Introduction to the Big Data Ecosystem (Hadoop and Spark)

Course Road Map

- **Module 1: Big Data Fundamentals**
- **Module 2: Data Acquisition and Storage**
- **Module 3: Data Access and Processing**
- **Module 4: Data Unification**
- **Module 5: Data Analysis**
- **Module 6: Big Data Deployment Options**

- **Lesson 2: Big Data Information Management System**
- **Lesson 3: Using Big Data Lite Virtual Machine and**
- **Lesson 4: Introduction to the Big Data Ecosystem**

Objectives

- After completing this lesson, you should be able to:
  - Define Hadoop and the *Hadoop Ecosystem*
  - List the Hadoop core components
  - Choose a Hadoop Distribution
  - List some of the other related projects in the Hadoop Ecosystem

Computer Clusters

- A computer rack (commonly called a rack) is a metal frame used to hold various hardware devices such as servers, hard disk drives, and other electronic equipment.
- A computer cluster is a single logical unit consisting of multiple computers (or racks) that are linked through a fast local area network (LAN).
- The components of a cluster, nodes (computers used as a servers), run their own instance of an operating system.
- A node typically includes CPU, memory, and disk(s) storage.

Distributed Computing

- Distributed computing is a technique that allows individual computers to be networked together.
- A distributed file system is a client/server application that allows clients to access and process data stored on the server as if it were stored on their own computer.
- File systems that manage the storage across a network of machines are called distributed file systems.

Apache Hadoop

- Apache Hadoop:
  - Is an open-source software framework for ***Distributed Storage*** and ***Distributed Processing*** of big data on clusters of commodity hardware
  - Is a batch and interactive data-processing system for enormous amounts of data
- Open source available:
  - From the Apache Hadoop Foundation
  - As distributions, such as:
    - Cloudera's Distribution Including Apache Hadoop (CDH)
    - Hortonworks (HDP)
    - MapR

Types of Analyses That Use Hadoop

- *Market analysis*
- *Product recommendations*
- Demand forecasting
- *Fraud detection*
- Text mining
- *Index building*
- Graph creation and analysis
- Pattern recognition
- Collaborative filtering
- Prediction models
- Sentiment analysis
- Risk assessment

Types of Data Generated

- Financial transactions
- Sensors data
- Server logs
- Analytics
- Email and text messages
- Social media

Apache Hadoop Core Components

A Hadoop cluster has:

- Distributed data using Apache Hadoop Distributed File System (HDFS)
- Distributed processing using one of the following:
  - Yet Another Resource Negotiator (YARN) MR2, an extensible framework job scheduling and cluster resource management
  - MapReduce Framework (MR1)

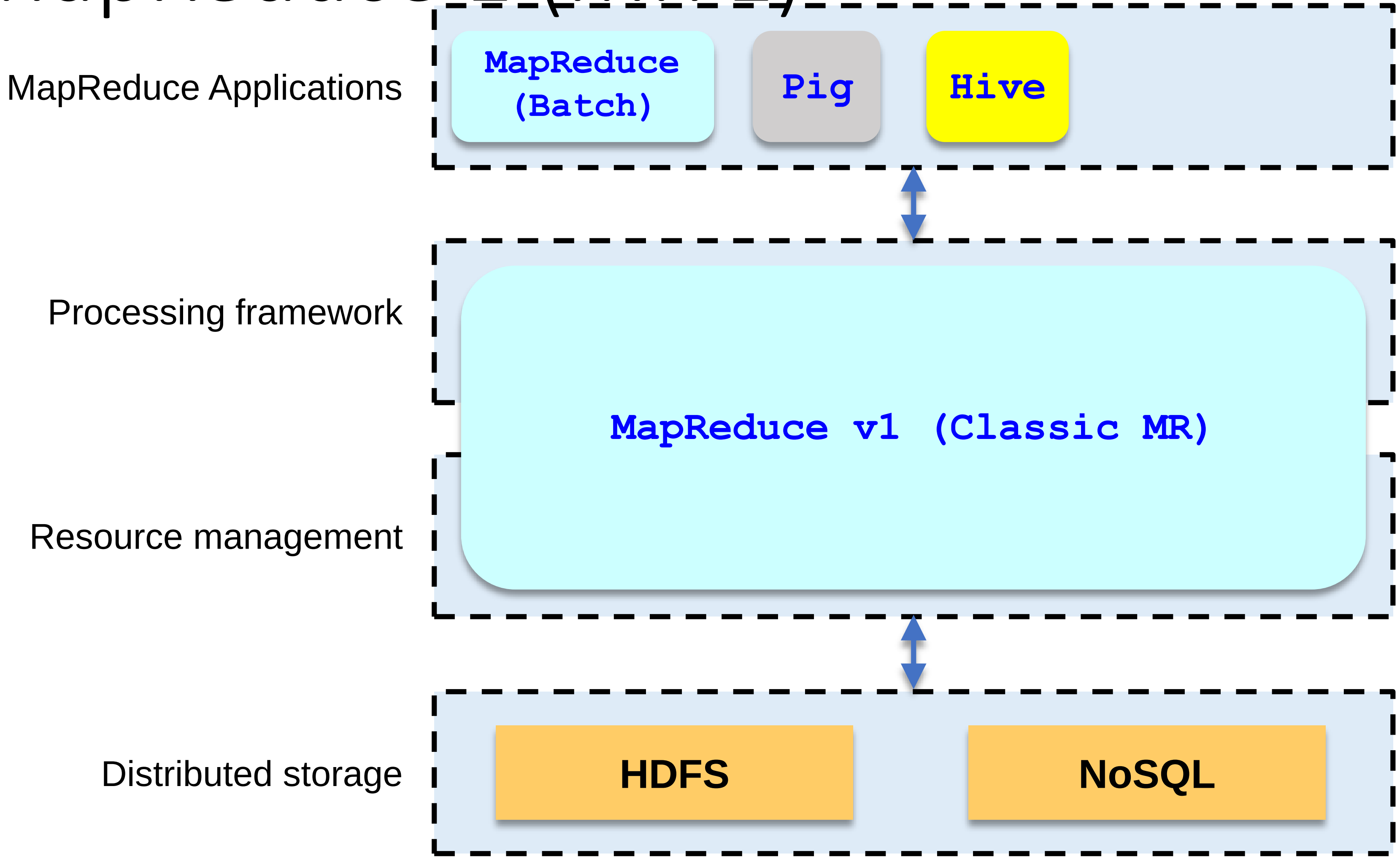
Apache Hadoop Core Components: HDFS

- Master-slave architecture
- Based on Google's File System (GFS) paper
- Stores and distributes data across the nodes in the cluster as data is loaded
- Redundant (reliability)
- Fault tolerant (high availability)
- Scalable (out instead of up)

Apache Hadoop Core Components: MapReduce Framework (MRv1)

- Is a programming model or framework for distributed computing
- Schedules and monitors tasks, and re-executes failed tasks
- Uses master-slave architecture
- Integrates with HDFS to provide the exact same benefits for distributed parallel data processing on the cluster
- ***Sends computations where the data is stored on local disks (data locality)***
- Hides complex "housekeeping" and distributed computing complexity tasks from the developer
- Supports <u>only</u> MapReduce applications

Running Applications Before Hadoop 2.x with MapReduce 1 (MR 1)



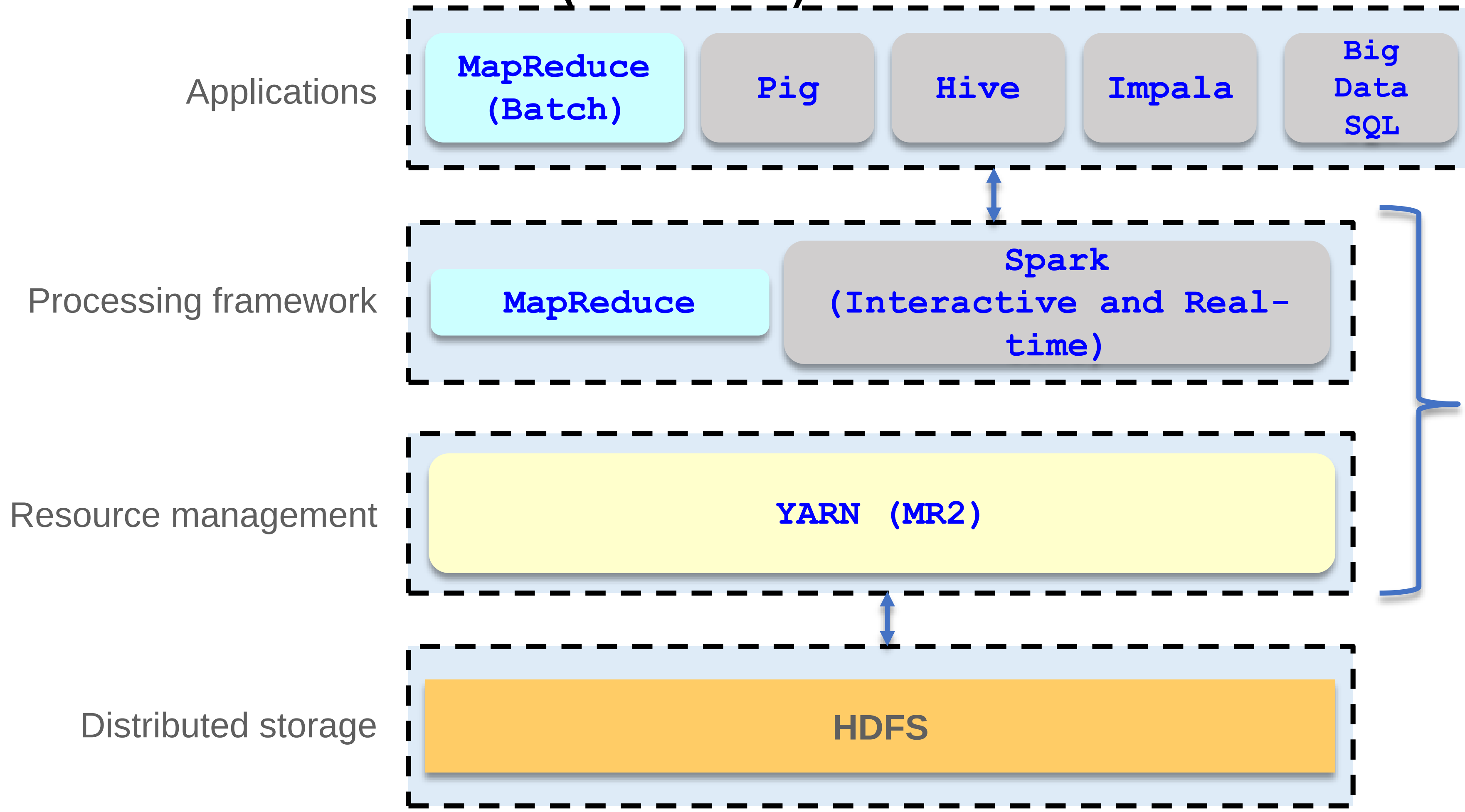
- Hadoop Cluster Resource Management *and*
- ***Batch*** Data Processing
- A single processing engine

Apache Hadoop Core Components: YARN (MR2)

- Is a subproject of Hadoop that separates resource management and processing components
- Is a resource-management framework for Hadoop that is independent of execution engines
- Provides a more efficient and flexible workload scheduling as well as a resource management facility, both of which ultimately enable Hadoop to run more than just MapReduce jobs such as Impala, Spark, and so on

YARN

Running Applications Starting with Hadoop 2.x With YARN (MR 2)



- Hadoop Cluster Resource Management and
- Batch, ***<u>Interactive</u>***, and ***<u>Real-Time</u>*** Data Processing
- Multiple data processing engines

Apache Hadoop Ecosystem

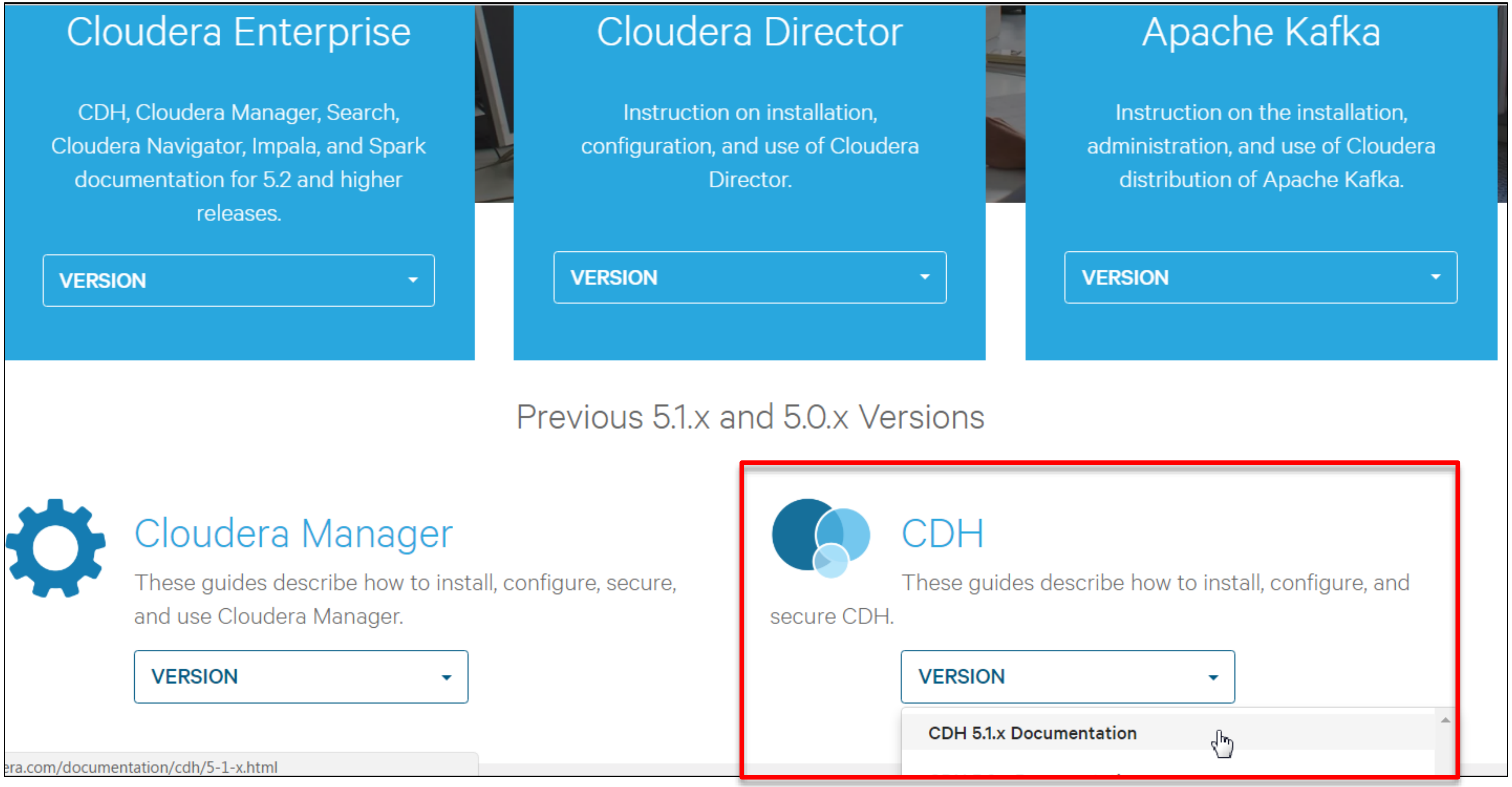
YARN
(MapReduce 2)

Hadoop Core Components:
- **HDFS (Storage)**
- **YARN (Processing and Resource Management, MR2)**
- **MapReduce (Distributed processing, MR1)**

Hadoop Ecosystem:
A *partial* list of
related projects (extend core Hadoop or make it easier to use)

http://www.cloudera.com/documentation.html

Additional Resources: Cloudera Distribution

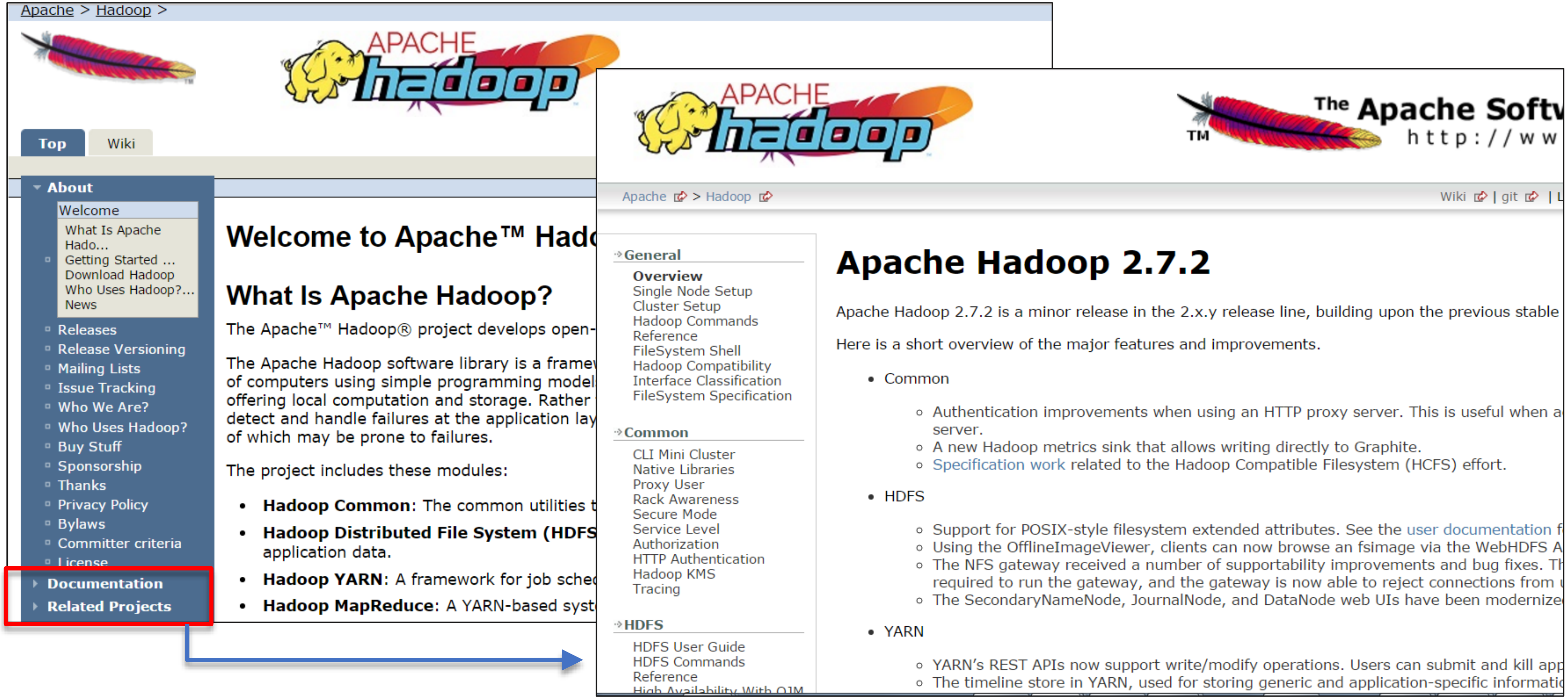


Big Data Appliance (BDA)

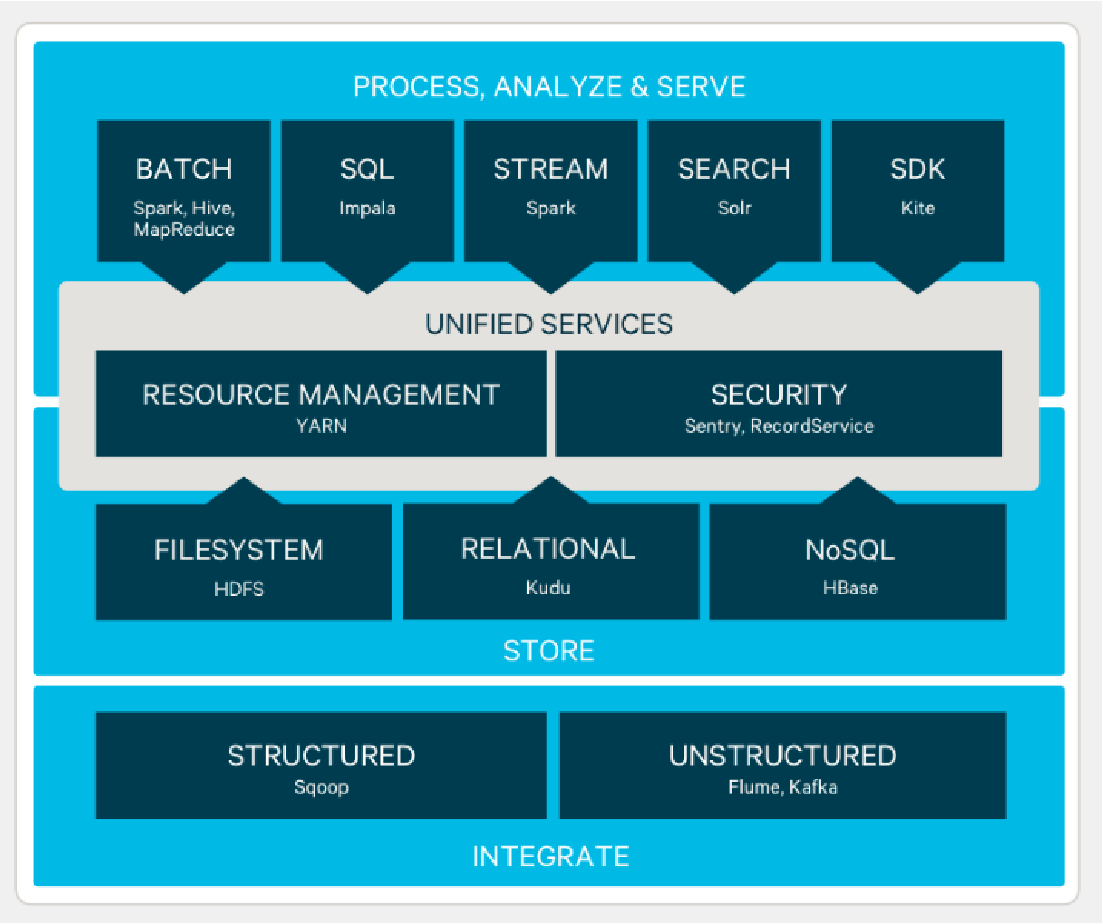
- The Oracle BDA:
  - Simplifies the deployment of Apache Hadoop
  - Optimizes the secure deployment of CDH clusters on an engineered system to manage and process the data
  - Is covered in a later lesson in this course

Additional Resources: Apache Hadoop

http://hadoop.apache.org/



CDH Architecture



Source: http://www.cloudera.com/products/apache-hadoop.html

CDH Components

Component	Description
Apache Hadoop	• A framework for executing applications on a large cluster of servers. It is built for massively parallel processing across a large number of nodes (servers). • Consists of the following core components: Hadoop Distributed File System (HDFS) and MapReduce
Hue (Hadoop User Experience)	• Is an open-source tool • Easy to use web front end for viewing files, running queries, performing searches, scheduling jobs, and more • Contains several applications to access a Hadoop cluster through a web front end
Apache Oozie	• Enables developers to create, edit, and submit workflows by using the Oozie dashboard • After considering the dependencies between jobs, the Oozie server submits those jobs to the server in the proper sequence.
Apache Spark	• It is an open source parallel data processing framework. • It complements Apache Hadoop. • It makes it easy to develop fast, unified Big Data applications combining batch, streaming, and interactive analytics on all your data.

CDH Architecture

Component	Description
Apache Solr (Cloudera Search)	• Cloudera Search is one of Cloudera's near-real-time access products and is powered by Solr. It enables nontechnical users to search and explore data stored in or ingested into Hadoop, Oracle NoSQL Database, and HBase. • Users do not need SQL or programming skills to use Cloudera Search because it provides a simple, full-text interface for searching.
Apache Hive	• Hive Metastore provides a metadata layer that describes the data stored in HDFS. • It provides a SQL layer to data on HDFS. It can run SQL queries on HDFS data. • It uses Map/Reduce for execution and HDFS for storage.
Apache Pig	• It is an analysis platform that provides a data flow language called Pig Latin. • It is an alternative abstraction on top of MapReduce.
Cloudera Impala	• The Impala server is a distributed, massively parallel processing (MPP) database engine. • It consists of different daemon processes that run on specific hosts within your CDH cluster. • The core Impala component is a daemon process that runs on each node of the cluster.

CDH Components

Component	Description
Apache Flume	• A distributed, reliable, available service for efficiently moving large amounts of data as it is generated • Ideal for collecting logs from diverse systems and inserting them in HDFS
Apache Sqoop	• Imports tables from an RDBMS into HDFS • Imports data from RDBMS into HDFS as delimited text files or sequence files • Generates a class file that can encapsulate a row of the imported data
Apache Hbase	• Is a NoSQL data store • Provides scalable inserts, efficient handling of sparse data, and a constrained data access model
Apache ZooKeeper	• ZooKeeper is a centralized service for maintaining configuration information, naming, distributed synchronization, and group services. • HBase cannot be active without ZooKeeper.
Apache Mahout	• Scalable machine-learning and data-mining algorithms
Apache Whirr	• Apache Whirr is a set of libraries for running cloud services. It provides: – A cloud-neutral way to run services – A common service APISmart defaults for services

Hadoop Major Timelines at a Glance

2002
Doug Cutting starts the Nutch project (Open source web crawler and indexer)

2003
Google publishes Google File System (GFS) paper.

2004
- Google publishes the MapReduce paper in 2004.
- Doug Cutting adds GFS and MapReduce support to Nutch in 2004/2005.

2006
Doug Cutting joins Yahoo!. Hadoop is released.

2007
- The NY Times converts 4 TB of archive to PDF for the Web over 100 Amazon EC2 compute cloud in less then 24 hours.
- Google sorts 1 TB in 68 seconds.

2008
Yahoo! wins Terasort competition: Fastest sort of 1 TB; 209 seconds over 910 nodes.

Cloudera is founded and Doug Cutting joins Cloudera.

2009
- Cloudera releases CDH.
- Amplab starts Mesos/Spark project.

Fastest sort of 1 TB; 62 sec over 1460 nodes

2010
Spark becomes an open source project.

HortonWorks is founded.

2013-2014
Spark becomes an Apache incubator, and then Apache top-level project and full integration with YARN.

2015
Spark wins 100TB sort.

Where to Go for More Information

Component	Website
Cloudera Manager	http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise/cloudera-manager.html
Apache Hadoop	http://hadoop.apache.org/
Apache Hadoop – Cloudera	**http://www.cloudera.com/products/apache-hadoop.html**
fuse-dfs	http://fuse.sourceforge.net/
Cloudera Hue	http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/4.2.0/Hue-2-User-Guide/hue2.html
Apache Oozie	http://oozie.apache.org/
Apache Hive	https://hive.apache.org/
Apache Pig	http://pig.apache.org
Apache Flume	http://flume.apache.org/
Apache Sqoop	http://sqoop.apache.org/
Apache HBase	http://hbase.apache.org/
Apache ZooKeeper	http://zookeeper.apache.org
Apache Mahout	http://mahout.apache.org
Apache Whirr	https://whirr.apache.org/

Summary

- In this lesson, you should have learned how to:
  - Define Hadoop and the *Hadoop Ecosystem*
  - Describe the Hadoop core components
  - Choose a Hadoop Distribution
  - List some of the other related projects in the Hadoop Ecosystem

Introduction to the Hadoop Distributed File System (HDFS)

Course Road Map

- Module 1: Big Data Fundamentals
- **Module 2: Data Acquisition and Storage**
  - Lesson 5: Introduction to the Hadoop Distributed File System (HDFS)
  - Lesson 6: Acquiring Data by Using CLI, Fuse, Flume, and Kafka
  - Lesson 7: Acquiring and Accessing Data by Using Oracle NoSQL Database
- Module 3: Data Access and Processing
- Module 4: Data Unification
- Module 5: Data Analysis
- Module 6: Big Data Deployment Options

Objectives

- After completing this lesson, you should be able to:
  - Describe the architectural components of HDFS
  - Interact with data stored in HDFS by using various methods

Agenda

- Understand the architectural components of HDFS
- Interact with data stored in HDFS
  - Hue
  - Hadoop client file system shell command-line interface (CLI)
  - WebHDFS
  - HttpFS

HDFS Design Principles and Characteristics

- **Master-slave architecture**
- **Fault-tolerant (HA)**
- **Redundant**
- **Supports MapReduce & Spark**
- **Scalable**
- **Commodity Hardware**

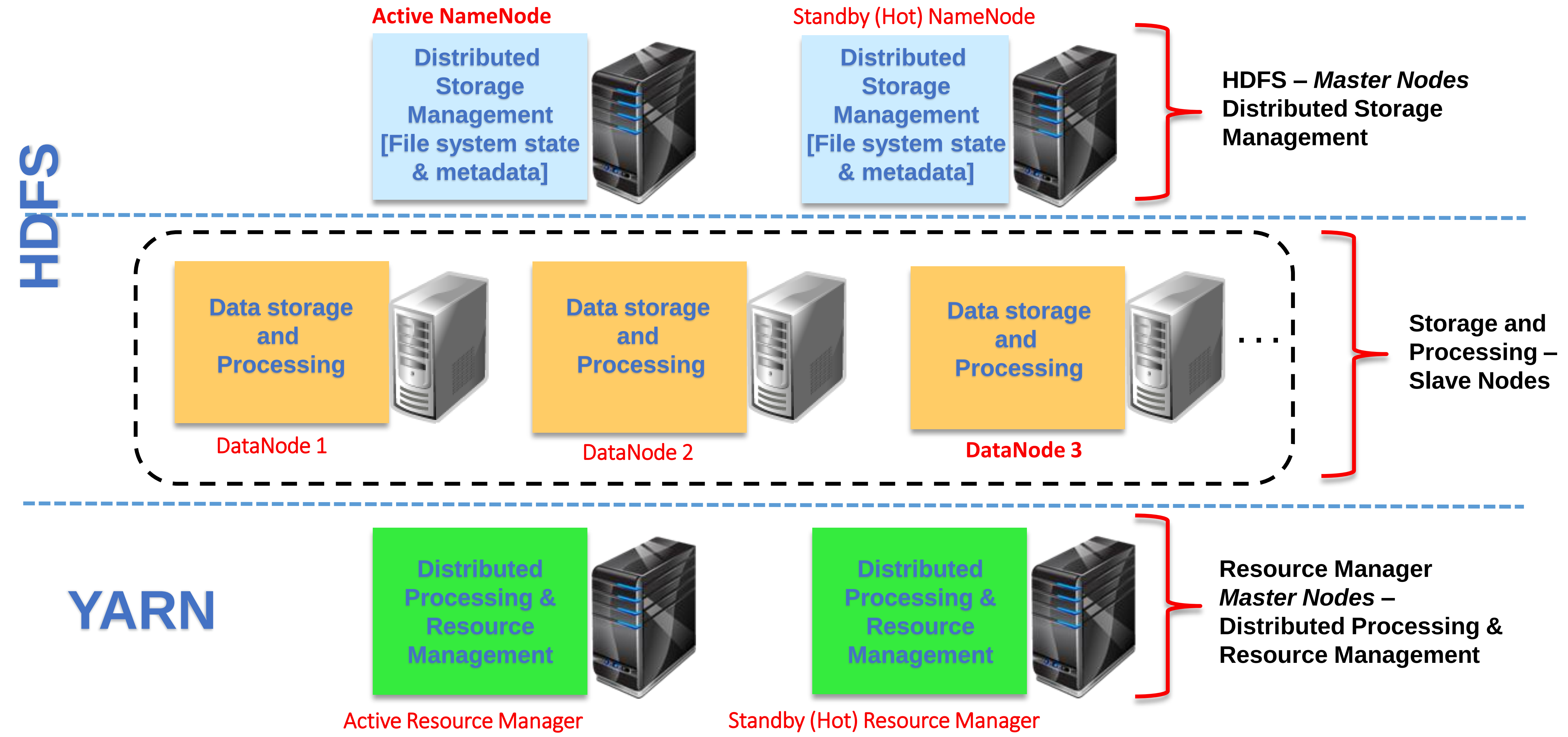
HDFS Key Definitions

Term	Description
Hadoop	A batch (MapReduce), interactive, or real-time (Spark and MR2) processing infrastructure that stores and distributes files and distributes work across a group of servers (nodes)
Hadoop Cluster	A collection of racks containing master and slave nodes
Blocks	HDFS breaks down a data file into blocks or “chunks” and stores the data blocks on different slave DataNodes in the Hadoop cluster.
Replication Factor	HDFS makes three copies of data blocks and stores them on different data nodes/racks in the Hadoop cluster.
NameNode (NN)	A service (daemon) that maintains a directory of all files in HDFS and tracks where data is stored in the HDFS cluster. It basically manages the file system’s metadata ***(does not contain actual data).***
DataNode (DN)	This is where the data is stored (HDFS) and processed (MapReduce). This is a slave node. HDFS stores the blocks or “chunks of data for a set of files on the data nodes.

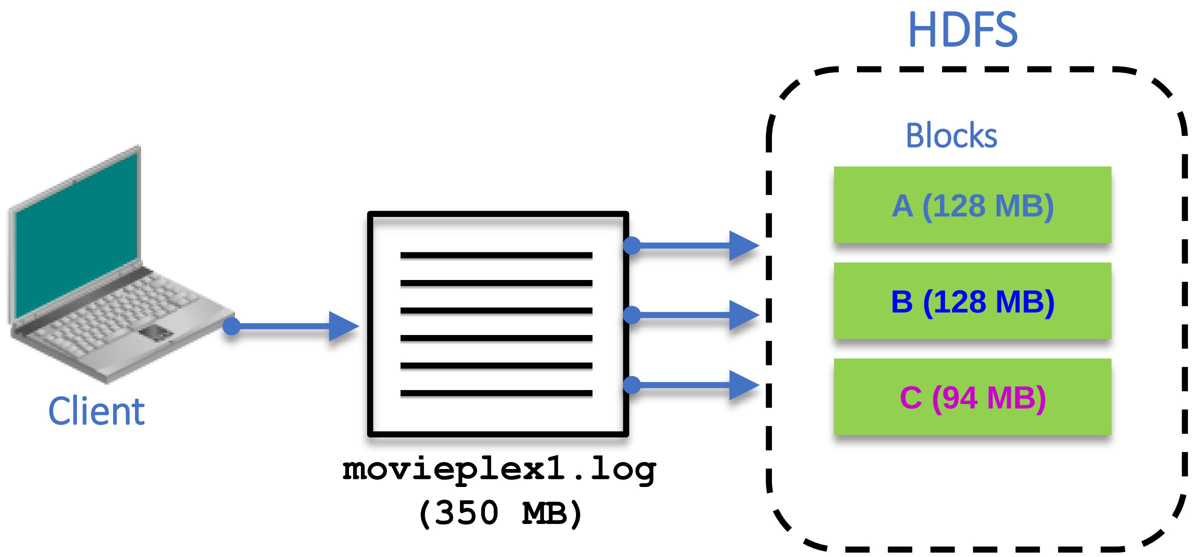
HDFS Deployments: High Availability (HA) and Non-HA

- Non-HA Deployment (Prior to Hadoop 2.0):
  - Uses the NameNode/Secondary NameNode architecture
  - The Secondary NameNode is not a failover for the NameNode.
  - The NameNode was the **Single Point of Failure (SPOF)** of the cluster prior to Hadoop 2.0 and CDH 4.0.
- HA Deployment (Hadoop 2.0 and later):
  - HDFS HA addresses the SPOF by running two redundant NameNodes in the same cluster to provide a **Fast Failover** if needed:
    - **Active NameNode**: Responsible for all client operations in the cluster
    - **Standby NameNode**: Acts as a **"hot" backup** to the Active NameNode, maintaining enough system information to provide a fast failover if necessary
  - **HA allows a fast failover** to a new NameNode in case a machine crashes.
- In this course, you will focus on the HDFS HA deployment option only.

Sample Hadoop High Availability (HA) Cluster



HDFS Files and Blocks

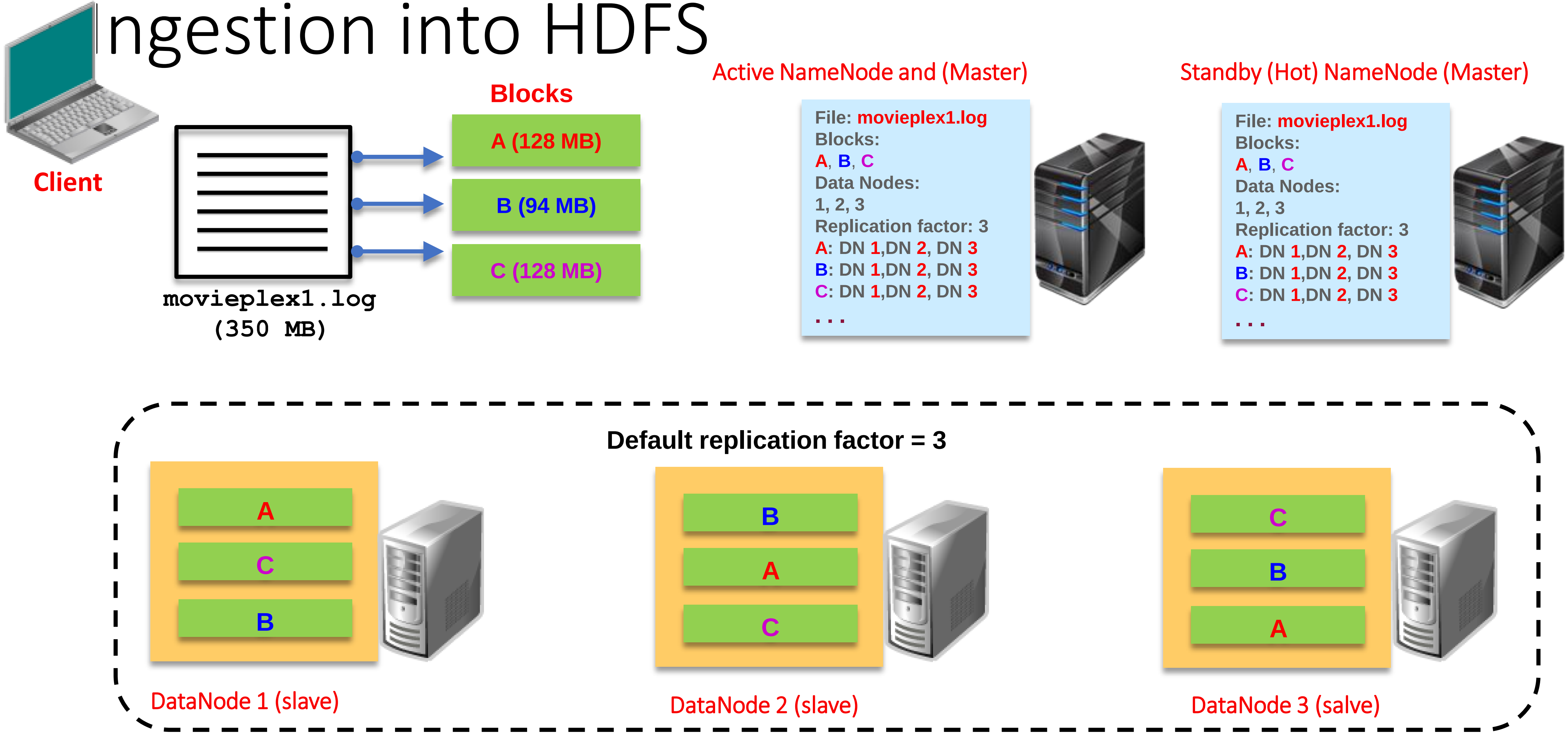


- Files in HDFS consist of blocks.
- HDFS blocks default to 128 MB in size
  (configurable).
- *Files are "chunked" into blocks as they are ingested (using Flume or Kafka) into HDFS.*

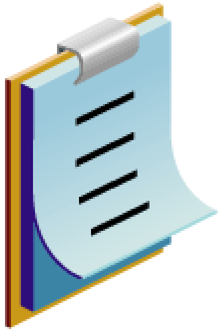
Assuming a default block size of 128 MB, HDFS ingests the **`movieplex1.log`** file into (3) blocks:

- A (128 MB)
- B **(128 MB)**
- C **(94 MB)**

Blocks are Replicated in the Cluster Upon Ingestion into HDFS



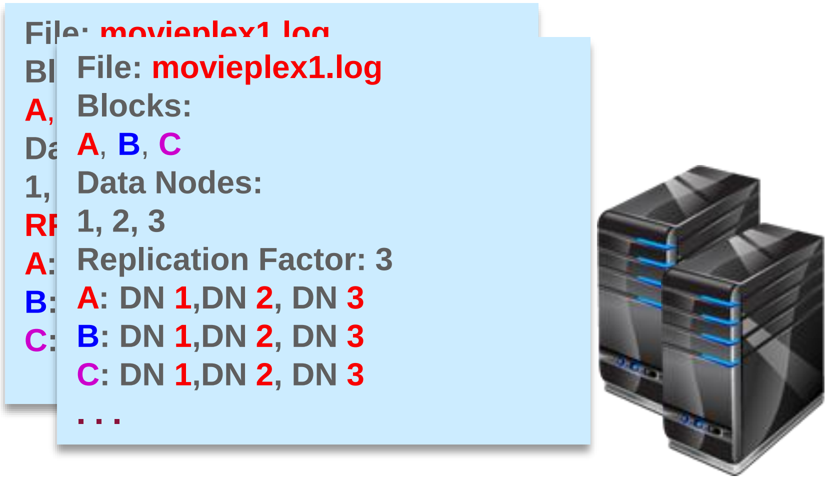
Active and Standby NameNodes Daemons

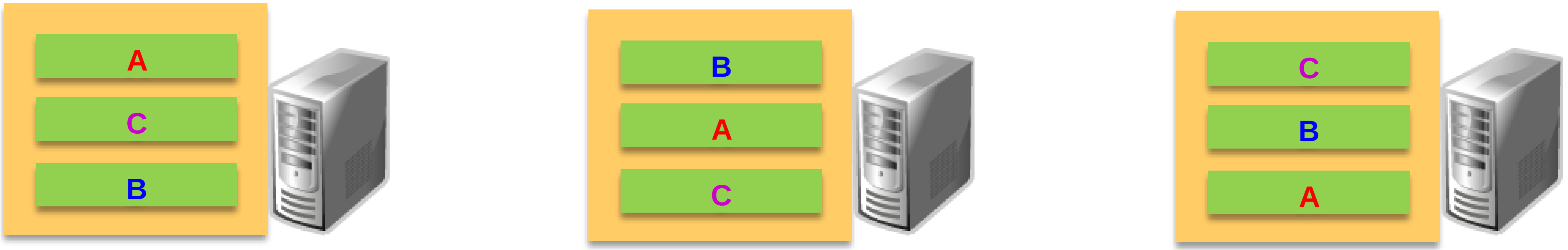


NameNode stores file system metadata such as:

- File information (name, updates, replication factor, etc.)
- File blocks information and locations
- Access rights to the file
- Number of files in the cluster
- Number of DataNodes in the cluster

Active NameNode and
Standby (Hot) NameNode (Masters)





DataNode 1 (slave)

DataNode 2 (slave)

