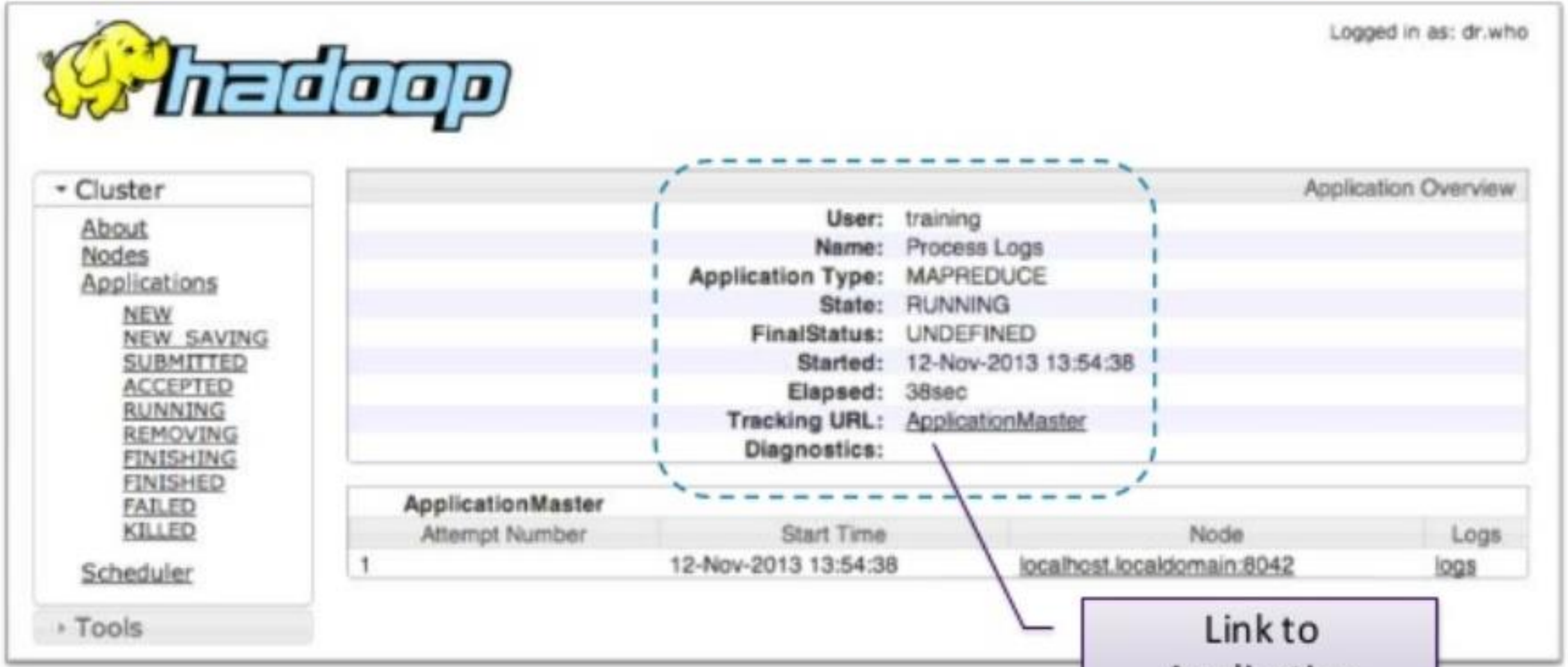
User Metrics for dr.who

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

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1384200217415_0009	training	Process Logs	MAPREDUCE	root.training	Tue, 12 Nov 2013 18:54:38 GMT	N/A	RUNNING	UNDEFINED		ApplicationMaster
application_1384200217415_0008	training	Average Word Length	MAPREDUCE	root.training	Mon, 11 Nov 2013 21:56:21 GMT	Mon, 11 Nov 2013 21:57:30 GMT	FINISHED	SUCCEEDED		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		History
application_1384200217415_0006	training	Process Logs	MAPREDUCE	root.training	Mon, 11 Nov 2013	Mon, 11 Nov 2013	FINISHED	SUCCEEDED		History

Resource Manager UI: Application Detail

`http://rmhost:8088/cluster/app/appid`



User:	training
Name:	Process Logs
Application Type:	MAPREDUCE
State:	RUNNING
FinalStatus:	UNDEFINED
Started:	12-Nov-2013 13:54:38
Elapsed:	38sec
Tracking URL:	ApplicationMaster
Diagnostics:	

