
Hadoop Architecture

Prof. Hyuk-Yoon Kwon

<https://sites.google.com/view/seoultech-bigdata>

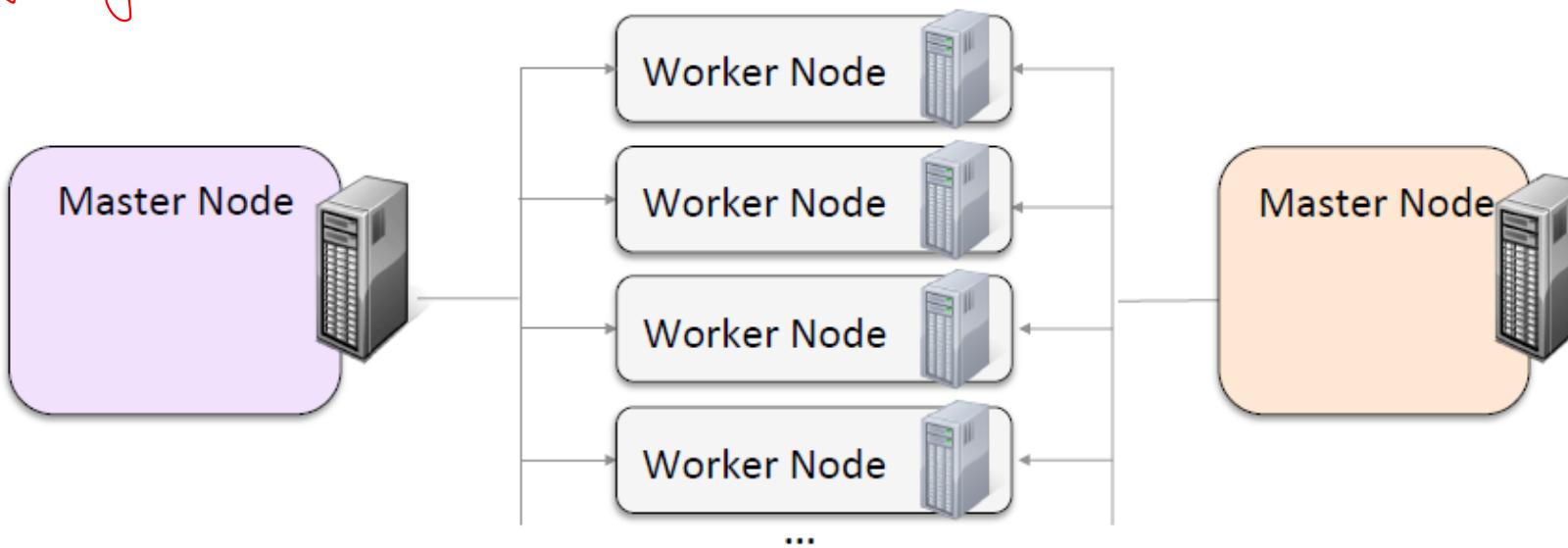
Hadoop Architecture and HDFS

In this chapter you will learn

- How Hadoop Distributed File System stores data across a cluster
- How to use HDFS using the Hue File Browser or the `hdfs` command
- How Hadoop YARN provides cluster resource management for distributed data processing *nodes, data, jobs*
- How to use Hue, the YARN Web UI or the `yarn` command to monitor your cluster

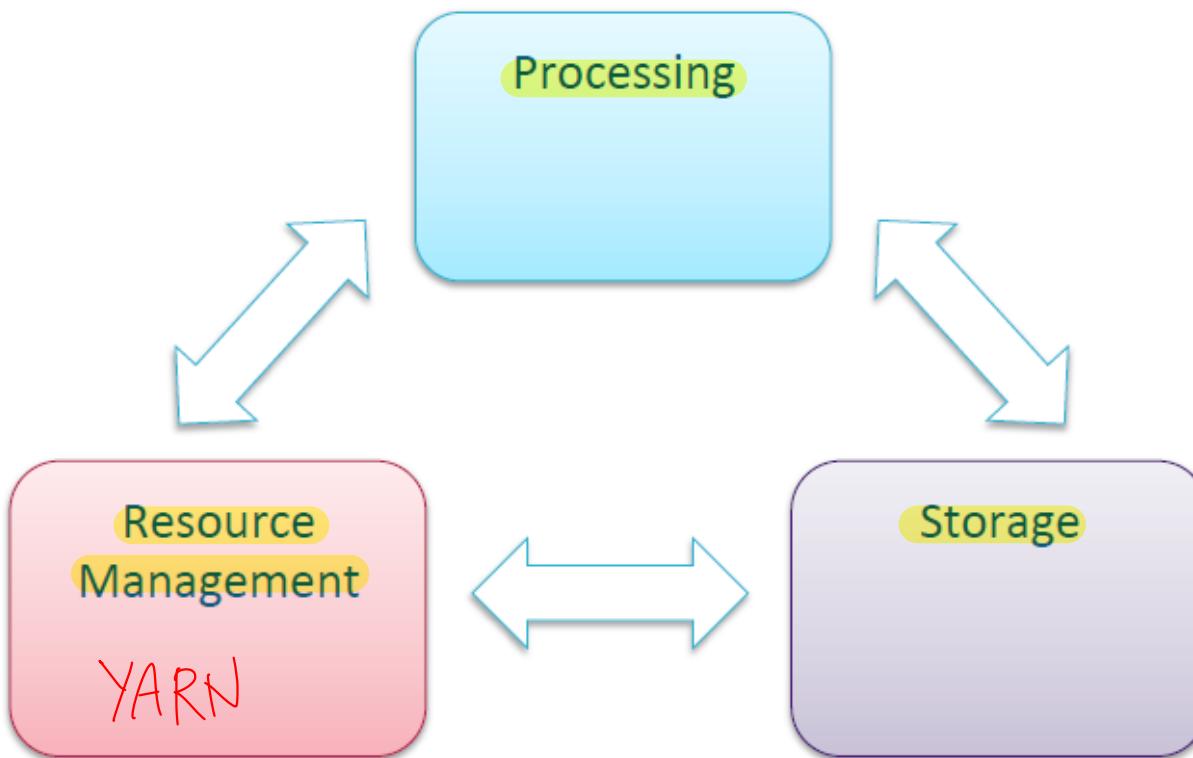
Hadoop Cluster Terminology

- A **cluster** is a group of computers working together
 - Provides data storage, data processing, and resource management
 - A **node** is an individual computer in the cluster
 - Master nodes manage distribution of work and data to worker nodes
 - A **daemon** is a program running on a node
 - Each Hadoop daemon performs a specific function in the cluster
- always running*



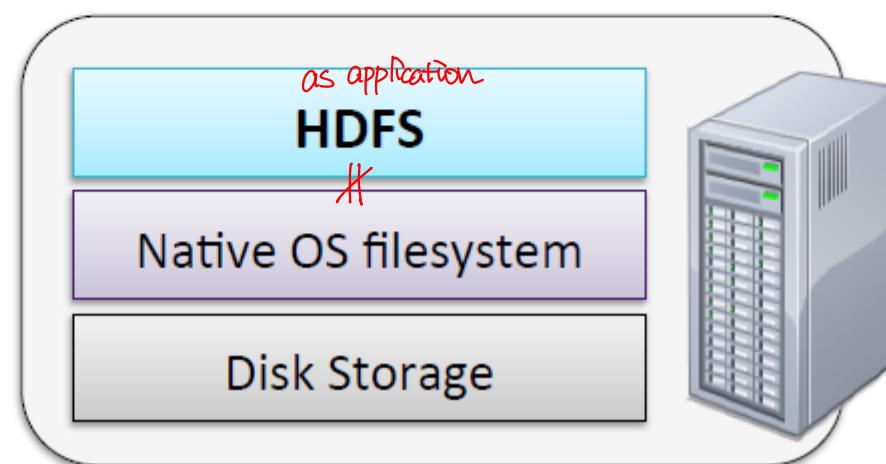
Cluster Components

- Three main components of a cluster
- Work together to provide distributed data processing
- We will start with the Storage component
 - HDFS



HDFS Basic Concepts (1) in user Space (not kernel)

- HDFS is a filesystem written in Java
 - Based on Google's GFS
- Sits on top of a native filesystem
 - Such as ext3, ext4, or xfs
- Provides redundant storage for massive amounts of data
 - Using readily-available, industry-standard computers



HDFS → { logical : file system
physical : not actual file system

HDFS Basic Concepts (2)

- HDFS performs **best with a 'modest' number of large files**
 - Millions, rather than billions, of files
 - Each file typically 100MB or more
- Files in HDFS are '**write once**' *modifying file = write new file*
 - No random writes to files are allowed
- HDFS is **optimized for large, streaming reads of files** *merge smaller files to large files for performance*
 - Rather than random reads