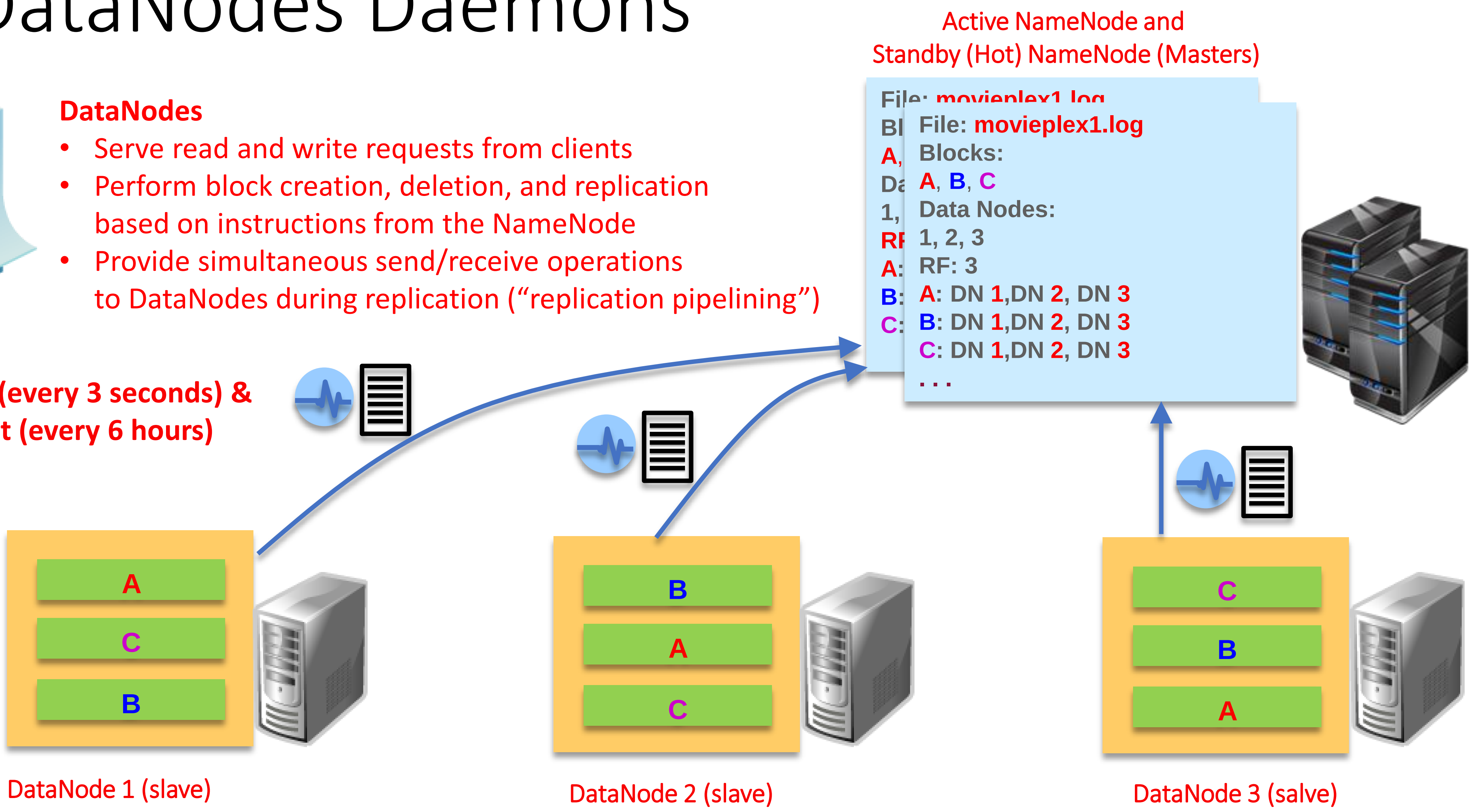
DataNode 3 (salve)

DataNodes Daemons

DataNodes

- Serve read and write requests from clients
- Perform block creation, deletion, and replication based on instructions from the NameNode
- Provide simultaneous send/receive operations to DataNodes during replication (“replication pipelining”)

Heartbeat (every 3 seconds) & Blockreport (every 6 hours)

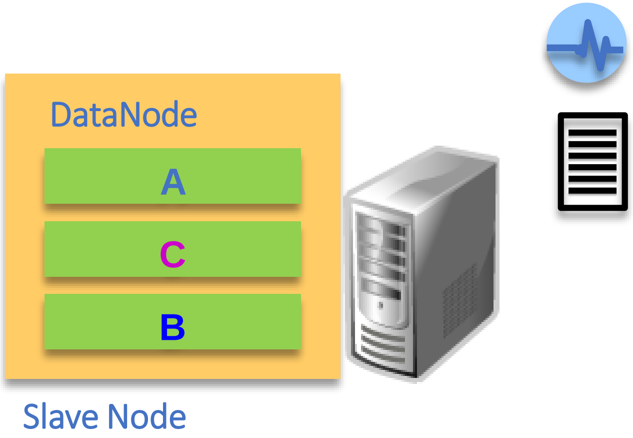


Functions of the NameNode

- Acts as the repository for all HDFS metadata
- Maintains the file system namespace
- Executes the directives for opening, closing, and renaming files and directories
- Stores the HDFS state in an image file (`fsimage`)
- Stores file system modifications in an edit log file (`edits`)
- On startup, merges the `fsimage` and `edits` files, and then empties `edits`
- Places replicas of blocks on multiple racks for fault tolerance
- Records the number of replicas (replication factor) of a file specified by an application

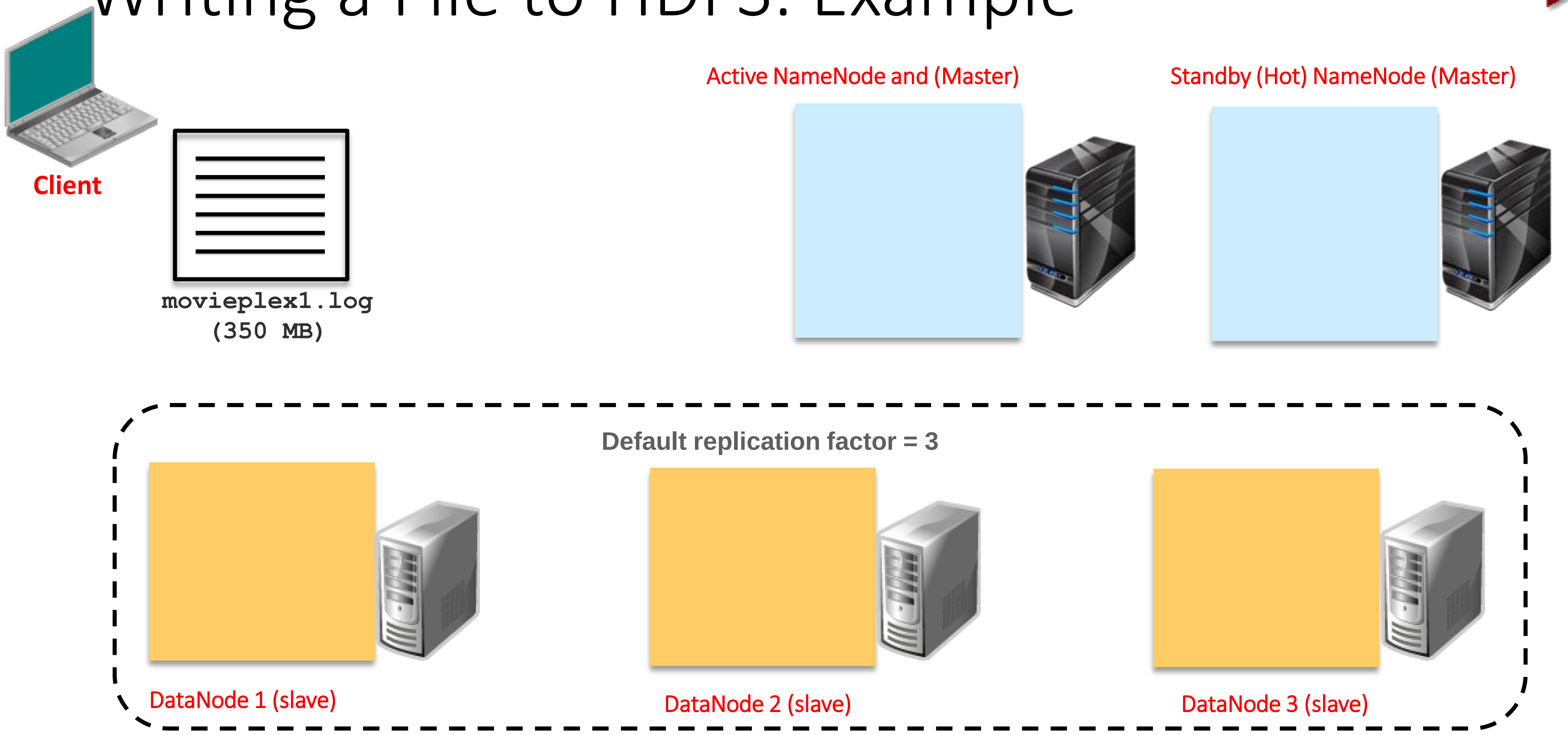
Functions of DataNodes

- DataNodes perform the following functions:
  - Serving read and write requests from the file system clients
  - Performing block creation, deletion, and replication based on instructions from the NameNode
  - Providing simultaneous send/receive operations to DataNodes during replication ("replication pipelining")

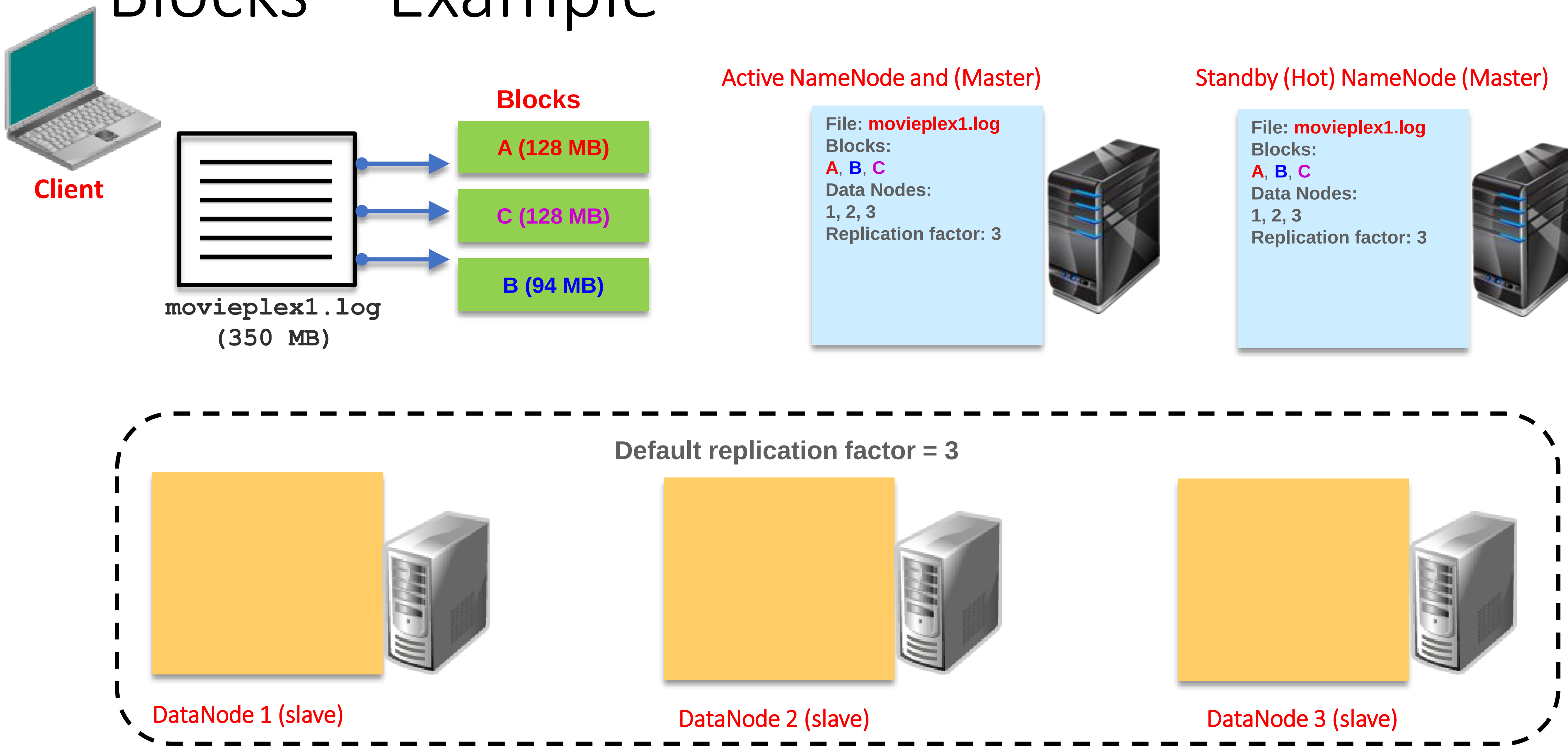


- **Heartbeat (I am alive!) every 3 seconds**
- **Blockreport (what data I am storing!) every 6 hours to NN**

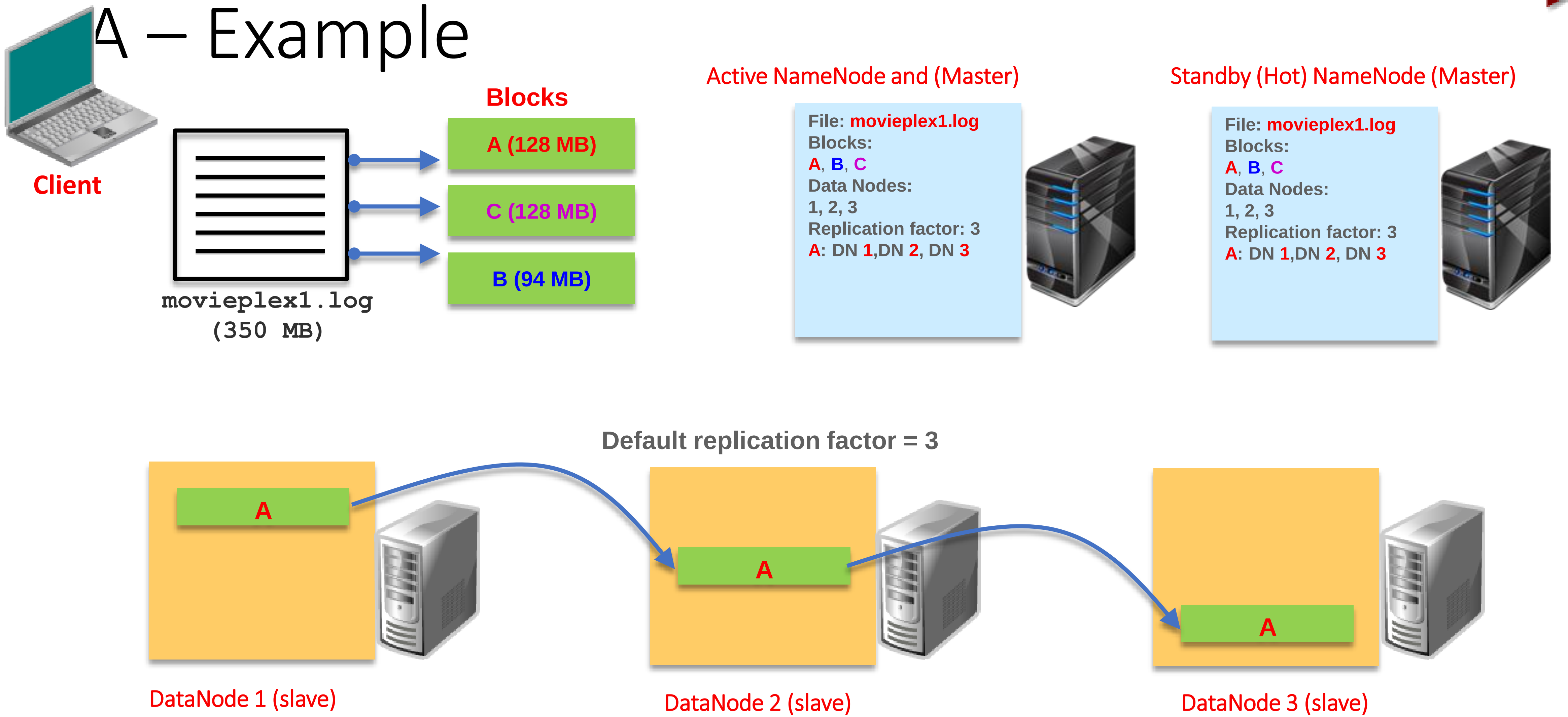
Writing a File to HDFS: Example



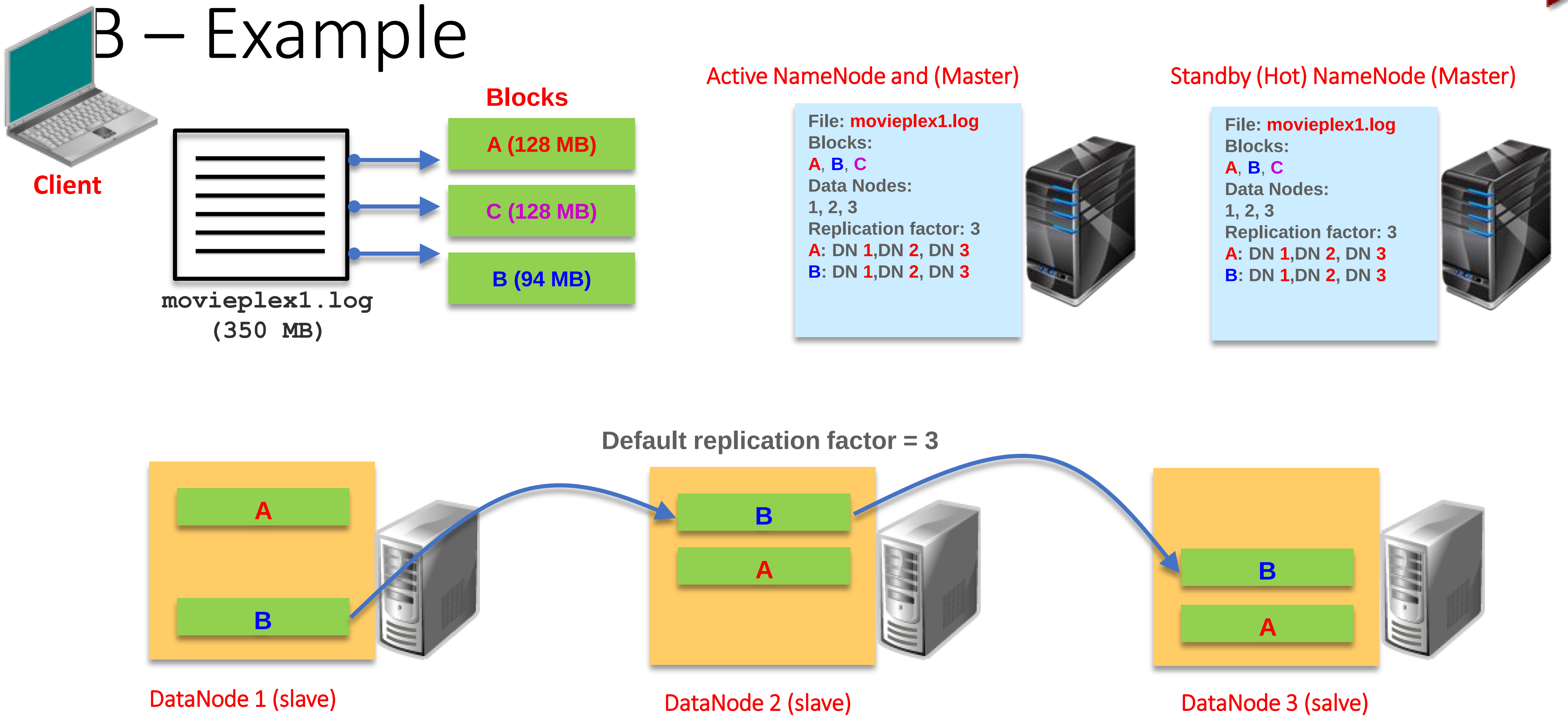
Writing a File to HDFS: File is “Chunked” into Blocks – Example



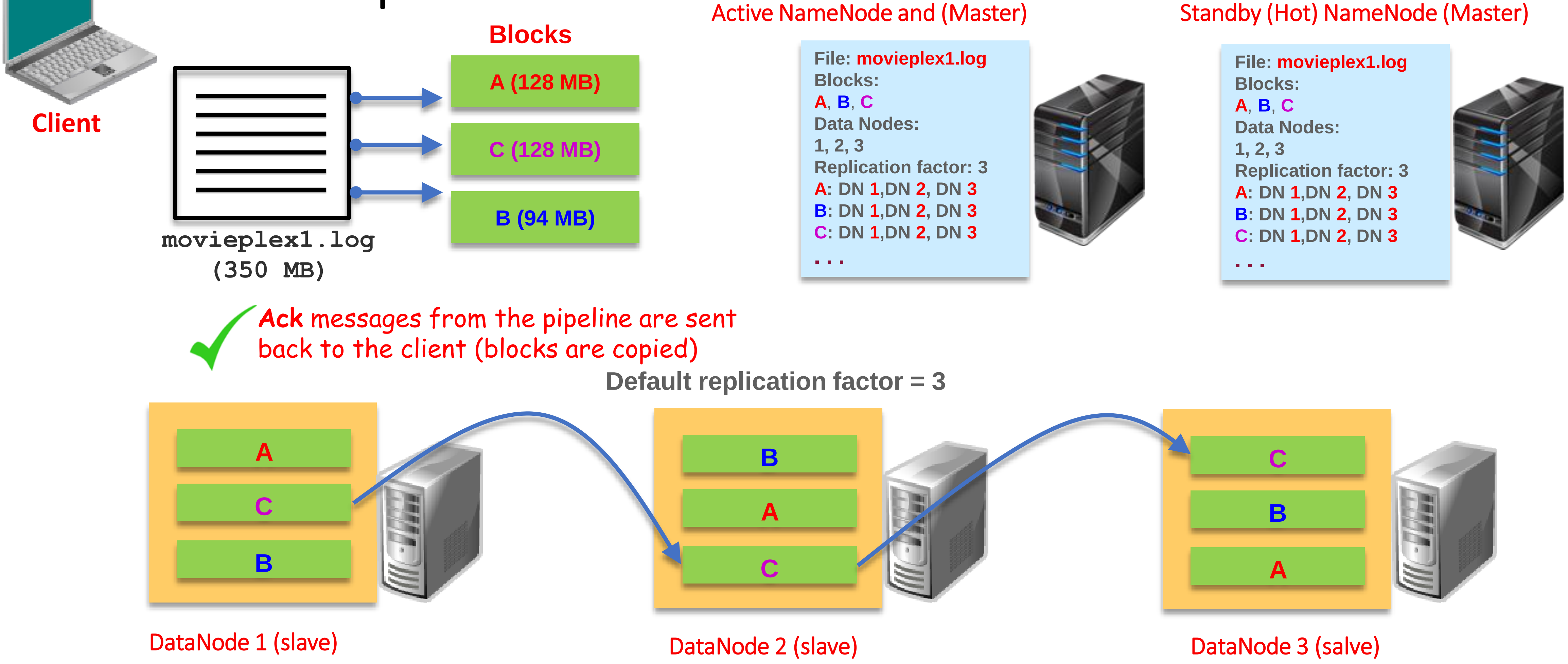
Writing a File to HDFS: Pipeline Created, Block A – Example



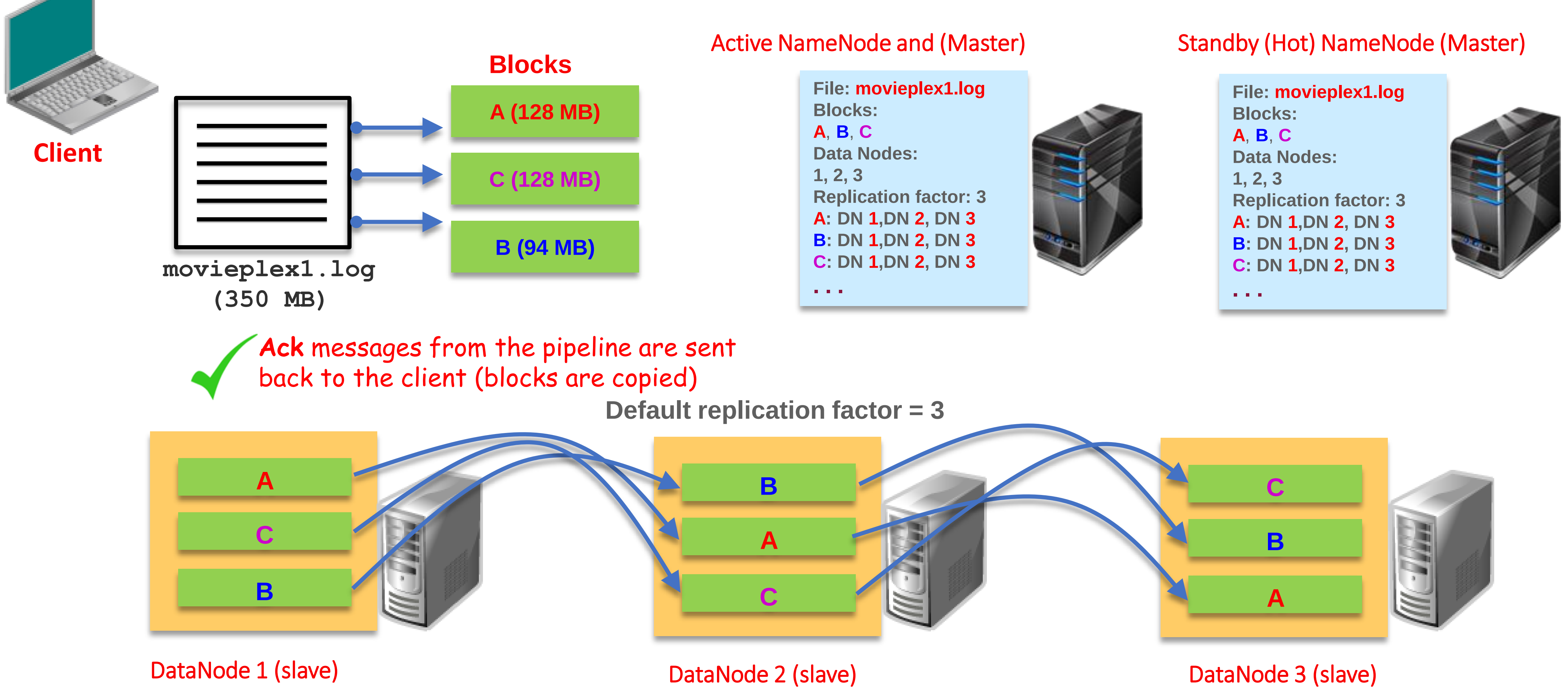
Writing a File to HDFS: Pipeline Created, Block B – Example



Writing a File to HDFS: Pipeline Created, Block C – Example



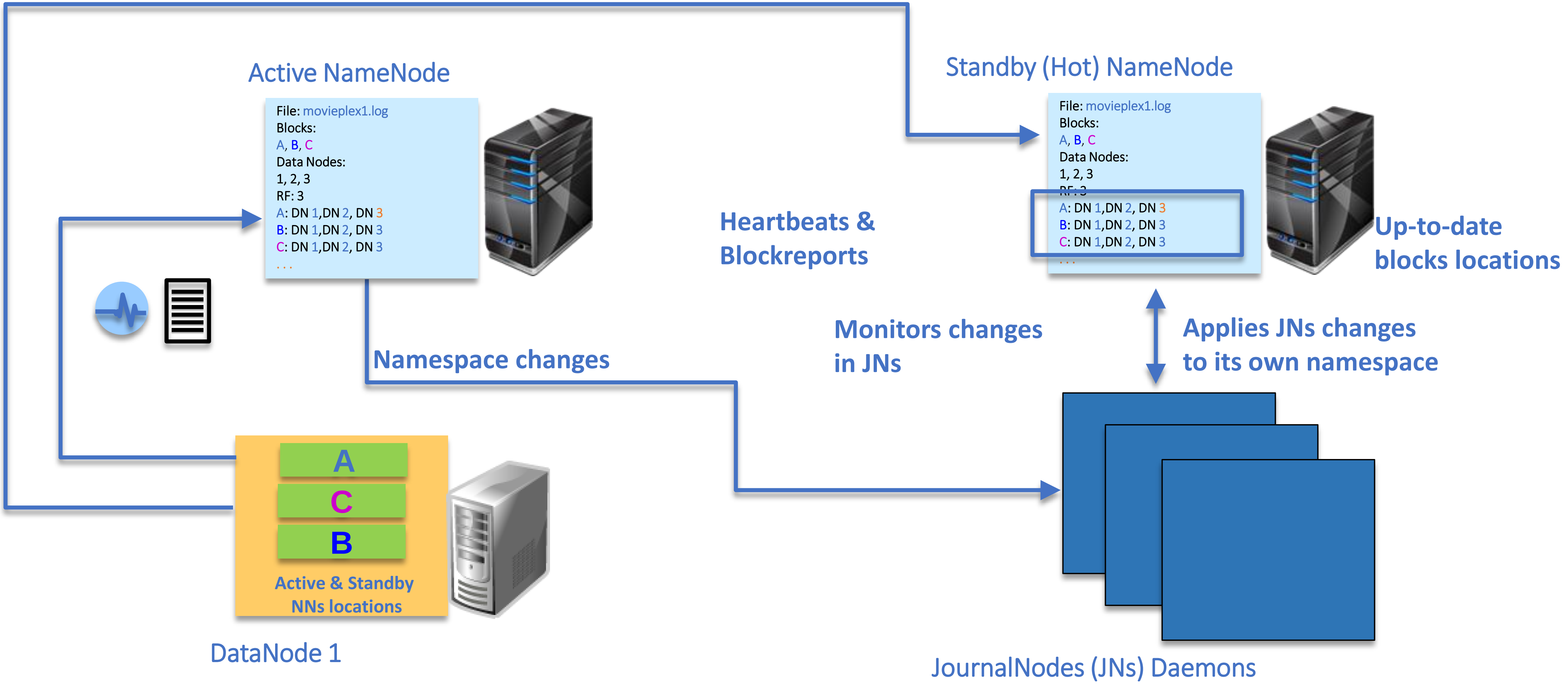
Writing a File to HDFS: Example



HDFS High Availability (HA) Using the Quorum Journal Manager (QJM)

- Prior to Hadoop 2.0.0, the NameNode was a single point of failure (SPOF) in an HDFS cluster.
- Each cluster had a single NameNode.
- The cluster is unavailable when the NameNode machine crashes or during software and hardware maintenance.
- HDFS HA addresses this problem by:
  - Running two redundant NameNodes in the same cluster:
    An **Active** NameNode and a **Hot Standby** NameNode
- HA provides fast failover to a new NameNode when the NameNode machine crashes or during regular software and hardware maintenance.
- Big Data Appliance (BDA) uses the HA implementation.

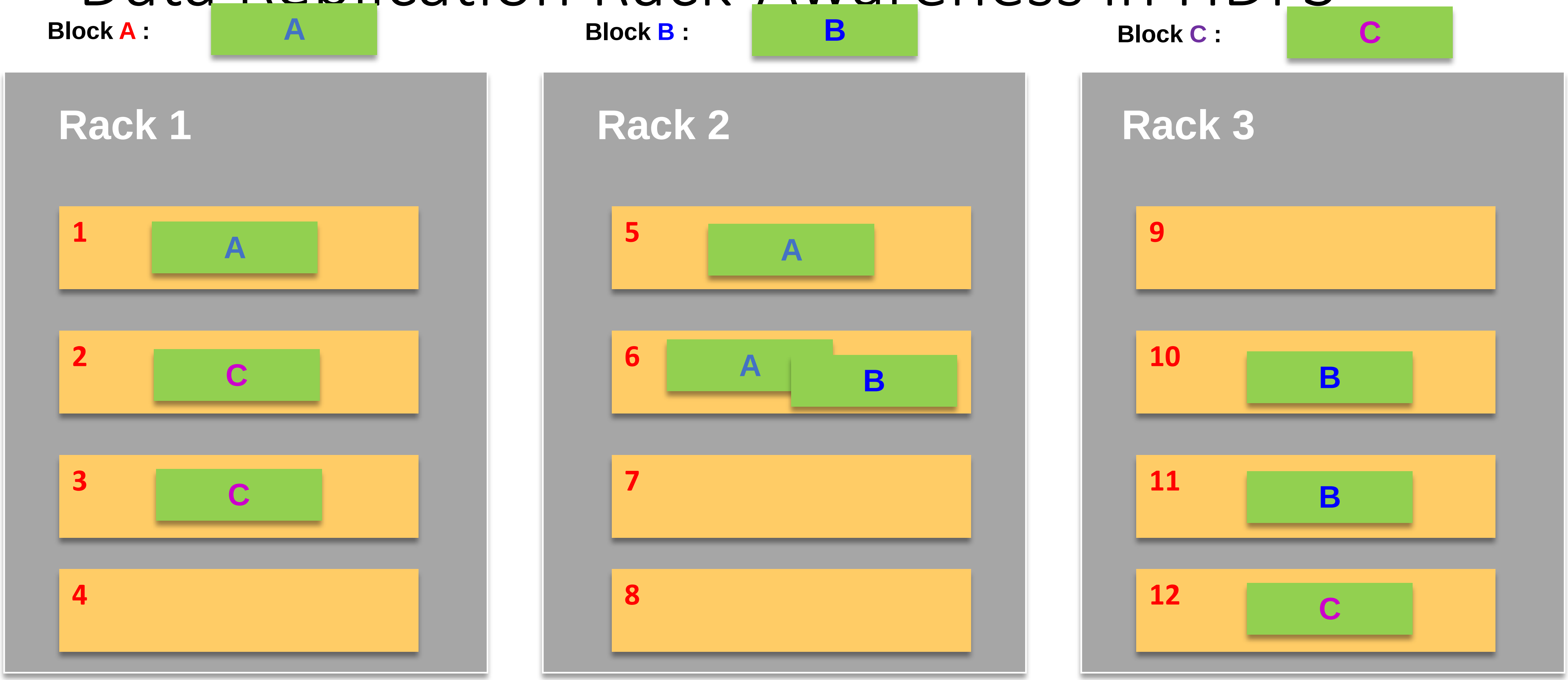
HDFS High Availability (HA) Using the Quorum Journal Manager (QJM) Feature



Enabling HDFS HA

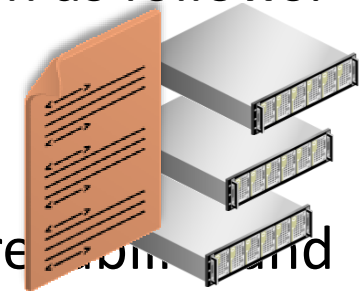
- Using Cloudera Manager:
  - Enable HA and Automatic Failover
- Using the command-line interface to configure automatic failover. Automatic failover adds the following components to an HDFS deployment:
  - A ZooKeeper quorum, which provides:
    - Failure detection
    - Active NameNode election
  - `ZKFailoverController` process (`ZKFC`), which provides:
    - Health monitoring
    - ZooKeeper session management
    - ZooKeeper-based election

Data Replication Rack-Awareness in HDFS

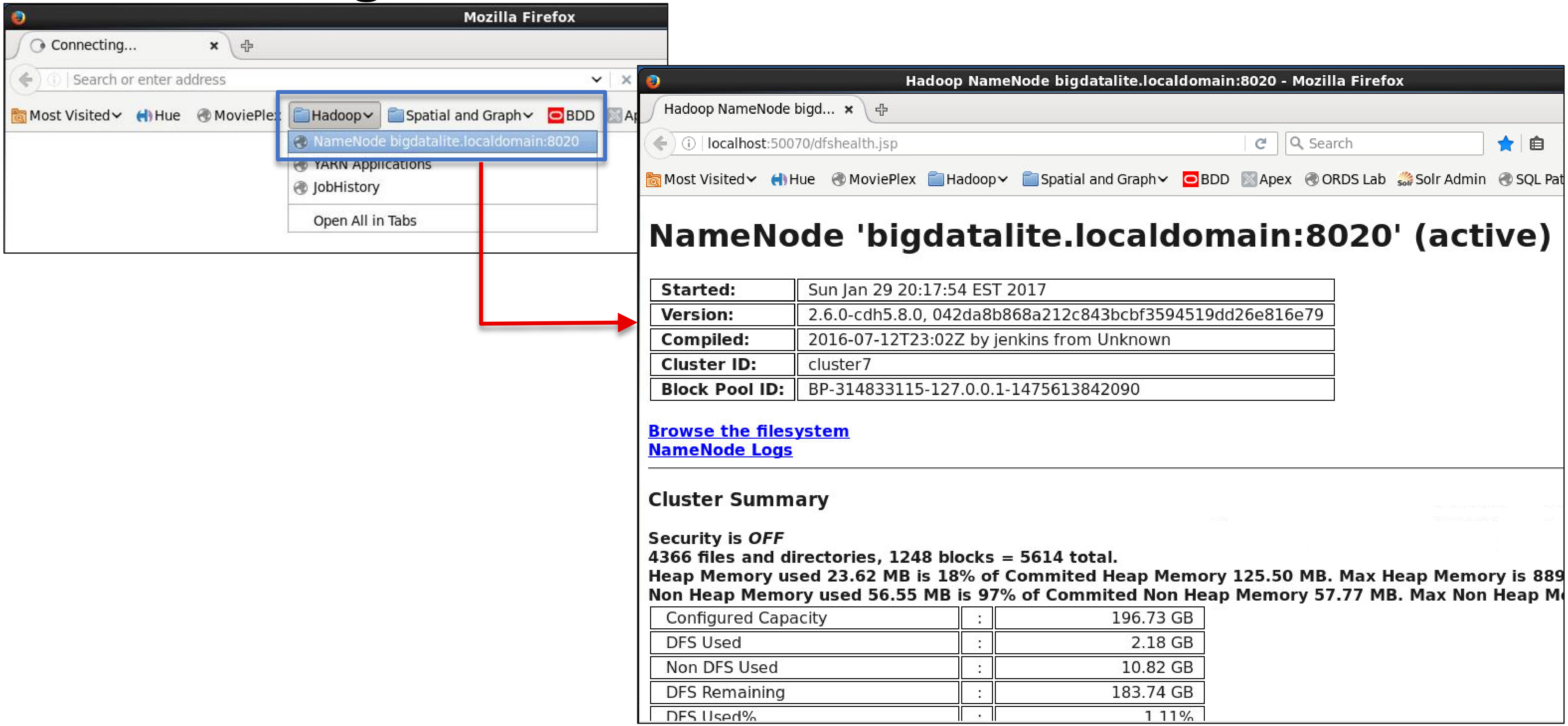


Data Replication Process

- The number of file replicas that will be maintained by HDFS (the “replication factor”) is stored in the NameNode.
  - If the factor is (3), the ***<u>HDFS Placement Policy</u>*** directs replication as follows:
    - One copy on one node in a local rack
    - One copy on a different remote rack
    - One copy on a different node in the same remote rack
  - This policy improves the write performance and ensures data reliability and availability.
  - *If the reader process requires data, HDFS makes sure that it pulls the nearest replica for the task, thereby reducing the read latency (data locality)*.



Accessing HDFS

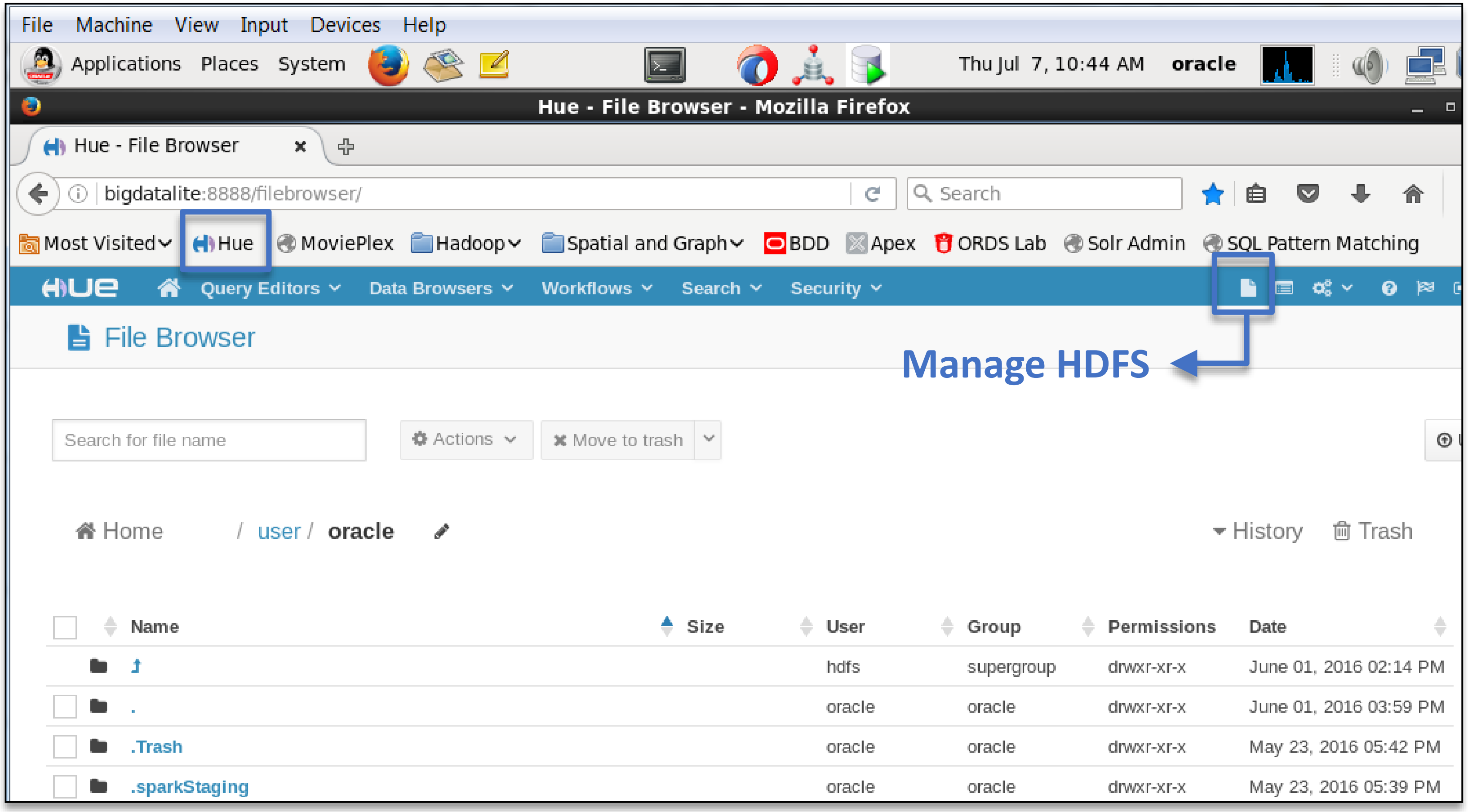


Agenda

- Understand the architectural components of HDFS
- Interact with data stored in HDFS
    - Hue
    - Hadoop client
    - WebHDFS
    - HttpFS

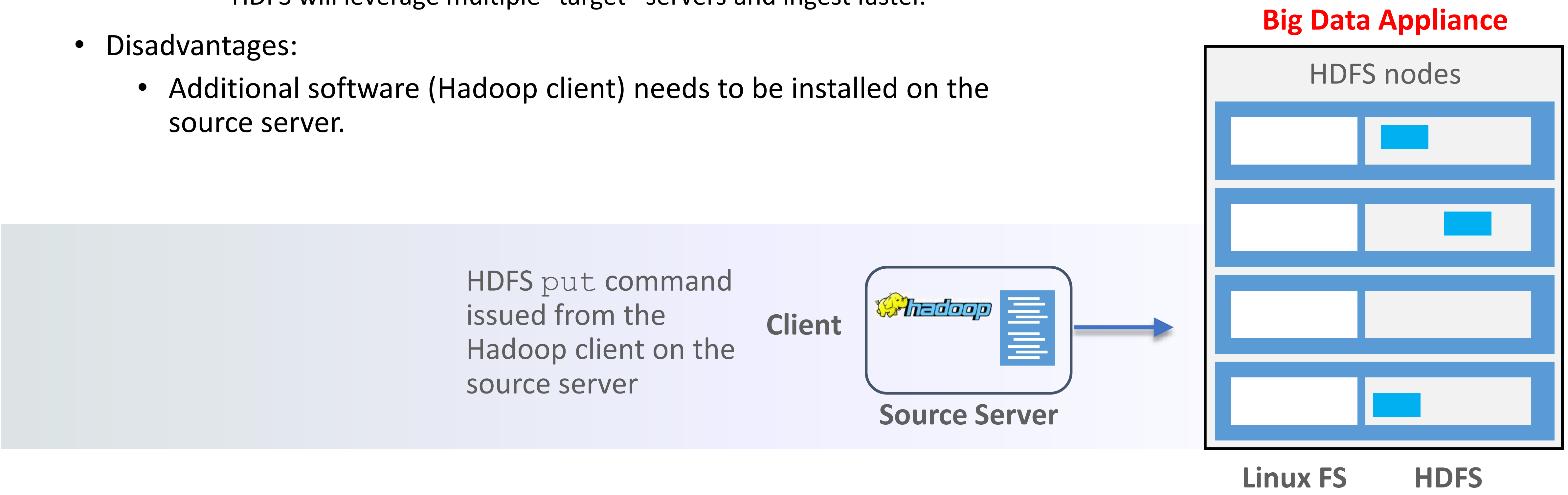
Using Cloudera Hue to Interact with HDFS

http://bda1node03.example.com:8888



Using Hadoop Client to Batch Load Data

- Advantages:
  - Enables direct HDFS writes without intermediate file staging on Linux FS
  - Easy to scale:
    - Initiate concurrent puts for multiple files.
    - HDFS will leverage multiple “target” servers and ingest faster.
- Disadvantages:
  - Additional software (Hadoop client) needs to be installed on the source server.