ApplicationMaster

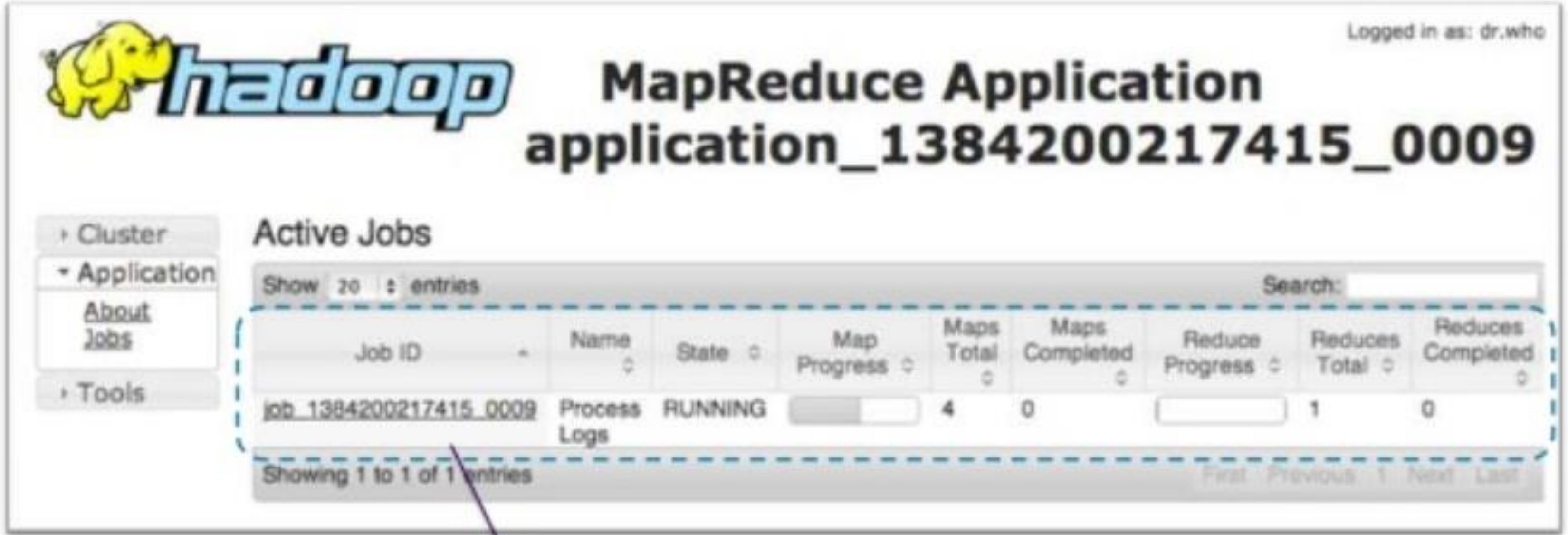
Attempt Number	Start Time	Node	Logs
1	12-Nov-2013 13:54:38	localhost.localdomain:8042	logs

Link to Application Master…

ANALYTIXLABS

MRAppMaster UI: Jobs

`http://rmhost:8088/proxy/appid`


Logged in as: dr.who

MapReduce Application application_1384200217415_0009

Active Jobs

Show 20 entries Search:

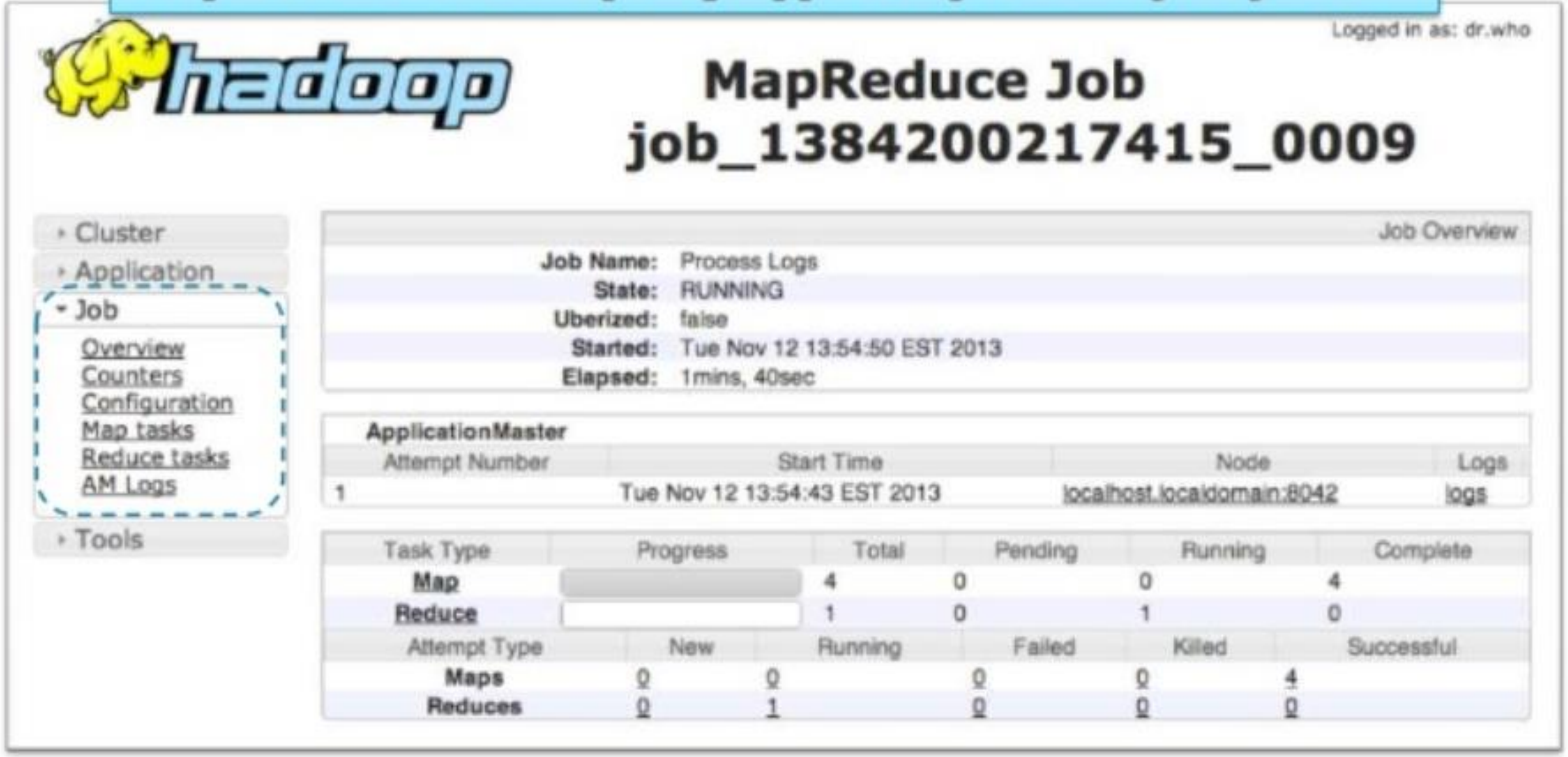
Job ID	Name	State	Map Progress	Maps Total	Maps Completed	Reduce Progress	Reduces Total	Reduces Completed
job_1384200217415_0009	Process Logs	RUNNING		4	0		1	0

Showing 1 to 1 of 1 entries

Link to Job Details...

MRAppMaster UI: Tasks

`http://rmhost:8088/proxy/appid/mapreduce/job/jobid`



MapReduce Job job_1384200217415_0009

Job Overview

Job Name:	Process Logs
State:	RUNNING
Uberized:	false
Started:	Tue Nov 12 13:54:50 EST 2013
Elapsed:	1mins, 40sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Tue Nov 12 13:54:43 EST 2013	localhost.localdomain:8042	logs