How Files Are Stored

minimizing the overhead to find the starting point of the file

huge size

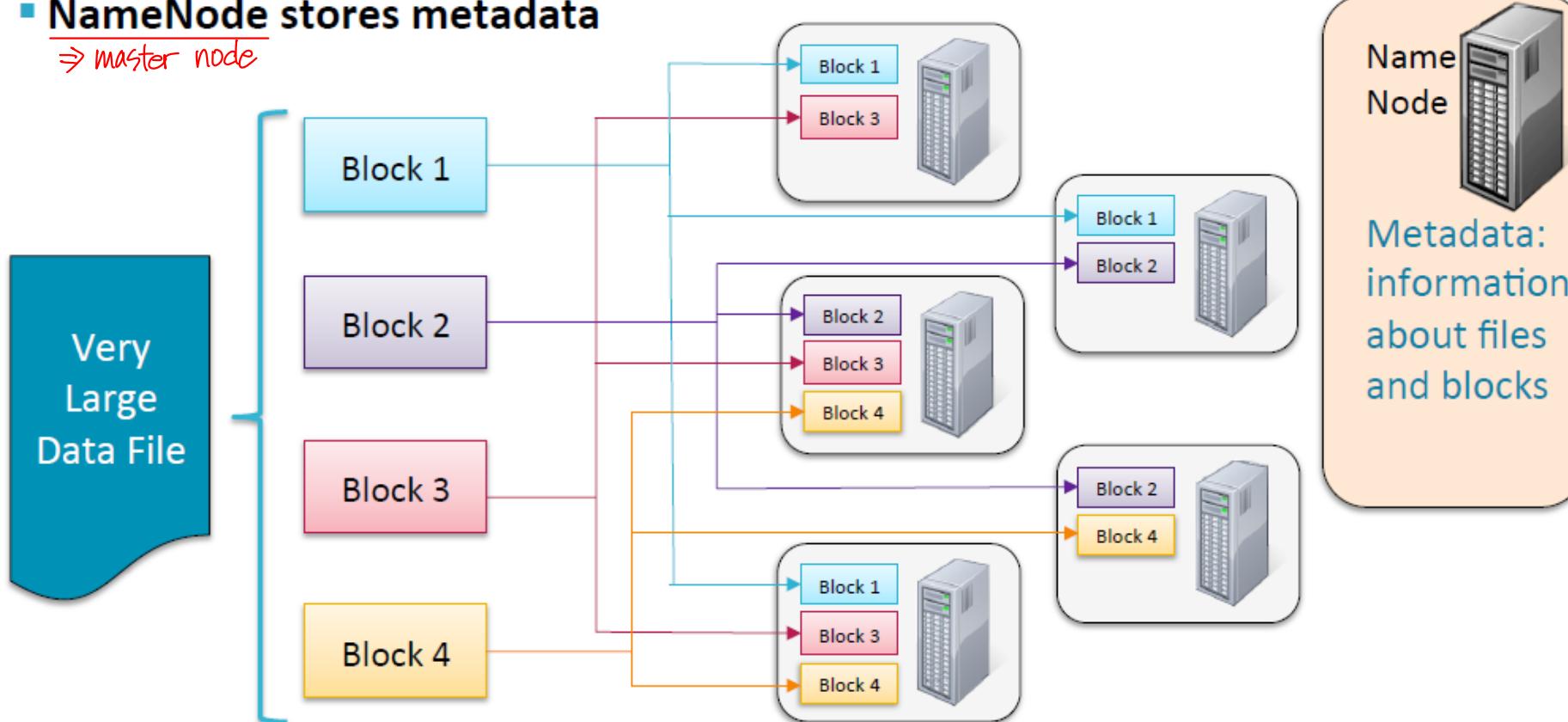
- Data files are split into 128MB blocks which are distributed at load time
default

- Each block is replicated on multiple data nodes (default 3x)

increase reliability of service
provide efficient service to user
maximize availability

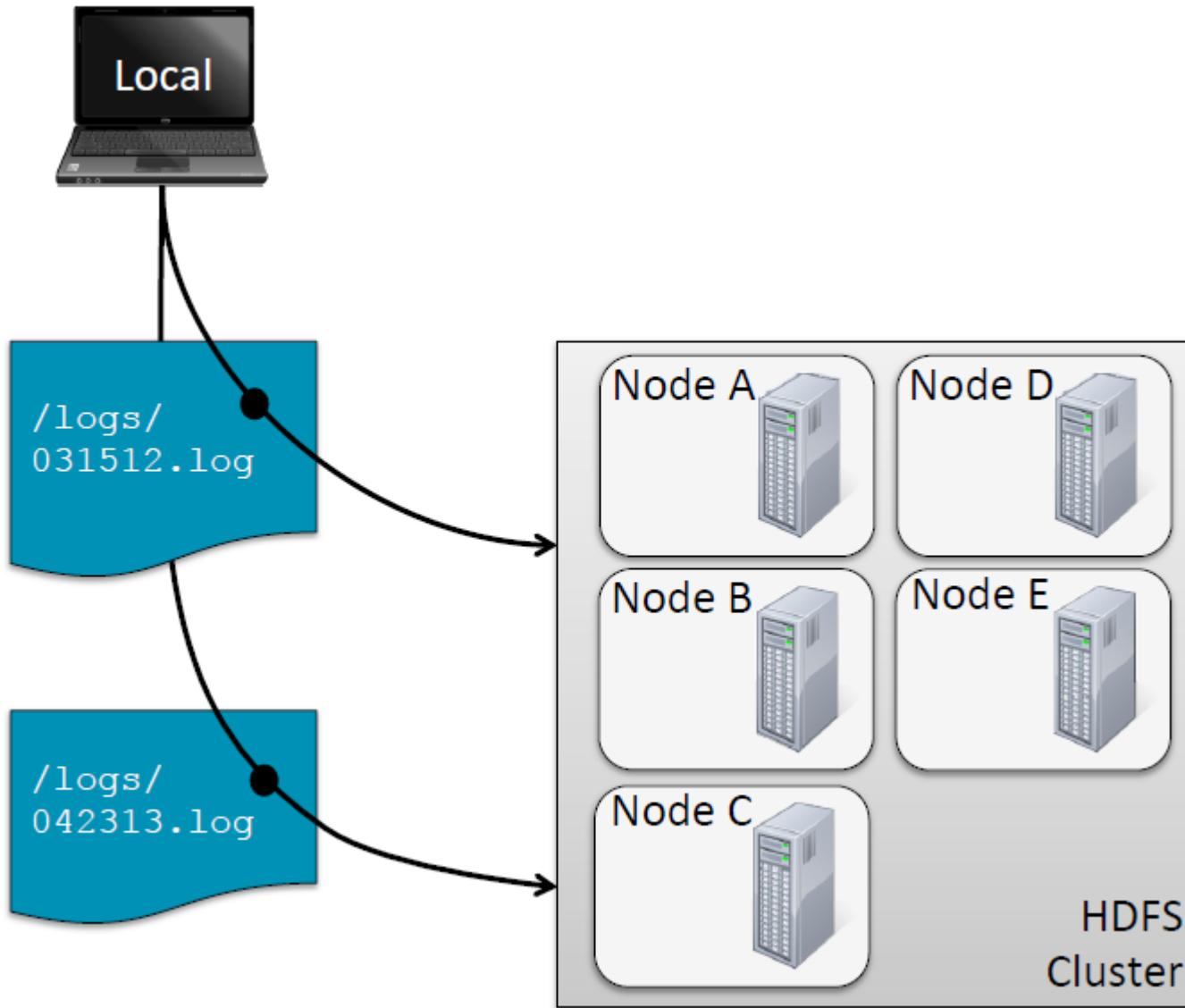
- NameNode stores metadata

→ master node

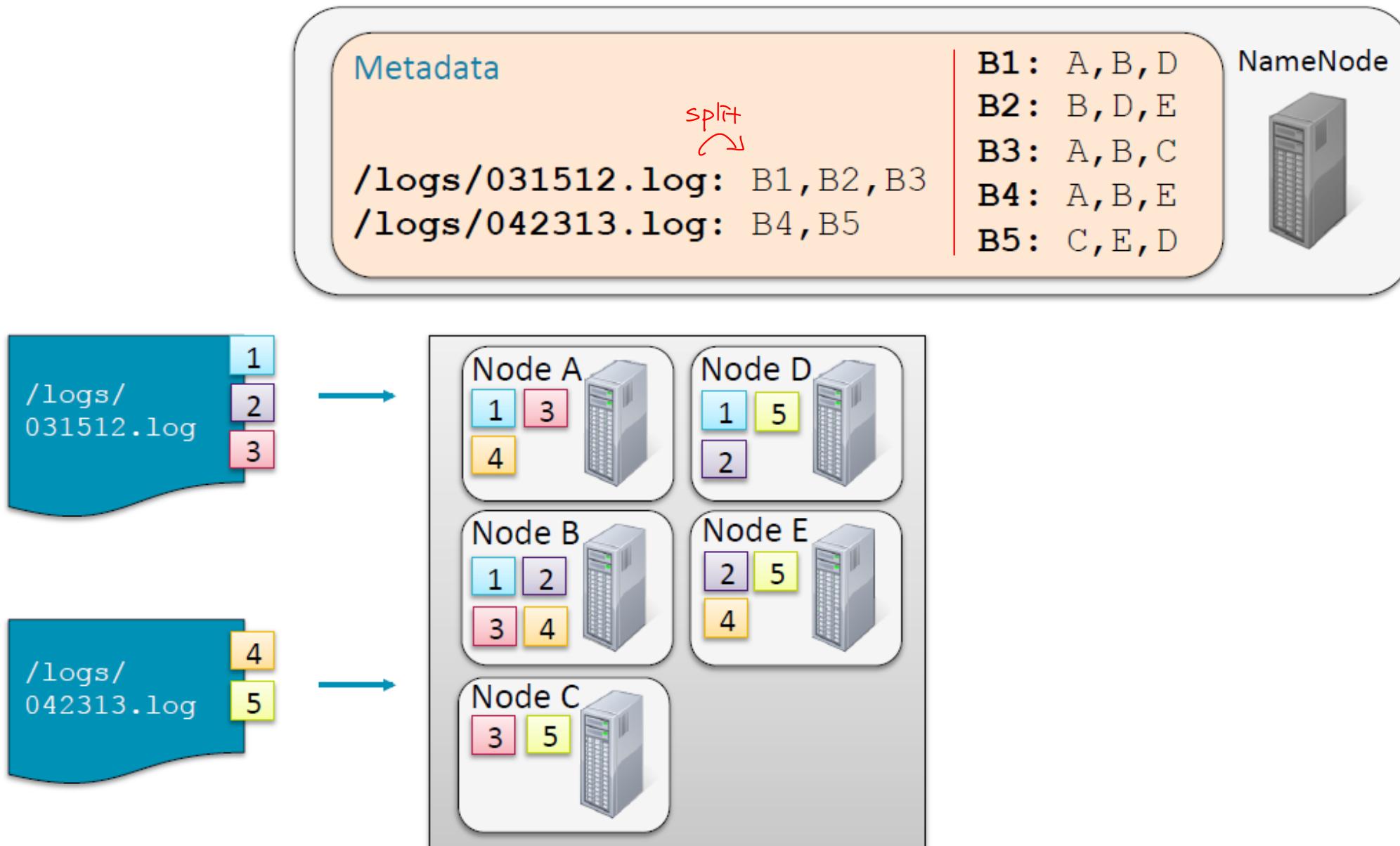


where the blocks are located in which nodes.
how split the large file into several blocks.