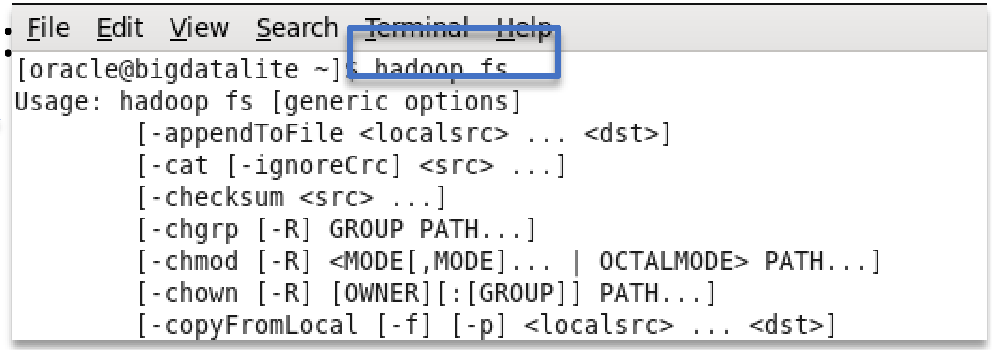


HDFS Commands

HDFS File System (FS) Shell Interface

- HDFS supports a traditional hierarchical file organization.
- You can use the **FS shell** command-line interface to interact with the data in HDFS.
- The syntax of this command set is similar to that of other shells.
  - You can create, remove, rename, and move directories/files.
- You can invoke FS shell as follows:

```
hadoop fs <args>
```



- The general command-line syntax is as follows:

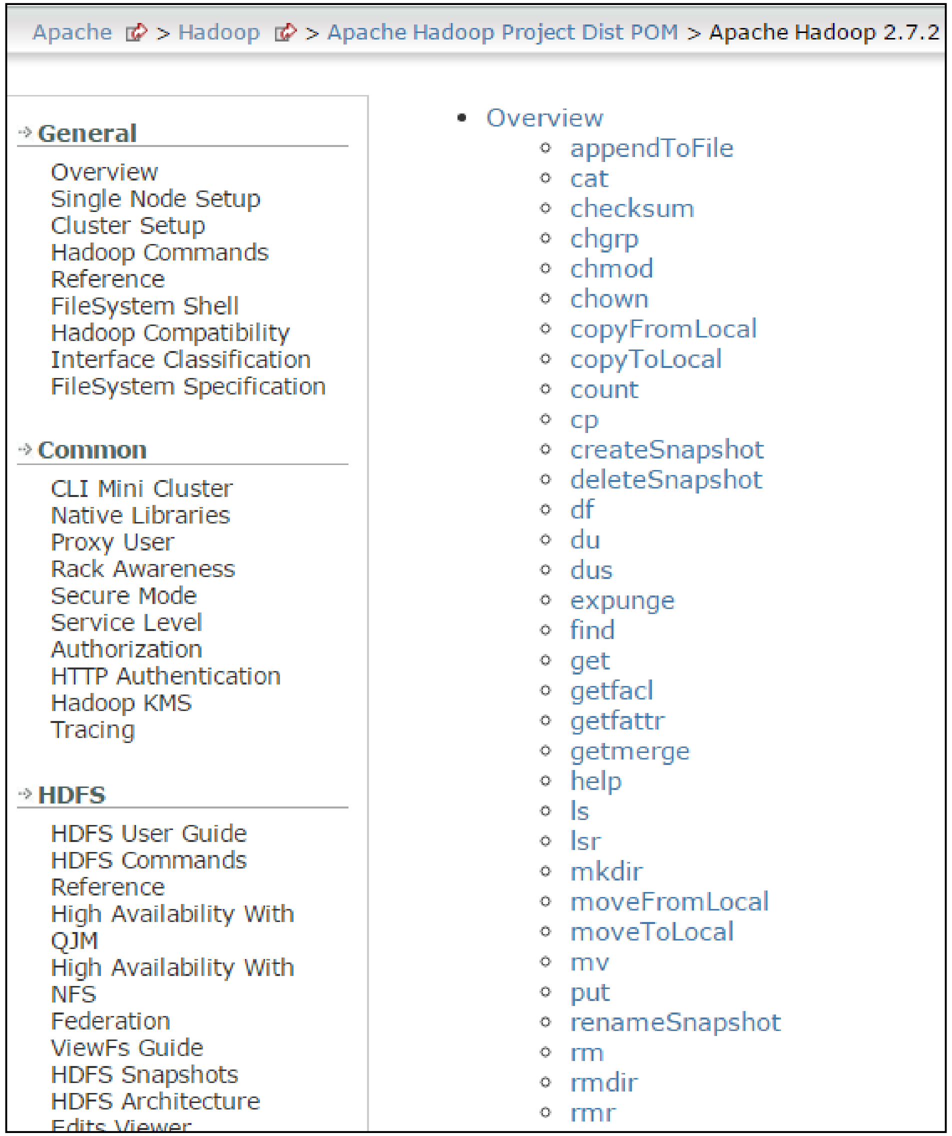
```
hadoop command [genericOptions] [commandOptions]
```

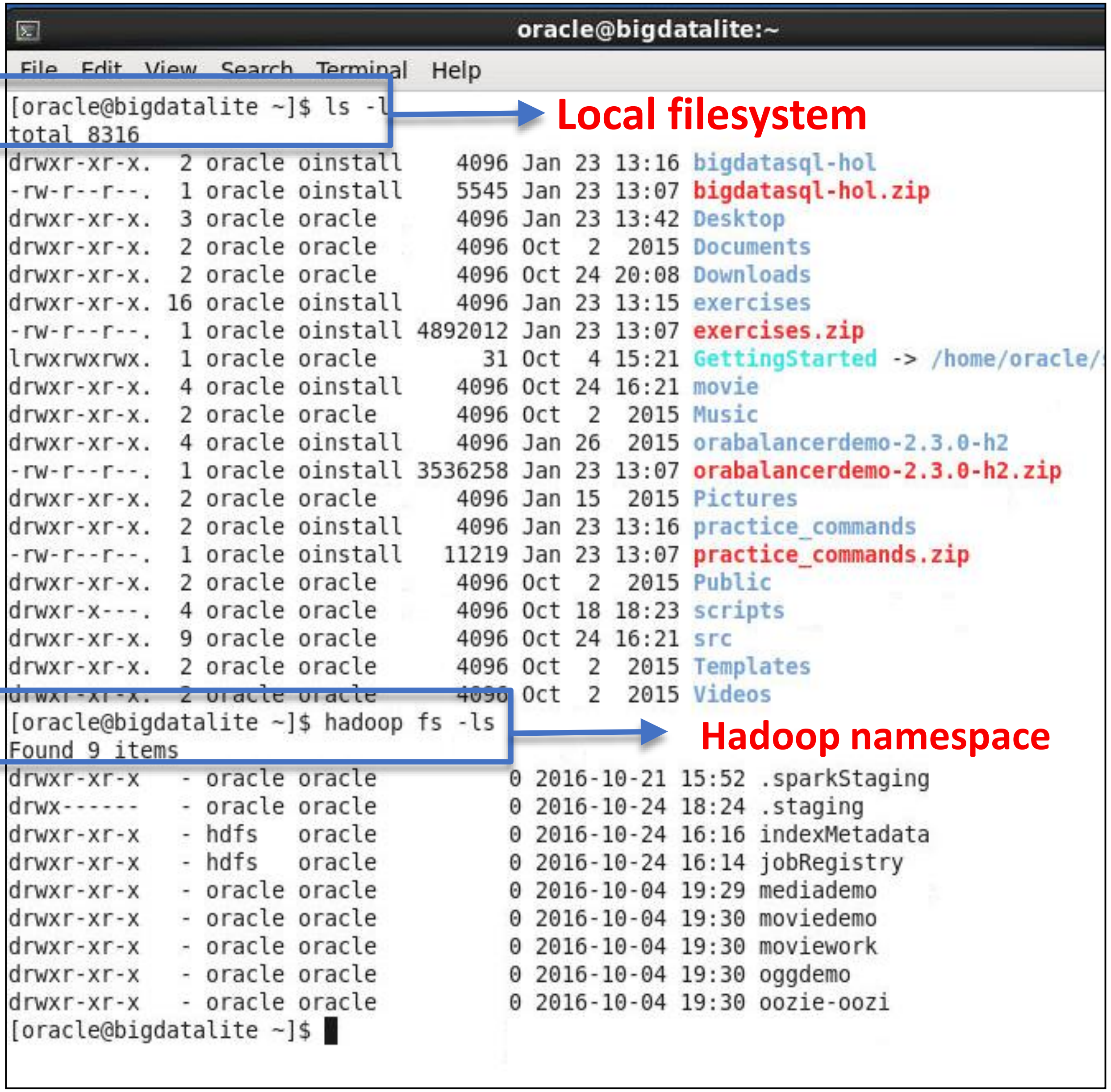
HDFS FS (File System) Shell Interface

```
hadoop fs -help
```

```
[oracle@bigdatalite ~]$ hadoop fs -help
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] <localsrc> ... <dst>]
        [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-count [-q] <path> ...]
        [-cp [-f] [-p] <src> ... <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] <path> ...]
        [-expunge]
        [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-getfacl [-R] <path>]
        [-getmerge [-nl] <src> <localdst>]
        [-help [cmd ...]]
        [-ls [-d] [-h] [-R] [<path> ...]]
        [-mkdir [-p] <path> ...]
        [-moveFromLocal <localsrc> ... <dst>]
        [-moveToLocal <src> <localdst>]
        [-mv <src> ... <dst>]
        [-put [-f] [-p] <localsrc> ... <dst>]
        [-renameSnapshot <snapshotDir> <oldName> <newName>]
        [-rm [-f] [-r|-R] [-skipTrash] <src> ...]
        [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
        [-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <acl_spec> <path>]]
```

FS Shell Commands





Sample FS Shell Commands

Command	Description
`ls`	Lists attributes of files and directories
`cat`	Copies source paths to `stdout`
`cp`	Copy files from source to destination in HDFS
`mv`	Moves files from source to destination. Moving files across file systems is not permitted.
`rm`	Deletes files specified. The `-r` option deletes the directory and its contents.
`put`	Copies files from the local file system to HDFS
`get`	Copies files from HDFS to the local file system
`mkdir`	Creates one or more HDFS directories
`rmdir`	Deletes a directory
`jar`	Runs a jar file. Users can bundle their MapReduce code in a JAR file and execute it using this command.
`version`	Prints the Hadoop version
`help`	Return usage output (available commands to use)

`ls` Command

```
hadoop fs -ls
```

```
oracle@bigdatalite:~
File  Edit  View  Search  Terminal  Help
[oracle@bigdatalite ~]$ hadoop fs -ls wordcount
Found 2 items
drwxr-xr-x   - oracle oracle          0 2015-03-10 02:45 wordcount/input
drwxr-xr-x   - oracle oracle          0 2015-03-10 04:09 wordcount/output
[oracle@bigdatalite ~]$ hadoop fs -ls wordcount/input
Found 2 items
-rw-r--r--   1 oracle oracle        518 2015-03-10 02:45 wordcount/input/file01
-rw-r--r--   1 oracle oracle        518 2015-03-10 02:45 wordcount/input/file02
[oracle@bigdatalite ~]$
```

directories

files

- For a file, it returns `stat` on the file with the following format:
  - `permissions number_of_replicas userid groupid filesize modification_date modification_time filename`
- For a directory, it returns a list of its direct children as in UNIX. A directory is listed as:
  - `permissions userid groupid modification_date modification_time dirname`

mkdir and copyFromLocal Commands

Create an HDFS directory named curriculum by using the mkdir command:

```
[oracle@bigdatalite ~]$ hadoop fs -mkdir curriculum
[oracle@bigdatalite ~]$ hadoop fs -ls
Found 9 items
drwx------   - oracle oracle          0 2014-08-25 05:55 .Trash
drwx------   - oracle oracle          0 2015-03-10 04:09 .staging
drwxr-xr-x   - oracle oracle          0 2015-03-24 09:38 curriculum
drwxr-xr-x   - oracle oracle          0 2014-01-12 18:15 moviedemo
drwxr-xr-x   - oracle oracle          0 2014-09-24 09:38 moviework
drwxr-xr-x   - oracle oracle          0 2014-09-08 15:50 oggdemo
drwxr-xr-x   - oracle oracle          0 2014-09-20 13:59 oozie-oozi
drwxr-xr-x   - oracle oracle          0 2015-03-24 00:57 test
drwxr-xr-x   - oracle oracle          0 2015-03-10 04:09 wordcount
[oracle@bigdatalite ~]$
```

Copy lab_05_01.txt from the local file system to the curriculum HDFS directory by using the copyFromLocal command:

```
[oracle@bigdatalite ~]$ cd Practice_Commands
[oracle@bigdatalite Practice_Commands]$ ls
lab_05_01.txt  lab_09_01.txt  lab_13_01.txt  lab_15_01.txt  lab_19_02.txt  lab_21_02.txt
lab_07_01.txt  lab_11_01.txt  lab_13_02.txt  lab_18_01.txt  lab_19_03.txt  lab_23_01.txt
lab_07_02.txt  lab_11_02.txt  lab_13_03.txt  lab_18_02.txt  lab_20_01.txt  lab_27_01.txt
lab_07_04.txt  lab_11_03.txt  lab_14_01.txt  lab_19_01.txt  lab_21_01.txt
[oracle@bigdatalite Practice_Commands]$ hadoop fs -copyFromLocal lab_05_01.txt curriculum/lab_05_01.txt
[oracle@bigdatalite Practice_Commands]$ hadoop fs -ls curriculum
Found 1 items
-rw-r--r--   1 oracle oracle        524 2015-03-24 10:14 curriculum/lab_05_01.txt
[oracle@bigdatalite Practice_Commands]$
```

`rm` and `cat` Commands

Delete the `curriculum` HDFS directory by using the `rm` command. Use the `-r` option to delete the directory and any content under it recursively:

```
[oracle@bigdatalite Practice_Commands]$ hadoop fs -rm -r curriculum
15/03/24 10:31:01 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interva
= 0 minutes.
Deleted curriculum
```

Display the contents of the `part-r-00000` HDFS file by using the `cat`

```
[oracle@bigdatalite ~]$ hadoop fs -cat  /user/oracle/wordcount/output/part-r-00000
and	12
awful	2
bank	2
company	4
cover	2
customer	6
disappointed	6
expensive	12
insurance	18
is	2
professional	2
protocols	2
service	12
staff	2
terrible	4
the	2
unreliable	6
very	6
with	4
worst	16
worthless	4
[oracle@bigdatalite ~]$
```

Using the `hdfs fsck` Command: Example

Use the `hdfs fsck` file system checking utility to perform health checks on the file system.

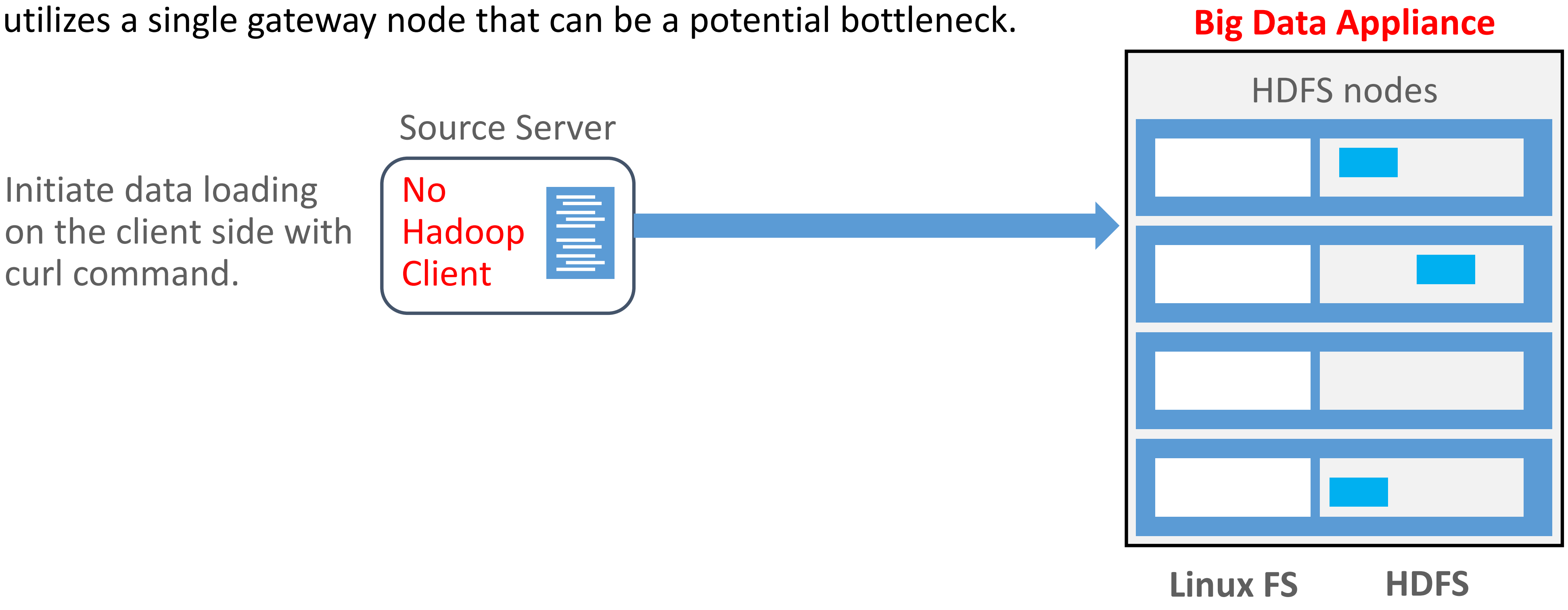
```
[oracle@bigdatalite ~]$ hdfs fsck  /user/oracle/wordcount/output/part-r-00000 -files -blocks
15/03/26 01:49:08 WARN ssl.FileBasedKeyStoresFactory: The property 'ssl.client.truststore.location' has no
t been set, no TrustStore will be loaded
Connecting to namenode via http://bigdatalite.localdomain:50070
FSCK started by oracle (auth:SIMPLE) from /127.0.0.1 for path /user/oracle/wordcount/output/part-r-00000 a
t Thu Mar 26 01:49:09 EDT 2015
/user/oracle/wordcount/output/part-r-00000 208 bytes, 1 block(s):  OK
0. BP-703742109-127.0.0.1-1398459391664:blk_1073754500_13678 len=208 repl=1

Status: HEALTHY
 Total size:    208 B
 Total dirs:    0
 Total files:   1
 Total symlinks:                0
 Total blocks (validated):      1 (avg. block size 208 B)
 Minimally replicated blocks:   1 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    1
 Average block replication:     1.0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          1
 Number of racks:               1
FSCK ended at Thu Mar 26 01:49:09 EDT 2015 in 0 milliseconds


The filesystem under path '/user/oracle/wordcount/output/part-r-00000' is HEALTHY
[oracle@bigdatalite ~]$ 
```

Agenda

- Understand the architectural components of HDFS
- Interact with data stored in HDFS
  - Hue
  - Hadoop client
  - WebHDFS
  - HttpFS

Loading Data with WebHDFS or HttpFS

- Advantages:
  - WebHDFS performance comparable with the Hadoop client
  - No additional software required on the client side
- Disadvantages:
  - Complex syntax (comparable with the Hadoop client)
  - HttpFS utilizes a single gateway node that can be a potential bottleneck.



hadoop fs -ls and LISTSTATUS

```
hadoop fs -ls
```

```
[oracle@bigdatalite ~]$ hadoop fs -ls
Found 8 items
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:42 .Trash
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:39 .sparkStaging
drwx------   - oracle oracle          0 2016-06-01 18:37 .staging
drwxr-xr-x   - oracle oracle          0 2016-06-01 18:59 mediademo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:02 moviedemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 moviework
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oggdemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oozie-oozi
[oracle@bigdatalite ~]$
```

```
curl -i
    "http://bigdatalite.localdomain:50070/webhdfs/v1/
    user/oracle?op=LISTSTATUS"
```

LISTSTATUS displays the same content of the `hadoop fs -ls` commend but in JSON format.


```
{"FileStatuses":{"FileStatus":[
{"accessTime":0,"blockSize":0,"childrenNum":1,"fileId":25974,"group":"oracle","l
ength":0,"modificationTime":1464050554815,"owner":"oracle","pathSuffix":".Trash"
,"permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":0,"fileId":24648,"group":"oracle","l
ength":0,"modificationTime":1464050368869,"owner":"oracle","pathSuffix":".sparkS
taging","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"}
,
{"accessTime":0,"blockSize":0,"childrenNum":0,"fileId":24580,"group":"oracle","l
ength":0,"modificationTime":1464820624005,"owner":"oracle","pathSuffix":".stagin
g","permission":"700","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":2,"fileId":68152,"group":"oracle","l
ength":0,"modificationTime":1464821985653,"owner":"oracle","pathSuffix":"mediade
mo","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":1,"fileId":17564,"group":"oracle","l
ength":0,"modificationTime":1463328175652,"owner":"oracle","pathSuffix":"moviede
mo","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":9,"fileId":17572,"group":"oracle","l
ength":0,"modificationTime":1463328181497,"owner":"oracle","pathSuffix":"moviewo
rk","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":1,"fileId":17611,"group":"oracle","l
ength":0,"modificationTime":1463328181552,"owner":"oracle","pathSuffix":"oggdemo
","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"},
{"accessTime":0,"blockSize":0,"childrenNum":0,"fileId":17615,"group":"oracle","l
ength":0,"modificationTime":1463328181651,"owner":"oracle","pathSuffix":"oozie-o
ozi","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"}
]}}
```

Uploading a Local File to an HDFS Directory with `hadoop fs`

Create an HDFS directory named `test11` using hadoop fs CLI:

```
[oracle@bigdatalite ~]$ hadoop fs -mkdir test11
[oracle@bigdatalite ~]$ hadoop fs -ls
Found 10 items
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:42 .Trash
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:39 .sparkStaging
drwx------   - oracle oracle          0 2016-06-01 18:37 .staging
drwxr-xr-x   - oracle oracle          0 2016-07-08 13:40 lauran
drwxr-xr-x   - oracle oracle          0 2016-06-01 18:59 mediademo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:02 moviedemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 moviework
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oggdemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oozie-oozi
drwxr-xr-x   - oracle oracle          0 2016-07-08 13:52 test11
[oracle@bigdatalite ~]$
```

Copying the local `test1.txt` file to HDFS directory `test11` using `hadoop fs` CLI:

```
hadoop fs -put test1.txt
    hdfs://bigdatalite.localdomain:8020/user/oracle/test11
```

```
[oracle@bigdatalite ~]$ hadoop fs -put test1.txt hdfs://bigdatalite.localdomain:
8020/user/oracle/test11
[oracle@bigdatalite ~]$ hadoop fs -ls test11
Found 1 items
-rw-r--r--   1 oracle oracle         16 2016-07-08 14:03 test11/test1.txt
[oracle@bigdatalite ~]$ hadoop fs -cat test11/test1.txt
This is test1.
[oracle@bigdatalite ~]$
```

Confirm file upload and view its content

Creating an HDFS Directory with WebHDFS

Creating an HDFS directory named `test21` by using WebHDFS:

```
curl -i -X PUT -L -H 'Content-Type:application/octet-stream'
"http://bigdatalite.localdomain:50070/webhdfs/v1/user/oracle/test21?op=
MKDIRS&user.name=oracle";
```

```
[oracle@bigdatalite ~]$ curl -i -X PUT -L -H 'Content-Type:application/octet-str
eam' "http://bigdatalite.localdomain:50070/webhdfs/v1/user/oracle/test21?op=MKDI
RS&user.name=oracle";
HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Fri, 08 Jul 2016 18:16:16 GMT
Date: Fri, 08 Jul 2016 18:16:16 GMT
Pragma: no-cache
Expires: Fri, 08 Jul 2016 18:16:16 GMT
Date: Fri, 08 Jul 2016 18:16:16 GMT
Pragma: no-cache
Content-Type: application/json
Set-Cookie: hadoop.auth="u=oracle&p=oracle&t=simple&e=1468037776103&s=3/Lz7/Bx0F
YL5SrugnxwayQFk5I="; Path=/; HttpOnly
Transfer-Encoding: chunked
Server: Jetty(6.1.26.cloudera.4)

{"boolean":true}[oracle@hadoop fs -ls
Found 11 items
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:42 .Trash
drwxr-xr-x   - oracle oracle          0 2016-05-23 20:39 .sparkStaging
drwx------   - oracle oracle          0 2016-06-01 18:37 .staging
drwxr-xr-x   - oracle oracle          0 2016-07-08 13:40 lauran
drwxr-xr-x   - oracle oracle          0 2016-06-01 18:59 mediademo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:02 moviedemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 moviework
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oggdemo
drwxr-xr-x   - oracle oracle          0 2016-05-15 12:03 oozie-oozi
drwxr-xr-x   - oracle oracle          0 2016-07-08 14:03 test11
drwxr-xr-x   - oracle oracle          0 2016-07-08 14:16 test21
[oracle@bigdatalite ~]$
```

Uploading a Local File to HDFS with WebHDFS

Creating an HDFS directory named `test21` by using WebHDFS:

```
curl -i -X PUT -L -H 'Content-Type:application/octet-stream'
"http://bigdatalite.localdomain:50070/webhdfs/v1/user/oracle/test21/tes
t1.txt?op=CREATE&user.name=oracle" -T test1.txt;
```

```
[oracle@bigdatalite ~]$ curl -i -X PUT -L -H 'Content-Type:application/octet-str
eam' "http://bigdatalite.localdomain:50070/webhdfs/v1/user/oracle/test21/test1.t
xt?op=CREATE&user.name=oracle" -T test1.txt;
HTTP/1.1 100 Continue

HTTP/1.1 307 TEMPORARY_REDIRECT
Cache-Control: no-cache
Expires: Fri, 08 Jul 2016 18:32:56 GMT
Date: Fri, 08 Jul 2016 18:32:56 GMT
Pragma: no-cache
Expires: Fri, 08 Jul 2016 18:32:56 GMT
Date: Fri, 08 Jul 2016 18:32:56 GMT
Pragma: no-cache
Set-Cookie: hadoop.auth="u=oracle&p=oracle&t=simple&e=1468038776897&s=L3iMT04D59
QuXkKU7UtgdVVnx44="; Path=/; HttpOnly
Location: http://bigdatalite.localdomain:50075/webhdfs/v1/user/oracle/test21/tes
t1.txt?op=CREATE&user.name=oracle&namenoderpcaddress=bigdatalite.localdomain:802
0&overwrite=false
Content-Type: application/octet-stream
Content-Length: 0
Server: Jetty(6.1.26.cloudera.4)

HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Location: hdfs://bigdatalite.localdomain:8020/user/oracle/test21/test1.txt
Content-Length: 0
Connection: close

[oracle@bigdatalite ~]$ hadoop fs -ls test21
Found 1 items
-rwxr-xr-x   1 oracle oracle         16 2016-07-08 14:32 test21/test1.txt
[oracle@bigdatalite ~]$
```

Creating an HDFS Directory and Loading Data by Using HttpFS

Creating an HDFS directory named `test31` by using HttpFS and uploading `test1.txt` to `/test31` HDFS directory

```
curl -i -X PUT -L -H 'Content-Type:application/octet-stream'
"http://bigdatalite.localdomain:14000/webhdfs/v1/user/oracle/test31/tes
t1.txt?op=CREATE&user.name=oracle" -T test1.txt;
```

```
[oracle@bigdatalite ~]$ curl -i -X PUT -L -H 'Content-Type:application/octet-str
eam' "http://bigdatalite.localdomain:14000/webhdfs/v1/user/oracle/test31/test1.t
xt?op=CREATE&user.name=oracle" -T test1.txt;
HTTP/1.1 100 Continue

HTTP/1.1 307 Temporary Redirect
Server: Apache-Coyote/1.1
Set-Cookie: hadoop.auth="u=oracle&p=oracle&t=simple-dt&e=1468040113065&s=rqBP4kl
EJMUyMa68Y71BLSbMHvc="; Path=/; HttpOnly
Location: http://bigdatalite.localdomain:14000/webhdfs/v1/user/oracle/test31/tes
t1.txt?op=CREATE&data=true&user.name=oracle
Content-Type: application/json
Content-Length: 0
Date: Fri, 08 Jul 2016 18:55:13 GMT

HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
Set-Cookie: hadoop.auth="u=oracle&p=oracle&t=simple-dt&e=1468040113091&s=HVP9fk8
EZEPYkoyLn6nK2i2qImE="; Path=/; HttpOnly
Content-Type: application/json
Content-Length: 0
Date: Fri, 08 Jul 2016 18:55:13 GMT

[oracle@bigdatalite ~]$ hadoop fs -ls test31
Found 1 items
-rwxr-xr-x   1 oracle oracle         16 2016-07-08 14:55 test31/test1.txt
(reverse-i-search)`':
```

HttpFS uses default port 14000

Summary

- In this lesson, you should have learned how to:
  - Describe the architectural components of HDFS
  - Use the **FS shell** command-line interface (CLI) to interact with data stored in HDFS

Practice 5: Overview

- In this practice, you load a JSON log file into HDFS. This log file was used to track activity in an online movie application.

Acquiring Data by Using CLI, Fuse DFS, Flume, and Kafka

Course Road Map

Module 1: Big Data Fundamentals

Module 2: Data Acquisition and Storage

Module 3: Data Access and Processing

Module 4: Data Unification

Module 5: Data Analysis

Module 6: Big Data Deployment Options

Lesson 5: Introduction to the Hadoop Distributed File System (HDFS)

Lesson 6: Acquiring Data by Using CLI, Fuse DFS, Flume, and Kafka

Lesson 07: Acquiring and Accessing Data by Using Oracle NoSQL Database

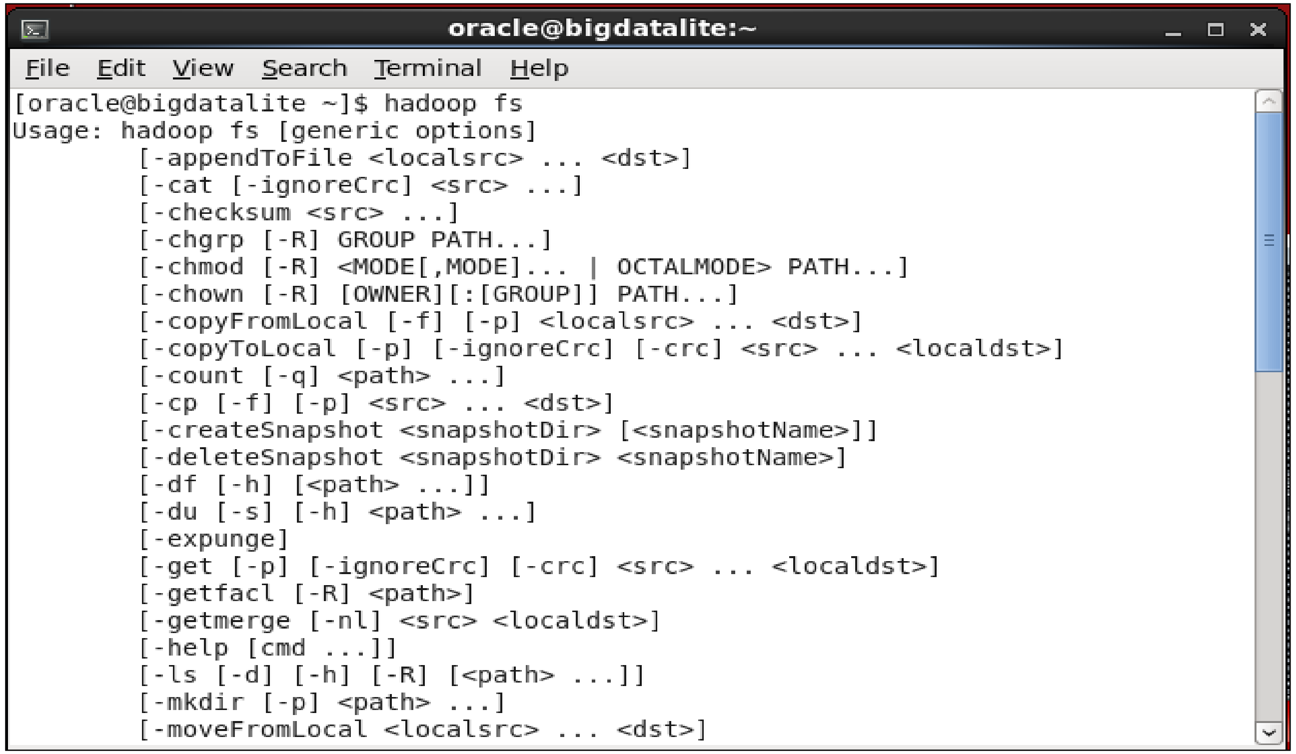
Objectives

- After completing this lesson, you should be able to describe:
  - Uses of the command-line interface (CLI)
  - Benefits of Fuse DFS
  - Architecture and features of Flume
  - Features and uses of Kafka

Reviewing the Command-Line Interface (CLI)

- View usage and generic options for `hadoop fs`:

```
$ hadoop fs
```




```
[oracle@bigdatalite ~]$ hadoop fs
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] <localsrc> ... <dst>]
        [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-count [-q] <path> ...]
        [-cp [-f] [-p] <src> ... <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] <path> ...]
        [-expunge]
        [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-getfacl [-R] <path>]
        [-getmerge [-nl] <src> <localdst>]
        [-help [cmd ...]]
        [-ls [-d] [-h] [-R] [<path> ...]]
        [-mkdir [-p] <path> ...]
        [-moveFromLocal <localsrc> ... <dst>]
```

Viewing File System Contents by Using the CLI

```
[oracle@bigdatalite ~]$ hadoop fs -ls /user/oracle/moviework/applog_json
Found 1 items
-rw-r--r--   1 oracle oracle   32557883 2014-09-13 13:39 /user/oracle/moviework/applog_json/movieapp_log_json.log
[oracle@bigdatalite ~]$ hadoop fs -tail /user/oracle/moviework/applog_json/movieapp_log_json.log
,"recommended":"Y","activity":2}
{"custid":1135508,"movieid":240,"genreid":8,"time":"2012-10-01:02:04:15","recommended":"Y","activity":5}
{"custid":1135508,"movieid":1092,"genreid":20,"time":"2012-10-01:02:10:23","recommended":"N","activity":5}
{"custid":1135508,"movieid":4638,"genreid":8,"time":"2012-10-01:02:10:54","recommended":"N","activity":7}
{"custid":1135508,"movieid":4638,"genreid":8,"time":"2012-10-01:02:16:49","recommended":"N","activity":7}
{"custid":1135508,"movieid":null,"genreid":null,"time":"2012-10-01:02:24:00","recommended":null,"activity":9}
{"custid":1135508,"movieid":240,"genreid":8,"time":"2012-10-01:02:31:12","recommended":"Y","activity":11,"price":2.99}
{"custid":1191532,"movieid":59440,"genreid":7,"time":"2012-10-01:03:11:35","recommended":"Y","activity":2}
{"custid":1191532,"movieid":null,"genreid":null,"time":"2012-10-01:03:15:29","recommended":null,"activity":9}
{"custid":1191532,"movieid":59440,"genreid":7,"time":"2012-10-01:03:19:24","recommended":"Y","activity":11,"price":3.99}
```

Loading Data by Using the CLI

- Put files into HDFS:

```
$ hadoop fs –put ?site.xml /u01/bigdatasql_config/bigatalite
```

What Is Fuse DFS?

- Enables access to HDFS as if it were just a file system folder

1. Install `hadoop-hdfs-fuse` on your file system.
2. Set up and test your mount point (non-HA installation).

```
$ mkdir -p <mount_point>
$ hadoop-fuse-dfs dfs://<name_node_hostname>:<namenode_port> <mount_point>
```

3. Clean up the test.

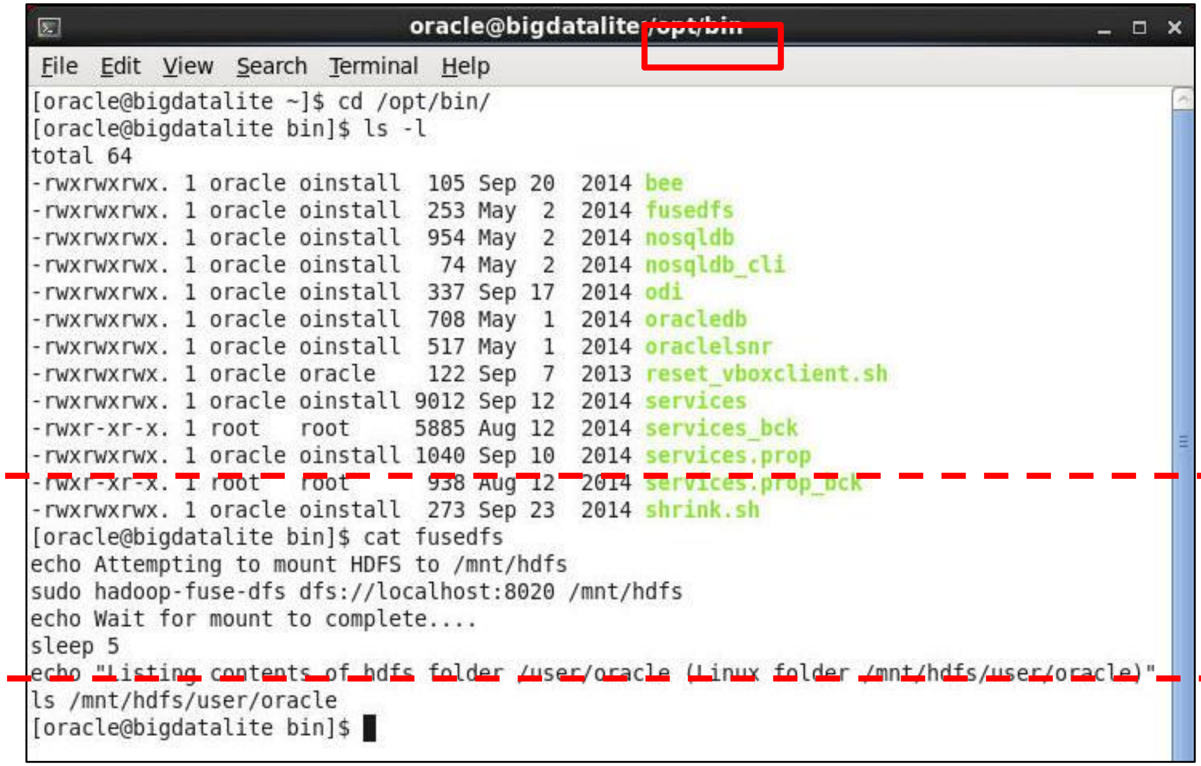
4. To add the system mount: open `/etc/fstab` and add lines to the bottom similar to these:

```
$ umount <mount_point>
```

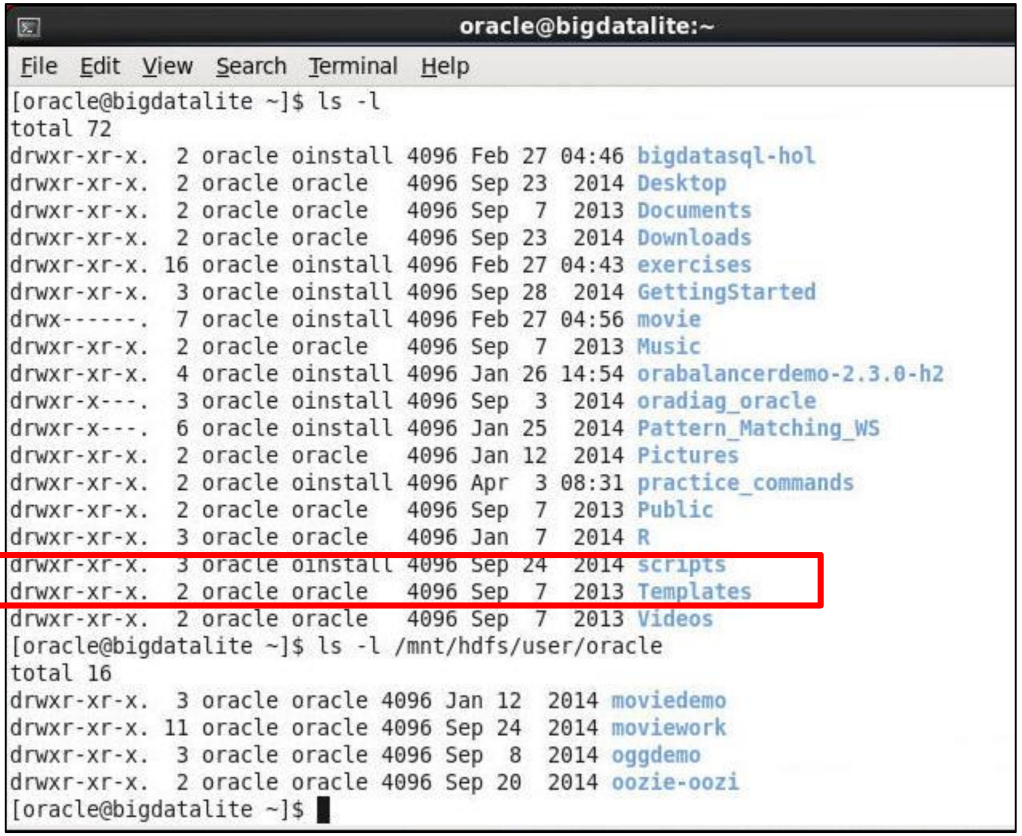
```
hadoop-fuse-dfs#dfs://<name_node_hostname>:<namenode_port> <mount_point>
fuse allow_other,usetrash,rw 2 0
```

- Result: Your system is now configured to allow you to use the `ls` command and use that mount point as if it were a normal system disk.