Task Type	Progress	Total	Pending	Running	Complete
Map		4	0	0	4
Reduce		1	0	1	0

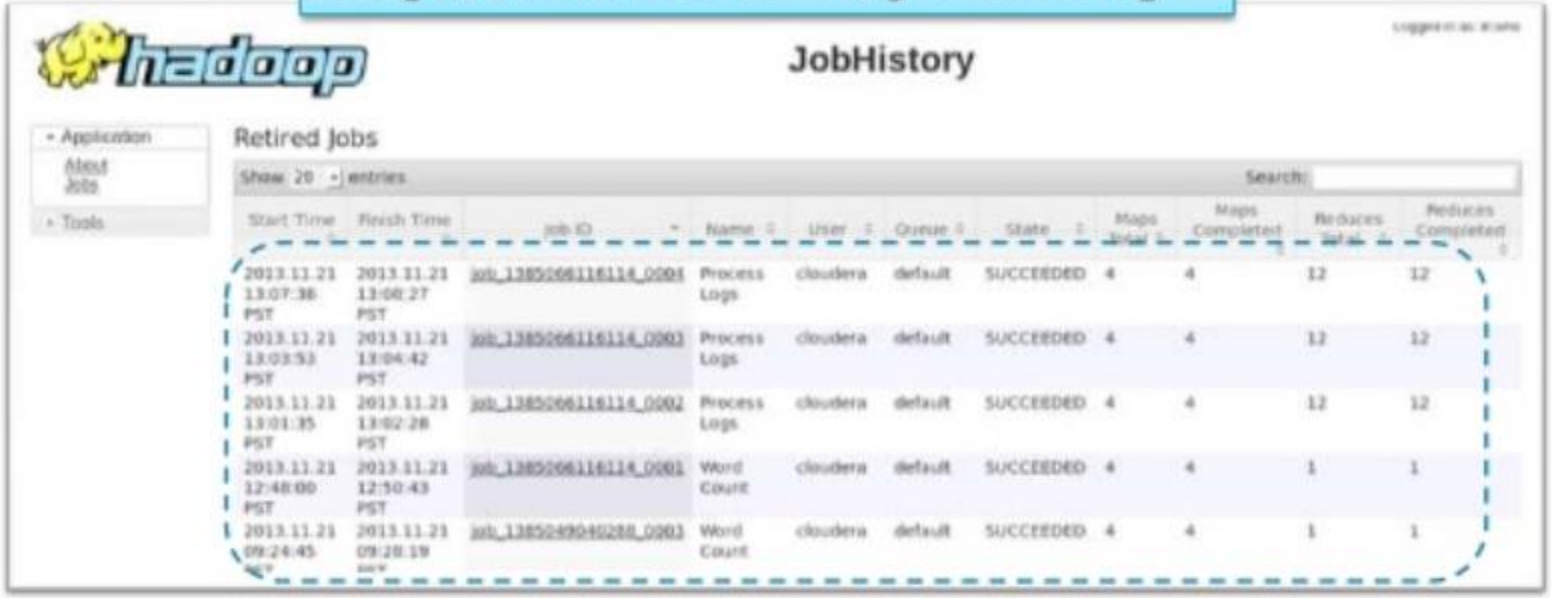
Attempt Type	New	Running	Failed	Killed	Successful
Maps	0	0	0	0	4
Reduces	0	1	0	0	0