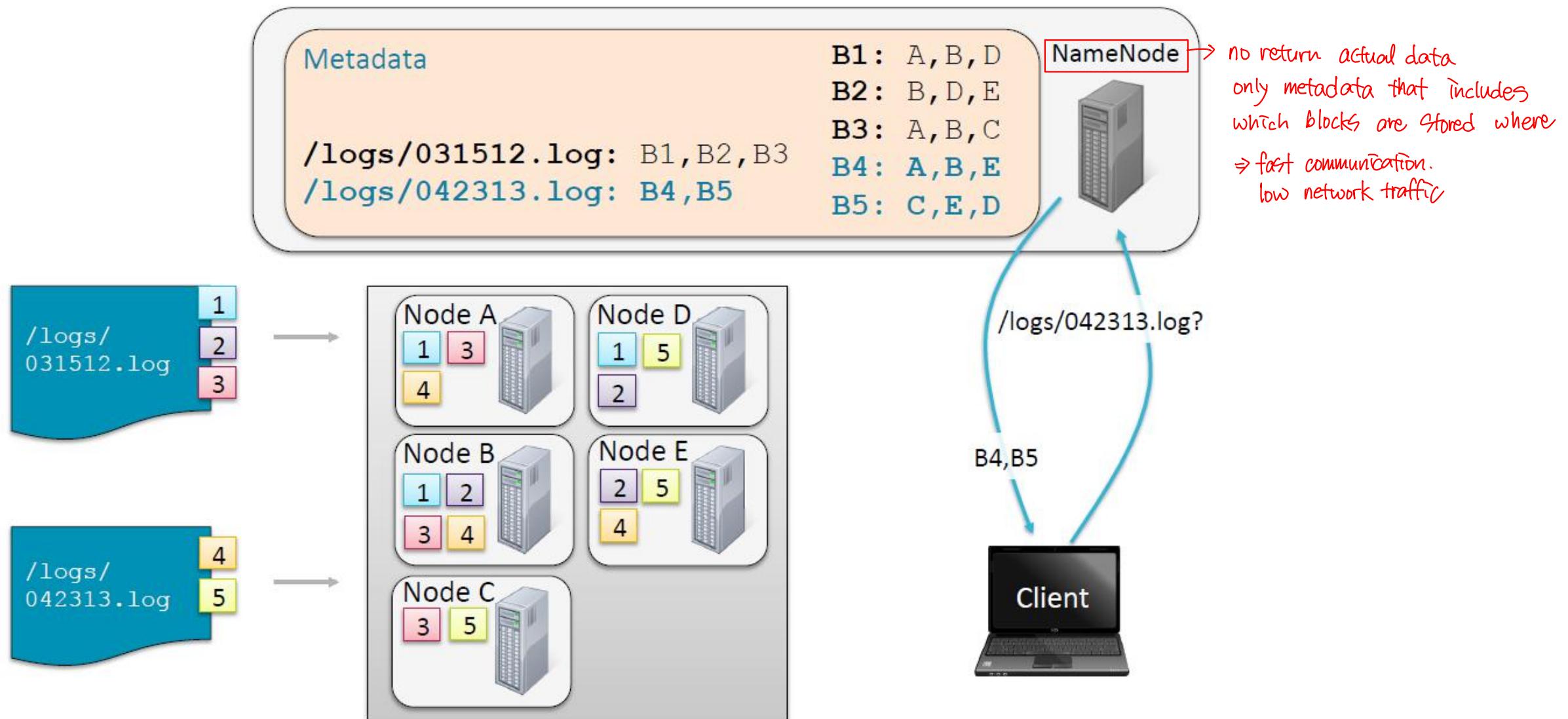
Example: Storing and Retrieving Files (1)



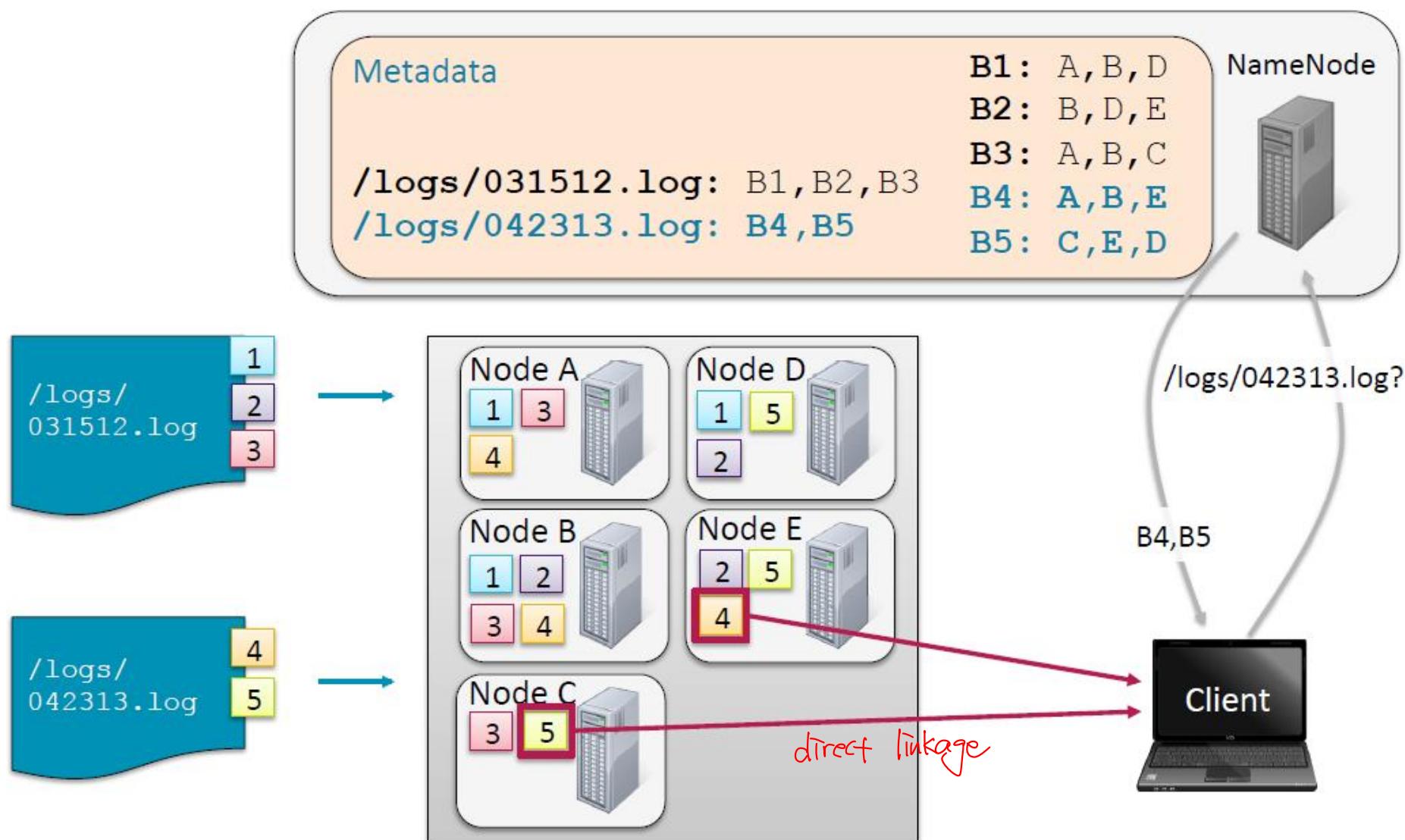
Example: Storing and Retrieving Files (2)



Example: Storing and Retrieving Files (3)



Example: Storing and Retrieving Files (4)



HDFS NameNode Availability

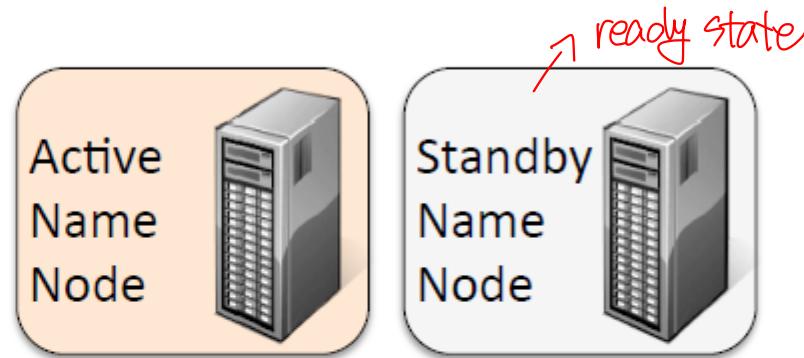
- The NameNode daemon must be running at all times
 - If the NameNode stops, the cluster becomes inaccessible

Name Node → managing all metadata for all datasets

- HDFS is typically set up for High Availability ... cost ↑

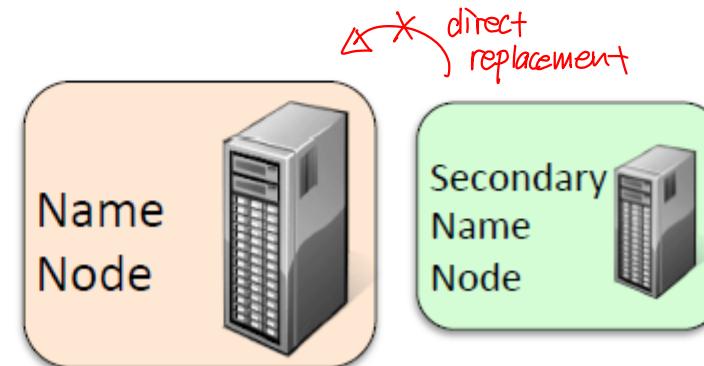
- Two NameNodes: Active and Standby

hot swap



- Small clusters may use 'Classic mode'

- One NameNode
- One "helper" node called the Secondary NameNode
 - Bookkeeping, not backup



Options for Accessing HDFS

- From the command line

- FsShell:

- \$ hdfs dfs

- In Spark

- By URI, e.g.