Enabling Fuse DFS on Big Data Lite



Using Fuse DFS

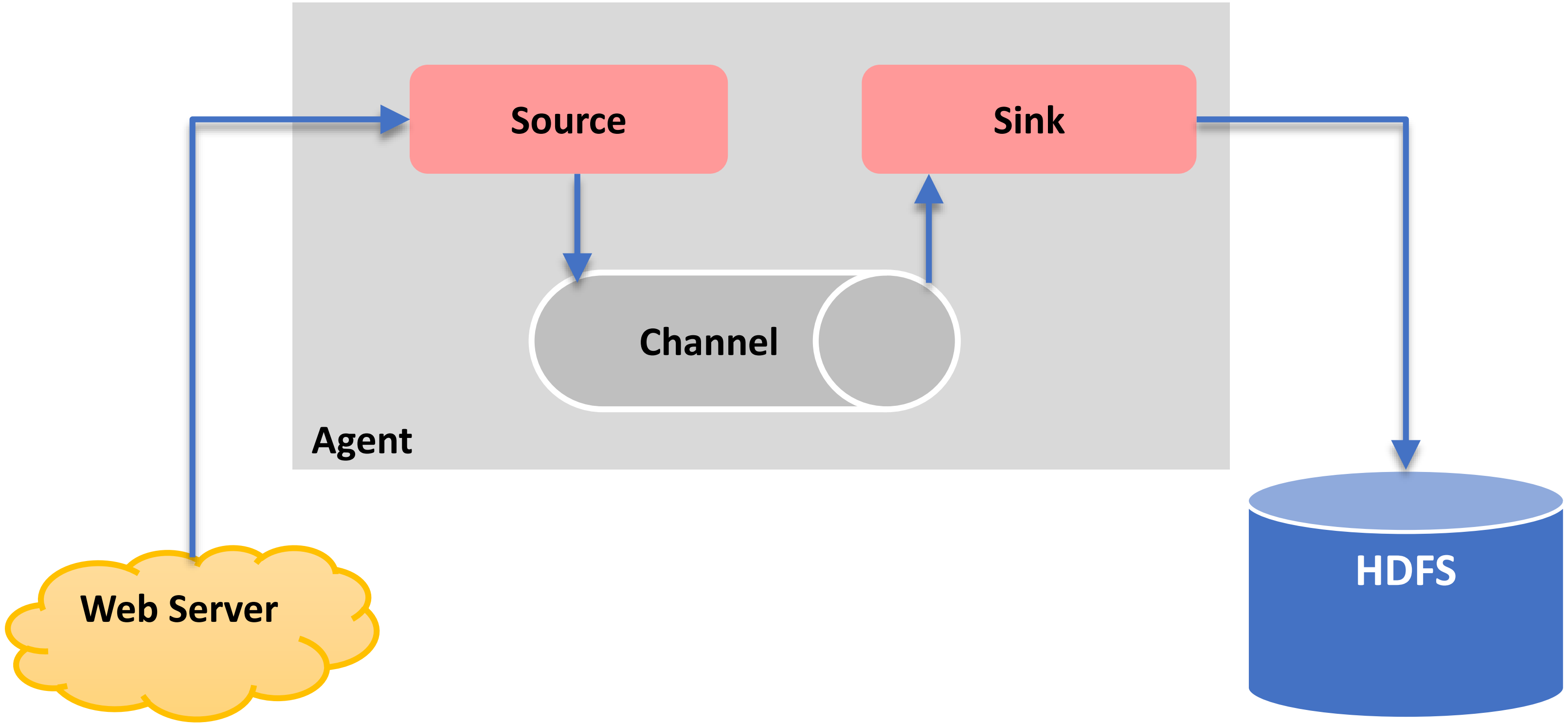


```
oracle@bigdatalite:~
File Edit View Search Terminal Help
[oracle@bigdatalite ~]$ ls -l
total 72
drwxr-xr-x.  2 oracle oinstall 4096 Feb 27 04:46 bigdatasql-hol
drwxr-xr-x.  2 oracle oracle   4096 Sep 23  2014 Desktop
drwxr-xr-x.  2 oracle oracle   4096 Sep  7  2013 Documents
drwxr-xr-x.  2 oracle oracle   4096 Sep 23  2014 Downloads
drwxr-xr-x. 16 oracle oinstall 4096 Feb 27 04:43 exercises
drwxr-xr-x.  3 oracle oinstall 4096 Sep 28  2014 GettingStarted
drwx------.  7 oracle oinstall 4096 Feb 27 04:56 movie
drwxr-xr-x.  2 oracle oracle   4096 Sep  7  2013 Music
drwxr-xr-x.  4 oracle oinstall 4096 Jan 26 14:54 orabalancerdemo-2.3.0-h2
drwxr-x---.  3 oracle oinstall 4096 Sep  3  2014 oradiag_oracle
drwxr-x---.  6 oracle oinstall 4096 Jan 25  2014 Pattern_Matching_WS
drwxr-xr-x.  2 oracle oracle   4096 Jan 12  2014 Pictures
drwxr-xr-x.  2 oracle oinstall 4096 Apr  3 08:31 practice_commands
drwxr-xr-x.  2 oracle oracle   4096 Sep  7  2013 Public
drwxr-xr-x.  3 oracle oracle   4096 Jan  7  2014 R
drwxr-xr-x.  3 oracle oinstall 4096 Sep 24  2014 scripts
drwxr-xr-x.  2 oracle oracle   4096 Sep  7  2013 Templates
drwxr-xr-x.  2 oracle oracle   4096 Sep  7  2013 Videos
[oracle@bigdatalite ~]$ ls -l /mnt/hdfs/user/oracle
total 16
drwxr-xr-x.  3 oracle oracle 4096 Jan 12  2014 moviedemo
drwxr-xr-x. 11 oracle oracle 4096 Sep 24  2014 moviework
drwxr-xr-x.  3 oracle oracle 4096 Sep  8  2014 oggdemo
drwxr-xr-x.  2 oracle oracle 4096 Sep 20  2014 oozie-oozi
[oracle@bigdatalite ~]$
```

What Is Flume?

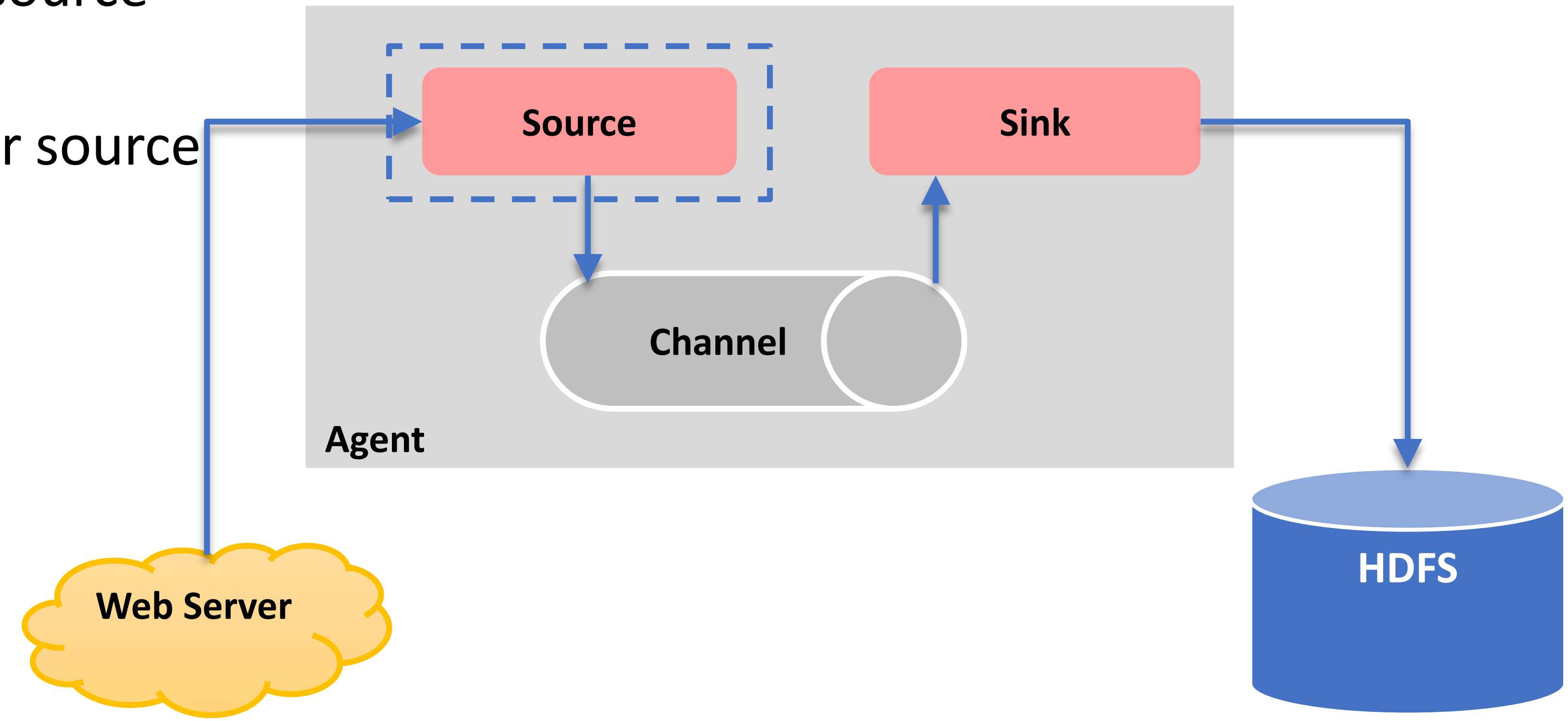
- Is a distributed service for collecting, aggregating, and moving large data to a centralized data store
- Was developed by Apache
- Has the following features:
  - Simple
  - Reliable
  - Fault tolerant
  - Used for online analytic applications

Flume: Architecture



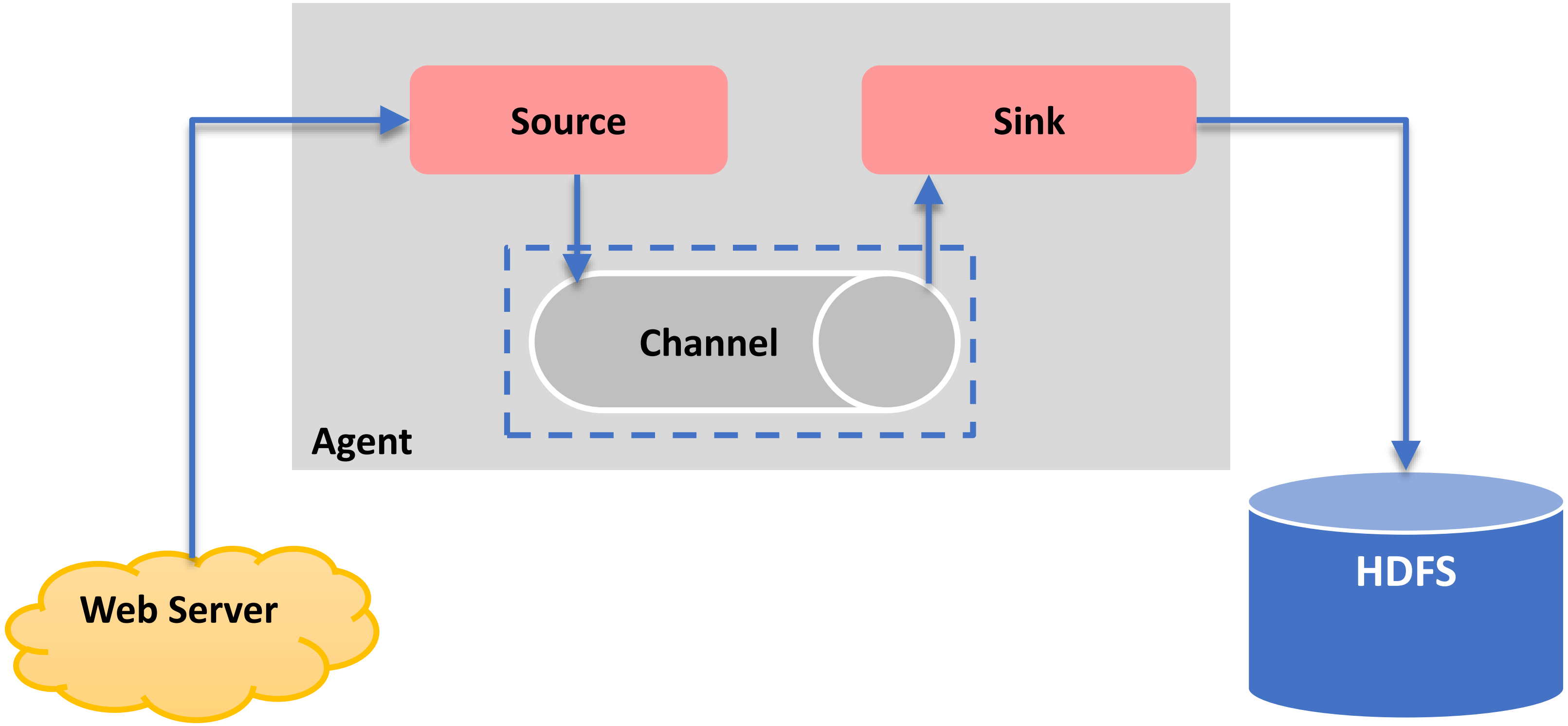
Flume Sources (Consume Events)

- Avro source
- Exec source
- Spooling Directory source
- Netcat source
- Sequence Generator source
- Syslog source
- HTTP source
- Custom source
- Scribe source



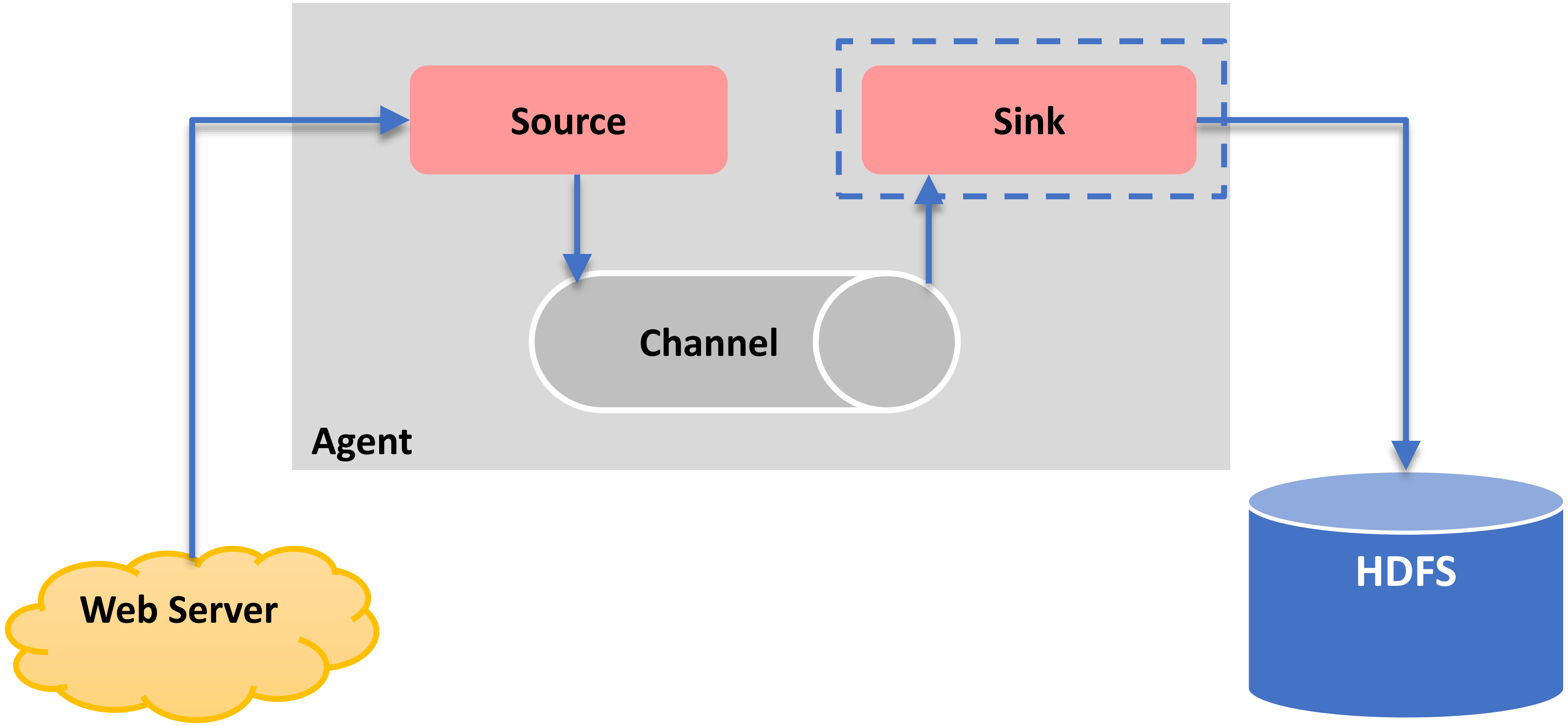
Flume Channels (Hold Events)

- Memory channel
- JDBC channel
- File channel
- Custom channel

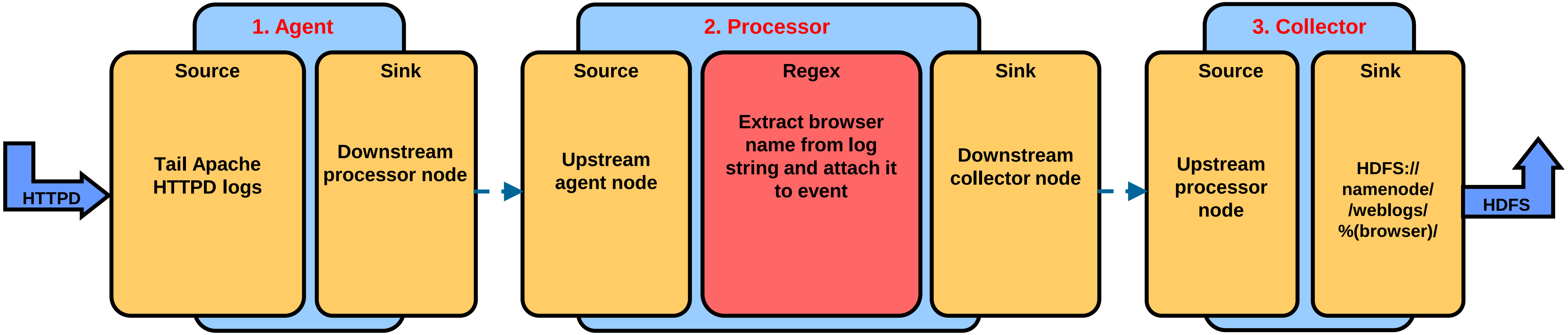


Flume Sinks (Deliver Events)

- HDFS sink
- Logger sink
- Avro sink
- IRC sink
- File Roll sink
- Null sink
- HBase sink
- AsyncHBaseSink
- ElasticSearchSink
- Custom sink



Flume: Data Flows



Configuring Flume

1. Create a configuration file (`flume.conf`).
2. Store the file in the `flume-ng/conf` directory.
3. Configure individual components.
4. (Optional) Edit `flume-env.sh`.
5. Verify the installation by running the following command:

```
$ flume-ng help
```

Exploring a `flume*.conf` File

```
[oracle@bigdatalite ~]$ cd /home/oracle/movie/moviedemo/scripts
[oracle@bigdatalite scripts]$ ls
1_start_movieapp.sh        flume_avro.conf       reset_nosqldb.sh
2_flume_tail_movielog.sh   flume_json.conf       tail_log.sh
3_nosqldb_console.sh       flume_movieagent.sh
[oracle@bigdatalite scripts]$ more flume_avro.conf
# Chain:  log -> json-to-avro interceptor -> memory channel -> hdfs sink
movieagent.sources=logFile
movieagent.channels=memoryChannel
movieagent.sinks=hdfs-sink

# Application log source
movieagent.sources.logFile.type=exec
movieagent.sources.logFile.command=tail -F /u01/Middleware/logs/activity.out
movieagent.sources.logFile.channels=memoryChannel
movieagent.sources.logFile.interceptors=jsonToAvro
movieagent.sources.logFile.interceptors.jsonToAvro.type=oracle.avro.ActivityJson
ToAvroInterceptor$Builder
movieagent.sources.logFile.interceptors.jsonToAvro.key=flume.avro.schema.url
movieagent.sources.logFile.interceptors.jsonToAvro.value=hdfs://bigdatalite.loca
ldomain/user/oracle/moviework/schemas/activity.avsc


# Memory Channel
movieagent.channels.memoryChannel.type=memory
movieagent.channels.memoryChannel.capacity=100

# HDFS Sink
movieagent.sinks.hdfs-sink.type=hdfs
movieagent.sinks.hdfs-sink.hdfs.path=hdfs://bigdatalite.localdomain/user/oracle/
moviework/applog_avro
movieagent.sinks.hdfs-sink.hdfs.filePrefix=streamed-movieapp
movieagent.sinks.hdfs-sink.hdfs.fileType=DataStream
movieagent.sinks.hdfs-sink.hdfs.rollInterval=60
movieagent.sinks.hdfs-sink.hdfs.fileSuffix=.avro
movieagent.sinks.hdfs-sink.channel=memoryChannel
movieagent.sinks.hdfs-sink.serializer=org.apache.flume.sink.hdfs.AvroEventSerial
izer$Builder
#movieagent.sinks.hdfs-sink.serializer.compressionCodec=snappy
[oracle@bigdatalite scripts]$
```

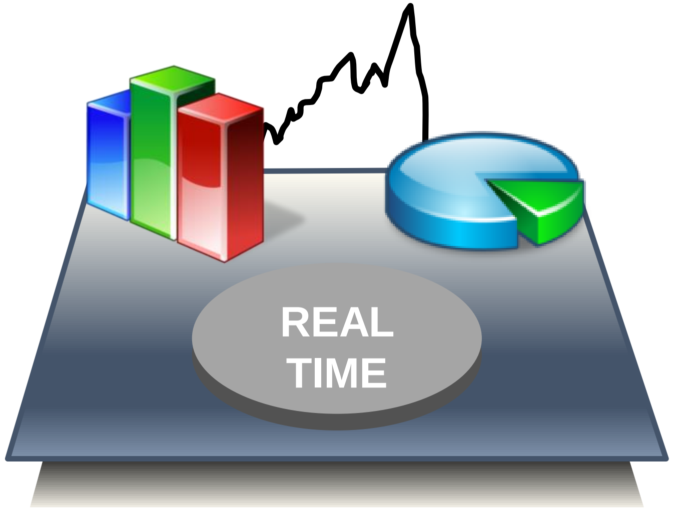
What Is Apache Kafka?

- It is a messaging system that collects and delivers high volumes of activity log data with low latency.
- Developed at LinkedIn, Kafka can handle over 500 billion events per day, spread over a number of data centers.
- In contrast to Flume, Kafka is about the online consumption of log data.
- Unlike traditional offline log processing systems, Kafka is designed to process log data in real-time.
- It enables data transfer between disparate data systems (such as relational, Hadoop, or NoSQL) that are geographically distributed.

APACHE kafka
A distributed streaming platform

Activity Log Processing in Real Time

- For Internet-based applications, log data is now required in real time.
- It enables user-oriented features such as:
  - Search
  - Ad targeting and reporting
  - Recommendations
  - Security against spam or unauthorized usage
  - Fraud detection
  - Newsfeed updates
- Log data is important for smooth running of the website, not just for offline log analysis.



Apache Kafka: Features

Publish and Subscribe

to streams of data like a messaging system.

Process

streams of data efficiently and in real time.

Store

streams of data safely in a distributed replicated cluster.

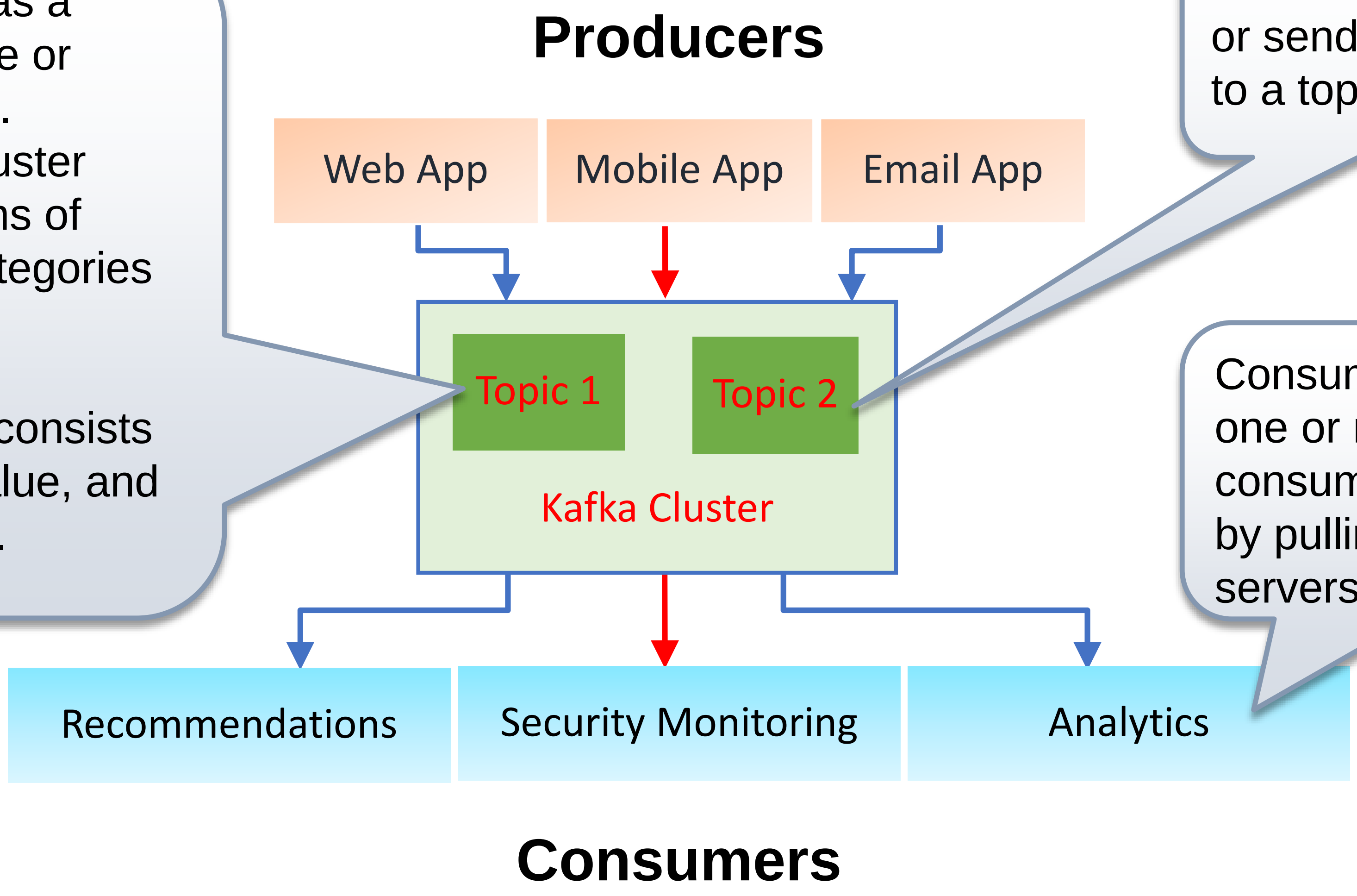
APACHE kafka

A distributed streaming platform

Apache Kafka: Basic Concepts

Kafka is run as a cluster on one or more servers.
The Kafka cluster stores streams of records in categories called *topics*.

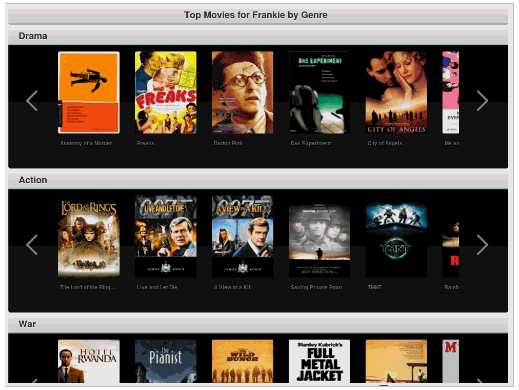
Each record consists of a key, a value, and a time stamp.



Producers publish or send messages to a topic.

Consumers subscribe to one or more topics and consume the messages by pulling data from the servers.

Using Apache Kafka: Oracle MoviePlex Examples



Example 1: User guest1 watched *The Incredibles.*

Stream of Movie Views

Topic 1

Topic 2

Kafka Cluster

Log Search

Analytics

Monitoring

Apache Spark streaming application subscribes to the user activity topic and pulls the event data.

On analysis, consumer makes a special offer:
You get a Superman lunch box!

Example 2: Events consumed from a web-metrics topic, when analyzed, point out multiple users using the same sign-in. **Check for fraud!**

Additional Resources

- http://flume.apache.org/index.html
- https://kafka.apache.org/

Summary

- In this lesson, you should have learned to describe:
  - Uses of the command-line interface (CLI)
  - Benefits of Fuse DFS
  - Architecture and features of Flume
  - Features and uses of Kafka

Practice 6: Overview

- In this practice, you use Flume to transfer JSON log data to HDFS.

Introduction to MapReduce and YARN Processing Frameworks

Course Road Map

- Module 1: Big Data Fundamentals
- Module 2: Data Acquisition and Storage
- **Module 3: Data Access and Processing**
- Module 4: Data Unification
- Module 5: Data Analysis
- Module 6: Big Data Deployment Options

- Lesson 8: Introduction to MapReduce and YARN Processing Frameworks
- Lesson 9: Resource Management Using YARN
- Lesson 10: Overview of Spark
- Lesson 11: Overview of Apache Hive
- Lesson 12: Overview of Cloudera Impala
- Lesson 13: Using Oracle XQuery for Hadoop
- Lesson 14: Overview of Solr

Agenda

- MapReduce process
- YARN architecture

Objectives

- After completing this lesson, you should be able to describe the following:
  - Identify the benefits of MapReduce, run a MapReduce job, and monitor the job
  - Review YARN architecture
  - Use YARN to monitor jobs and to manage resources in your Hadoop cluster

Apache Hadoop Core Components

- Apache Hadoop is a system for large-scale and distributed data processing. It has two core components:
  - Distributed storage with HDFS (covered in the lesson titled "Introduction to the HDFS")
  - Distributed and parallel processing with MapReduce or Spark framework
- MapReduce is a batch-oriented software framework that enables you to write applications that will process large amounts of data in parallel, on large clusters of commodity hardware, and in a reliable and fault-tolerant manner.
- The Apache Spark framework is a cluster-computing platform designed to be fast and general-purpose. It improves the MapReduce functionality (covered later).
- Both MapReduce and Spark are managed by YARN (the acronym for "Yet Another Resource Negotiator").

MapReduce Framework: Features

- Integrates with HDFS and provides the same benefits for parallel data processing
- Parallelizes and distributes computations to where the data is stored (data locality)
- The framework:
  - Schedules and monitors tasks, and re-executes failed tasks
  - Hides complex distributed computing tasks from the developer
  - Enables developers to focus on writing the Map and Reduce functions

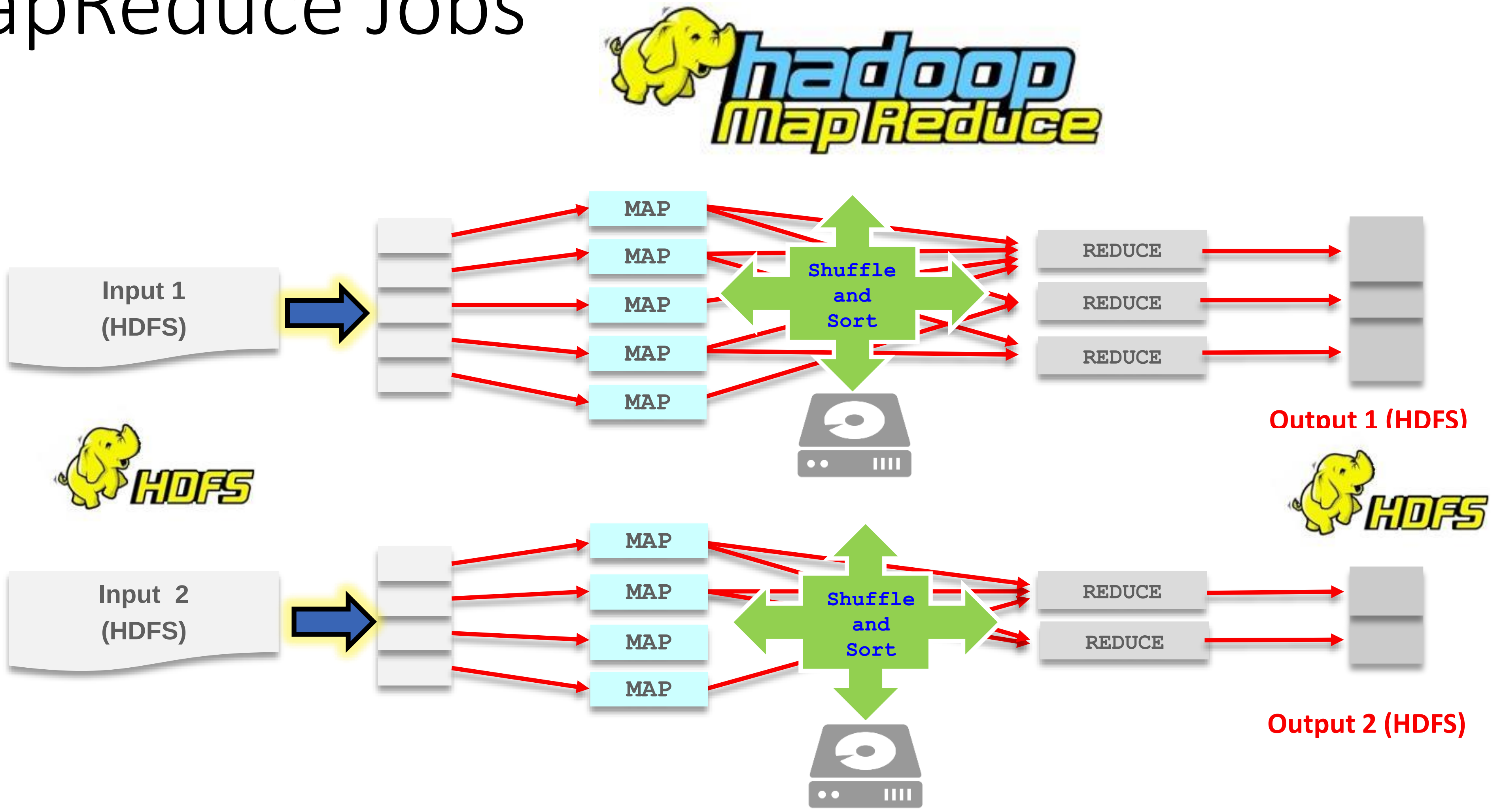
MapReduce Job

- A MapReduce job is a unit of work requested by a client.
- The job consists of:
  - Input data (usually stored in HDFS)
  - A MapReduce program (user-defined)
- Hadoop runs the job by dividing it into tasks:
  - Map tasks (user-defined)
  - Shuffle and sort tasks (MapReduce)
  - Reduce tasks (user-defined)
- Hadoop divides the input data into fixed-size pieces called *input splits.*
  - Hadoop creates one map task for each input split
  - Each input split runs the user-defined map function for each *record* in the input split

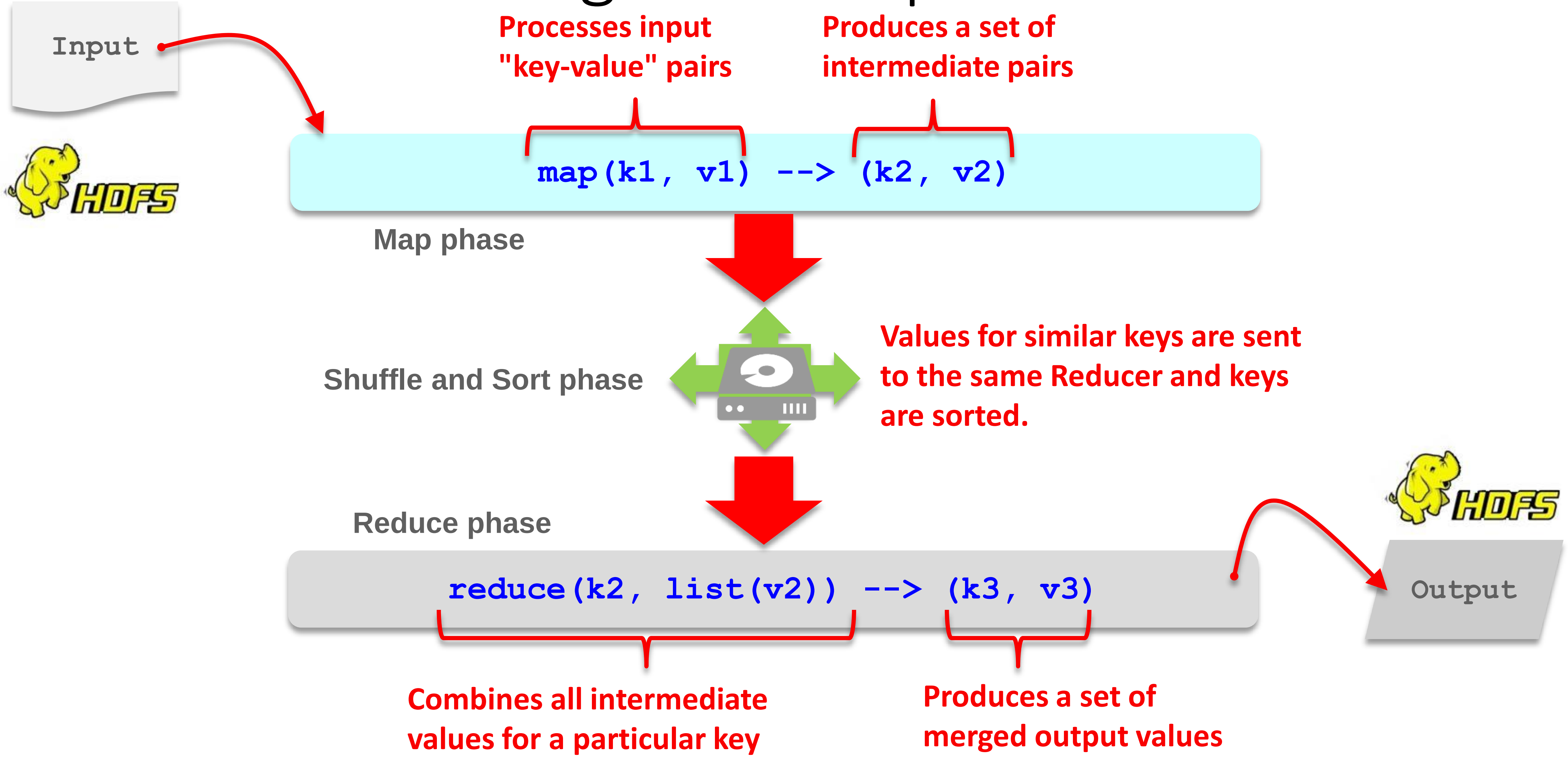
Benefits of MapReduce

- MapReduce provides:
  - Automatic parallelization and distribution of large data sets that are stored and distributed on Hadoop cluster slave nodes
  - Fault-tolerance
  - I/O scheduling
  - Status and monitoring

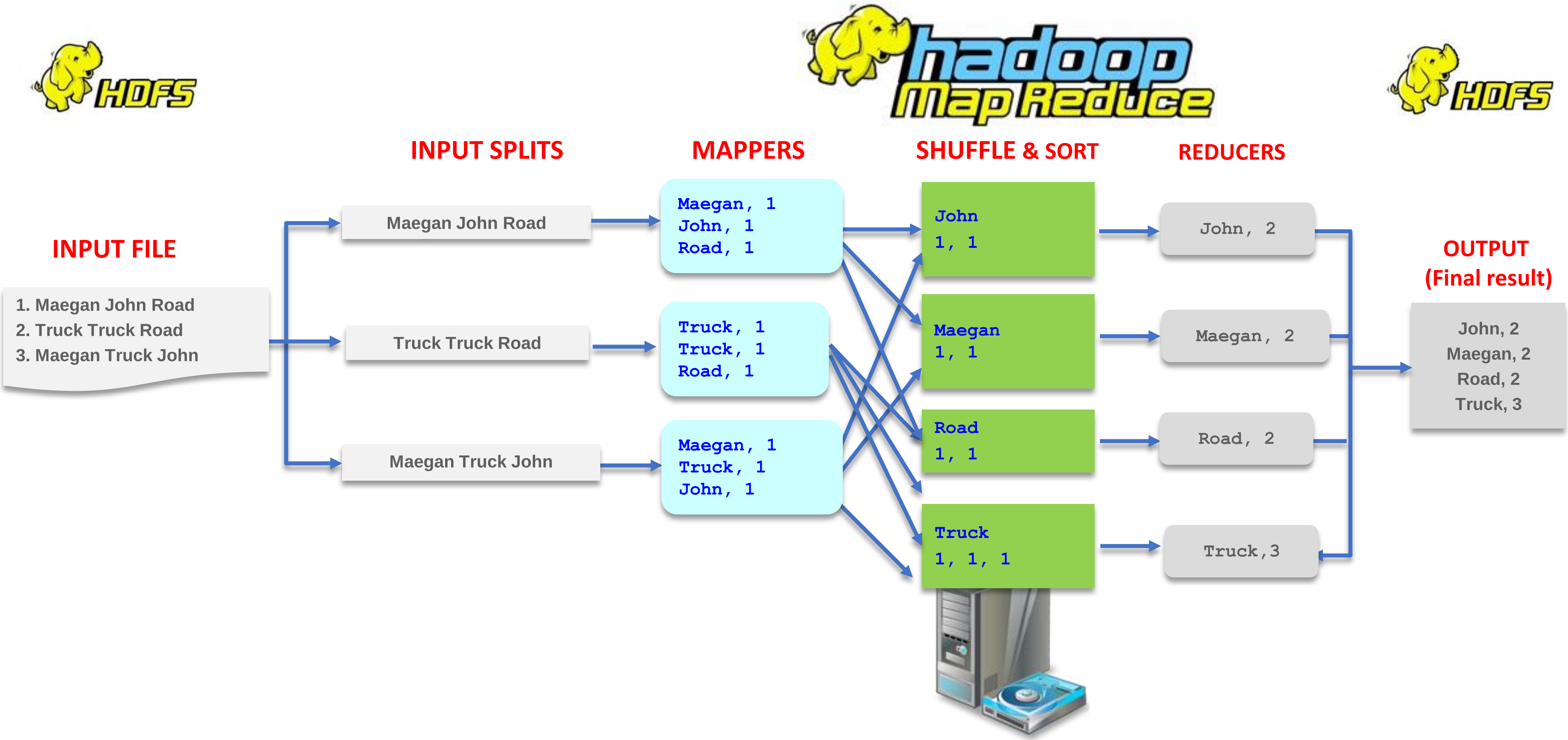
MapReduce Jobs



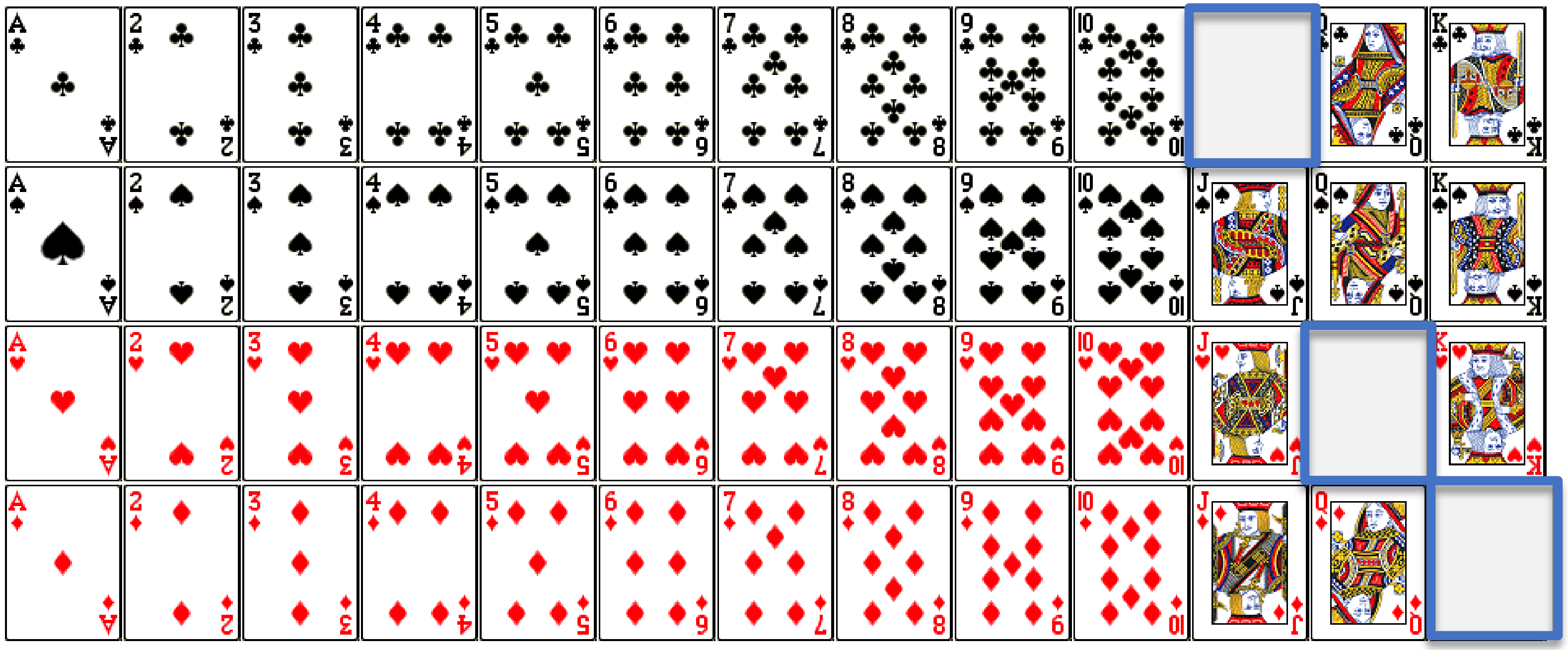
Parallel Processing with MapReduce



Word Count Process: Example 1

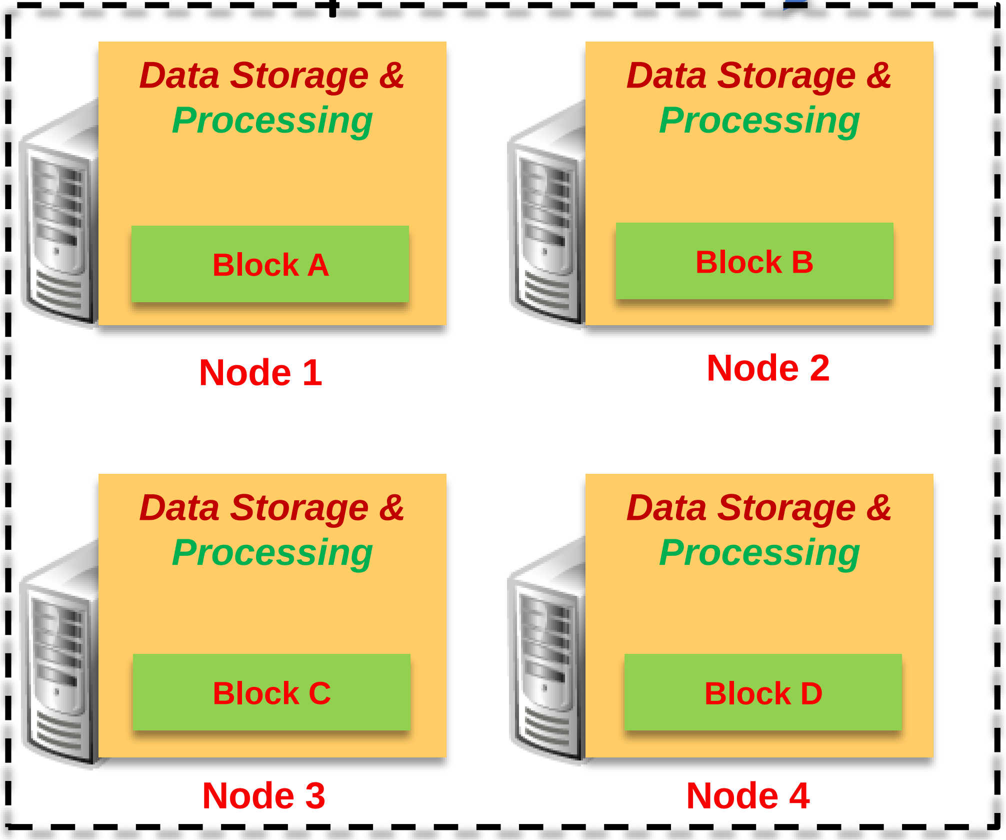


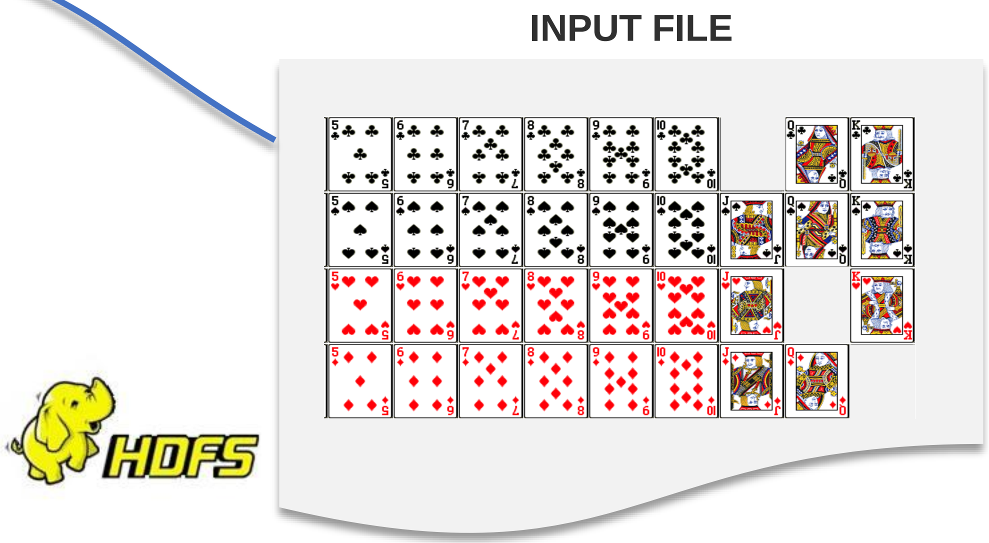
MapReduce Mechanics: Deck of Cards Example



Suppose three face cards are removed from the deck of playing cards.
How do you find the suits whose face cards are short by using MapReduce?