MR Job History Server

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

MR Job History Server

`http://rmhost:19888/jobhistory`

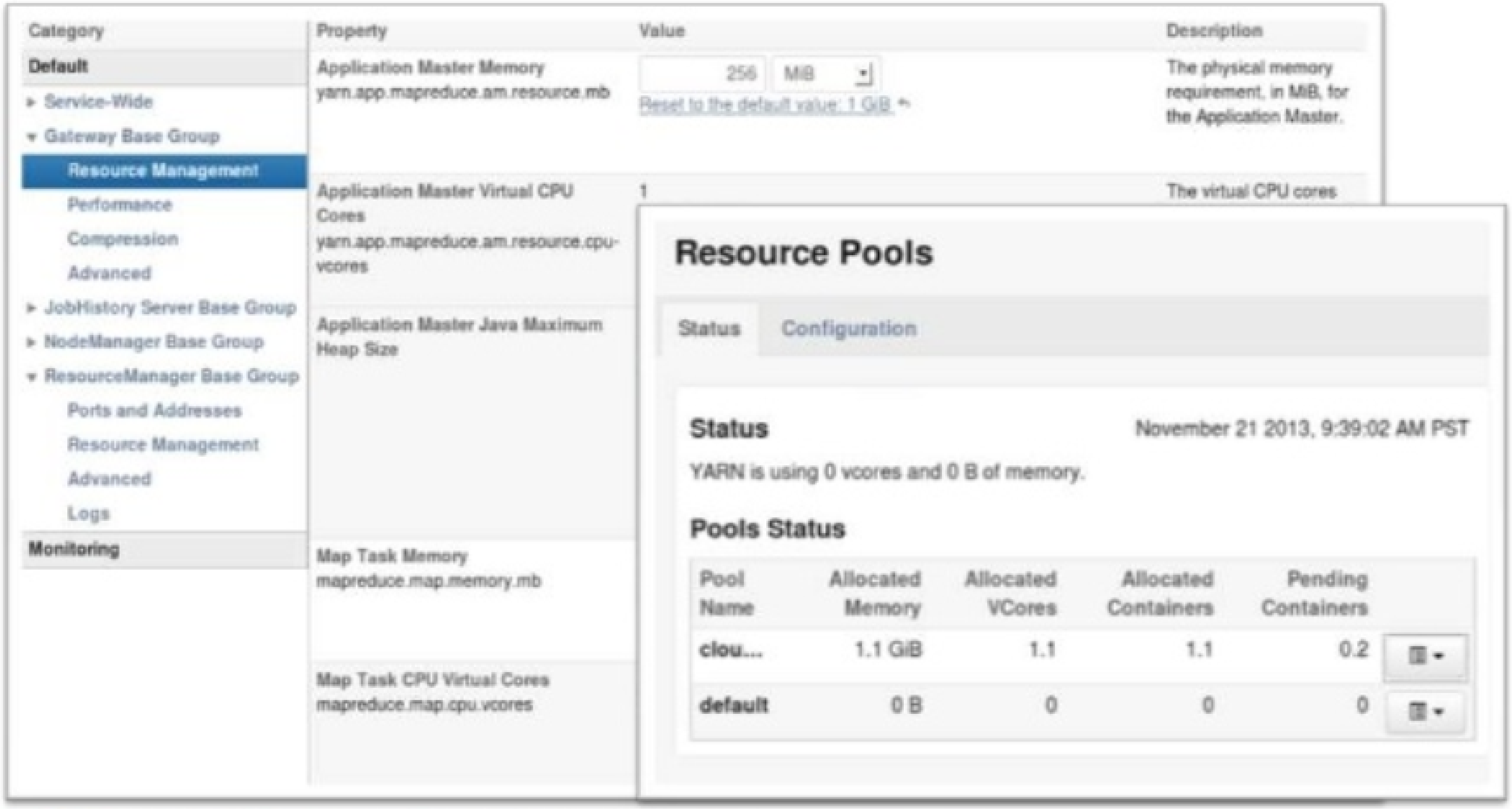


Retired Jobs

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

Cloudera Manager

- Full support for MR2 added in CM 5



Category

Default

- ▸ Service-Wide
- ▾ Gateway Base Group
  - Resource Management
  - Performance
  - Compression
  - Advanced
- ▸ JobHistory Server Base Group
- ▸ NodeManager Base Group
- ▾ ResourceManager Base Group
  - Ports and Addresses
  - Resource Management
  - Advanced
  - Logs

Monitoring

Property	Value	Description
Application Master Memory yarn.app.mapreduce.am.resource.mb	256 MiB Reset to the default value: 1 GiB	The physical memory requirement, in MiB, for the Application Master.
Application Master Virtual CPU Cores yarn.app.mapreduce.am.resource.cpu-vcores	1	The virtual CPU cores
Application Master Java Maximum Heap Size		
Map Task Memory mapreduce.map.memory.mb		
Map Task CPU Virtual Cores mapreduce.map.cpu.vcores		