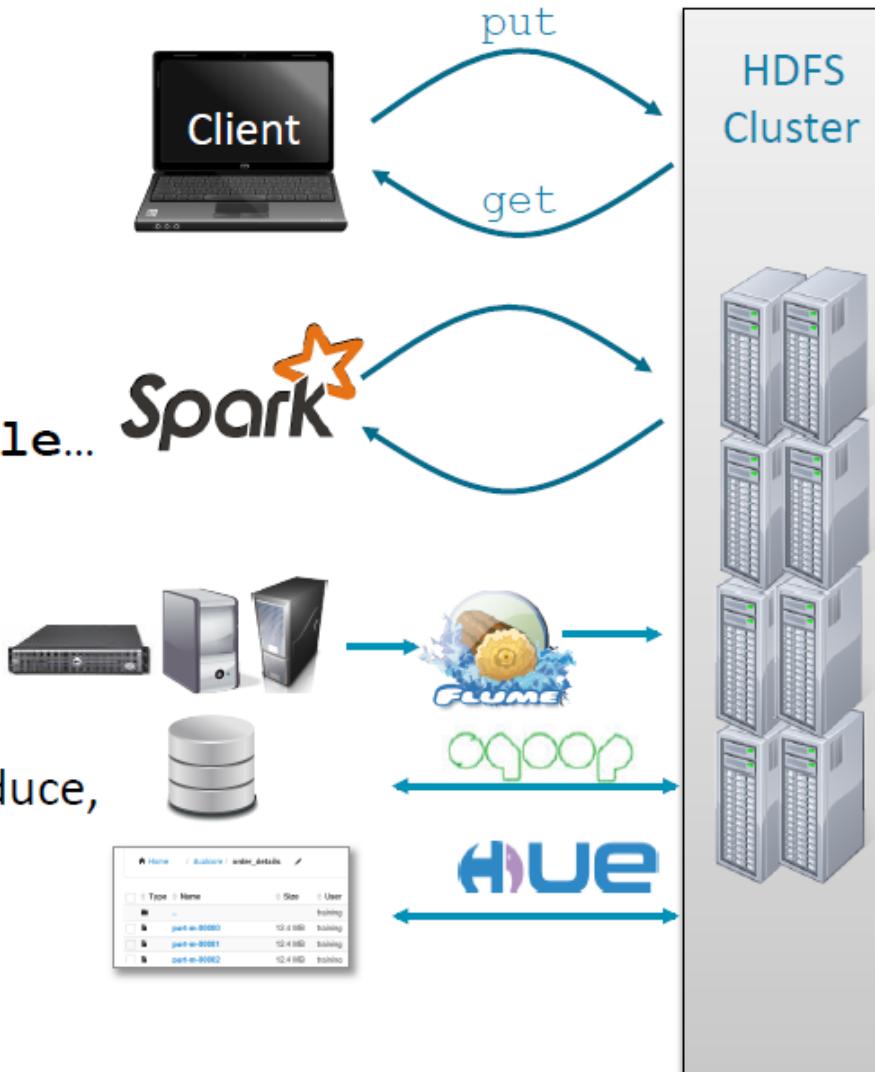
- hdfs://nnhost:port/file...

- Other programs

- Java API

- Used by Hadoop MapReduce,
Impala, Hue, Sqoop,
Flume, etc.

- RESTful interface



HDFS Command Line Examples (1)

- Copy file `foo.txt` from local disk to the user's directory in HDFS

```
$ hdfs dfs -put local foo.txt hdfs foo.txt
```

– This will copy the file to `/user/username/foo.txt`

- Get a directory listing of the user's home directory in HDFS

```
$ hdfs dfs -ls
```

- Get a directory listing of the HDFS root directory

```
$ hdfs dfs -ls /
```

HDFS Command Line Examples (2)

- Display the contents of the HDFS file /user/fred/bar.txt

```
$ hdfs dfs -cat /user/fred/bar.txt
```

- Copy that file to the local disk, named as baz.txt

```
$ hdfs dfs -get /user/fred/bar.txt baz.txt
```

- Create a directory called input under the user's home directory

```
$ hdfs dfs -mkdir input
```

Note: copyFromLocal is a synonym for put; copyToLocal is a synonym for get

HDFS Command Line Examples (3)

- Delete the directory `input_old` and all its contents

```
$ hdfs dfs -rm -r input_old
```

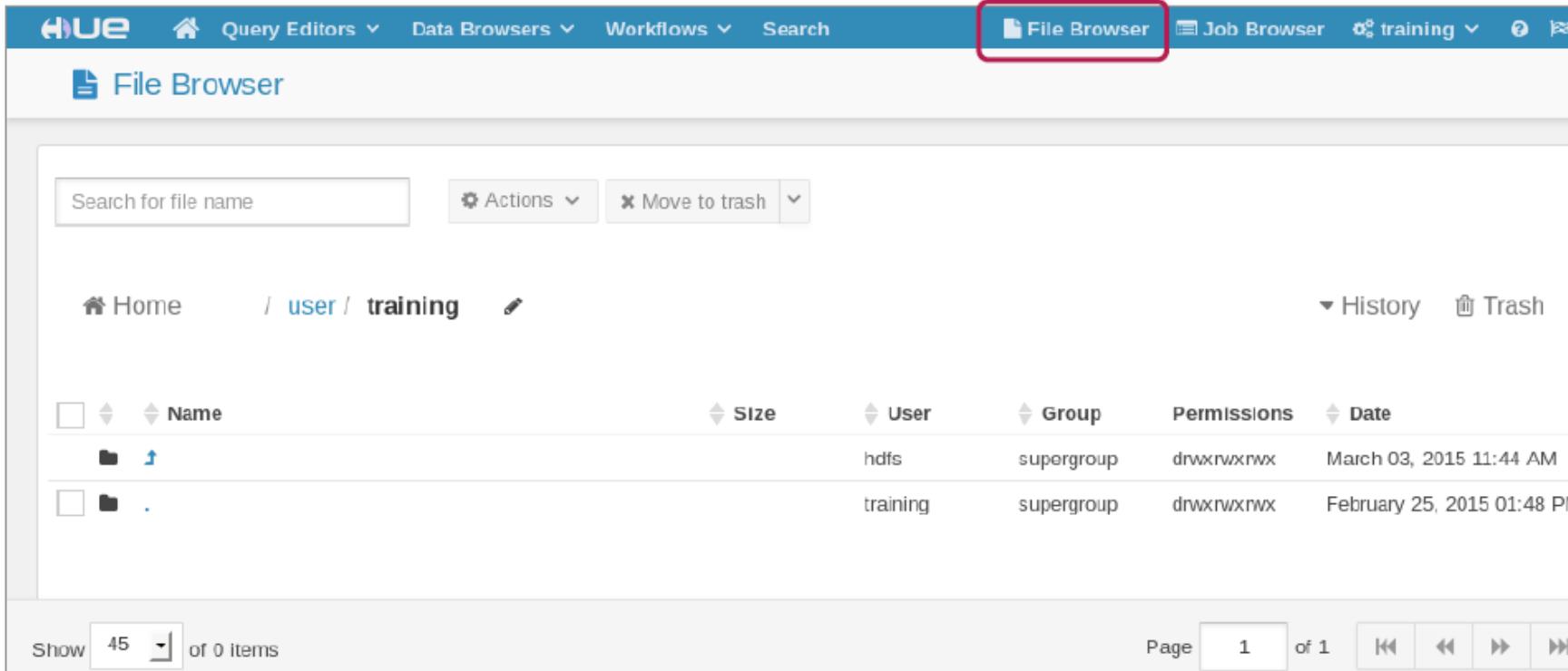
 
recursively

Practice1: Access HDFS with Command Line

- In this homework lab you will
 - Create a **/loudacre** base directory for course homework
 - Practice uploading and viewing data files
- Please refer to the Homework description

The Hue HDFS File Browser

- The File Browser in Hue lets you view and manage your HDFS directories and files
 - Create, move, rename, modify, upload, download and delete directories and files
 - View file contents



The screenshot shows the Hue web interface with the 'File Browser' tab highlighted by a red box. The top navigation bar includes links for 'Query Editors', 'Data Browsers', 'Workflows', 'Search', 'File Browser' (highlighted), 'Job Browser', and 'training'. Below the navigation is a search bar labeled 'File Browser' and a 'File Browser' tab. The main content area displays a file listing for the directory '/user/training'. The table headers are 'Name', 'Size', 'User', 'Group', 'Permissions', and 'Date'. The data shows two entries: a folder named 'user' (size 0, owner hdfs, group supergroup, permissions drwxrwxrwx, created March 03, 2015) and a folder named '.' (size 0, owner training, group supergroup, permissions drwxrwxrwx, created February 25, 2015). Navigation controls at the bottom allow for page selection (Page 1 of 1) and navigation arrows.