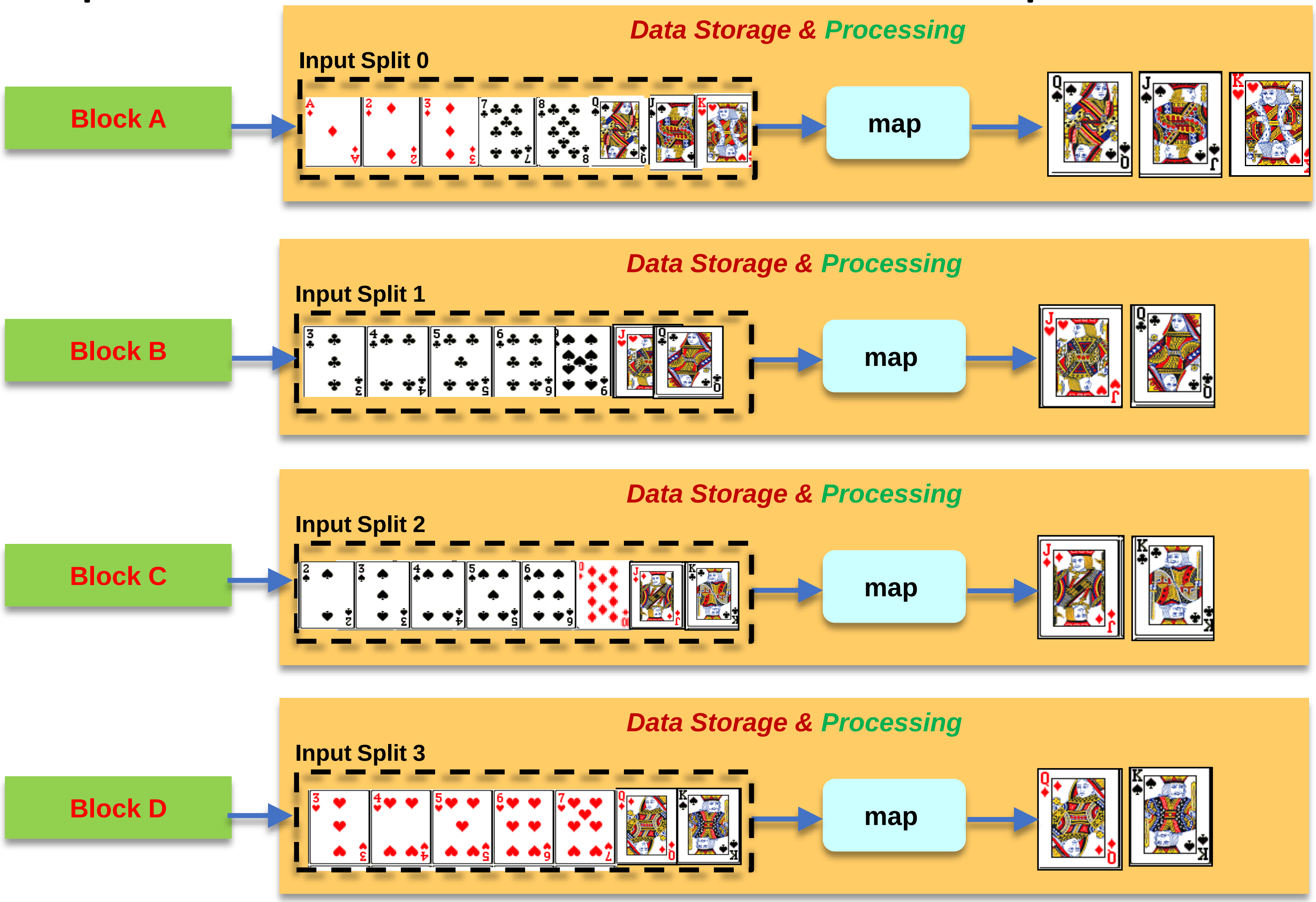
MapReduce Mechanics Example: Assumptions





1. The deck of cards was an input file stored in HDFS as blocks on several slave nodes for parallelism and fault-tolerance (four blocks in this example).
2. Nodes operate on their local data.

MapReduce Mechanics: Map Phase

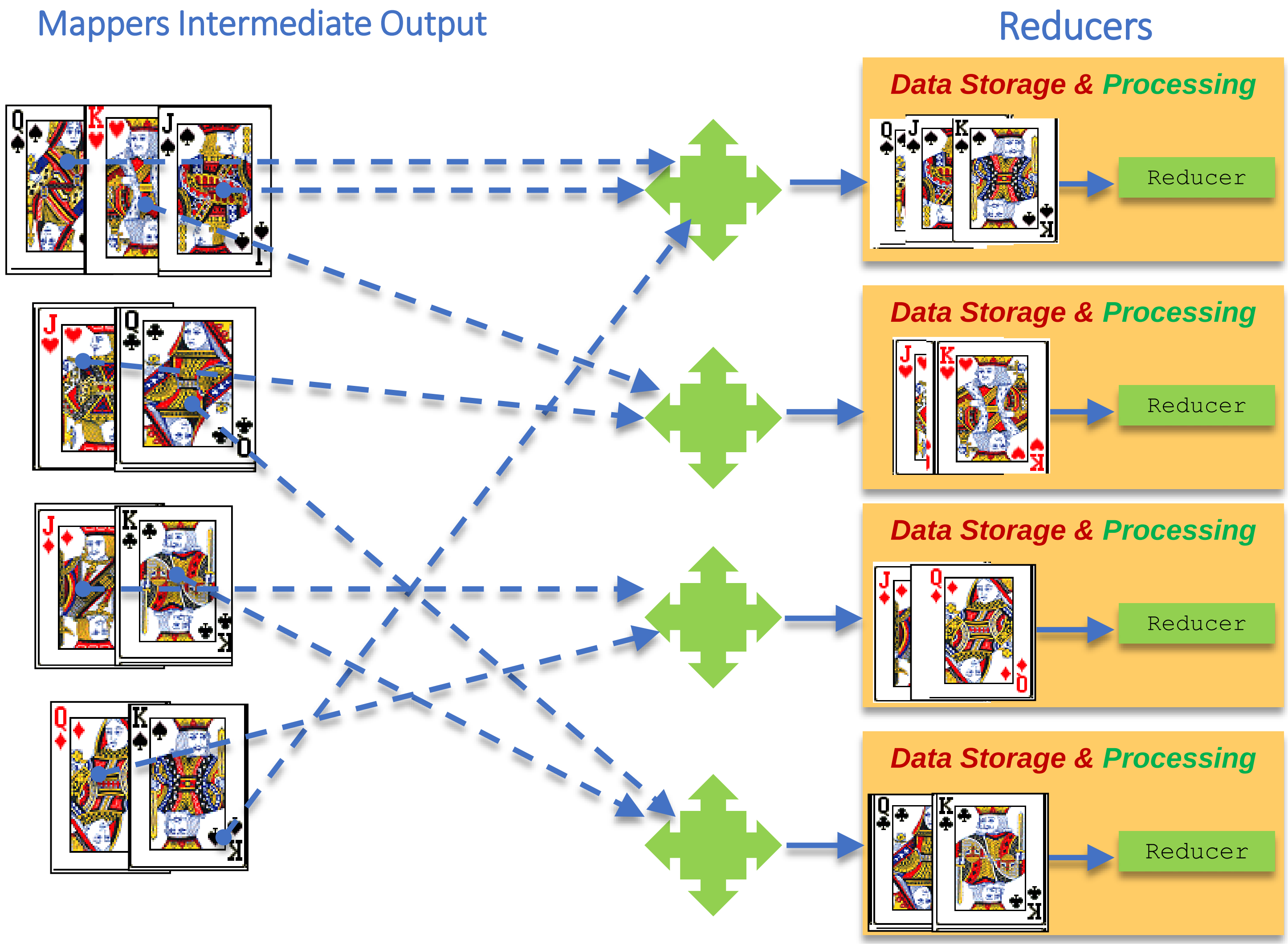



```
If face_card:
emit(suit, card)
```

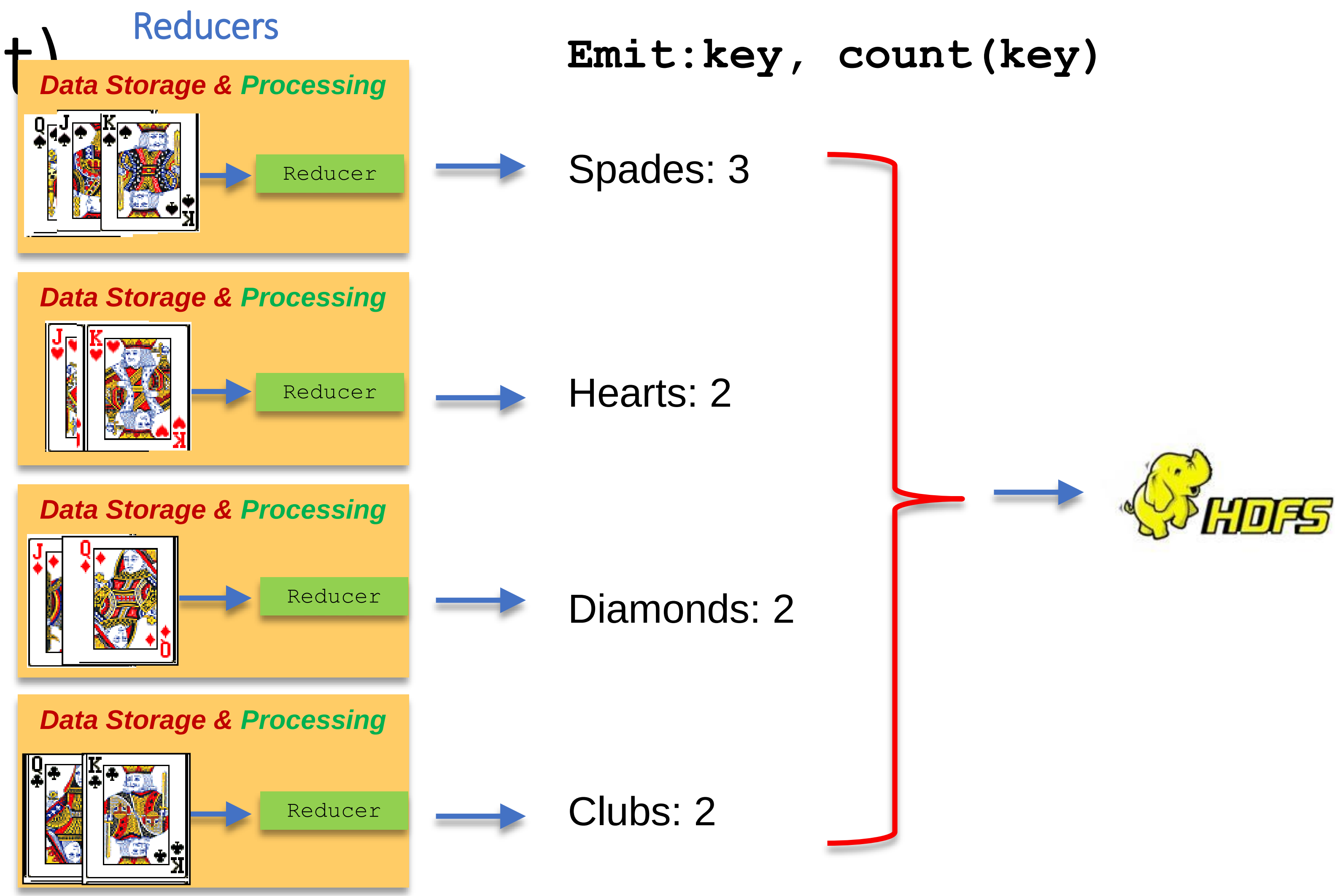
Key
Value

MapReduce Mechanics: Shuffle and Sort Phase

Intermediate data is shuffled and sorted for delivery (stored on local disk) to the reduce tasks.



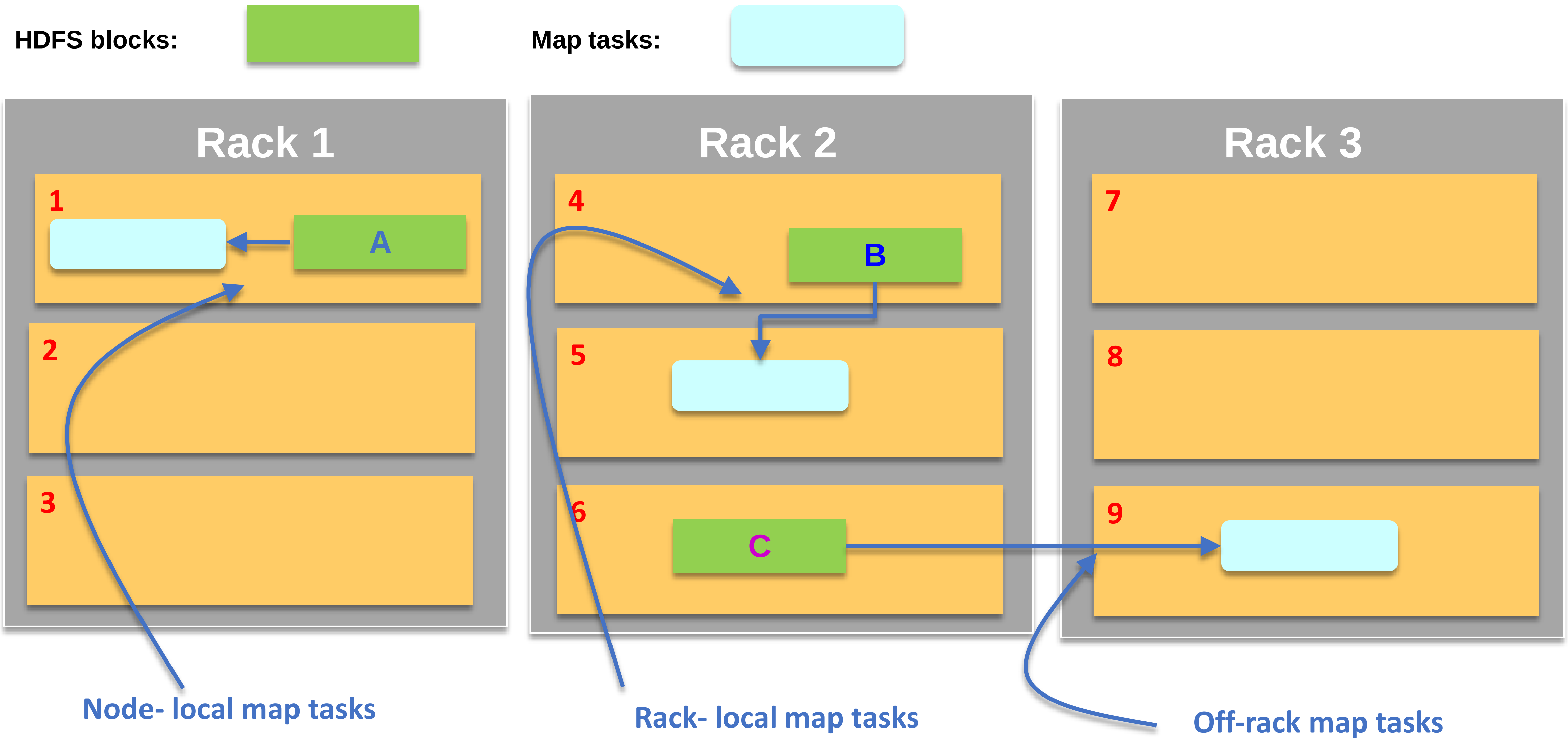
MapReduce Mechanics: Reduce Phase (Result)



Interacting with MapReduce

- MapReduce code can be written in Java, C, and scripting languages.
- Higher-level abstractions (Hive, Pig) enable easy interaction.
  - Optimizers construct MapReduce jobs.
- Code is:
  - Submitted to the JobTracker daemons on the Master node and executed by the TaskTrackers on the Slave nodes with the older MR1.
    - Runs only MapReduce applications
    - MRv1 was the key processing framework prior to YARN.
  - Submitted to the ResourceManager daemon on the Master node and executed by the ApplicationMasters on the Slave nodes with the YARN (MR2).
    - Runs MapReduce applications, Impala, and Spark
    - Adds the advantage of YARN where ResourceManager is now a scheduler and a resource manager
    - YARN introduced both flexibility and scalability to the processing framework.
    - Most Hadoop implementations use YARN.

Data Locality Optimization in Hadoop



Submitting a MapReduce Job

```
hadoop jar WordCount.jar WordCount
    /user/oracle/wordcount/input
    /user/oracle/wordcount/output
```

Code	Description
`hadoop jar`	Tells the client to submit job to the JobTracker
`WordCount.jar`	The JAR file that contains the Map and Reduce code
`WordCount`	The name of the class that contains the main method where processing starts
`/user/oracle/wordcount/input`	The input directory
`/user/oracle/wordcount/output`	A single HDFS output path. All final output will be written to this directory.

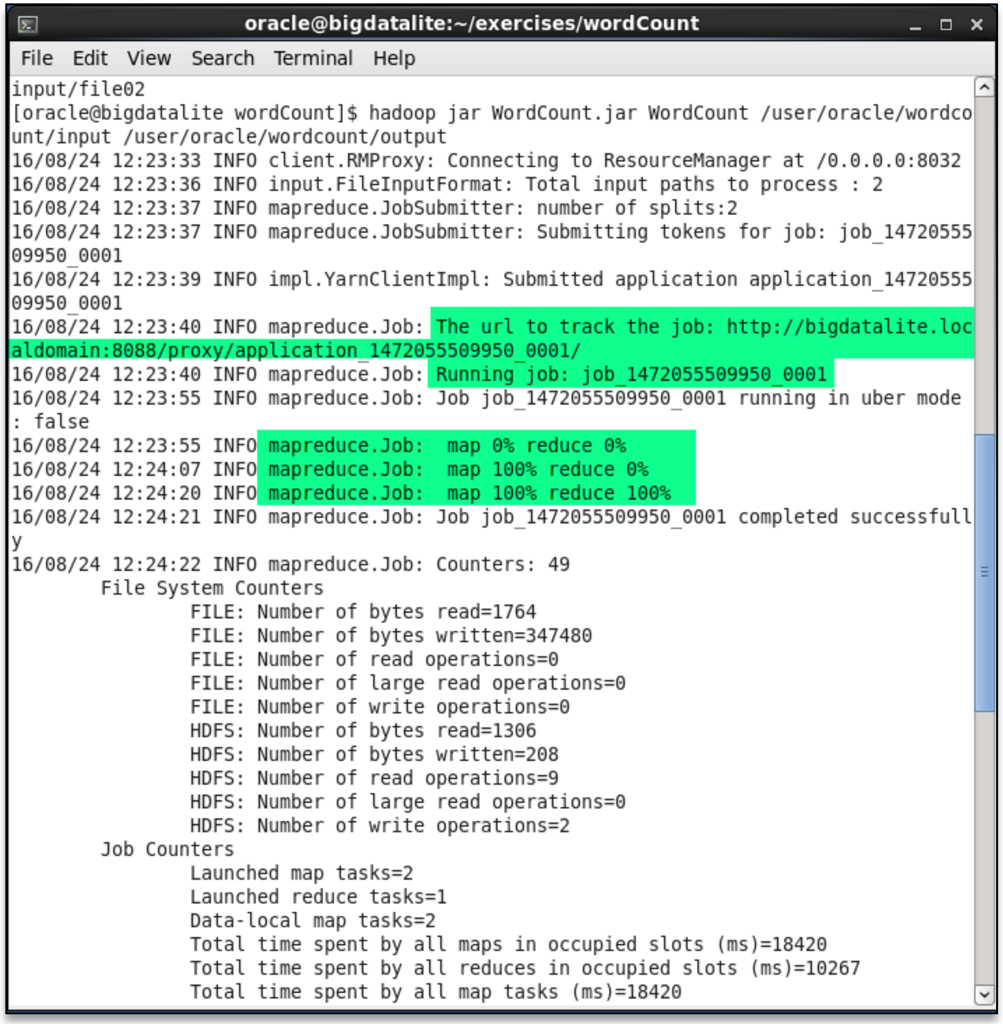
Submitting a `WordCount` MapReduce Job: Reviewing the Input Data Files

```
cat file01 file02
```

```
[oracle@bigdatalite wordCount]$ cat file01 file02
very disappointed and very expensive
expensive and unreliable insurance
worthless insurance and expensive
worst customer service
worst insurance company
worst professional staff and unreliable insurance company
insurance is very expensive
worst insurance cover
terrible service
disappointed with the expensive insurance service
worthless and expensive
awful customer service
terrible worst service
worst bank and worst customer service
worst insurance
disappointed with protocols
unreliable insurance
best service I recommend it
good professionals and efficient insurance
I will recommend it
good customer service
best insurance I found I recommend it[oracle@bigdatalite wordCount]$
```

Submitting the `WordCount` MapReduce Job

```
hadoop jar WordCount.jar WordCount \
/user/oracle/wordcount/input \
/user/oracle/wordcount/output
```

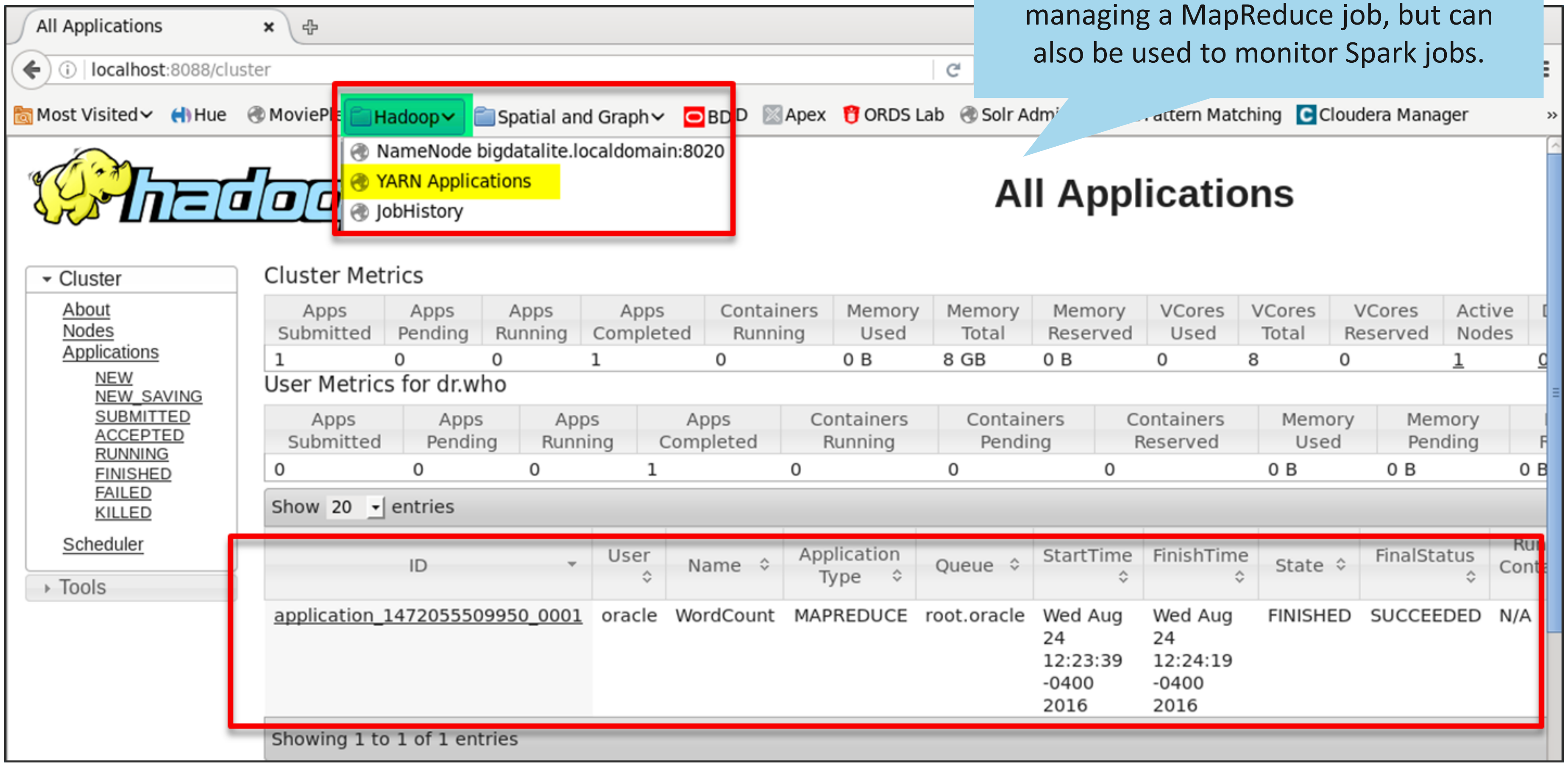


```
oracle@bigdatalite:~/exercises/wordCount
File  Edit  View  Search  Terminal  Help
input/file02
[oracle@bigdatalite wordCount]$ hadoop jar WordCount.jar WordCount /user/oracle/wordco
unt/input /user/oracle/wordcount/output
16/08/24 12:23:33 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/08/24 12:23:36 INFO input.FileInputFormat: Total input paths to process : 2
16/08/24 12:23:37 INFO mapreduce.JobSubmitter: number of splits:2
16/08/24 12:23:37 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14720555
09950_0001
16/08/24 12:23:39 INFO impl.YarnClientImpl: Submitted application application_14720555
09950_0001
16/08/24 12:23:40 INFO mapreduce.Job: The url to track the job: http://bigdatalite.loc
aldomain:8088/proxy/application_1472055509950_0001/
16/08/24 12:23:40 INFO mapreduce.Job: Running job: job_1472055509950_0001
16/08/24 12:23:55 INFO mapreduce.Job: Job job_1472055509950_0001 running in uber mode
: false
16/08/24 12:23:55 INFO mapreduce.Job:  map 0% reduce 0%
16/08/24 12:24:07 INFO mapreduce.Job:  map 100% reduce 0%
16/08/24 12:24:20 INFO mapreduce.Job:  map 100% reduce 100%
16/08/24 12:24:21 INFO mapreduce.Job: Job job_1472055509950_0001 completed successfull
y
16/08/24 12:24:22 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=1764
                FILE: Number of bytes written=347480
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1306
                HDFS: Number of bytes written=208
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=18420
                Total time spent by all reduces in occupied slots (ms)=10267
                Total time spent by all map tasks (ms)=18420
```

Monitoring MapReduce Jobs by Using the YARN Resource Manager Web UI

`http://localhost:8088/cluster`

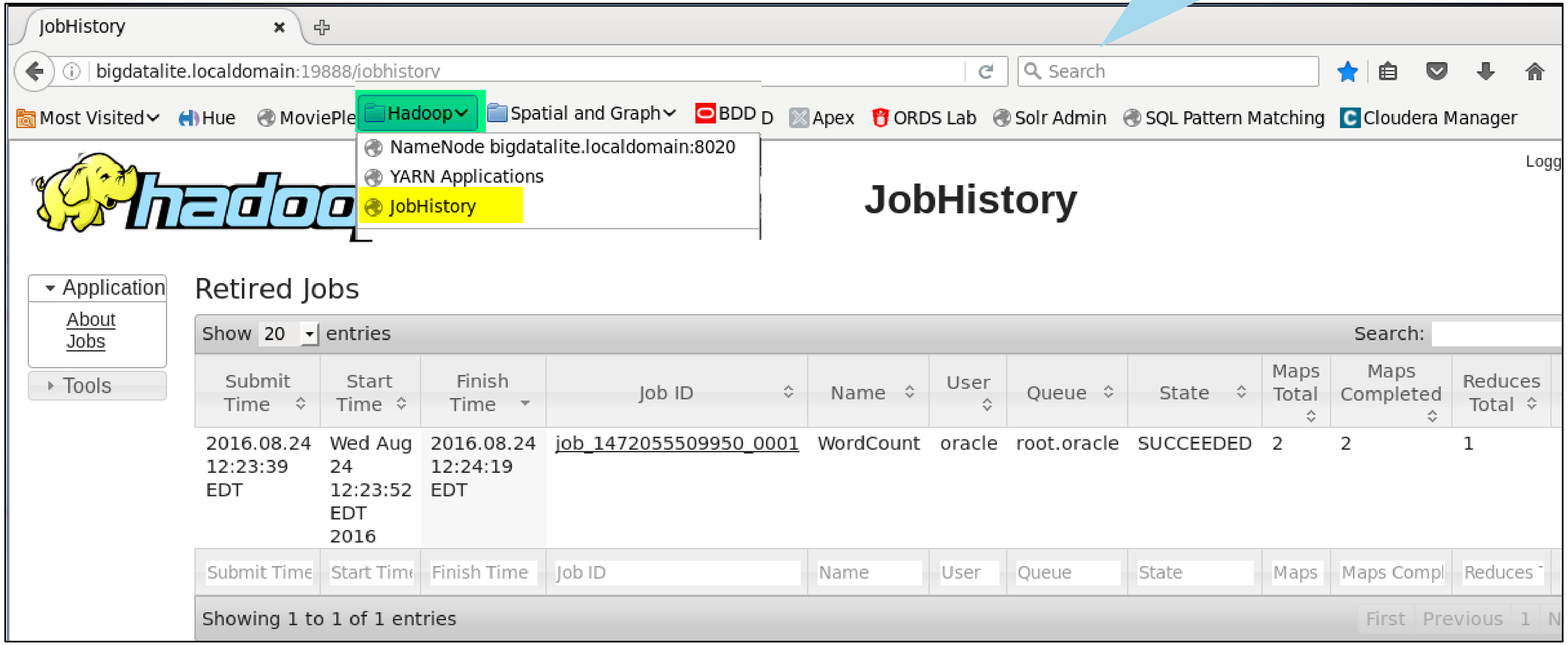
YARN (covered later in this lesson) is used to manage resources on a Hadoop cluster. In this example, YARN is managing a MapReduce job, but can also be used to monitor Spark jobs.



Monitoring MapReduce Jobs by Using the JobHistory Server Web UI

http://bigdatalite.localdomain:19888/jobhistory

MapReduce JobHistoryServer archives jobs’ metrics and can be accessed through the JobHistory Web UI or Hue.



Viewing the `WordCount.java` Program

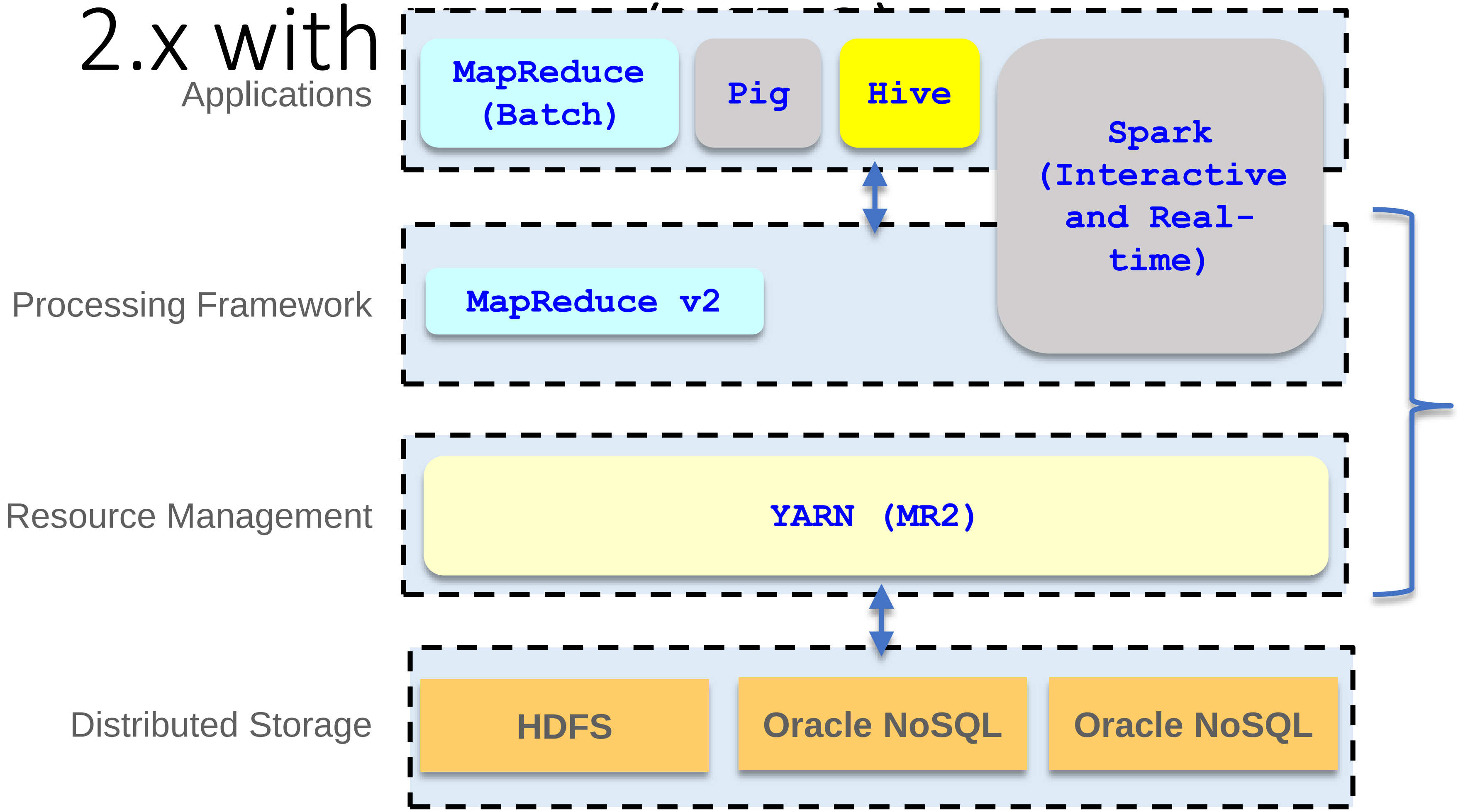
```
[oracle@bigdatalite ~]$ hadoop fs -ls /user/oracle/wordcount/output
Found 2 items
-rw-r--r--   1 oracle oracle          0 2016-08-24 12:24 /user/oracle/wordcount/
output/_SUCCESS
-rw-r--r--   1 oracle oracle        208 2016-08-24 12:24 /user/oracle/wordcount/
output/part-r-00000
[oracle@bigdatalite ~]$
```

```
[oracle@bigdatalite ~]$ hadoop fs -cat /user/oracle/wordcount/output/part-r-0000
0
and     12
awful   2
bank    2
company 4
cover   2
customer        6
disappointed    6
expensive       12
insurance       18
is      2
professional    2
protocols       2
service 12
staff   2
terrible        4
the     2
unreliable      6
very    6
with    4
worst   16
worthless       4
[oracle@bigdatalite ~]$
```

Agenda

- MapReduce process
- YARN architecture

Running Applications Starting with Hadoop 2.x with



- Hadoop cluster Resource Management
- Batch, ***<u>Interactive</u>***, and ***<u>Real-Time</u>*** data processing
- Multiple data processing engines

YARN Architecture

- YARN splits up the two major functionalities of the JobTracker, resource management and job scheduling/monitoring, into separate new daemons.
  - A global ResourceManager (RM)
  - A per-application ApplicationMaster (AM)
- The single RM and NodeManager (NM) on each slave node, form the data-computation framework
- The RM is the ultimate authority that arbitrates resources among all the applications in the system.
- The per-application AM negotiates resources from the RM and works with the NodeManager(s) to execute and monitor the tasks.