Resource Pools

Status | Configuration

Status November 21 2013, 9:39:02 AM PST

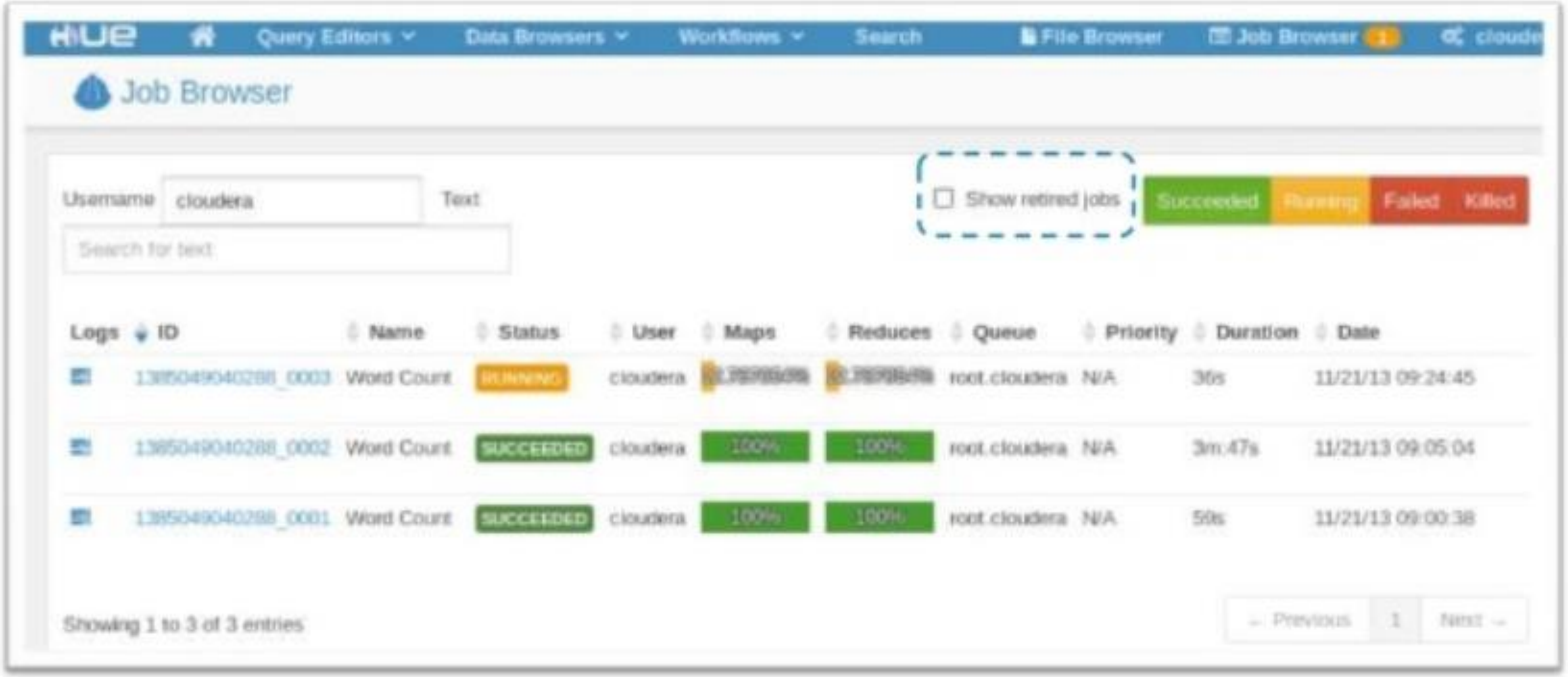
YARN is using 0 vcores and 0 B of memory.

Pools Status

Pool Name	Allocated Memory	Allocated VCores	Allocated Containers	Pending Containers
clou...	1.1 GiB	1.1	1.1	0.2
default	0 B	0	0	0

Hue

- **Hue supports browsing MRv2 jobs**
  - Connects to Job History Server for “retired” jobs

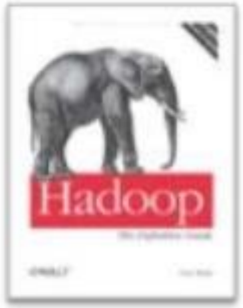


Logs	ID	Name	Status	User	Maps	Reduces	Queue	Priority	Duration	Date
▤	1385049040288_0003	Word Count	RUNNING	cloudera	[illegible]	[illegible]	root.cloudera	N/A	36s	11/21/13 09:24:45
▤	1385049040288_0002	Word Count	SUCCEEDED	cloudera	100%	100%	root.cloudera	N/A	3m:47s	11/21/13 09:05:04
▤	1385049040288_0001	Word Count	SUCCEEDED	cloudera	100%	100%	root.cloudera	N/A	59s	11/21/13 09:00:38

Showing 1 to 3 of 3 entries

Where To Learn More

- ***Hadoop: The Definitive Guide,* 3rd Edition**
  - Chapter 6 – focuses on how MR is implemented on YARN



- **Cloudera Blog posts –** blog.cloudera.com/blog/category/yarn
  - Migrating to MapReduce 2 on YARN
  - Writing Hadoop Programs That Work Across Releases
  - and more…

Contact Us

Visit us on: http://www.analytixlabs.in/

For more information, please contact us: http://www.analytixlabs.co.in/contact-us/

Or email: info@analytixlabs.co.in

Call us we would love to speak with you: (+91) 9910509849

Join us on:

Twitter - http://twitter.com/#!/AnalytixLabs

Facebook - http://www.facebook.com/analytixlabs

LinkedIn - http://www.linkedin.com/in/analytixlabs

Blog - http://www.analytixlabs.co.in/category/blog/