Name	Size	User	Group	Permissions	Date
user	0	hdfs	supergroup	drwxrwxrwx	March 03, 2015 11:44 AM
.	0	training	supergroup	drwxrwxrwx	February 25, 2015 01:48 PM

HDFS Recommendations

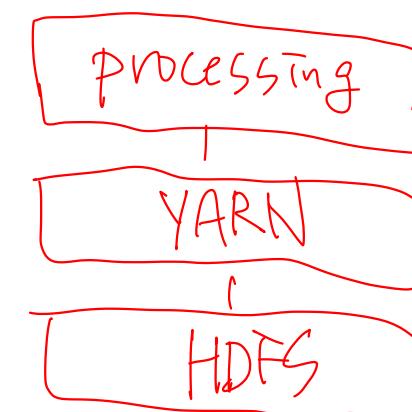
- HDFS is a repository for all your data
 - Structure and organize carefully!
- Best practices include
 - Define a standard directory structure
 - Include separate locations for staging data
- Example organization
 - `/user/`... – data and configuration belonging only to a single user
 - `/etl` – Work in progress in Extract/Transform/Load stage
 - `/tmp` – Temporary generated data shared between users
 - `/data` – Data sets that are processed and available across the organization for analysis
 - `/app` – Non-data files such as configuration, JAR files, SQL files, etc.

Practice2: Access HDFS with Hue

- In this homework lab you will
 - Create a **/loudacre** base directory for course homework
 - Practice uploading and viewing data files
- Please refer to the Homework description

What is YARN?

- YARN = Yet Another Resource Negotiator
- YARN is the Hadoop processing layer that contains
 - A resource manager
 - A job scheduler
- YARN allows multiple data processing engines to run on a single Hadoop cluster
 - Batch programs (e.g. Spark, MapReduce)
 - Interactive SQL (e.g. Impala)
 - Advanced analytics (e.g. Spark, Impala)
 - Streaming (e.g. Spark Streaming)



YARN Daemons

responsibility to maintain all resources. in distributed env.

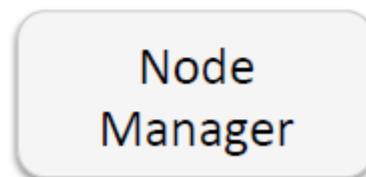
■ Resource Manager (RM)

- Runs on master node
- Global resource scheduler
- Arbitrates system resources between competing applications \Rightarrow optimize overall cluster utilization
considering node status
- Has a pluggable scheduler to support different algorithms (capacity, fair scheduler, etc.)



■ Node Manager (NM)

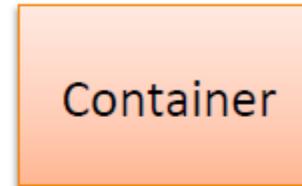
- Runs on slave nodes
- Communicates with RM



Running an Application in YARN

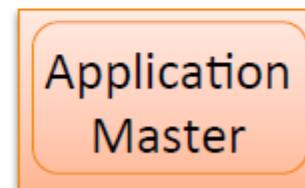
- **Containers** execute App in Hadoop \Rightarrow creating container (RM)

- 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

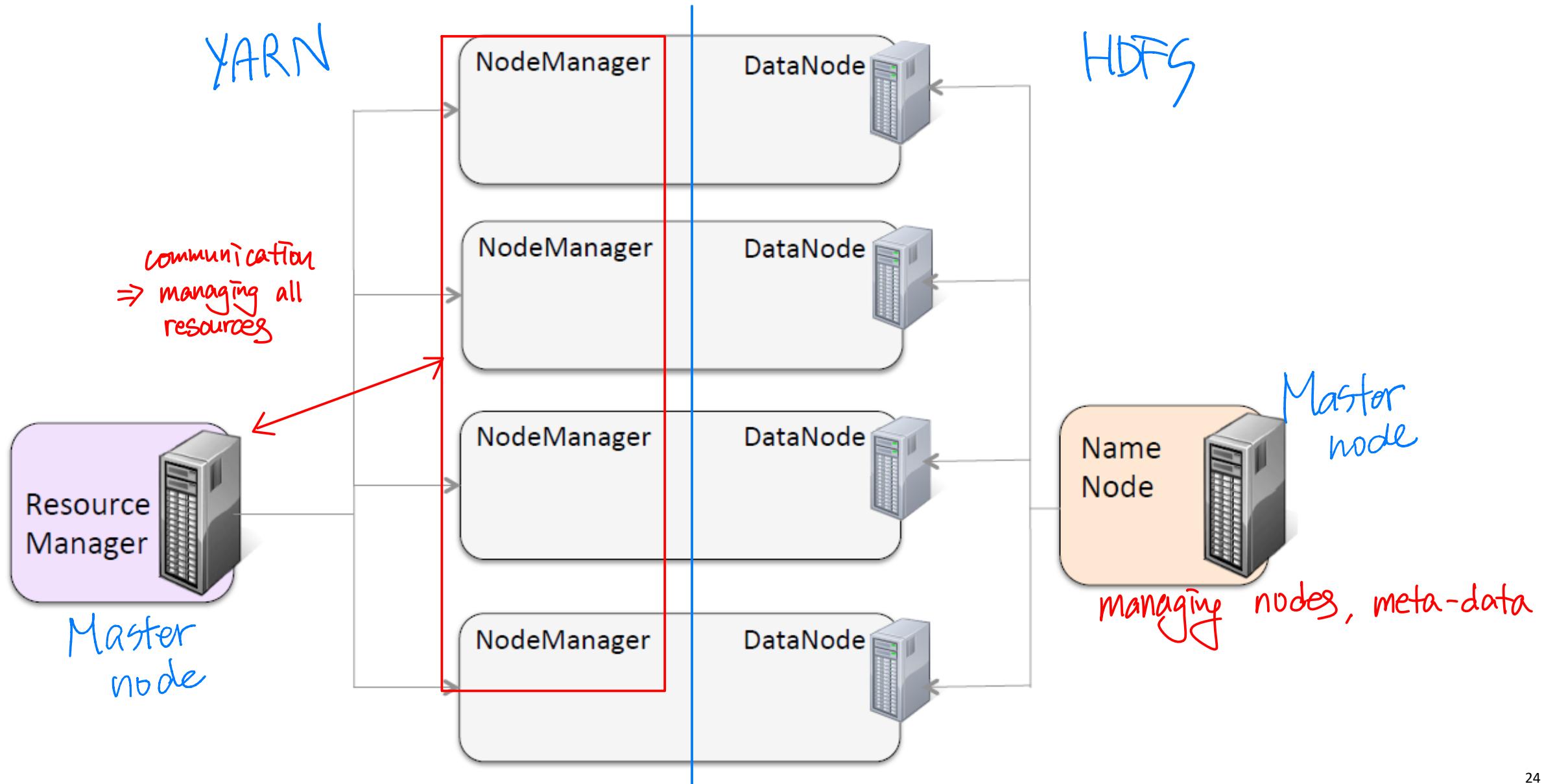


- **Application Master (AM)** managing overall app process (Start ~ end)

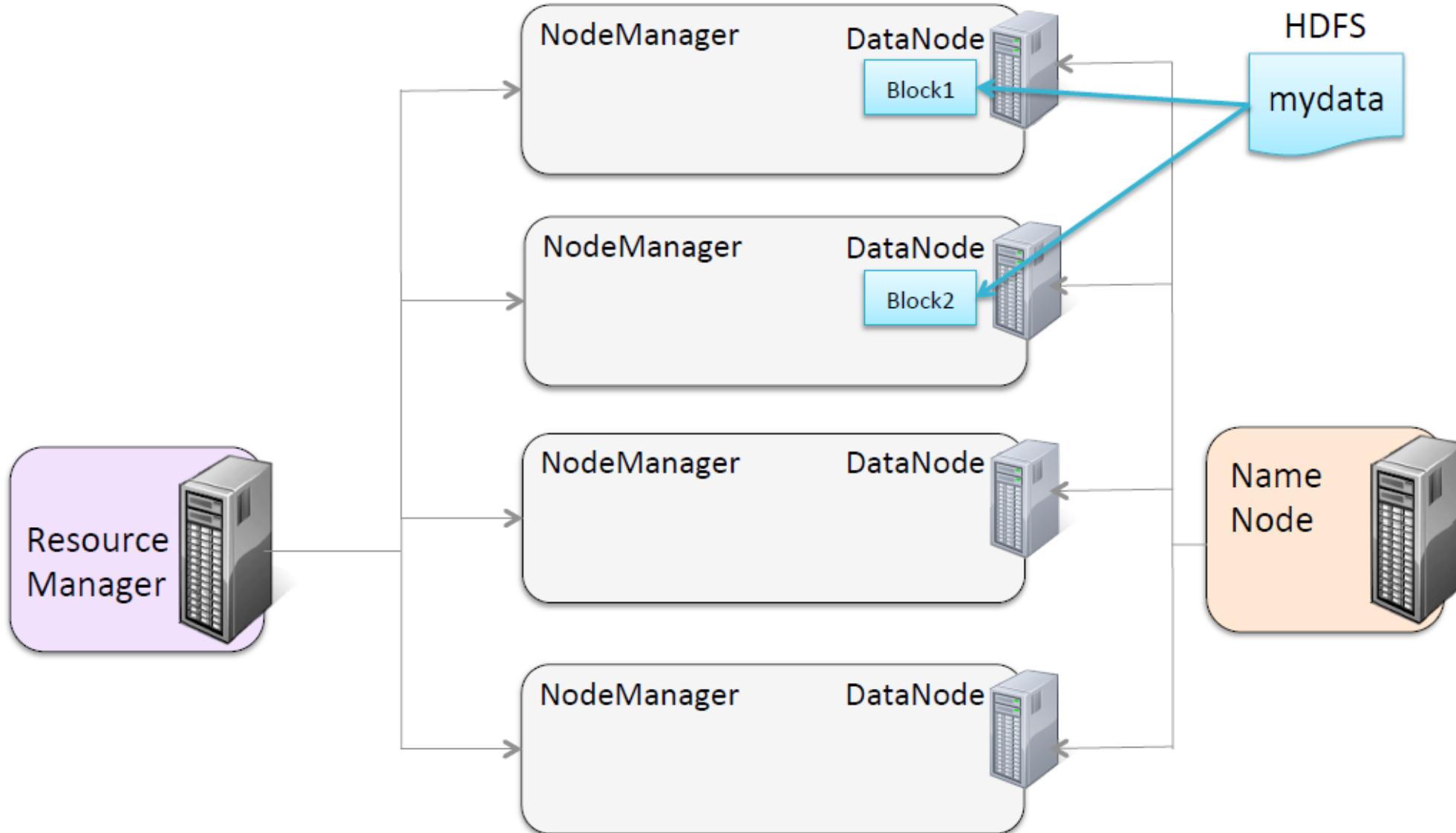
- One per application (for each)
- Framework/application specific
- Runs in a container
- Requests more containers to run application tasks