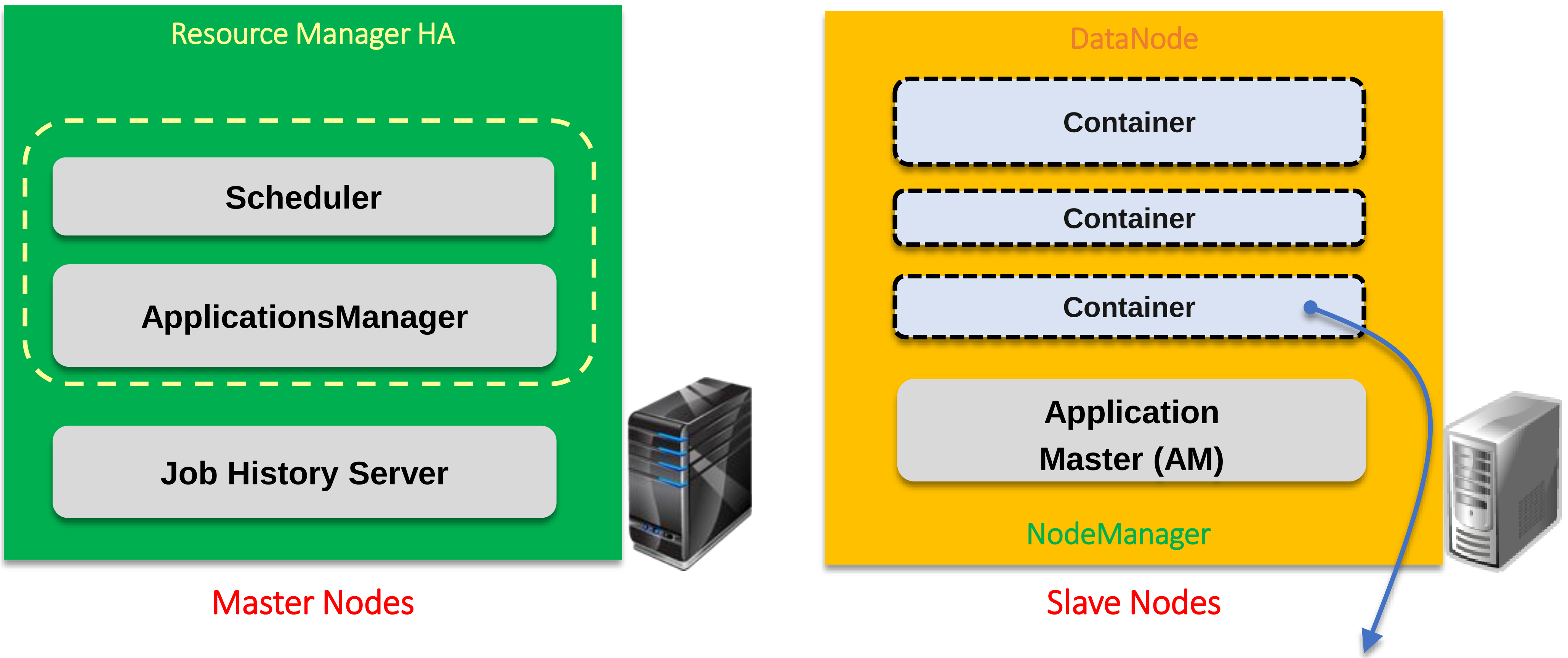
YARN: Features

- Scalability
- Compatibility with MapReduce
- Improved cluster utilization
- Support for workloads other than MapReduce

YARN Daemons

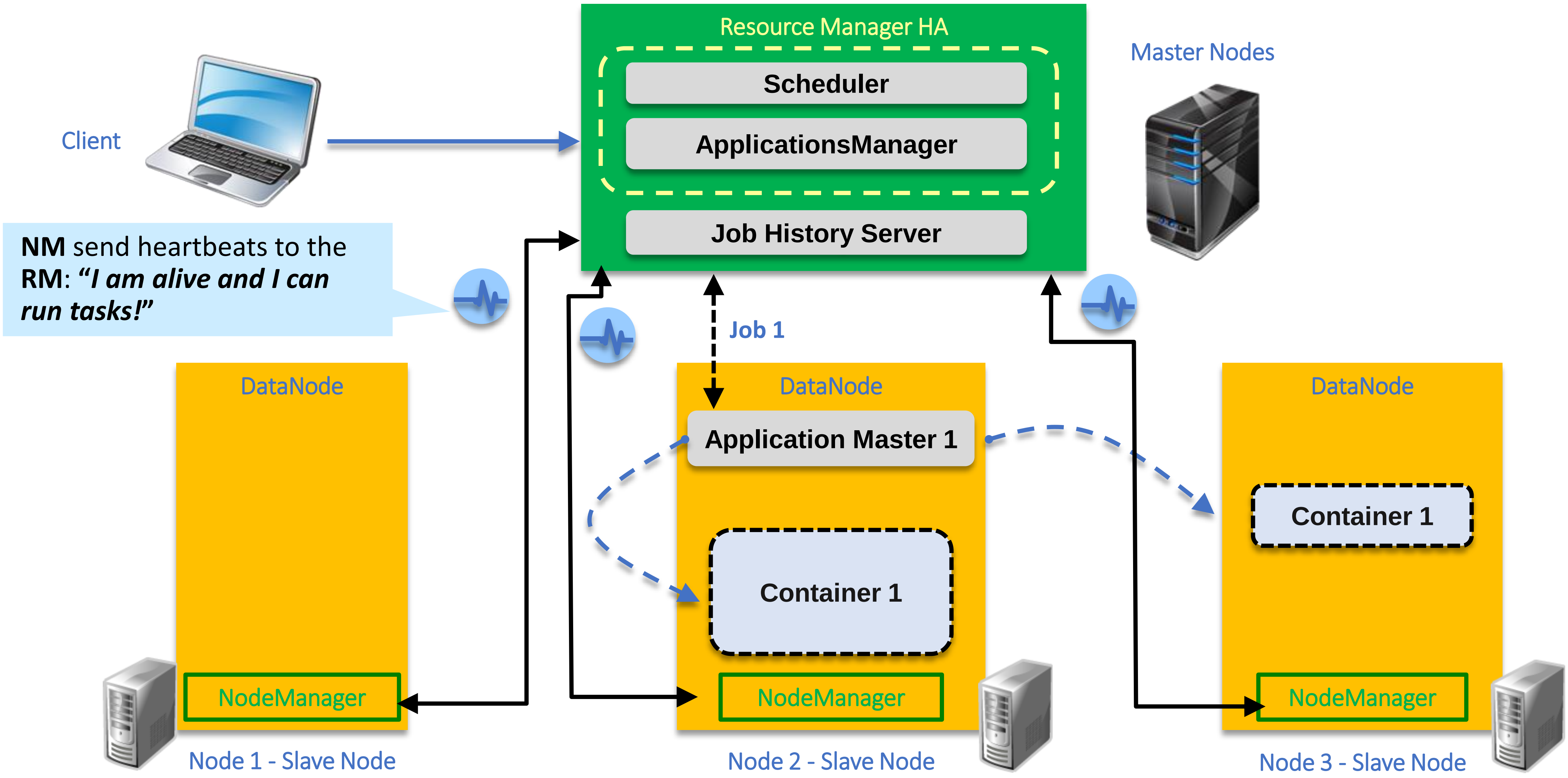
Component	Description
ResourceManager (RM)	• A dedicated scheduler that allocates resources to the requesting applications • **The RM has two main components: Scheduler and ApplicationsManager** • It is a critical component of the cluster and runs on a dedicated master node.
NodeManager (NM)	• Each slave node in the cluster has an NM daemon, which acts as a slave for the RM. • Each NM tracks the available data processing resources and usage (cpu, memory, disk, network) on its slave node and sends regular reports to the RM.
ApplicationMaster (AM)	• The per-application AM is responsible for negotiating resources from the RM and working with the NM(s) to execute and monitor the tasks. • Runs on a slave node
Container	• A container is a collection of all the resources necessary to run an application: CPU, memory, network bandwidth, and disk space. • Runs on a slave node in a cluster.
Job History Server	• Archives jobs and metadata

YARN Architecture

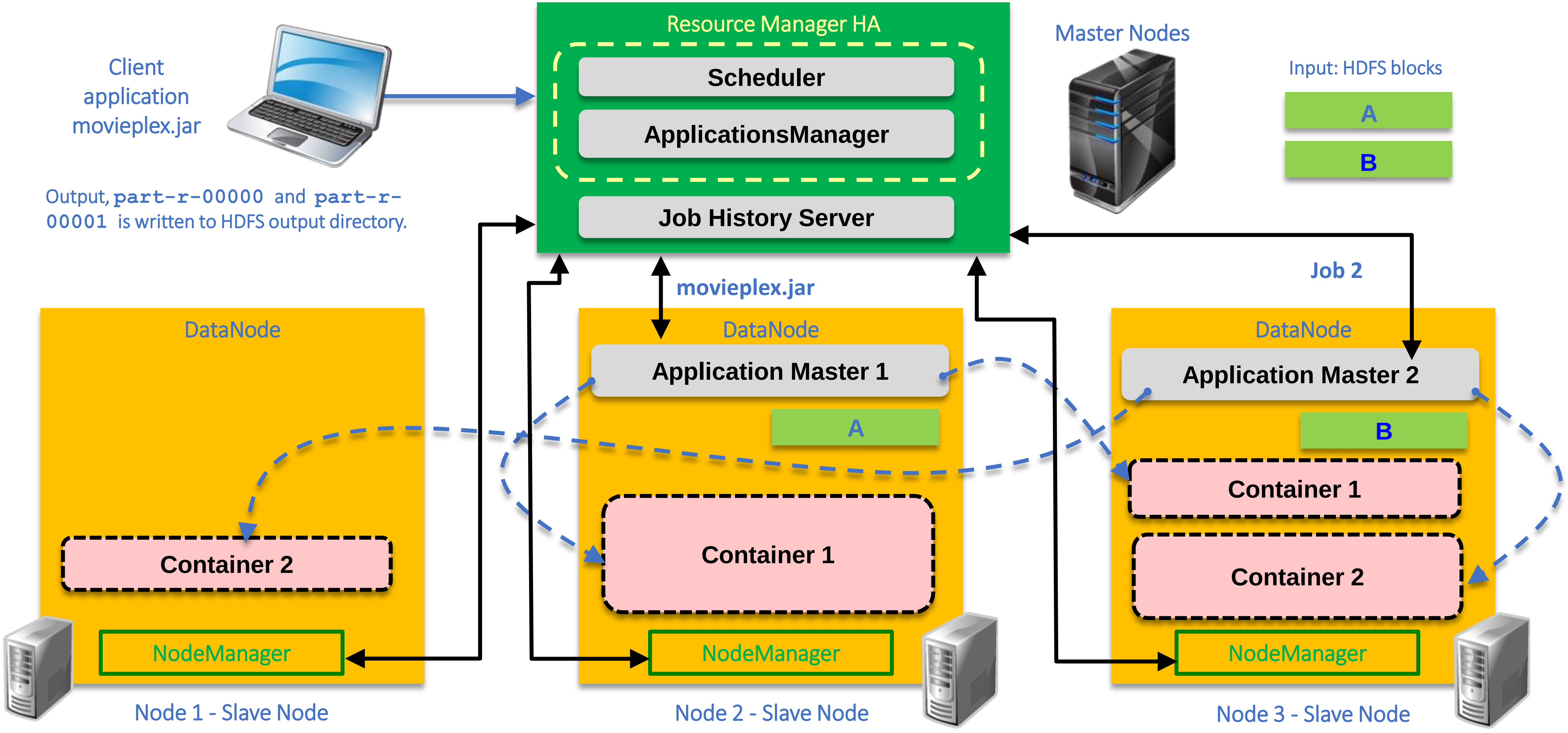


Application Master Containers specifications use the physical world approach such as CPU and cores, memory, disk space, and network Bandwidth (instead Of the Map and Reduce slots in MapReduce V1)

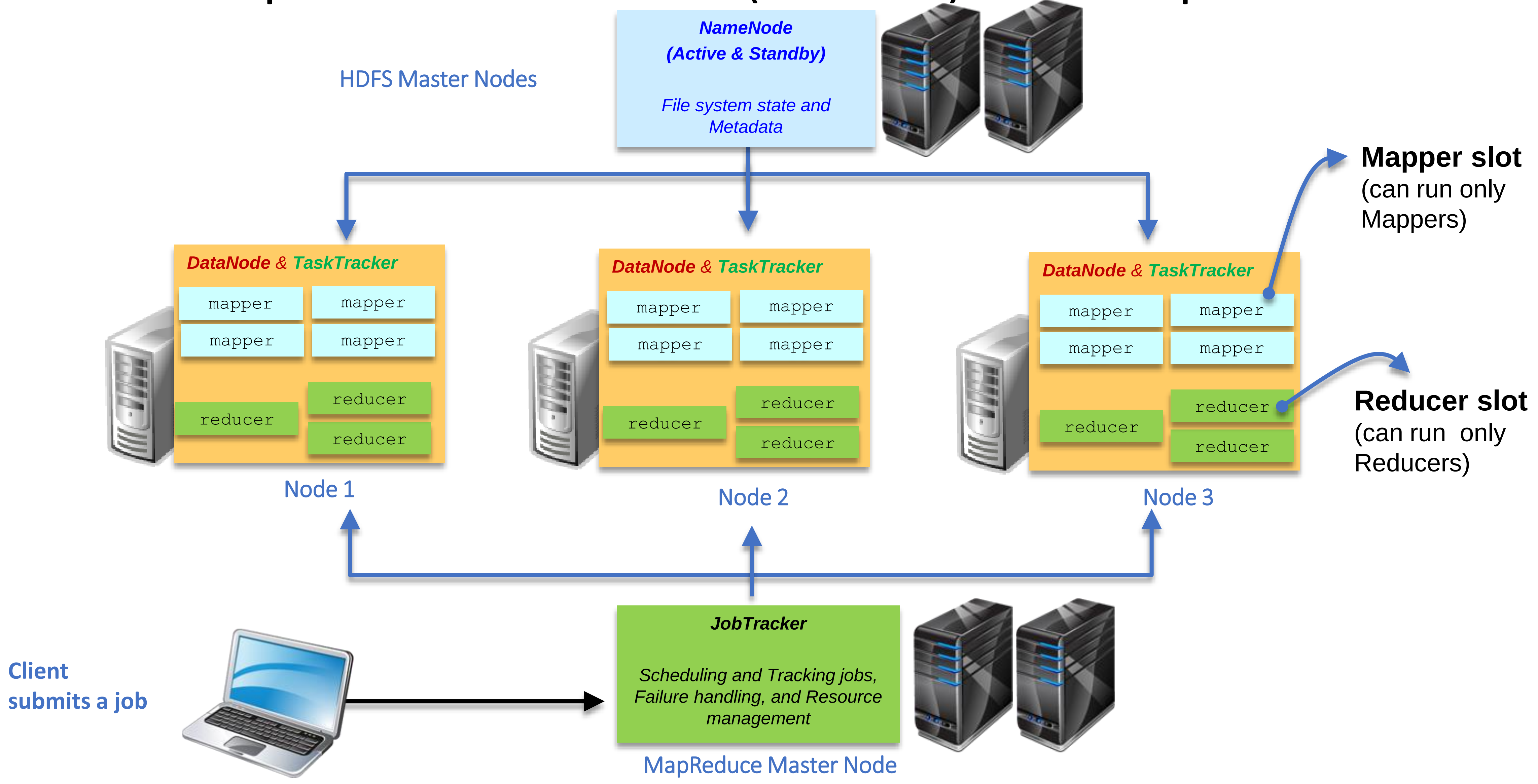
YARN (MRv2) Architecture



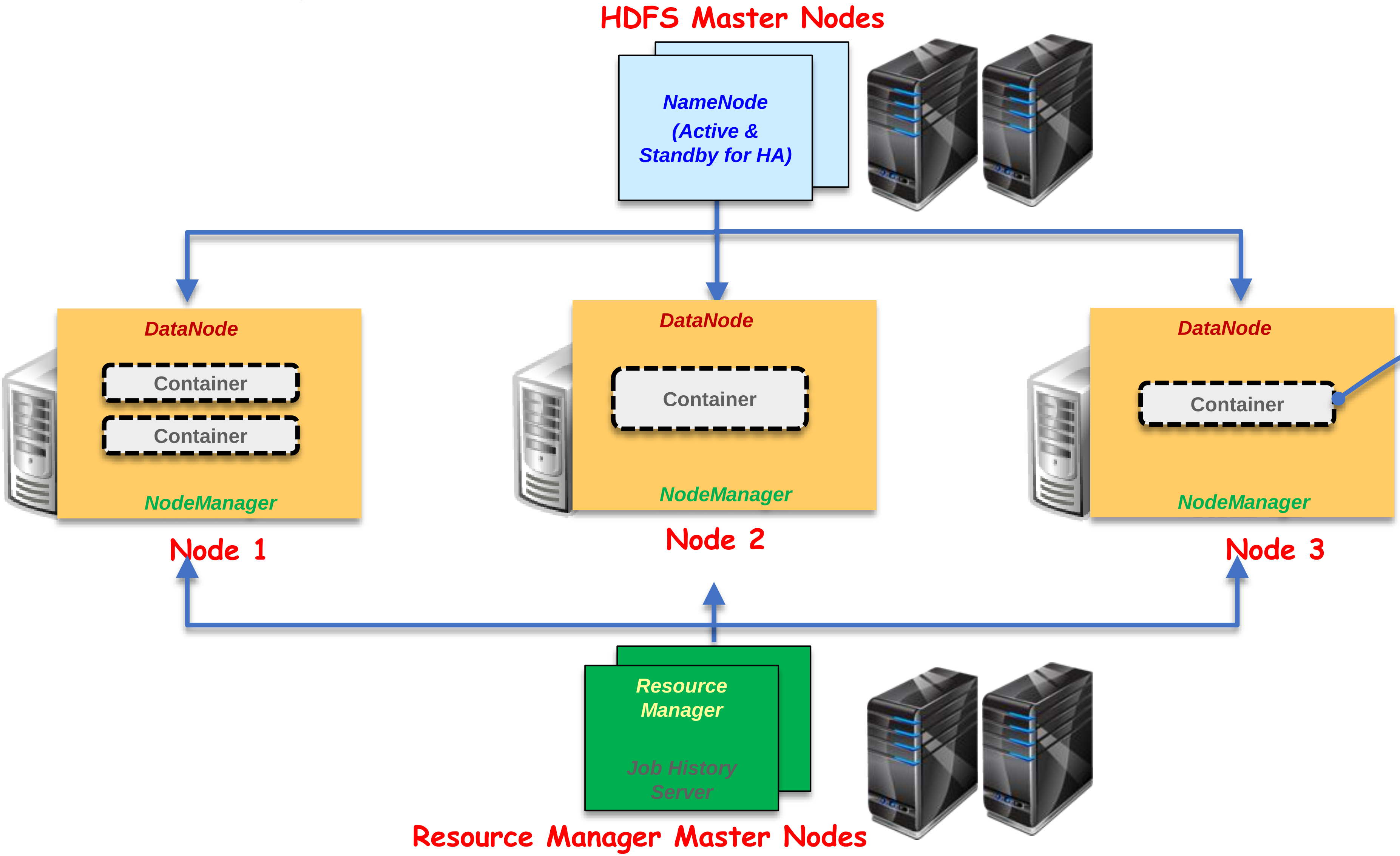
YARN Application Workflow



Hadoop Basic Cluster (MRv1): Example



Hadoop Basic Cluster YARN (MRv2): Example



Physical world approach such as CPU and cores, memory, etc.
(can run either Mappers or Reducers)

Summary

- In this lesson, you should have learned how to:
  - Identify the benefits of MapReduce, run a MapReduce job, and monitor the job
  - Identify the MapReduce version 1 and YARN (MR2) architectures
  - Use YARN to monitor jobs and to manage resources in your Hadoop cluster

Practice 8: Overview

- In these practices, you:
  - Compile a `WordCount.java` program, which runs on a Hadoop Cluster
  - Upload the files on which you run the WordCount into the Hadoop Distributed File System (HDFS)
  - Run the `WordCount.java` program and view the results
  - Monitor a job with the Resource Manager and JobHistory Web UIs

Resource Management Using YARN

Course Road Map

- Module 1: Big Data Fundamentals
- Module 2: Data Acquisition and Storage
- **Module 3: Data Access and Processing**
  - Lesson 8: Introduction to MapReduce
  - **Lesson 9: Resource Management Using YARN**
  - Lesson 10: Apache Spark
  - Lesson 11: Overview of Apache Hive
  - Lesson 12: Overview of Cloudera Impala
  - Lesson 13: Using Oracle XQuery for Hadoop
  - Lesson 14: Overview of Solr
- Module 4: Data Unification
- Module 5: Data Analysis
- Module 6: Big Data Deployment Options

Objectives

- After completing this lesson, you should be able to do the following:
  - Use YARN to manage resources efficiently
  - Use the YARN `application` command

Agenda

- Job Scheduling in YARN
- YARN `application` command

Job Scheduling in YARN

- YARN provides a pluggable model to schedule policies. The scheduler is responsible for deciding where and when to run tasks. YARN supports the following pluggable schedulers:
  - First In, First Out (FIFO)
    - Allocates resources based on arrival time
  - Capacity Scheduler
    - Allocates resources to pools, with FIFO scheduling within each pool
    - Default in Hadoop
  - Fair Scheduler
    - Allows YARN applications to share resources in large clusters fairly
    - This course focuses on the Fair Scheduler
    - Default in CDH5 (used in Oracle BDA)

YARN Fair Scheduler

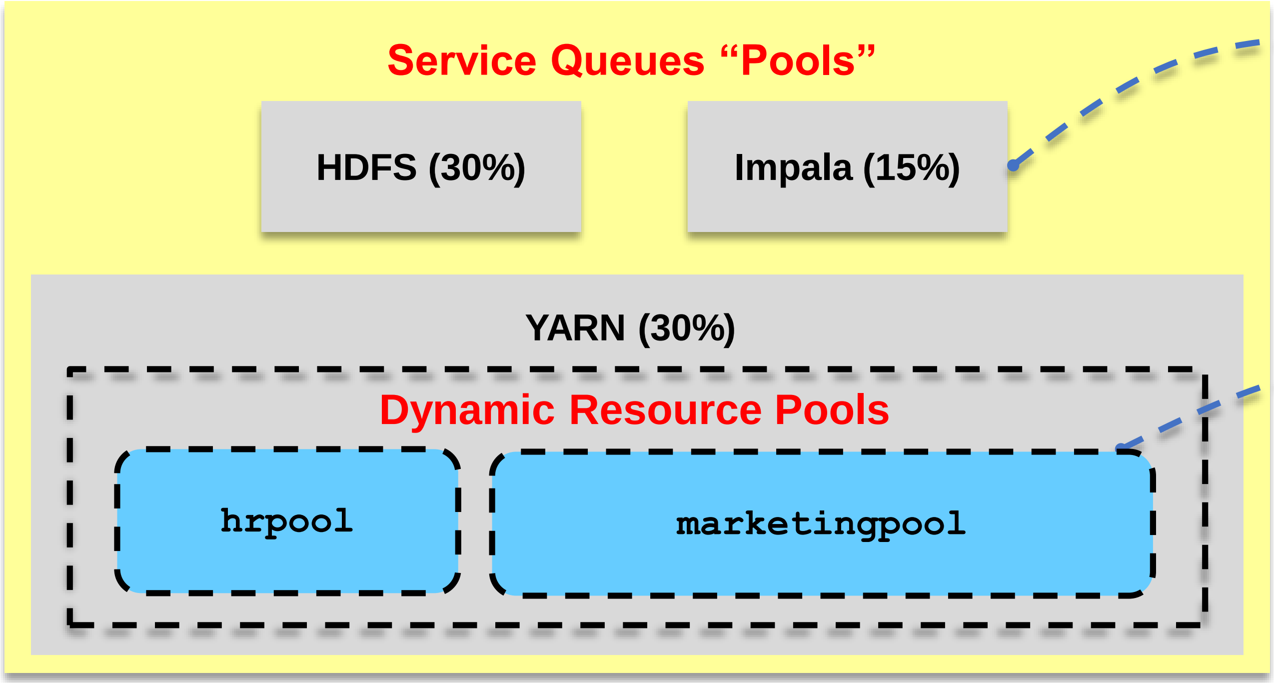
- It is a pluggable scheduler that allows YARN applications to share resources in large clusters fairly.
- In the YARN Fair Scheduler, the terms “queue” and “pool” are used interchangeably.
- It assigns resources to applications so that all applications get an equal share of allocated resources over time.
- It bases scheduling fairness decisions only on memory but you can configure it to be based on both memory and CPU.
- It organizes applications into “queues” (pools), and shares resources fairly between these queues. Every application belongs to a queue (“default” queue is the default).
- It grants YARN containers to the queue with the least amount of allocated resources.

YARN Fair Scheduler

- It works also with application priorities. The priorities are used as weights to determine the fraction of total resources that each application must get.
- If an application specifically lists a queue in a container resource request, the request is submitted to that queue.
- The Fair Scheduler supports moving a running application to a different queue, for example:
  - Moving an important application to a higher priority queue
  - Moving an unimportant application to a lower priority queue
- The Fair Scheduler also enables you to assign guaranteed minimum shares to queue.

Cloudera Manager Resource Management: Features

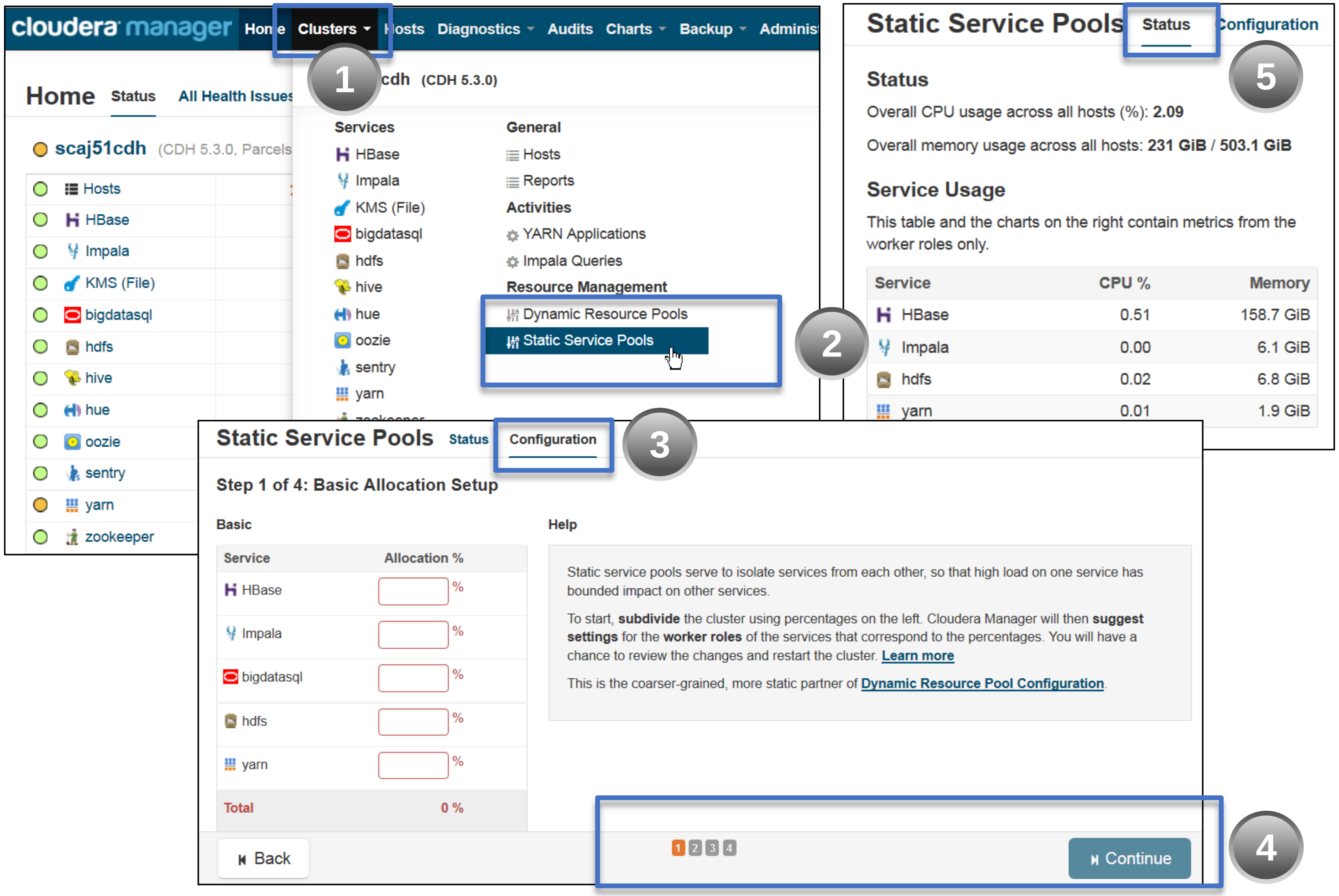
- Cloudera Manager provides the following features to assist you with allocating cluster resources to services:
  - Static allocation (percentage of cluster resources)
  - Dynamic allocation



Gray boxes represent statically allocated resource pools.

Blue boxes represent dynamically allocated resource pools.

Static Service Pools



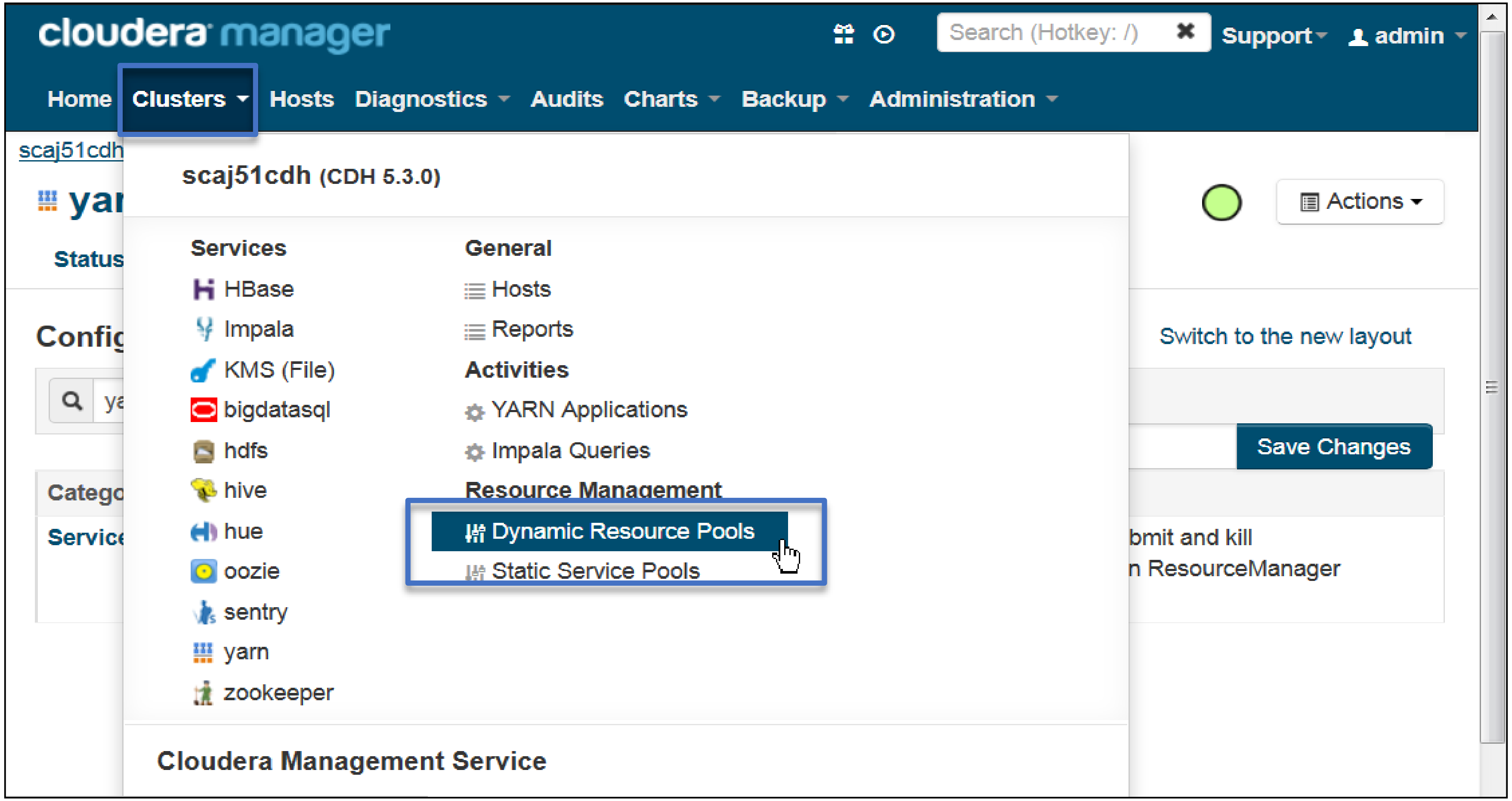
Working with the Fair Scheduler

- Creating Dynamic Resource Pools
- Assigning Priorities to Dynamic Resource Pools
- Monitoring Jobs

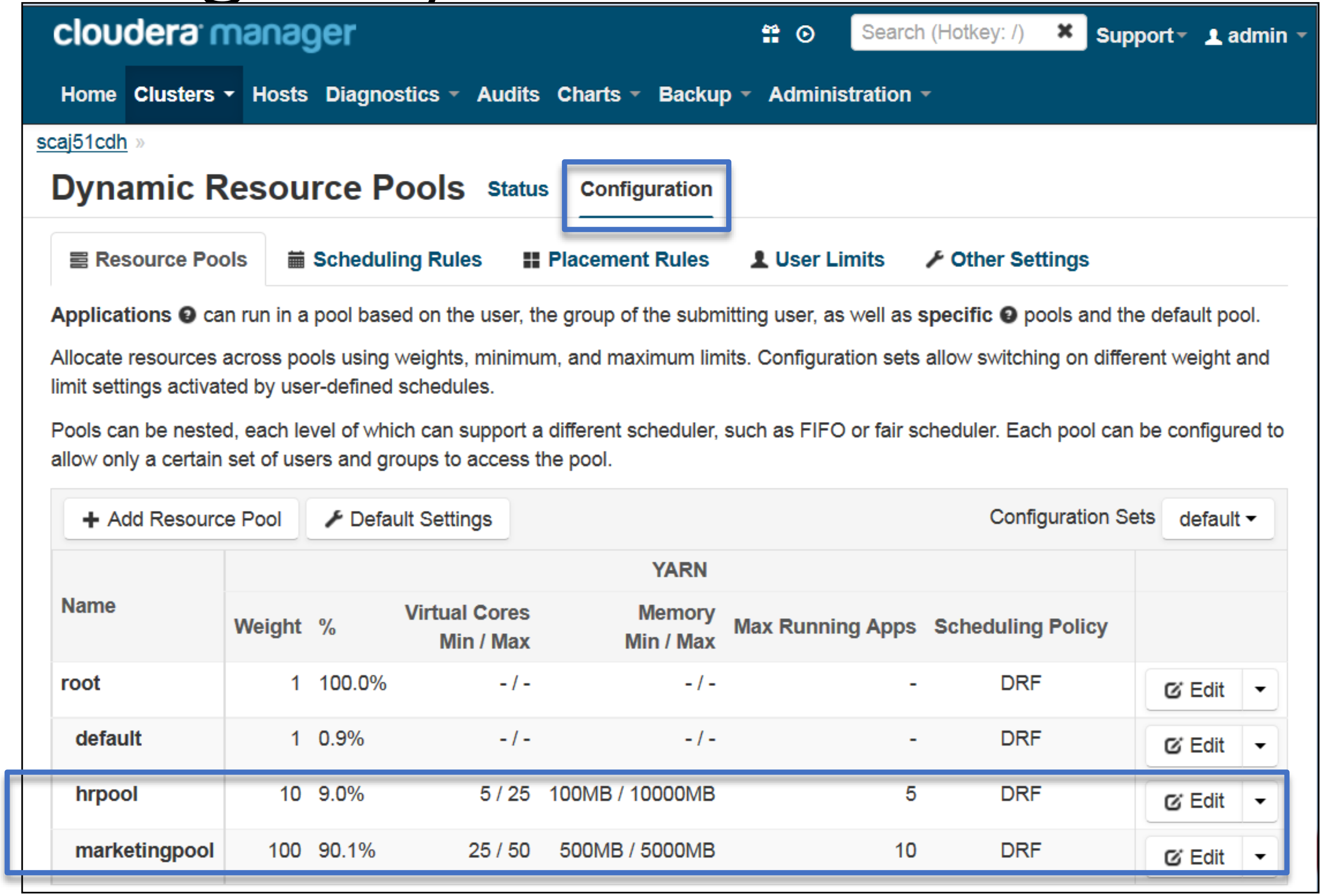
Cloudera Manager Dynamic Resource Management: Example

```
# Linux commands for creating the groups and
# users in the new groups which are used in the
# on the next few slides.
groupadd marketing
groupadd hr
groupadd development
useradd -r -g marketing bob
useradd -r -g hr lucy
. . .
```

Cloudera Manager Dynamic Resource Management: Example



Cloudera Manager Dynamic Resource Management: Example



Cloudera Manager Dynamic Resource Management: Example

Name	YARN						
	Weight	%	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Scheduling Policy	
root	1	100.0%	- / -	- / -	-	DRF	Edit
default	1	0.9%	- / -	- / -	-	DRF	Edit
hrpool	10	9.0%	5 / 25	100MB / 10000MB	5	DRF	Edit
marketingpool	100	90.1%	25 / 50	500MB / 5000MB	10	DRF	Edit

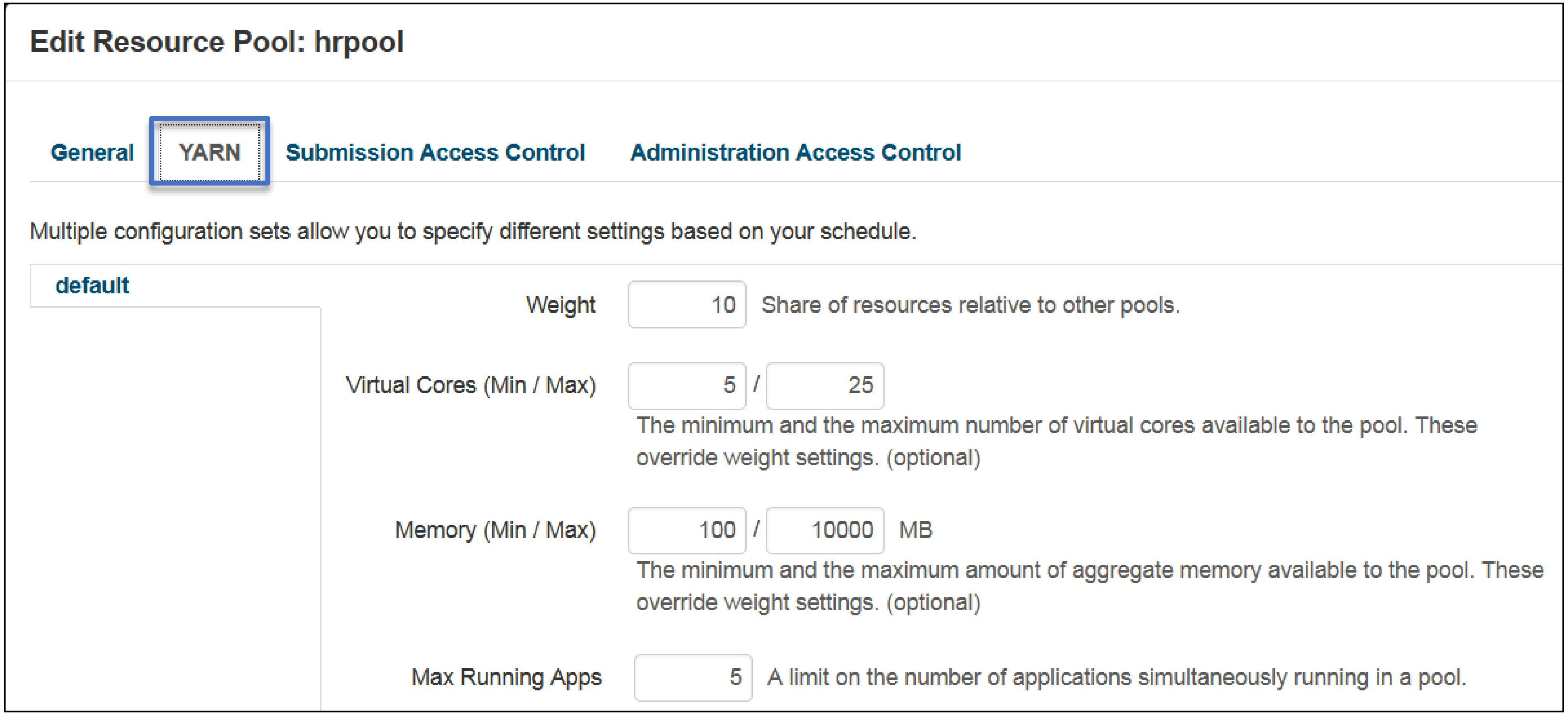
Edit Resource Pool: hrpool

General | YARN | Submission Access Control | Administration Access Control

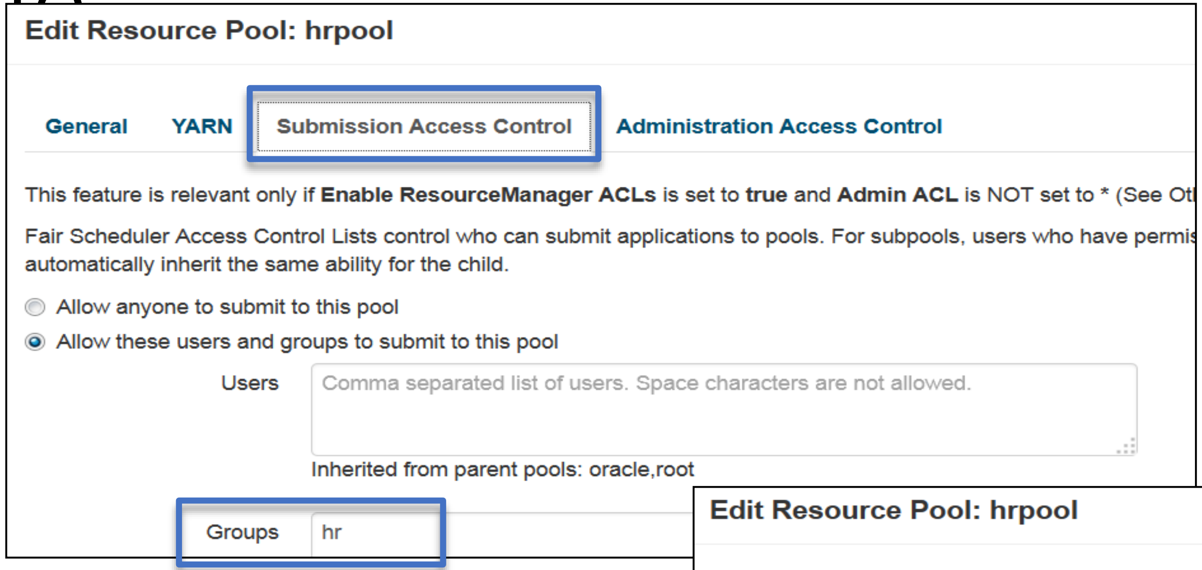
Resource Pool Name: hrpool — Alphanumeric characters only.

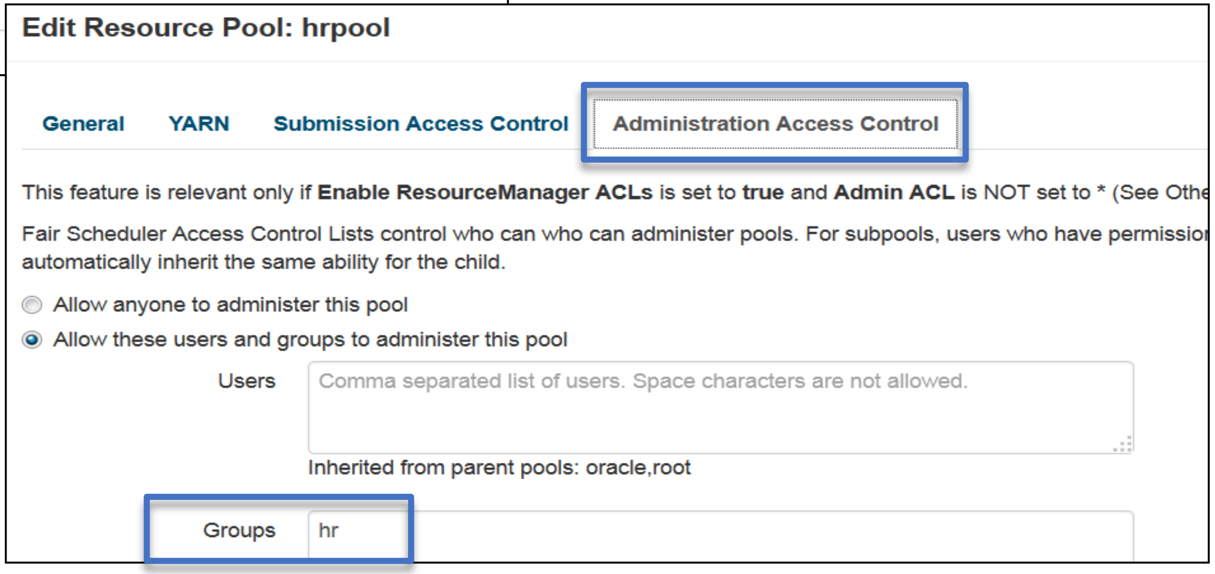
Scheduling Policy:
- (•) DRF: Dominant Resource Fairness. Schedules resources fairly based on both CPU and memory. (Recommended)
- ( ) FAIR: Schedules resources fairly based only on memory.
- ( ) FIFO: First in, first out.

Cloudera Manager Dynamic Resource Management: Example



Cloudera Manager Dynamic Resource Management: Example





Submitting a Job to `hrpool` by the `lucy` User from the `hr` Group

```
[oracle@scaj51bda12 jars]$ id lucy
uid=478(lucy) gid=1005(hr) groups=1005(hr)
[oracle@scaj51bda12 jars]$ kinit lucy
Password for lucy@DEV.ORACLE.COM:
```

`lucy` belongs to the **`hr`** group; therefore, she can submit the job to **`hrpool`**.

```
$ hadoop jar hadoop-mapreduce-examples-2.5.0-cdh5.3.0.jar teragen -
Dmapred.map.tasks=5 -D mapreducqueuename=hrpool 1000000
/user/hr/d2
```

```
15/02/26 10:08:59 INFO mapreduce.Job:  map 0% reduce 0%
15/02/26 10:09:09 INFO mapreduce.Job:  map 40% reduce 0%
15/02/26 10:10:06 INFO mapreduce.Job:  map 60% reduce 0%
15/02/26 10:10:34 INFO mapreduce.Job:  map 100% reduce 0%
15/02/26 10:10:34 INFO mapreduce.Job: Job job_1424929332007_0001 completed successfully
15/02/26 10:10:34 INFO mapreduce.Job: Counters: 31
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=576800
                FILE: Number of read operations=0
```

Monitoring the Status of the Submitted MapReduce Job

output directory

```
[oracle@scaj51bda12 jars]$ hadoop fs -ls /user/hr/d2
Found 6 items
-rw-r--r--   3 lucy hive          0 2015-02-26 10:10 /user/hr/d2/_SUCCESS
-rw-r--r--   3 lucy hive   20000000 2015-02-26 10:10 /user/hr/d2/part-m-00000
-rw-r--r--   3 lucy hive   20000000 2015-02-26 10:10 /user/hr/d2/part-m-00001
-rw-r--r--   3 lucy hive   20000000 2015-02-26 10:09 /user/hr/d2/part-m-00002
-rw-r--r--   3 lucy hive   20000000 2015-02-26 10:09 /user/hr/d2/part-m-00003
-rw-r--r--   3 lucy hive   20000000 2015-02-26 10:10 /user/hr/d2/part-m-00004
[oracle@scaj51bda12 jars]$
```

Logged in as: dr.who

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	1	0	0 B	159 GB	0 B	0	248	0	4	0	0	0	0

User Metrics for dr.who

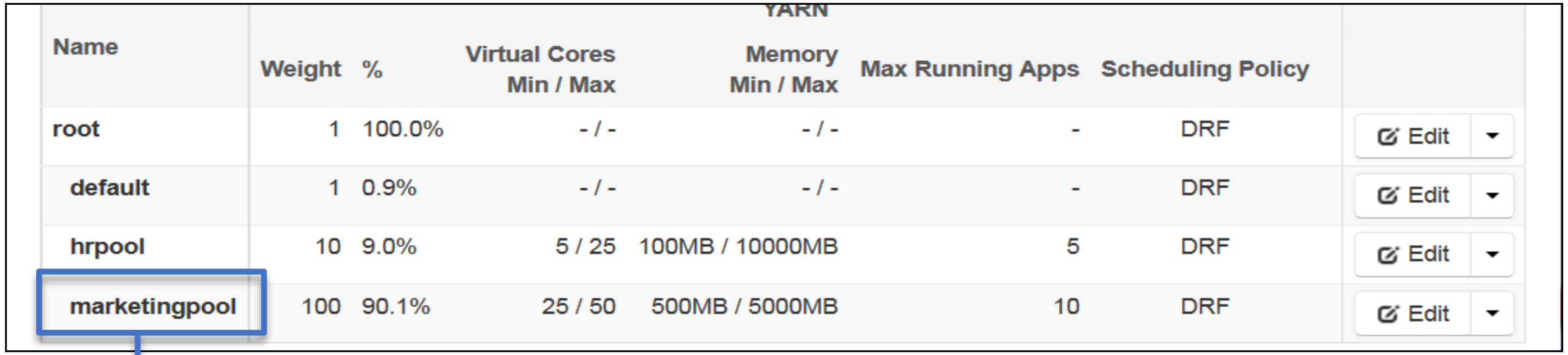
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	0	1	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries Search:

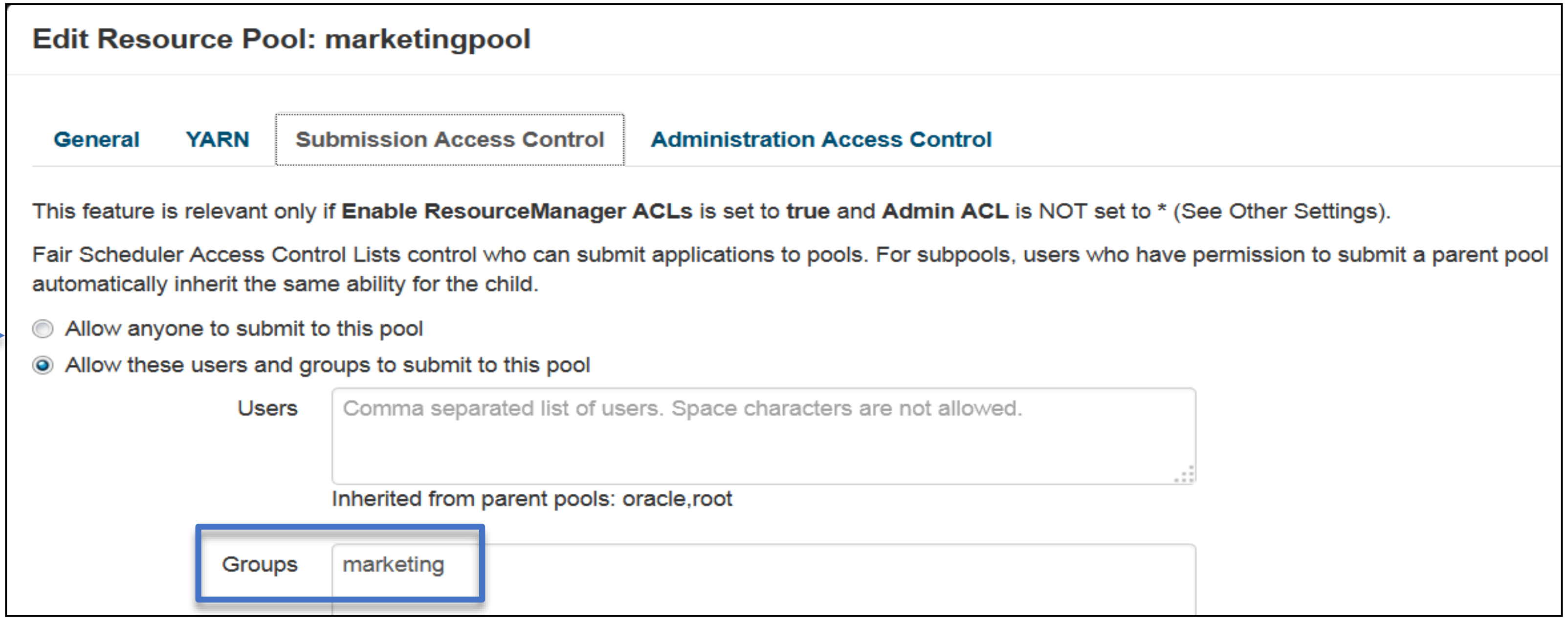
ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1424929332007_0001	lucy	TeraGen	MAPREDUCE	root.hrpool	Thu Feb 26 13:07:21 -0500 2015	Thu Feb 26 13:10:32 -0500 2015	FINISHED	SUCCEEDED		History

The job was submitted to the `root.hrpool`.

Examining `marketingpool`



Name	Weight	%	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Scheduling Policy	
	YARN						
root	1	100.0%	- / -	- / -	-	DRF	Edit
default	1	0.9%	- / -	- / -	-	DRF	Edit
hrpool	10	9.0%	5 / 25	100MB / 10000MB	5	DRF	Edit
marketingpool	100	90.1%	25 / 50	500MB / 5000MB	10	DRF	Edit



Edit Resource Pool: marketingpool

General | YARN | Submission Access Control | Administration Access Control

This feature is relevant only if **Enable ResourceManager ACLs** is set to **true** and **Admin ACL** is NOT set to * (See Other Settings).

Fair Scheduler Access Control Lists control who can submit applications to pools. For subpools, users who have permission to submit a parent pool automatically inherit the same ability for the child.

- ○ Allow anyone to submit to this pool
- ◉ Allow these users and groups to submit to this pool

Users: Comma separated list of users. Space characters are not allowed.

Inherited from parent pools: oracle,root

Groups: marketing

Submitting a Job to `marketingpool` by the `lucy` User from the `hr` Group

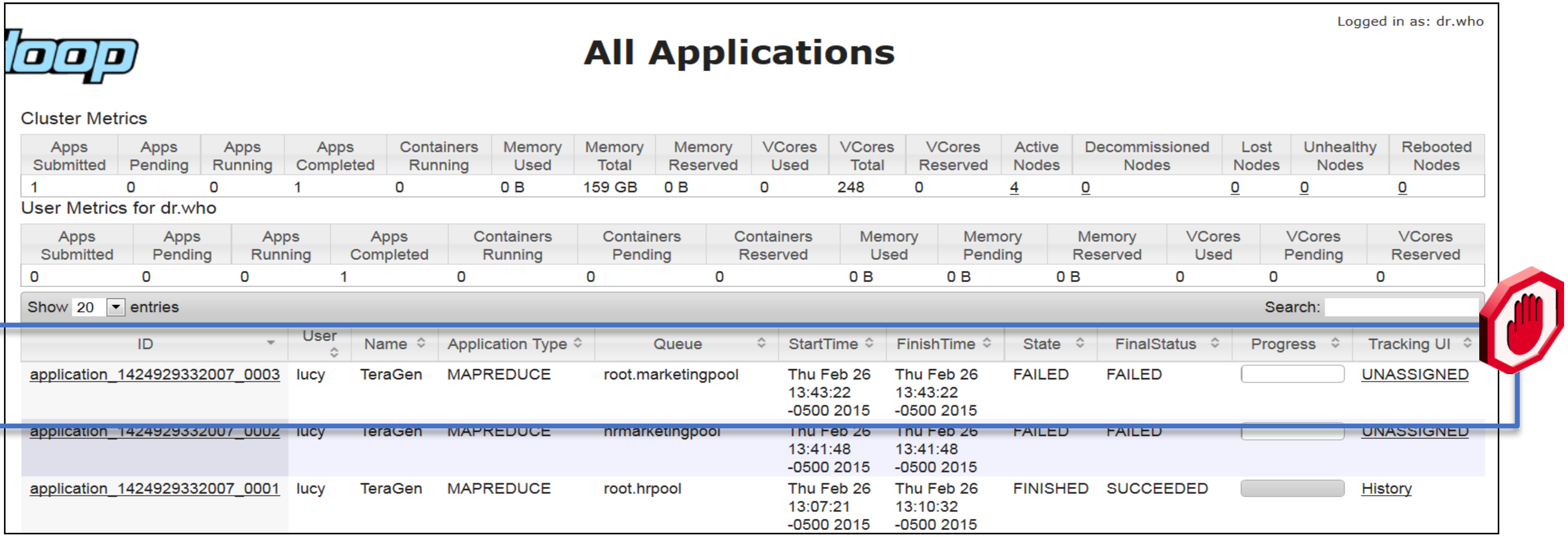
```
[oracle@scaj51bda12 jars]$ id lucy
uid=478(lucy) gid=1005(hr) groups=1005(hr)
[oracle@scaj51bda12 jars]$ kinit lucy
Password for lucy@DEV.ORACLE.COM:
```

`lucy` belongs to the **`hr`** group; therefore, she CANNOT submit the job to **`marketingpool`**.

```
$ hadoop fs -rm -r /user/hr/d2
$ hadoop jar hadoop-mapreduce-examples-2.5.0-cdh5.3.0.jar teragen
-D mapred.map.tasks=5 -D mapreduce.job.queuename=marketingpool
1000000 /user/hr/d2
```

```
15/02/26 10:43:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1424929332007_0003
15/02/26 10:43:21 WARN token.Token: Cannot find class for token kind kms-dt
15/02/26 10:43:21 WARN token.Token: Cannot find class for token kind kms-dt
Kind: kms-dt, Service: 192.168.42.121:16000, Ident: 00 04 6c 75 63 79 04 79 61 72 6e 00 8a 01 4b c7 33 46 ab 8a 01 4b
 eb 3f ca ab 3c 1d
15/02/26 10:43:21 INFO mapreduce.JobSubmitter: Kind: HDFS_DELEGATION_TOKEN, Service: ha-hdfs:scaj51cdh-ns, Ident: (HD
FS_DELEGATION_TOKEN token 202 for lucy)
15/02/26 10:43:22 INFO impl.YarnClientImpl: Submitted application application_1424929332007_0003
15/02/26 10:43:22 INFO mapreduce.JobSubmitter: Cleaning up the staging area /user/lucy/.staging/job_1424929332007_000
3
15/02/26 10:43:22 WARN security.UserGroupInformation: PriviledgedActionException as:lucy@DEV.ORACLE.COM (auth:KERBERO
S) cause:java.io.IOException: Failed to run job : User lucy cannot submit applications to queue root.marketingpool
java.io.IOException: Failed to run job : User lucy cannot submit applications to queue root.marketingpool
```

Monitoring the Status of the Submitted MapReduce Job


Logged in as: dr.who

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	1	0	0 B	159 GB	0 B	0	248	0	4	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	VCores Used	VCores Pending	VCores Reserved
0	0	0	1	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1424929332007_0003	lucy	TeraGen	MAPREDUCE	root.marketingpool	Thu Feb 26 13:43:22 -0500 2015	Thu Feb 26 13:43:22 -0500 2015	FAILED	FAILED		UNASSIGNED
application_1424929332007_0002	lucy	TeraGen	MAPREDUCE	hrmarketingpool	Thu Feb 26 13:41:48 -0500 2015	Thu Feb 26 13:41:48 -0500 2015	FAILED	FAILED		UNASSIGNED
application_1424929332007_0001	lucy	TeraGen	MAPREDUCE	root.hrpool	Thu Feb 26 13:07:21 -0500 2015	Thu Feb 26 13:10:32 -0500 2015	FINISHED	SUCCEEDED		History

Submitting a Job to marketingpool by the bob User from the marketing Group

```
[oracle@scaj51bda12 jars]$ id bob
uid=477(bob) gid=1007(marketing) groups=1007(marketing)
[oracle@scaj51bda12 jars]$ kinit bob
Password for bob@DEV.ORACLE.COM:
[oracle@scaj51bda12 jars]$
```

bob belongs to the **marketing** group; therefore, he can submit the job to **marketingpool.**

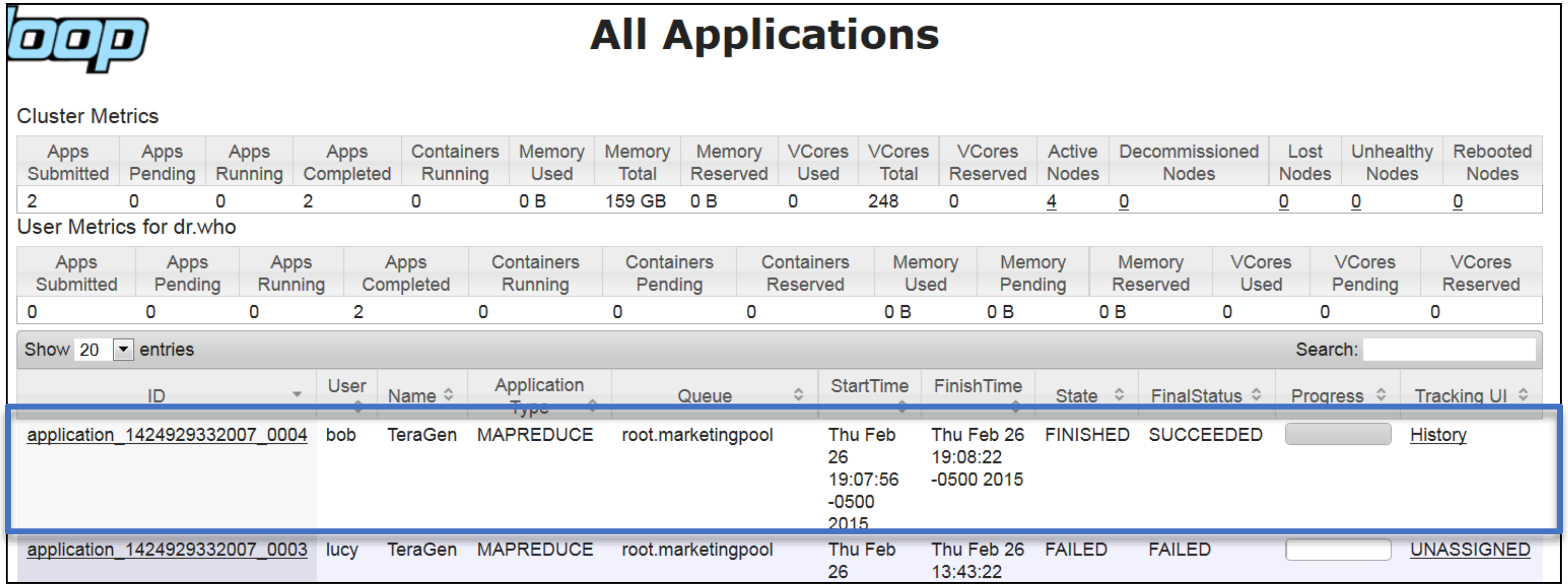
```
$ hadoop fs -rm -r /user/marketing/d1
$ hadoop jar hadoop-mapreduce-examples-2.5.0-cdh5.3.0.jar teragen
-Dmapred.map.tasks=5 -D mapreduce.job.queuename=marketingpool
1000000 /user/marketing/d1
```

```
15/02/26 16:07:57 INFO mapreduce.Job: The url to track the job: http://scaj51bda
12.us.oracle.com:8088/proxy/application_1424929332007_0004/
15/02/26 16:07:57 INFO mapreduce.Job: Running job: job_1424929332007_0004
15/02/26 16:08:09 INFO mapreduce.Job: Job job_1424929332007_0004 running in uber
 mode : false
15/02/26 16:08:09 INFO mapreduce.Job:  map 0% reduce 0%
15/02/26 16:08:19 INFO mapreduce.Job:  map 80% reduce 0%
15/02/26 16:08:23 INFO mapreduce.Job:  map 100% reduce 0%
15/02/26 16:08:23 INFO mapreduce.Job: Job job_1424929332007_0004 completed succe
ssfully
```

Monitoring the Status of the Submitted MapReduce Job

output directory

```
[oracle@scaj51bda12 jars]$ hadoop fs -ls /user/marketing/d1
Found 6 items
-rw-r--r--   3 bob marketing          0 2015-02-26 16:08 /user/marketing/d1/_SUC
CESS
-rw-r--r--   3 bob marketing   20000000 2015-02-26 16:08 /user/marketing/d1/part
-m-00000
-rw-r--r--   3 bob marketing   20000000 2015-02-26 16:08 /user/marketing/d1/part
-m-00001
-rw-r--r--   3 bob marketing   20000000 2015-02-26 16:08 /user/marketing/d1/part
-m-00002
-rw-r--r--   3 bob marketing   20000000 2015-02-26 16:08 /user/marketing/d1/part
-m-00003
-rw-r--r--   3 bob marketing   20000000 2015-02-26 16:08 /user/marketing/d1/part
-m-00004
[oracle@scaj51bda12 jars]$
```



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
2	0	0	2	0	0 B	159 GB	0 B	0	248	0	4	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	VCores Used	VCores Pending	VCores Reserved
0	0	0	2	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1424929332007_0004	bob	TeraGen	MAPREDUCE	root.marketingpool	Thu Feb 26 19:07:56 -0500 2015	Thu Feb 26 19:08:22 -0500 2015	FINISHED	SUCCEEDED		History
application_1424929332007_0003	lucy	TeraGen	MAPREDUCE	root.marketingpool	Thu Feb 26	Thu Feb 26 13:43:22	FAILED	FAILED		UNASSIGNED

Delay Scheduling

- YARN schedulers attempt to honor locality container requests.
- If the requested node is not available, delaying the granting of an alternative resource for a few seconds can increase the chances of getting a container on the requested node.
- Delay scheduling is supported by both Capacity and Fair Schedulers.

Agenda

- Job Scheduling in YARN
- YARN `application` command

YARN `application` Command

- You can use the `yarn application` command to list and kill applications:

```
$ yarn application <options>
```

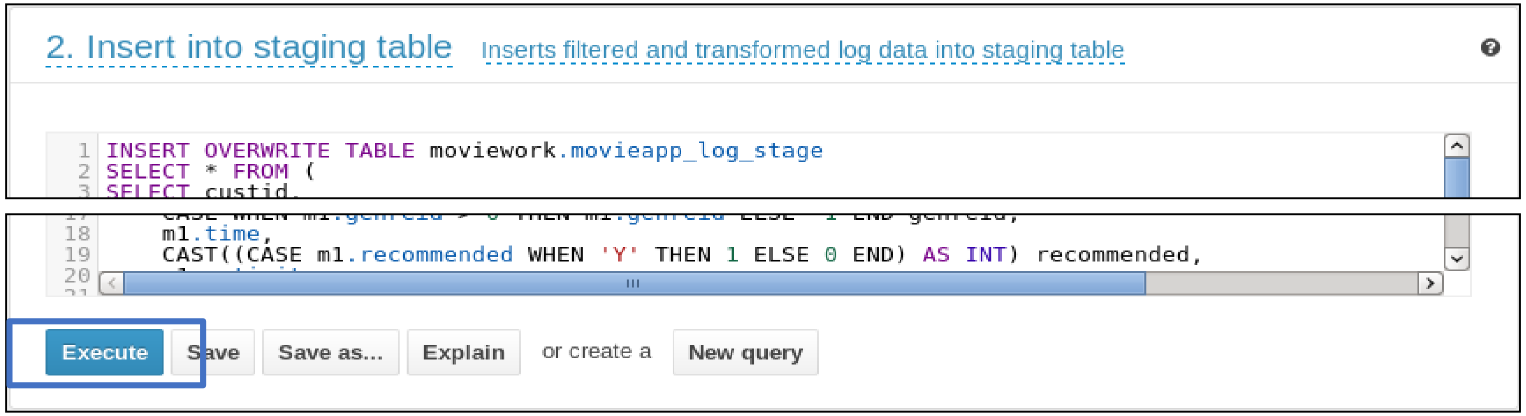
<options>	Description
`-list`	Lists applications from the Resource Manager
`-appStates`	Works with `-list` to filter applications based on input comma-separated list of application states. The options are: `ALL`, `NEW`, `NEW_SAVING`, `SUBMITTED`, `ACCEPTED`, `RUNNING`, `FINISHED`, `FAILED`, `KILLED`.
`-status ApplicationID`	Prints the status of the application
`- kill ApplicationID`	Kills the application

YARN `application` Command: Example

```
$ yarn application -list
```

```
[oracle@bigdatalite ~]$ yarn application -list
15/02/12 09:00:00 INFO client.RMProxy: Connecting to ResourceManager at localhos
t/127.0.0.1:8032
Total number of applications (application-types: [] and states: [SUBMITTED, ACCE
PTED, RUNNING]):0
                Application-Id      Application-Name        Application-Type
     User            Queue                    State             Final-State
      Progress                       Tracking-URL
[oracle@bigdatalite ~]$
```

1



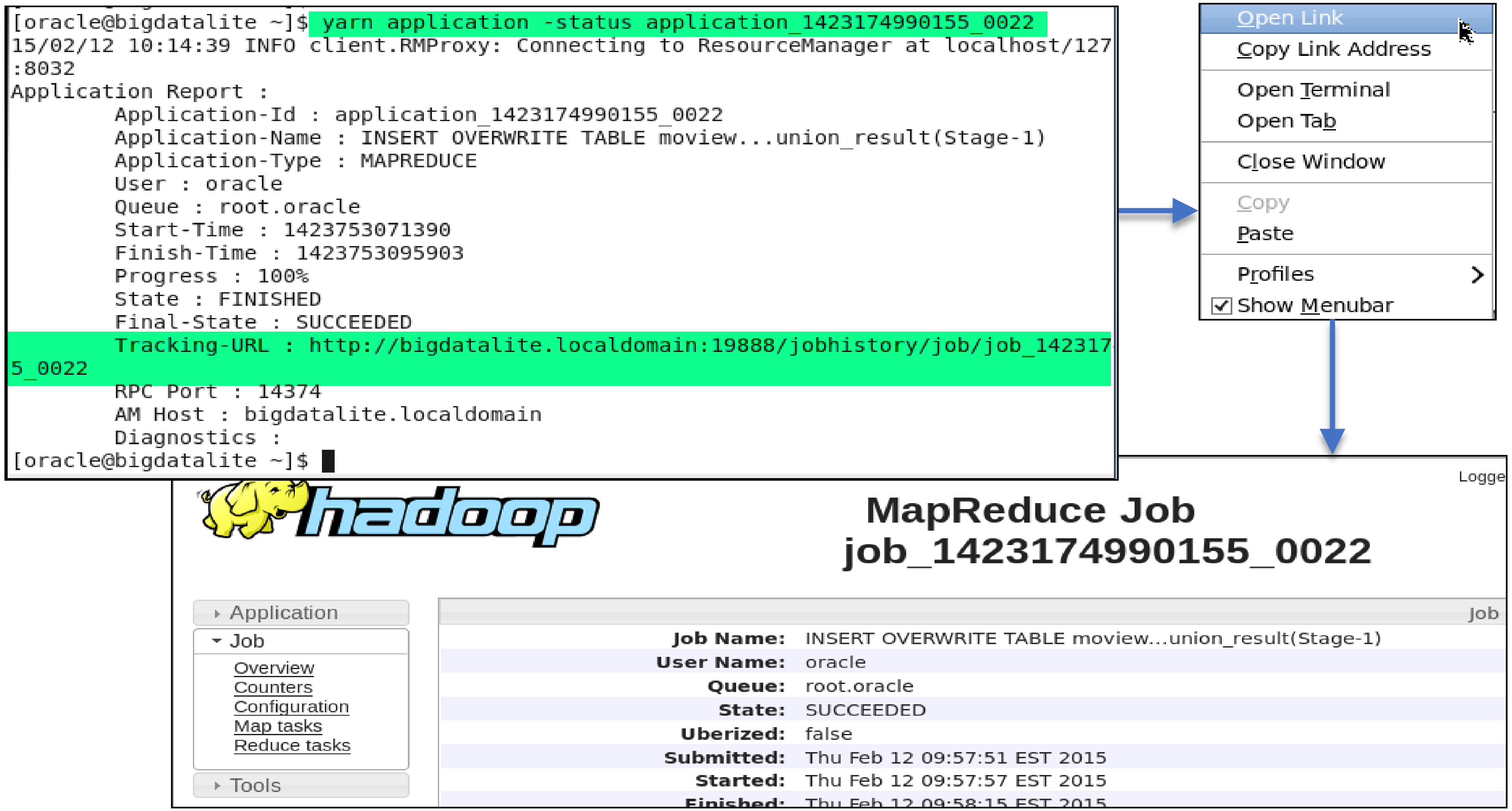
2

```
[oracle@bigdatalite ~]$ yarn application -list
15/02/12 09:57:57 INFO client.RMProxy: Connecting to ResourceManager at localhost/127
:8032
Total number of applications (application-types: [] and states: [SUBMITTED, ACCEPTED,
NG]):1
                Application-Id      Application-Name        Application-Type
    Queue                    State             Final-State              Progress
            Tracking-URL
application_1423174990155_0022  INSERT OVERWRITE TABLE moview...union_result(Stage-1)
        MAPREDUCE             oracle       root.oracle                RUNNING
DEFINED              5% http://bigdatalite.localdomain:54444
```

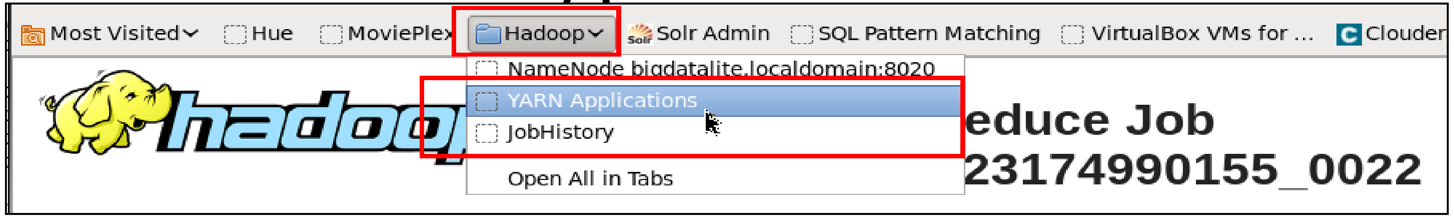
3

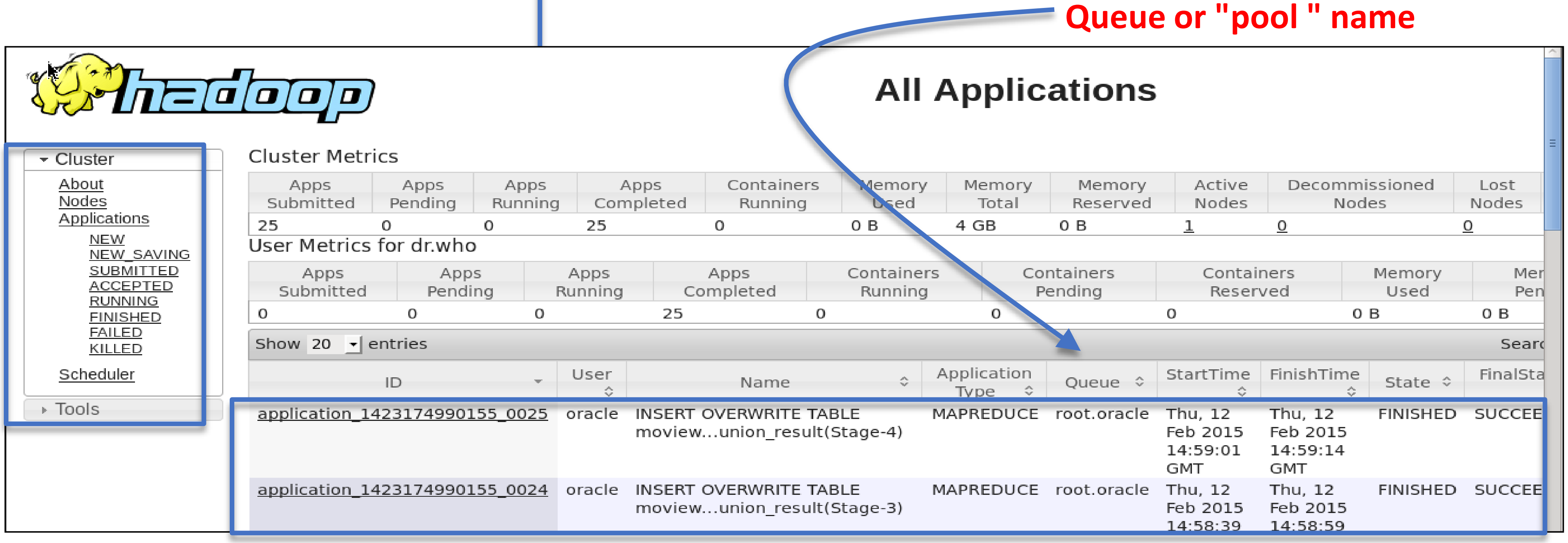
YARN `application` Command: Example

```
$ yarn application -status <job_id>
```

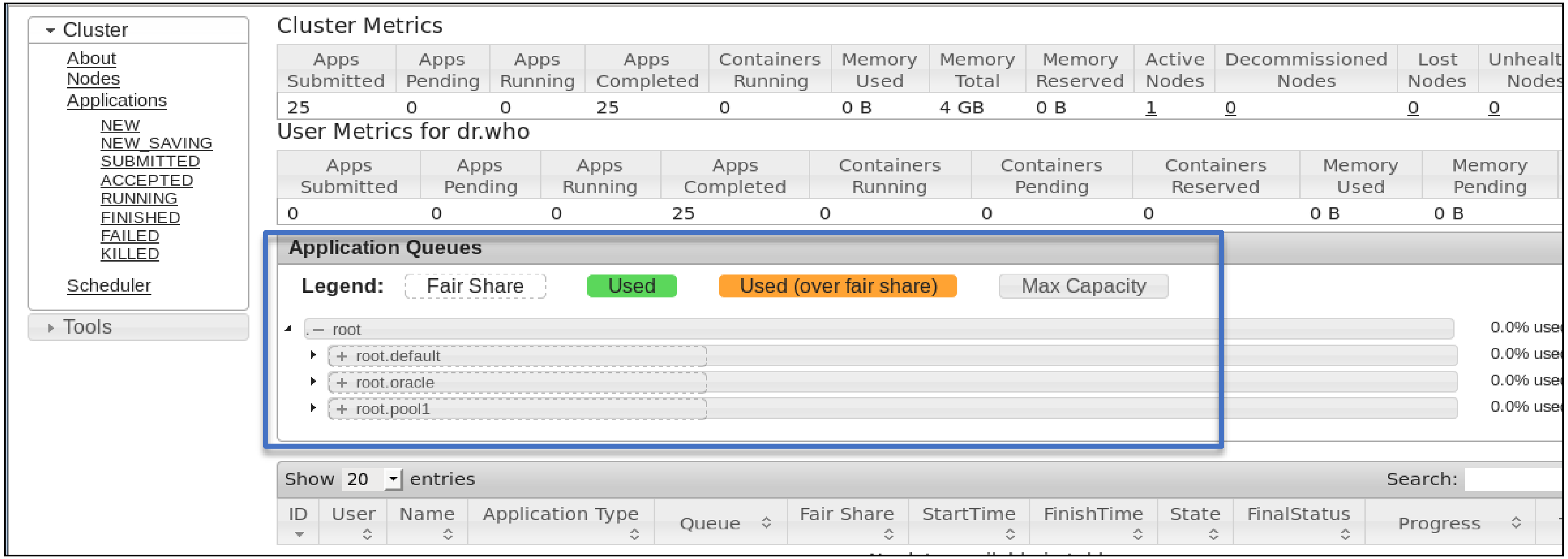


Monitoring an Application by Using the ResourceManager Web UI





Scheduler: BDA Example



Summary

- In this lesson, you should have learned how to:
  - Use YARN to manage resources efficiently
  - Use the YARN `application` command

Practice 9

- Monitoring a Running Job by Using the CLI
- Monitoring a Running Job by Using the Resource Manager UI

Overview of Spark

Course Road Map

- Module 1: Big Data Fundamentals
- Module 2: Data Acquisition and Storage
- **Module 3: Data Access and Processing**
- Module 4: Data Unification
- Module 5: Data Analysis
- Module 6: Big Data Deployment Options

- Lesson 8: Introduction to MapReduce and YARN Processing Frameworks
- Lesson 9: Resource Management Using YARN
- **Lesson 10: Overview of Spark** ✓
- Lesson 11: Overview of Apache Hive
- Lesson 12: Overview of Cloudera Impala
- Lesson 13: Using Oracle XQuery for Hadoop
- Lesson 14: Overview of Solr

Objectives

- After completing this lesson, you should be able to:
  - Describe Spark and identify its benefits
  - Identify Spark's components and architecture
  - Interact with Spark
  - Run a Spark job and monitor the job by using YARN's Resource Manager Web UI

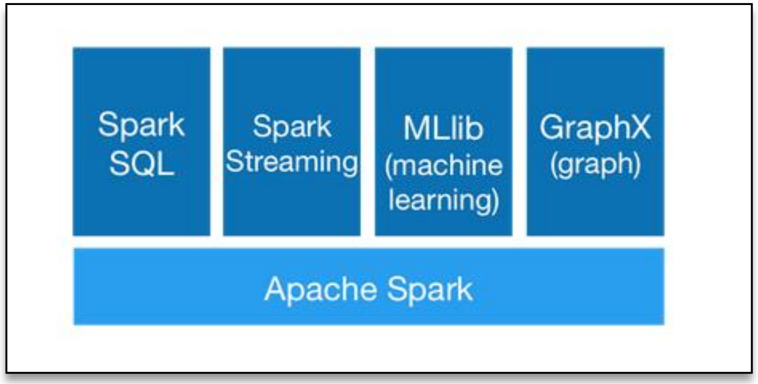
What Is Apache Spark?

- A fast and general engine for large-scale (and multi-stage processing routines) data processing
- An open-source, parallel data-processing framework with a proven scalability
- Makes it easy to develop fast, unified big data applications combining batch, streaming, and interactive analytics on all your data
- Hides complex distributed computing tasks from the developer so that they can focus on writing the Map and Reduce functions (abstraction layer)
- An alternative to using traditional MapReduce on Hadoop, which does not work well for interactive queries or real-time, low-latency applications
  - A major disadvantage of Hadoop's MapReduce implementation was its persistence of intermediate data to disk between the Map and Reduce processing phases.

APACHE Spark™ *Lightning-fast cluster computing*

Benefits of Using Spark

- Speed (much faster than MapReduce)
- Ease of use (higher level of abstraction than MapReduce)
- Sophisticated analytics
- Runs on various cluster managers: YARN, stand-alone (built-in) scheduler, and Mesos
- A cross-platform application capable of running on Windows and Linux
- Native integration with programming interfaces such as Java, Python, Scala, SQL, and R



Source: Apache Spark page on http://spark.apache.org/

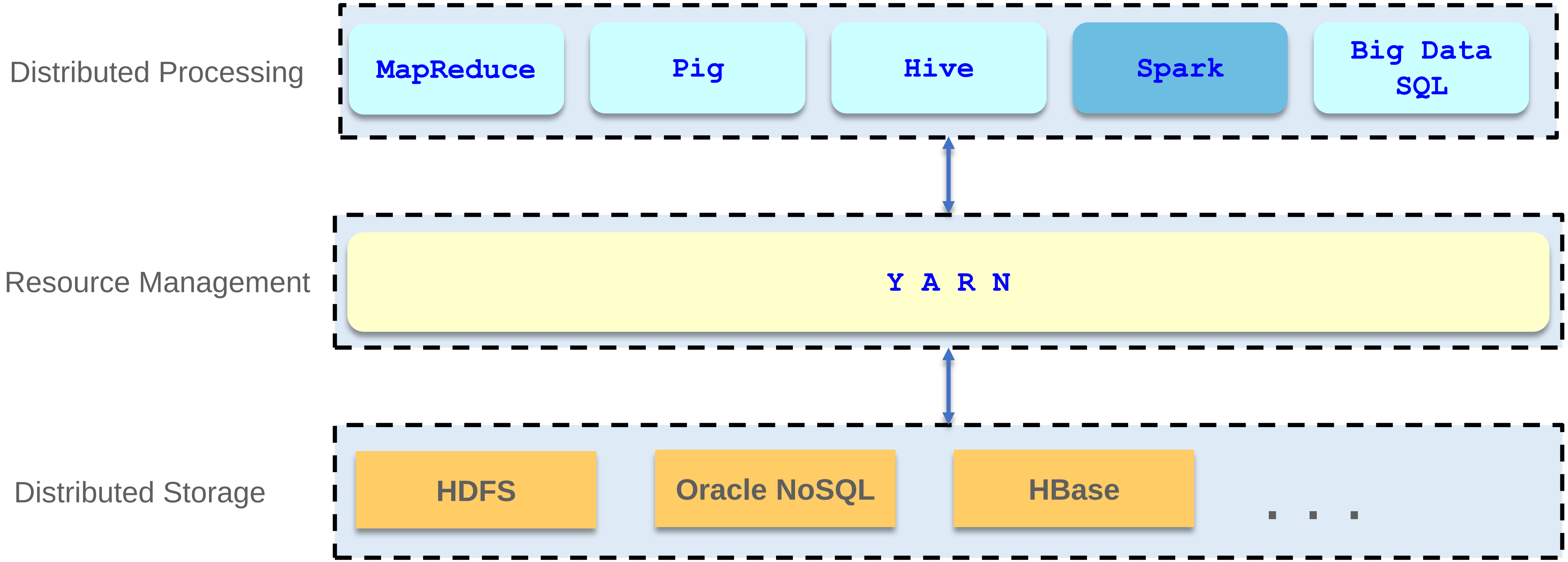
Spark Versus MapReduce

Apache Spark	Hadoop MapReduce
Spark stores data in memory.	MapReduce stores data on disk.
Spark can be used to perform streaming, batch processing, and machine learning all in the same cluster.	MapReduce is used for generating reports that help find answers to historical queries. It performs only batch processing.
Spark is written in Scala.	MapReduce is written in Java.
Spark executes jobs 10 to 100 times faster than Hadoop MapReduce.	Hadoop MapReduce doesn’t leverage the memory of the Hadoop cluster to the maximum.
Spark is easy to program.	MapReduce is difficult to program.

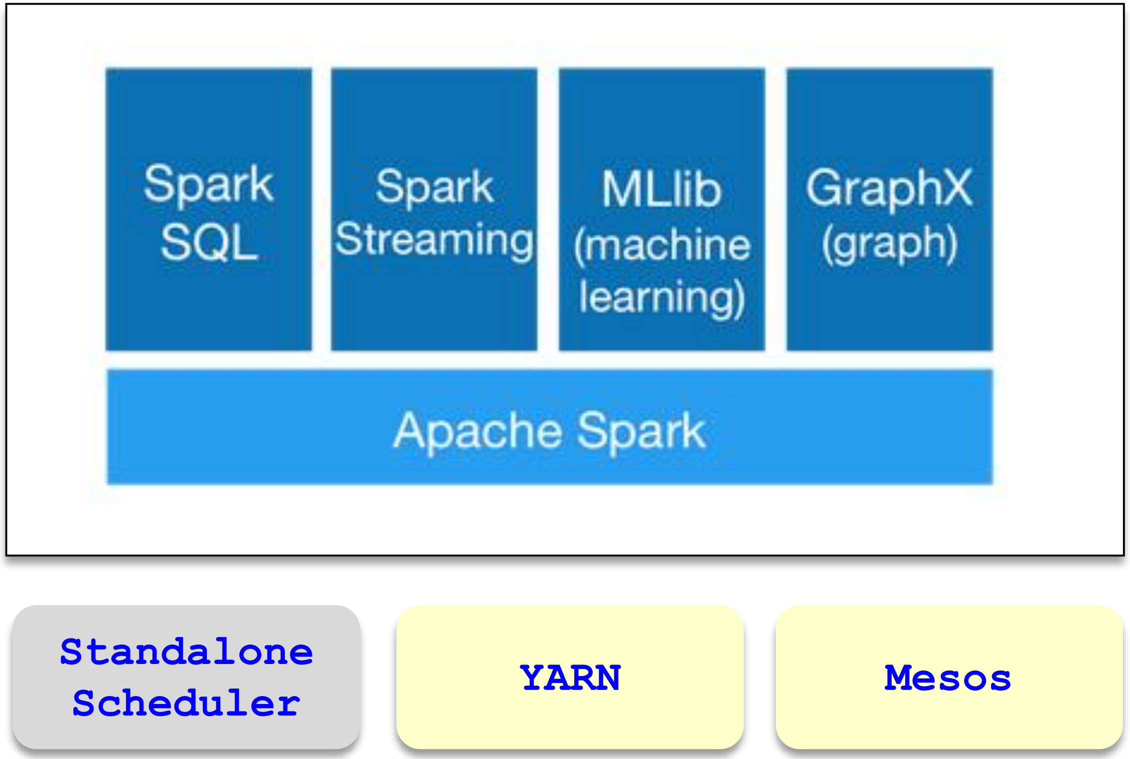
How is Spark Used?