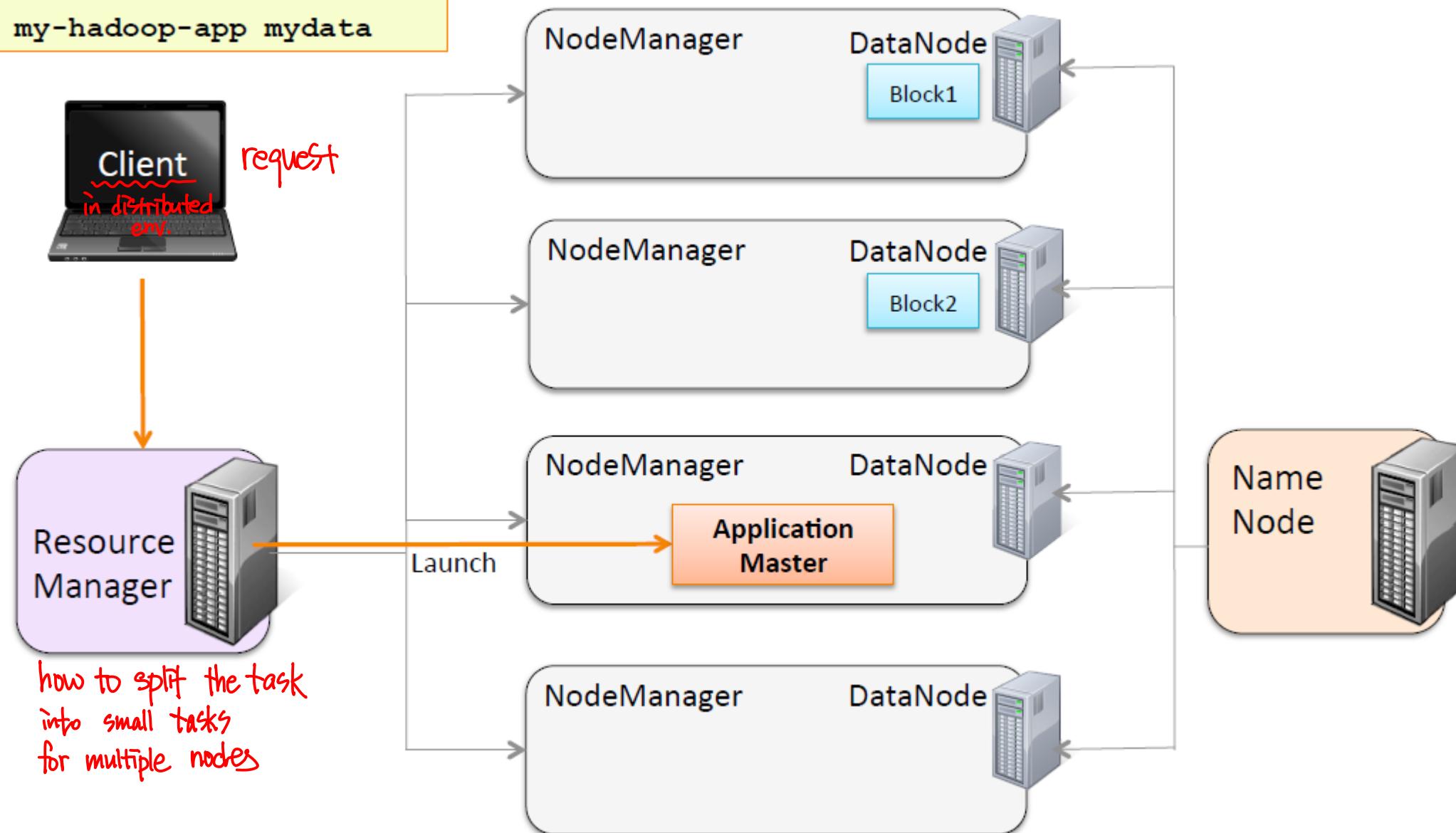
Running an Application on YARN (1)



Running an Application on YARN (2)

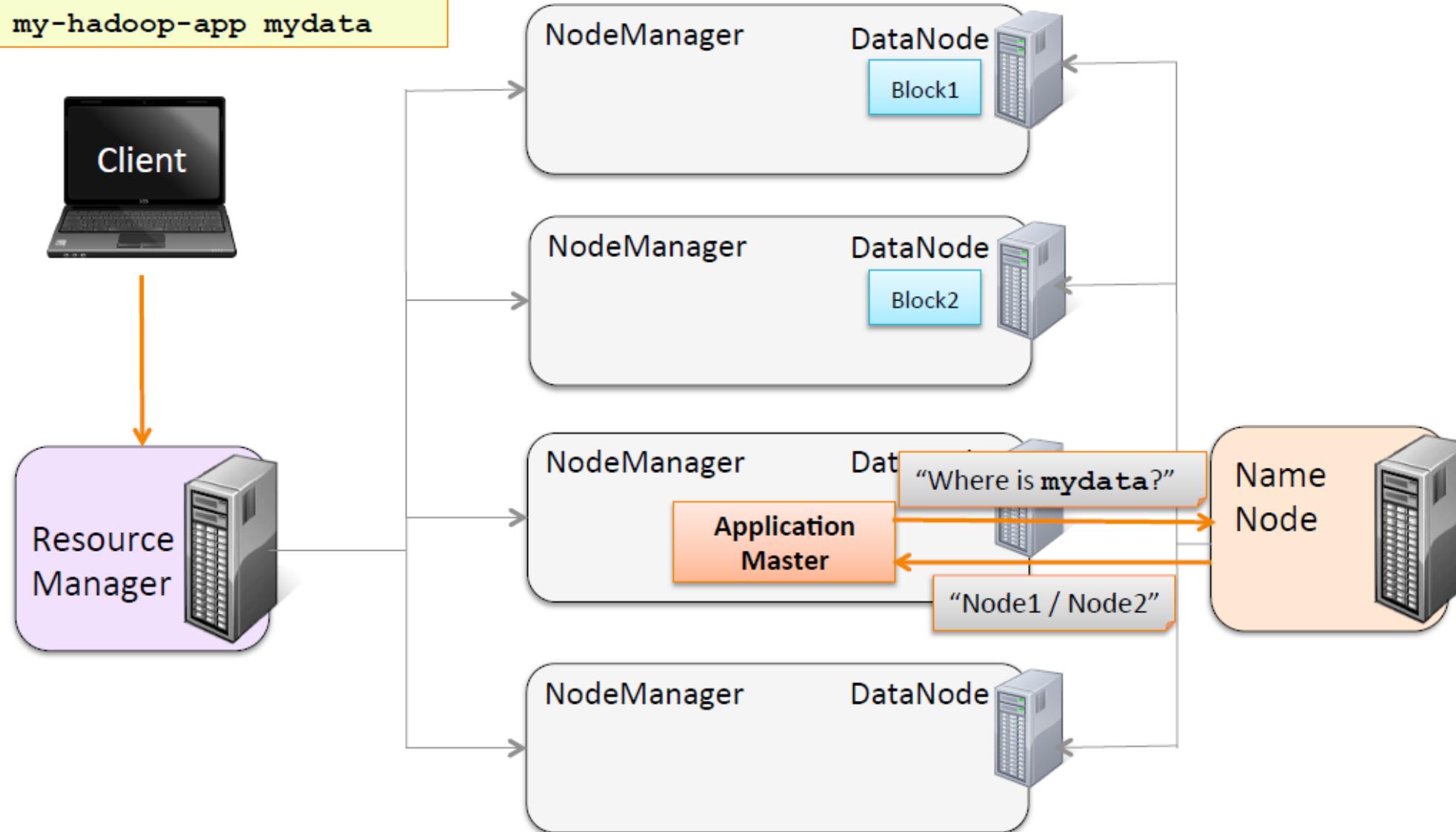


Running an Application on YARN (3)



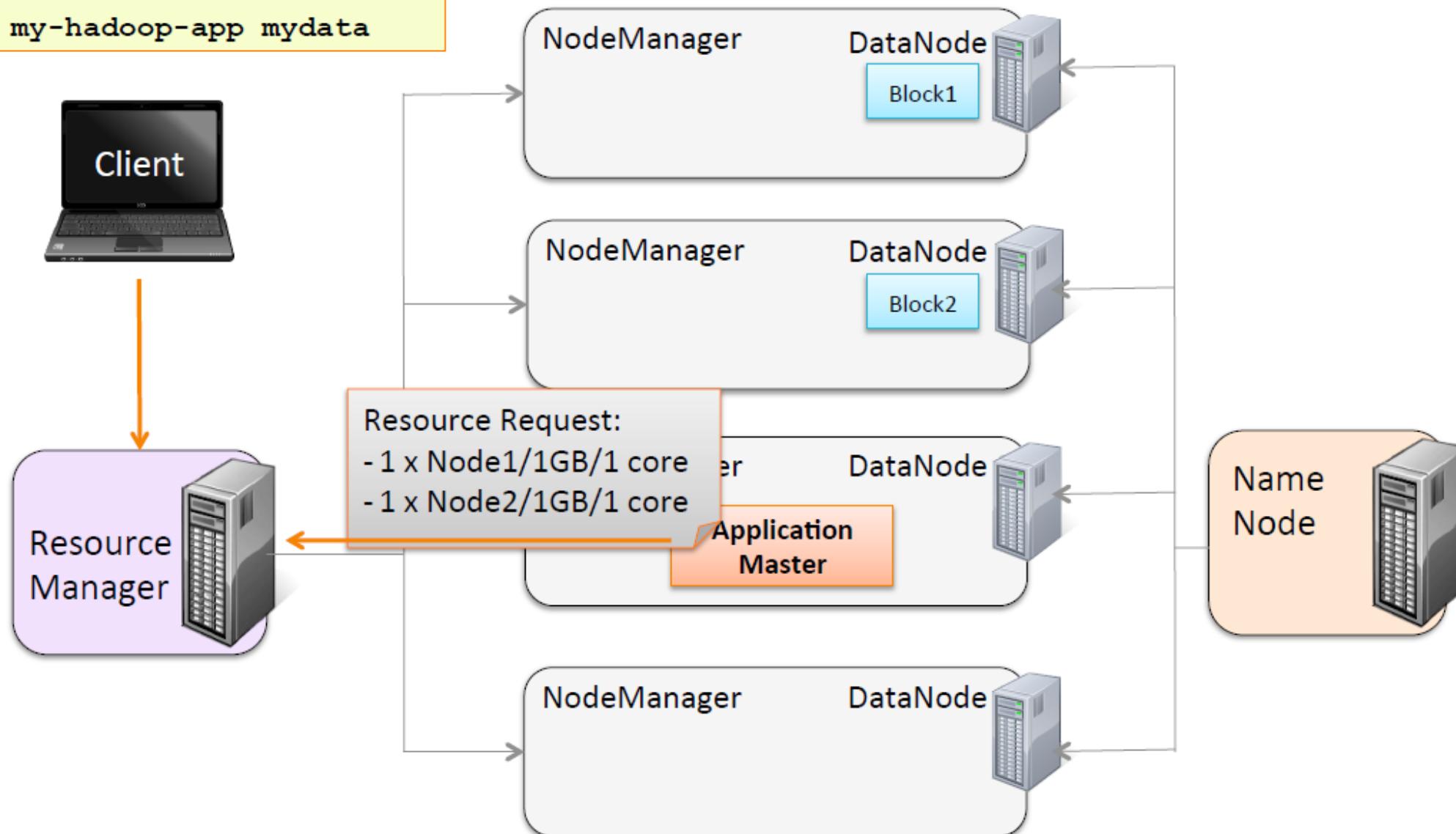
Running an Application on YARN (4)

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



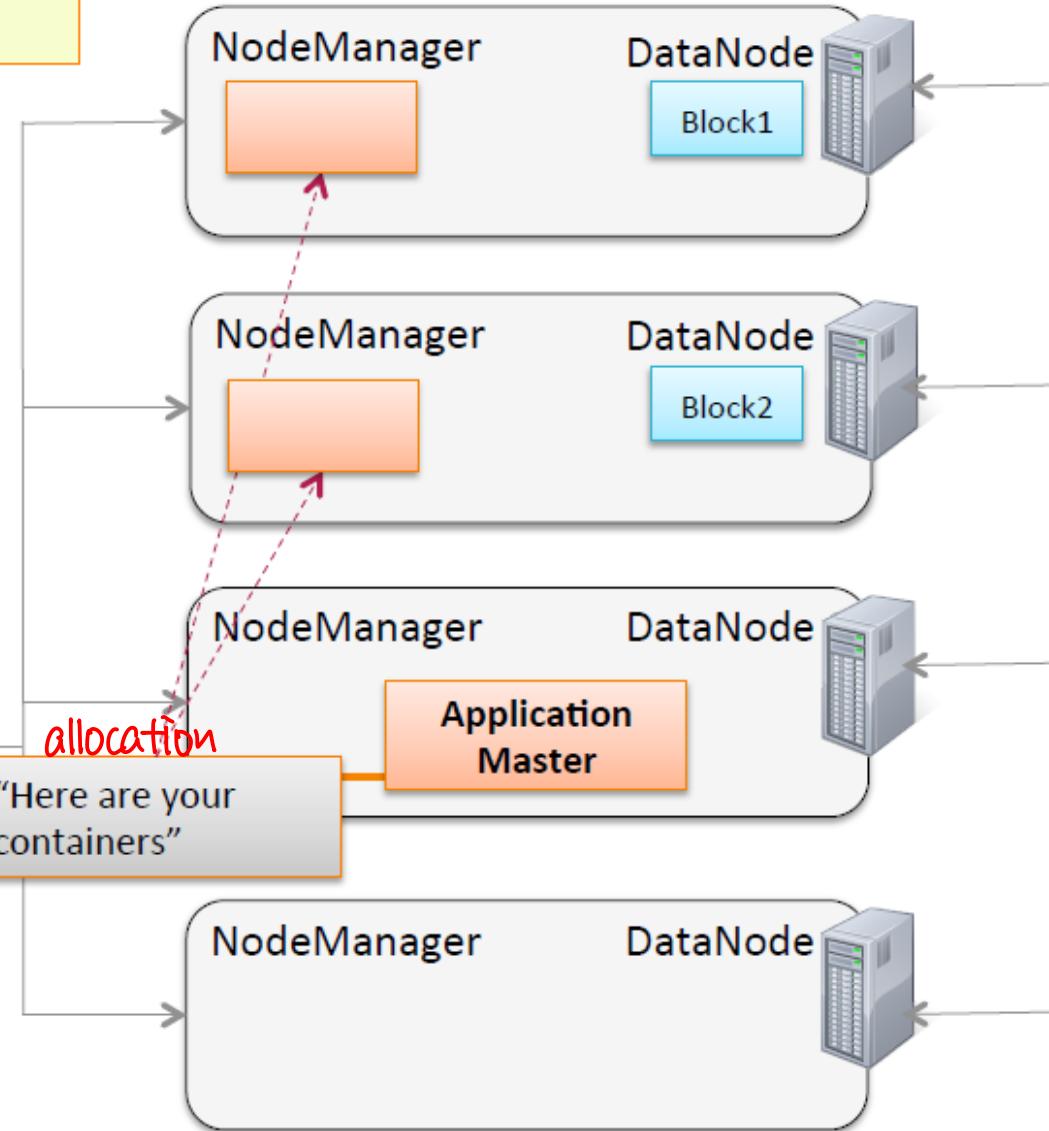
Running an Application on YARN (5)

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

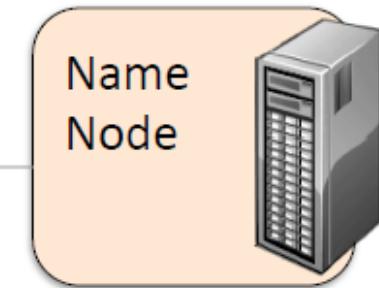


Running an Application on YARN (6)

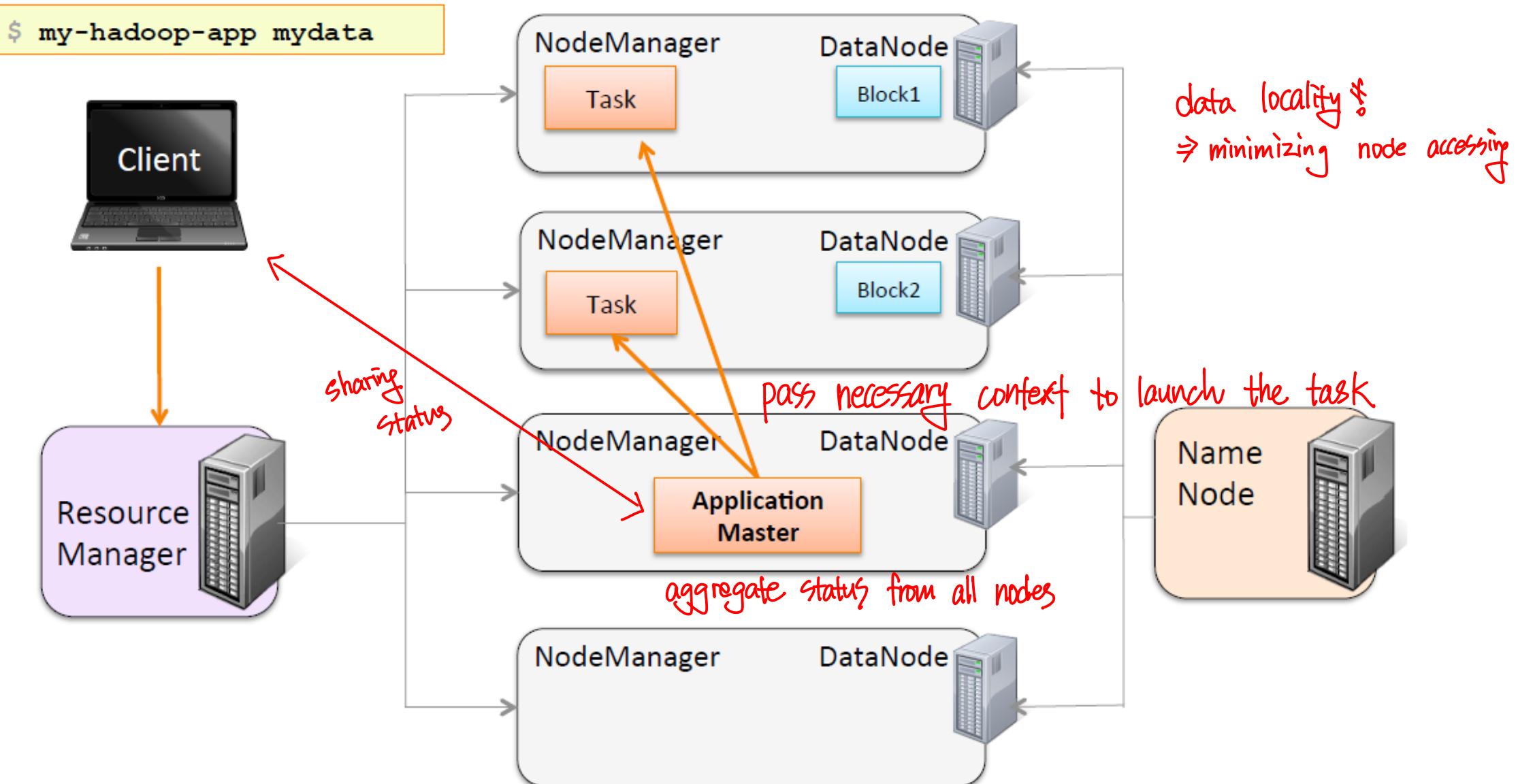
```
$ my-hadoop-app mydata
```



ideal
data storing node
= running app node
''' reducing cost'''

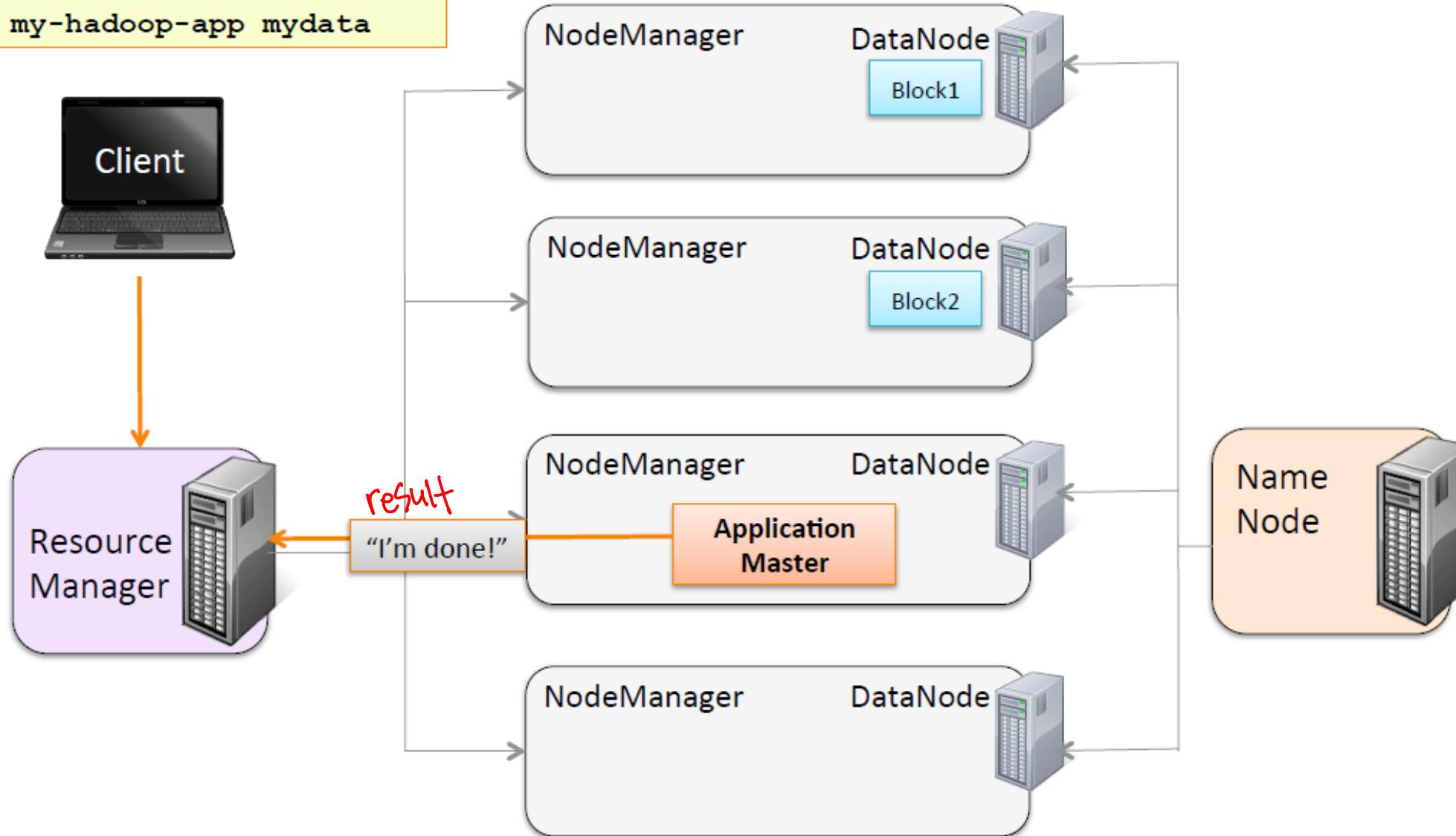


Running an Application on YARN (7)



Running an Application on YARN (8)

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



Working with YARN

- Developers need to be able to
 - Submit jobs (applications) to run on the YARN cluster
 - Monitor and manage jobs
- Hadoop includes three major YARN tools for developers
 - The Hue Job Browser
 - The YARN Web UI
 - The YARN command line

Master node : Resource Manager

Worker node : Node Manager } Containers
 (Application Master)

The Hue Job Browser

- The Hue Job Browser allows you to

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

The screenshot shows the Hue Job Browser interface. At the top, there's a navigation bar with links for 'Query Editors', 'Data Browsers', 'Workflows', and 'Search'. On the far right of the top bar, there's a user icon with a red box around it, indicating one active session. Below the bar, the title 'Job Browser' is displayed next to a blue circular icon.

On the left, there are search filters: 'Username' set to 'training', 'Text' input field containing 'Search for text', and a row of buttons for filtering by status: 'Succeeded' (green), 'Running' (orange, highlighted), 'Failed' (red), and 'Killed' (dark grey).

The main area is a table listing the running jobs:

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

At the bottom, a message says 'Showing 1 to 2 of 2 entries' and there are navigation buttons for 'Previous', '1', and '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

The screenshot shows the Hadoop Resource Manager UI for the 'Nodes of the cluster' page. The top navigation bar includes the Hadoop logo, the title 'Nodes of the cluster', and a 'Cluster Overview' link. A user is logged in as 'dr.who'. On the left, a sidebar menu is open, showing the 'Cluster Metrics' section with various counters for submitted and pending applications, running containers, and memory usage. Below this is the 'User Metrics for dr.who' section, which is currently empty. The main content area displays a table of nodes in the cluster, with two entries listed:

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

Annotations on the page include a blue dashed box around the sidebar and a purple box with the text 'link to Node Manager UI' pointing to the 'Nodes' link in the sidebar.

link to Node Manager UI

List of each node in cluster

Resource Manager UI: Applications

The screenshot shows the Hadoop Resource Manager UI for managing applications. The top navigation bar includes the Hadoop logo, the title "All Applications", and a "Cluster Overview" section. A user is logged in as "dr.who". On the left, there's a sidebar with sections for Cluster (About, Nodes, Applications, Scheduler), Tools, and a User Metrics table for "dr.who". The main area displays "Cluster Metrics" and a table of running 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

Applications Table

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	<div style="width: 50%;"></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 style="width: 100%;"></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 style="width: 100%;"></div>	History
application_1384200217415_0006	training	Process Logs	MAPREDUCE	root.training	Mon, 11 Nov 2013	Mon, 11 Nov 2013	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History

Link to Application Details... (next slide)

List of running and recent applications

Resource Manager UI: Application Detail

The screenshot shows the Hadoop Resource Manager Application Detail UI. The top navigation bar includes the Hadoop logo and the text "Logged in as: dr.who". On the left, a sidebar menu under "Cluster" lists "About", "Nodes", "Applications" (with sub-options: NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING, REMOVING, FINISHING, FINISHED, FAILED, KILLED), and "Scheduler". Below this is a "Tools" section. The main content area is titled "Application Overview" and displays the following details for a running application:

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:	(link)

Below this is a table titled "ApplicationMaster" showing one entry:

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

Two callout boxes point to specific features:

- A box points to the "Tracking URL" link with the text: "Link to Application Master (UI depends on specific framework)"
- A box points to the "Logs" column header with the text: "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



 **JobHistory** Logged in as: dr.who

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

- Command to configure and view information about the YARN cluster
 - **yarn <command>**
- Most YARN command line tools are for administrators rather than developers
- Some helpful commands for developers
 - **yarn application**
 - Use **-list** to see running applications
 - Use **-kill** to kill a running application
 - **yarn logs -applicationId <app-id>**
 - View the logs of the specified application

Practice3: Run a YARN Job

- **In this homework, you will**
 - Use the YARN Web UI to view your YARN cluster “at rest”
 - Submit an application to run on the cluster
 - Monitor the job using both the YARN UI and Hue
- **Please refer to the Homework description**

Essential Points

- HDFS is the storage layer for Hadoop
- Chunks data into blocks and distributes them across the cluster when data is stored
- Slave nodes run DataNode daemons, managed by a single NameNode on a master node
- Access HDFS using Hue, the `hdfs` command or via the HDFS API
- 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
- Monitor jobs using Hue, the YARN Web UI or the `yarn` command