- Real-time data analysis
  - Fraud detection
  - Website monitoring
  - Network monitoring
  - Web clicks and advertising
- Customer segmentation for marketing purposes, and sentiment analysis
- Machine learning and advanced analytics

Spark Architecture



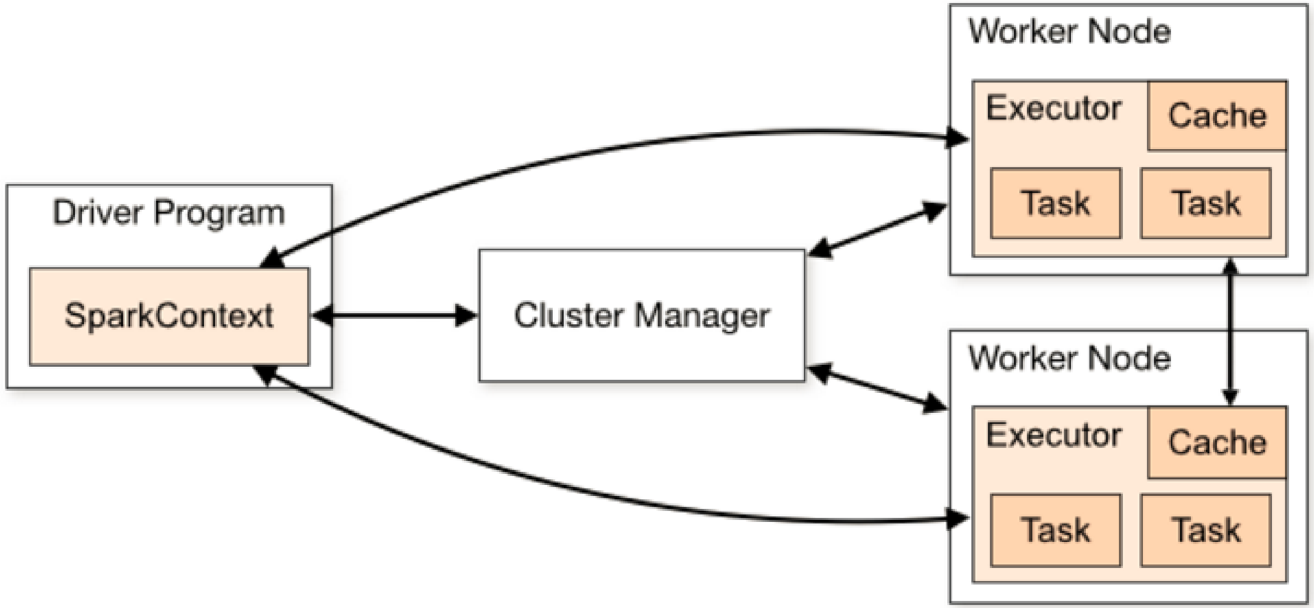
Spark Architecture



Available Cluster Managers/Schedulers

Spark Application Components

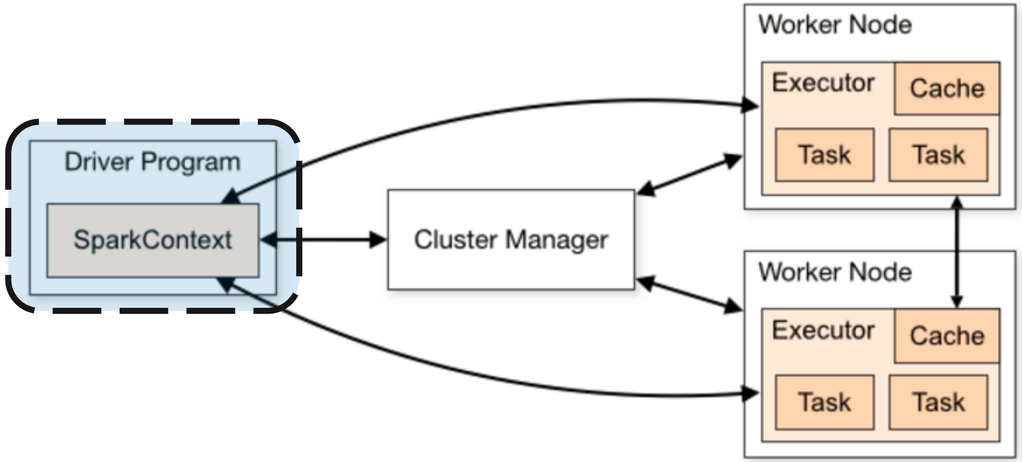
- Spark Driver
- Spark Master
- Cluster Manager
- Executors



Source: Apache Spark page on http://spark.apache.org/docs/latest/cluster-overview.html

Spark Driver

- Runs in the Application Master container (if you are using YARN)
- Creates the `SparkContext`, which is the application instance representing the connection to the Spark master and Spark worker or slave nodes in the cluster
- Plans, schedules, and monitors the Spark application
- Hadoop injects the Spark code into the driver (similar to what happens in MapReduce)
- Controls the job or the interactive query that you are running
- Returns the application's results to the cl



Spark Driver

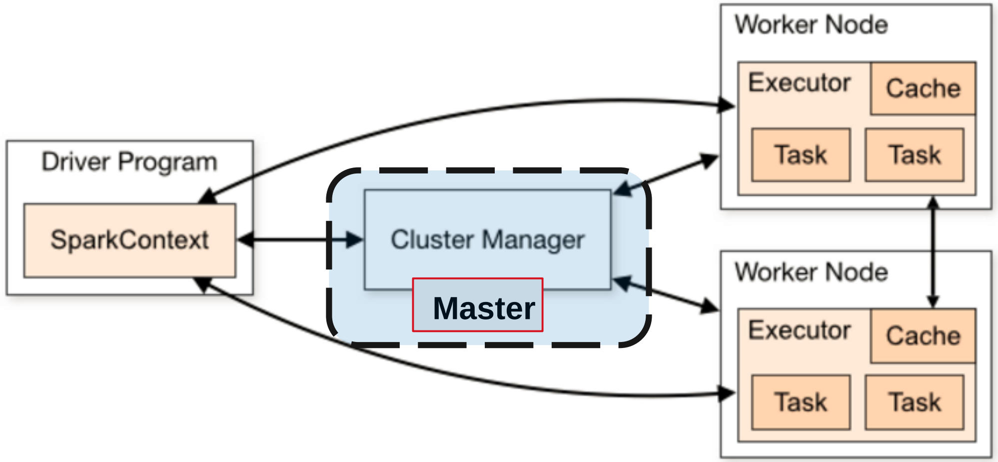
The driver:

- Takes the input and plans the execution of the Spark program
- Uses all of the requested transformations (data manipulation operations) and actions (requests for output or a prompt to execute the program) and creates a directed acyclic graph (DAG).
  - The DAG consists of *tasks* and *stages*.
    - Tasks are the smallest unit of schedulable work in a Spark program.
    - Stages are sets of tasks that can be run together. Stages are dependent on one another.
- Coordinates the running of stages and tasks defined in the DAG

•

Spark Master

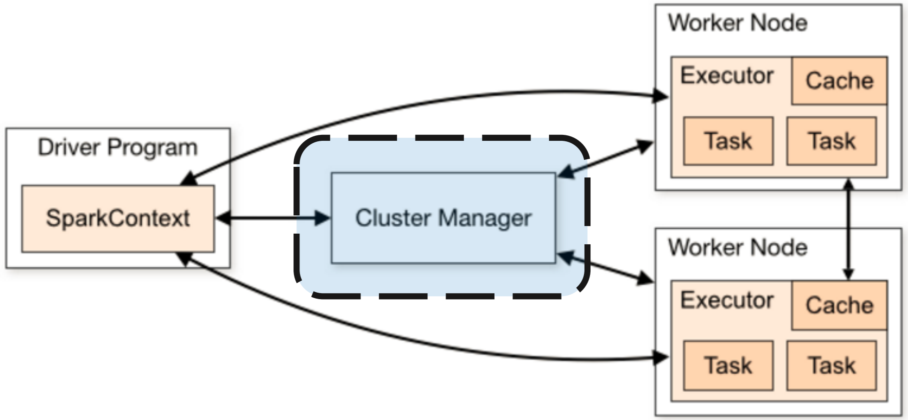
- The Spark master and the cluster manager are the central processes that monitor, reserve, and allocate the distributed cluster resources (containers in the case of YARN) on which the executors run.
- The Spark master and the cluster manager can be separate processes or they can be combined into one process, as is the case when running Spark in Standalone mode.
- The Spark master is the process that requests resources in the cluster and makes these available to the Spark driver.
- In either deployment mode, the master negotiates resources or containers with worker nodes and tracks their status and monitors their progress.



Cluster Manager

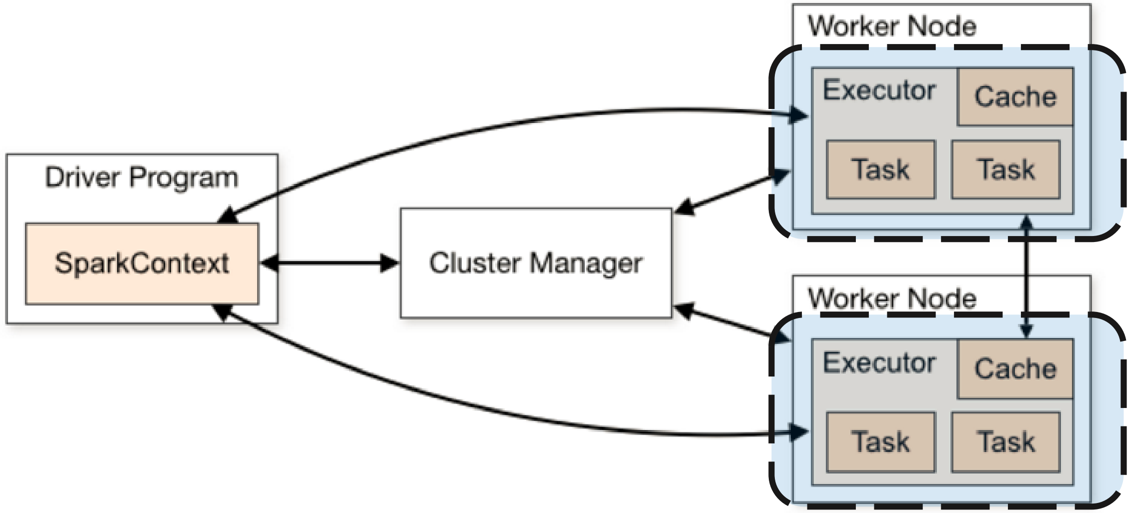
- Spark currently supports three cluster managers:
  - Standalone
    - The Master process is also a simple cluster manager.
  - YARN (ResourceManager)
    - You can submit Spark applications in YARN by using two submission modes:
      - `yarn-client`
      - `yarn-cluster`
  - Apache Mesos
    - A general cluster manager that can also run Hadoop MapReduce and service applications

Note: Apache Mesos is not covered in this course

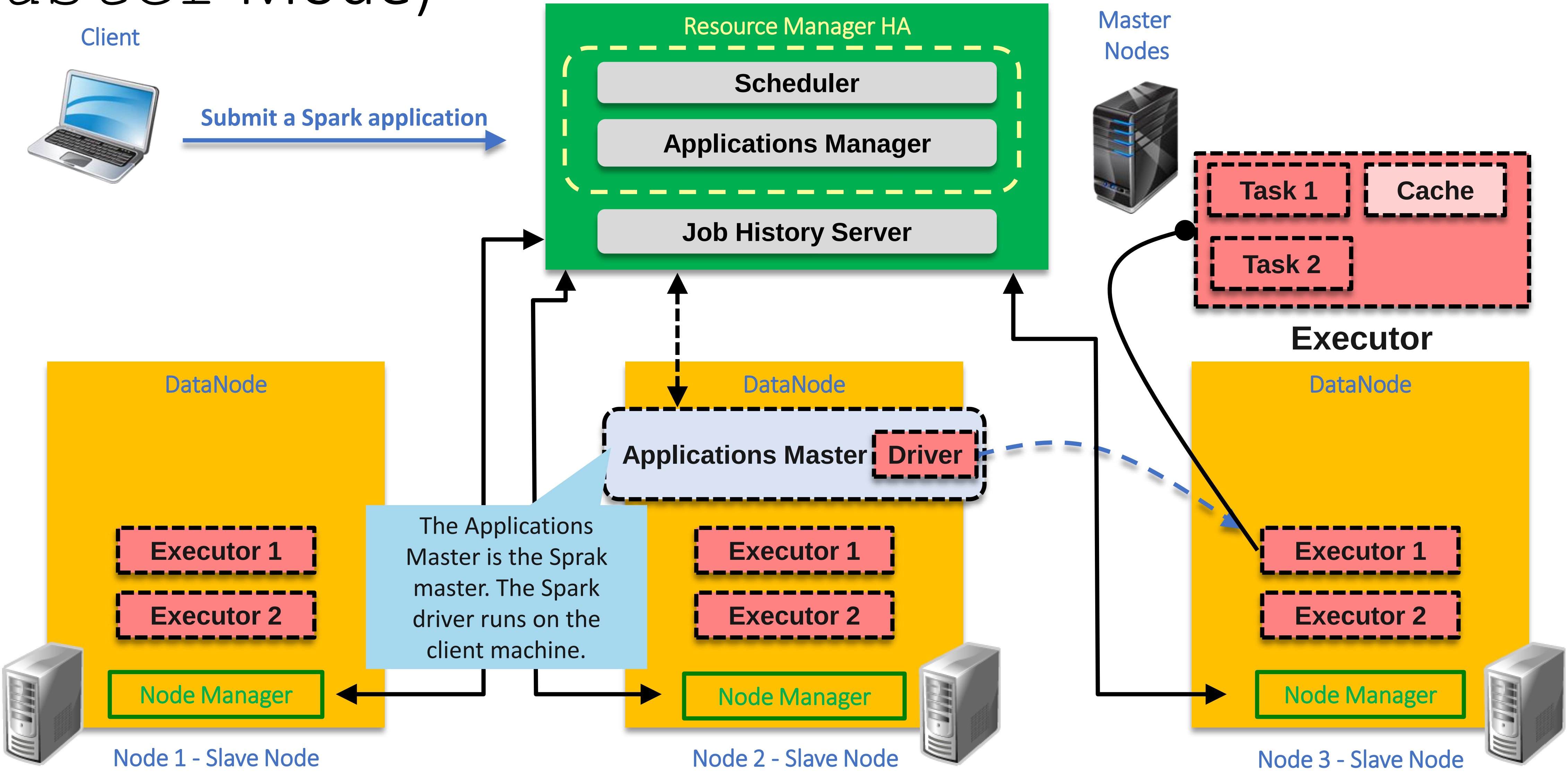


Executors

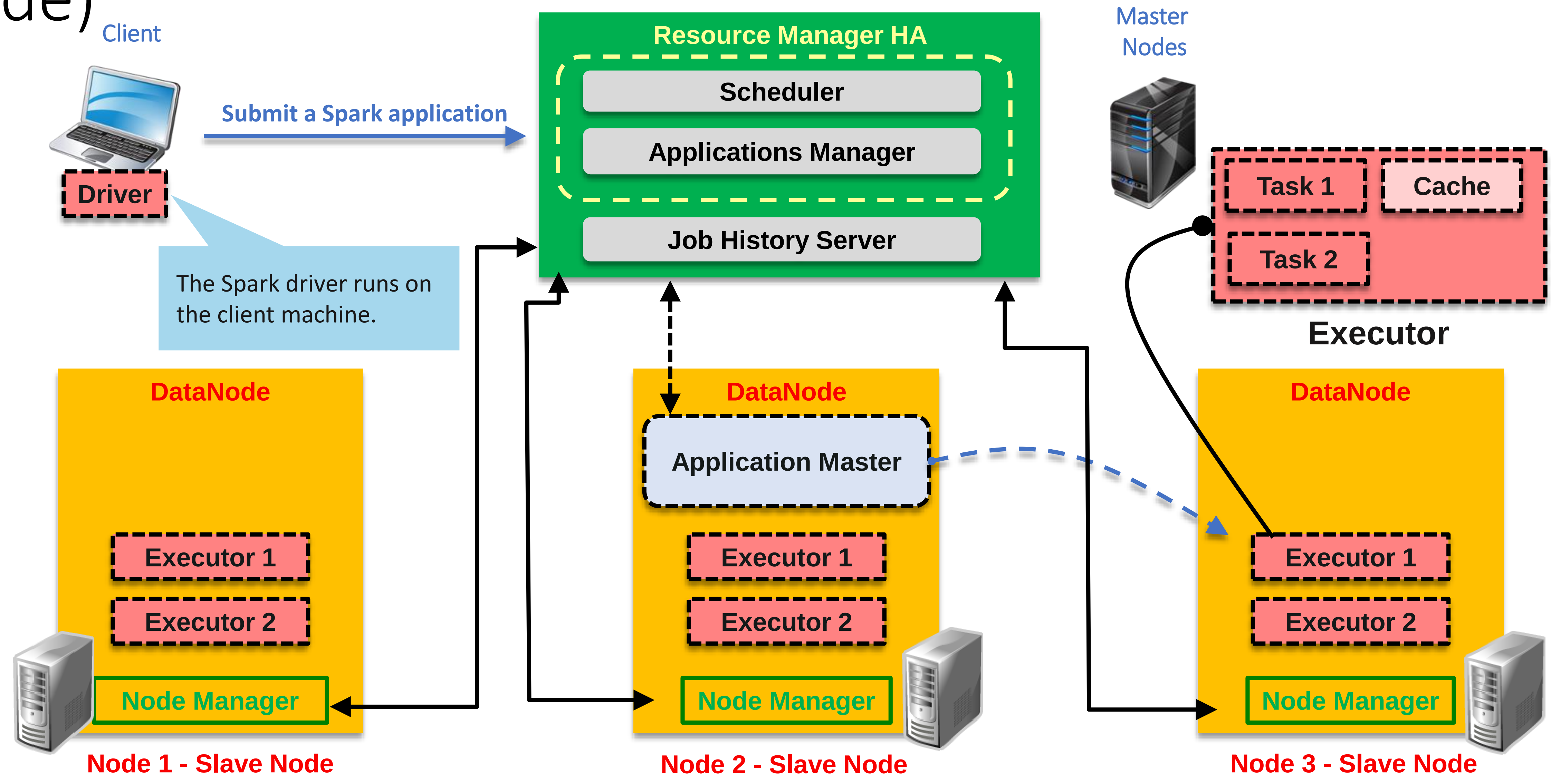
- Spark executors are the host processes on which tasks from a Spark DAG are run.
- Executors reserve CPU and memory resources on slave nodes or workers in a Spark cluster.
- Executors are dedicated to a specific Spark application and terminated when the application completes.



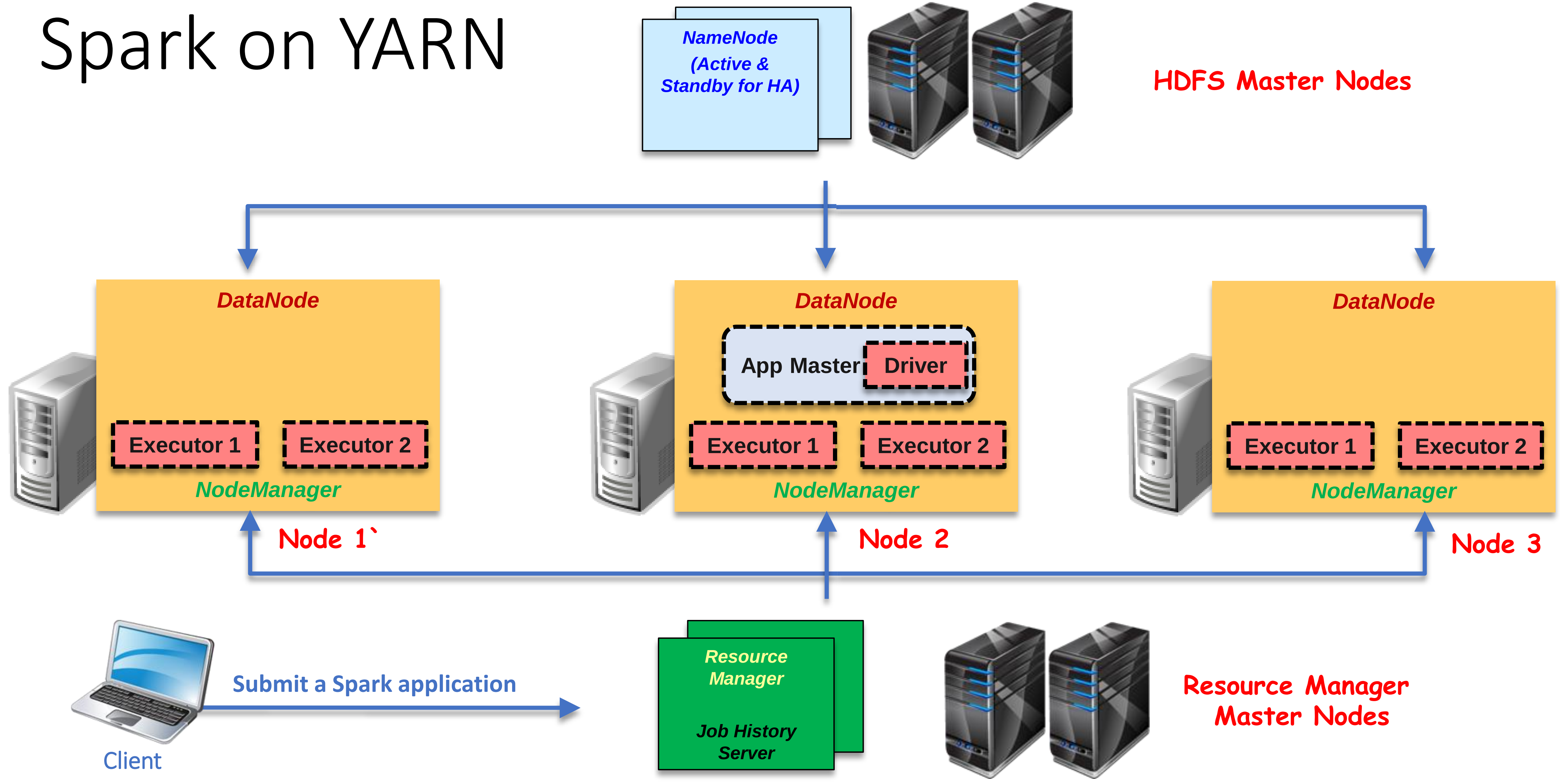
Running a Spark Application on YARN (`yarn-cluster` Mode)



Running a Spark Application on YARN (`yarn-client` Mode)



Spark on YARN



Resilient Distributed Dataset (RDD)

- Spark implements a distributed, fault tolerant, ***in-memory*** structure called a Resilient Distributed Dataset (RDD). RDD is the basic level of abstraction in Spark.
- Spark maximizes the use of memory across multiple machines, improving overall performance. Spark's reuse of these in-memory structures makes it well suited for iterative, machine learning operations, and interactive queries.
- RDD represents an immutable, partitioned collection of elements that can be operated on in parallel.
- RDD is input for your Spark jobs
- Spark provides two ways to create RDDs:
  - Loading an external data set (such as files, SQL or NoSQL datastores), or programmatically
  - Parallelizing a collection in your driver program

Resilient Distributed Dataset (RDD)

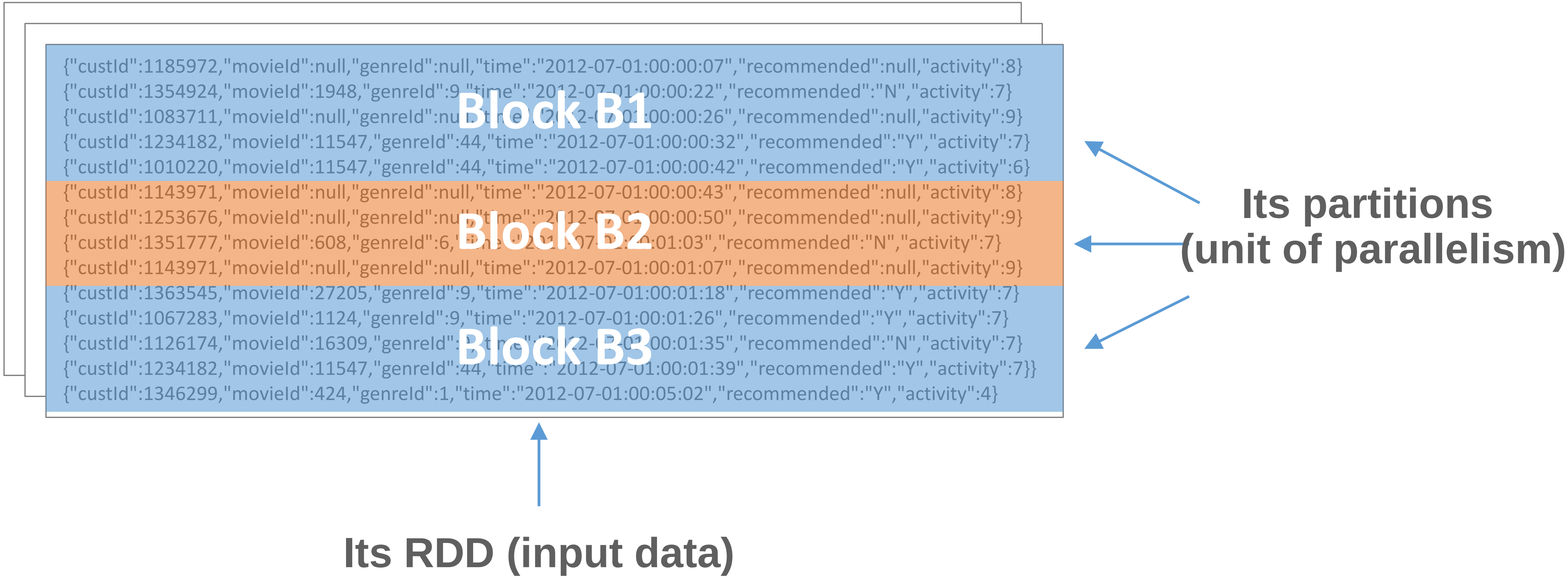
- Resilient:
  - If a node that is performing an application in Spark is lost, the data set can be reconstructed using the sequence of steps required to create the RDD.
- Distributed:
  - Data in RDDs is divided into one or many partitions and distributed as in-memory collections of objects across worker nodes in the cluster.
- Data set:
  - RDDs are data sets that consist of records.
  - Records are uniquely identifiable data collections within a data set.
  - Records could be a collection of fields similar to a row in a table in a relational database, a line of text in a file, or some other formats.

RDDs

- An RDD is the most fundamental data object used in Spark programming.
- RDDs are data sets within a Spark application. This includes:
  - The initial data set(s) loaded
  - Any intermediate data sets
  - The final resultant data set(s)
- Most Spark programming consists of creating new RDDs by performing operations on existing RDDs.
- RDDs are typically stored in memory and not on disk, a big advantage over MapReduce where intermediate data is stored on disk.

RDD Terminology

Example: 1TB File: hdfs://namespace-ns/tmp/example-data/file.dat



RDD Operations

Data manipulation operations

Actions: Requests for output or a prompt to execute the program

Transformations

- Map
- Filter
- Sample
- Union
- Groupbykey
- Reducebykey
- Join & cache

Parallel Operations

- Reduce
- Collect
- Count
- Save
- lookupkey

Commonly Used Transformations

Transformations	Description
`map()`	Returns a new RDD by applying a function to each element of the source
`Filter()`	Creates a new RDD by passing in the supplied function used to filter the results
`reduceByKey()`	Returns data set (K, V) pairs where the value for each key is aggregated using the given reduce function
`flatMap()`	Returns a new RDD by applying a function to each element of the source. The function returns a sequence rather than a single item.

Sample Actions

Actions	Description
`count()`	Returns the number of elements in the data set
`reduce()`	Aggregates the elements of the data set using a function func (which takes two arguments and returns one)
`collect()`	Returns all the elements of the data set as an array at the driver program
`take(n)`	Returns an array with the first n elements of the data set
`first()`	Returns the first element of the data set (similar to take(1))

RDD Example

```
val files = sc.textFile("input.txt")
```

File: input.txt

Hello welcome to Hadoop World
Hello welcome to Spark
This is a Scala program

RDD: files

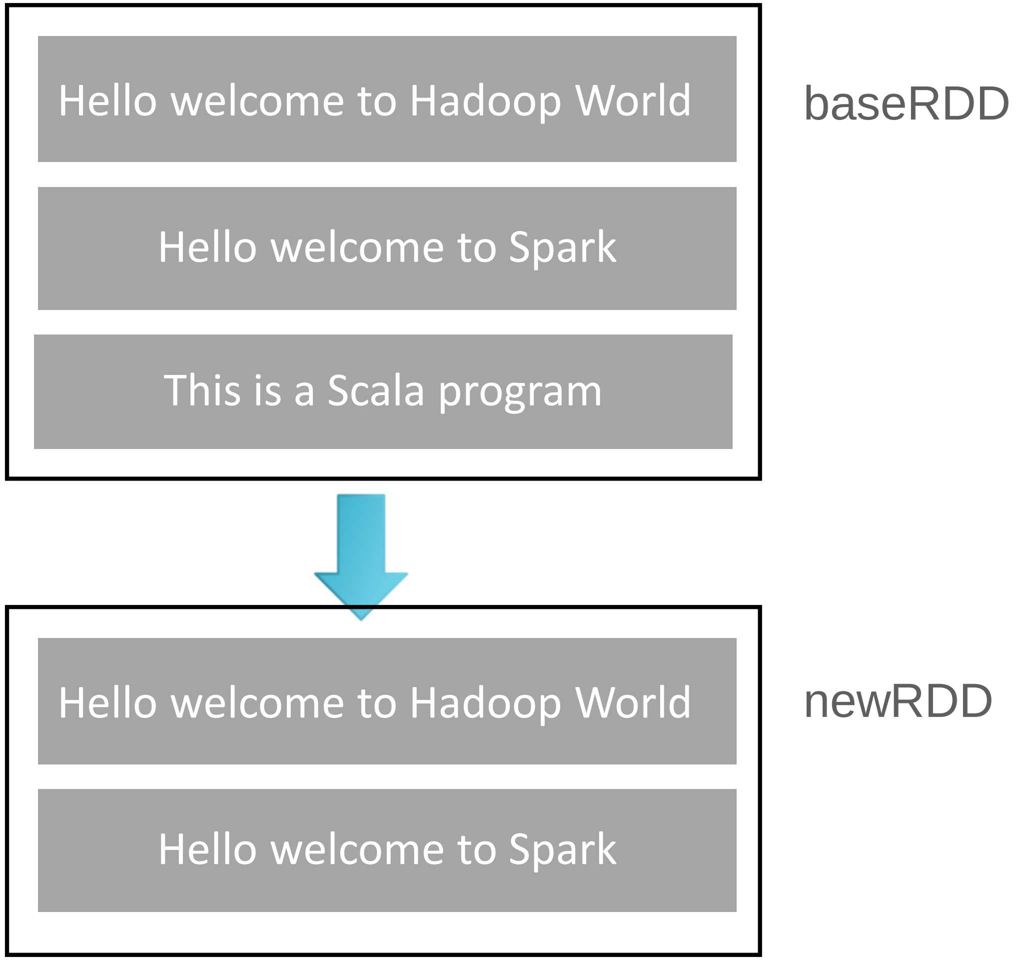
Hello welcome to Hadoop World

Hello welcome to Spark

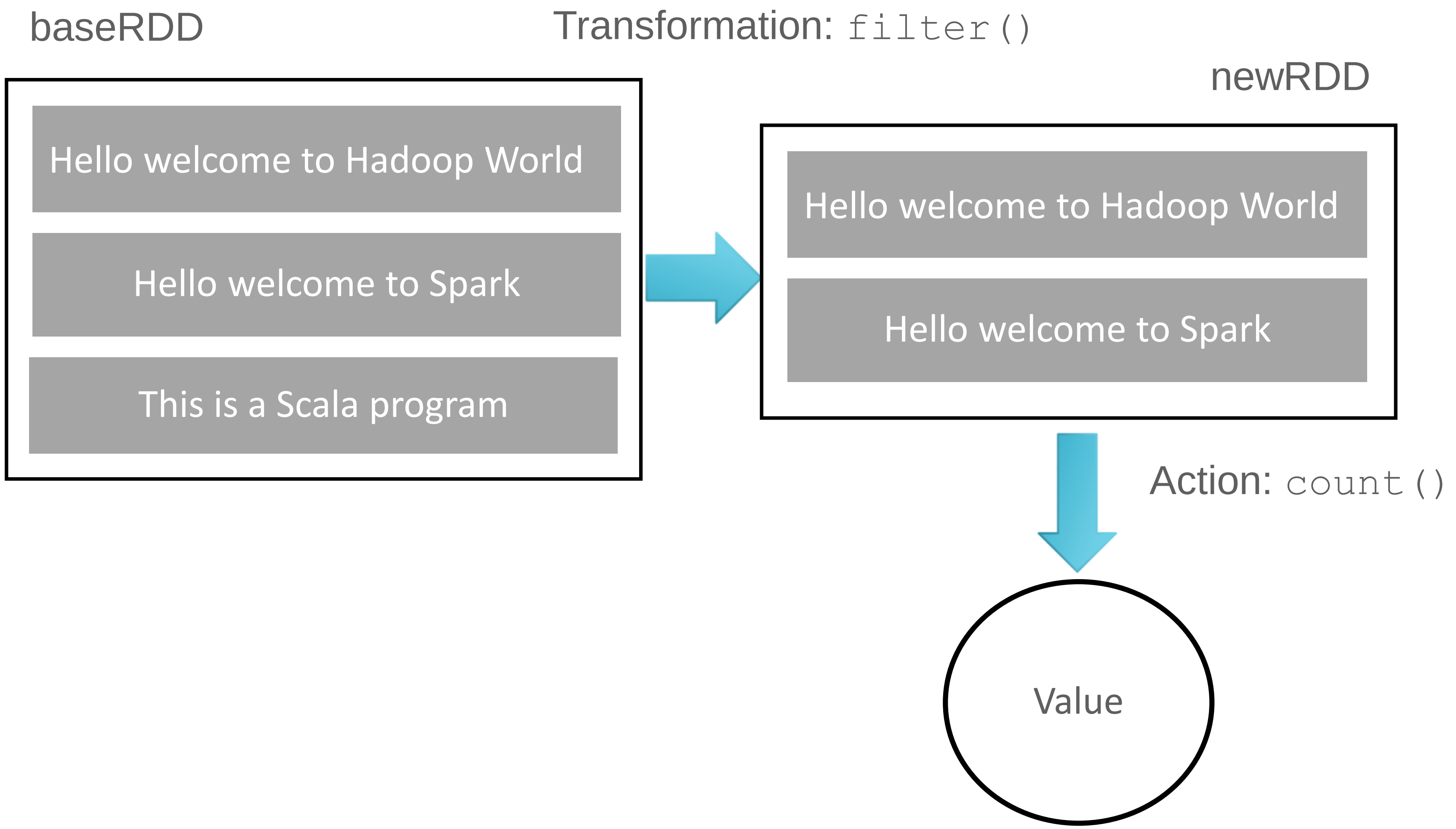
This is a Scala program

RDD Transformation: `Filter()` example

`Filter lines that start with "H"`



RDD Action: `Count()` Example



Spark Interactive Shells

- There are two interactive Spark shells available to execute the Spark

spark-shell is used for Scala.

pyspark is used for Python.

```
oracle@bigdatalite:~
File  Edit  View  Search  Terminal  Help
[oracle@bigdatalite ~]$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_101)
Type in expressions to have them evaluated.
Type :help for more information.
17/03/02 16:54:03 WARN util.Utils: Your hostname, bigdatalite.localdomain resolv
es to a loopback address: 127.0.0.1; using 10.247.45.235 instead (on interface e
th0)
17/03/02 16:54:03 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to ano
ther address
Spark context available as sc (master = yarn-client, app id = application_148573
9108130_0004).
SQL context available as sqlContext.

scala> 
```

```
oracle@bigdatalite:~
File  Edit  View  Search  Terminal  Help
[oracle@bigdatalite ~]$ pyspark
Python 2.6.6 (r266:84292, Aug 18 2016, 08:36:59)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Python version 2.6.6 (r266:84292, Aug 18 2016 08:36:59)
SparkContext available as sc, HiveContext available as sqlContext.
>>> 
```

Scala Language: Overview

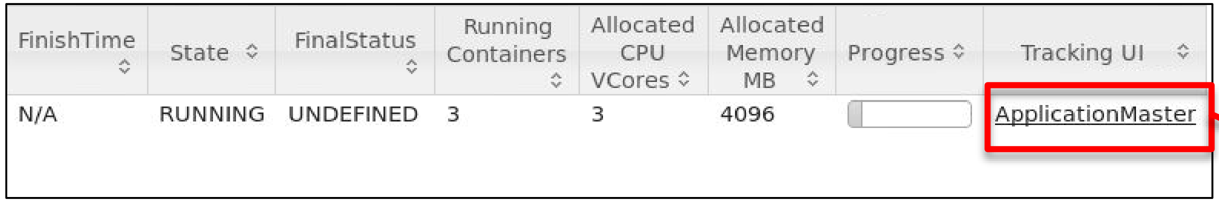
- Scala programming language can be used for implementing Spark.
  - The acronym for “Scalable Language”
  - A pure-bred object-oriented language
  - Runs on the JVM
  - Reliable for large mission critical systems

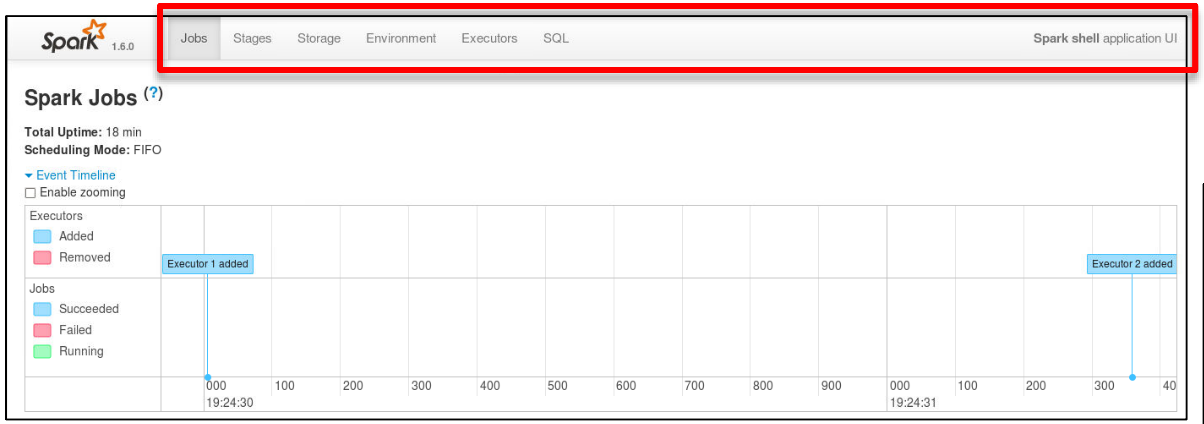
Starting Interactive Spark-Shell for Scala

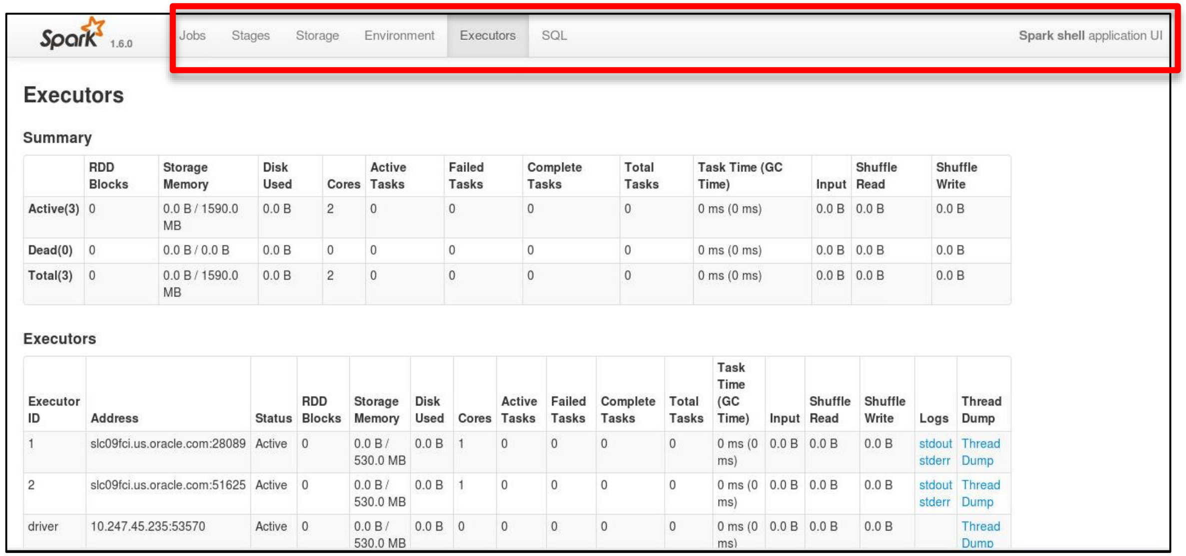
```
[oracle@bigdatalite ~]$ cd /home/oracle/exercises/spark
[oracle@bigdatalite spark]$ pwd
/home/oracle/exercises/spark
[oracle@bigdatalite spark]$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_101)
Type in expressions to have them evaluated.
Type :help for more information.
17/03/09 19:24:18 WARN util.Utils: Your hostname, bigdatalite.localdomain resolves to a loopback addres
s: 127.0.0.1; using 10.247.45.235 instead (on interface eth0)
17/03/09 19:24:18 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Spark context available as sc (master = yarn-client, app id = application_1485739108130_0016).
SQL context available as sqlContext.

scala>
```

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Progress	Tracking UI
application_1485739108130_0016	oracle	Spark shell	SPARK	root.oracle	Thu Mar 9 19:24:21 -0500 2017	N/A	RUNNING	UNDEFINED	3	3	4096		ApplicationMaster
application_1485739108130_0015	oracle	Spark shell	SPARK	root.oracle	Thu Mar 9 19:23:15 -0500 2017	Thu Mar 9 19:23:26 -0500 2017	FINISHED	SUCCEEDED	N/A	N/A	N/A		History

Using the ResourceManager UI to Monitor a Spark Application







Word Count Example by Using Interactive Scala

```
// Open the two input files and get their file handles.
val files =
     sc.textFile("hdfs://bigdatalite:8020/user/oracle/wordcount/input")

// Set the execution context for the data.
val counts = files.flatMap(line => line.split(" ")) .map(word => (word,
     1)) .reduceByKey(_ + _)

// Execute the counts and save the output to sp-out directory.
counts.saveAsTextFile("hdfs://bigdatalite:8020/user/oracle/wordcount/sp-
     out")

// print out the counts to the console.
counts.collect().foreach(println)
```

Creates a new RDD from HDFS

Executes the word Count transformation

Applies the action

Print the output

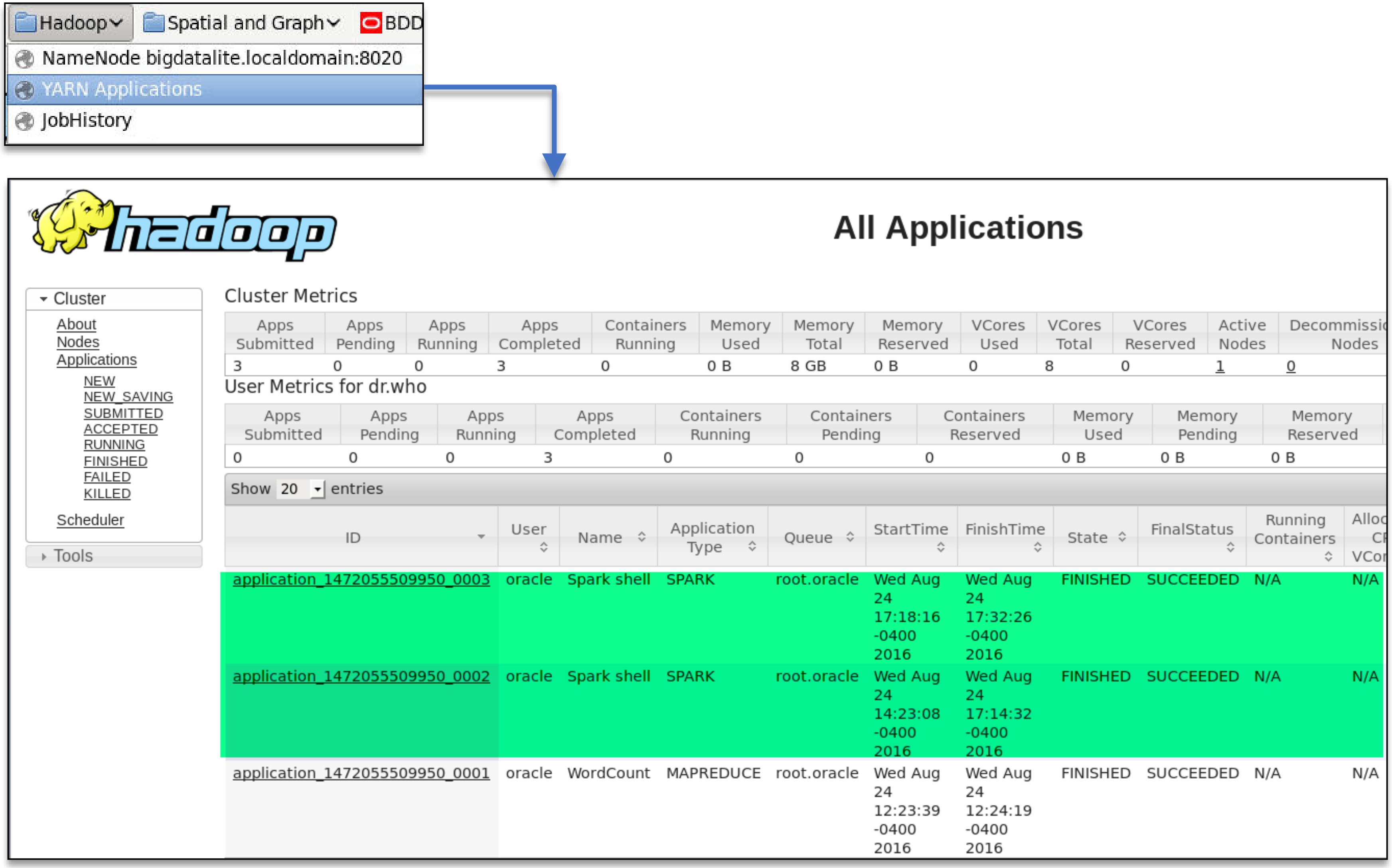
```
[oracle@bigdatalite ~]$ hadoop fs -ls /user/oracle/wordcount/input
Found 2 items
-rw-r--r--   1 oracle oracle        518 2016-08-24 12:07 /user/oracle/wordcount/input/file01
-rw-r--r--   1 oracle oracle        518 2016-08-24 12:08 /user/oracle/wordcount/input/file02
[oracle@bigdatalite ~]$ hadoop fs -ls /user/oracle/wordcount/sp-out
Found 3 items
-rw-r--r--   1 oracle oracle          0 2016-08-24 17:21 /user/oracle/wordcount/sp-out/_SUCCESS
-rw-r--r--   1 oracle oracle         89 2016-08-24 17:21 /user/oracle/wordcount/sp-out/part-00000
-rw-r--r--   1 oracle oracle        161 2016-08-24 17:21 /user/oracle/wordcount/sp-out/part-00001
[oracle@bigdatalite ~]$
```

Check the output directory.

Word Count Example by Using Interactive Scala: Output

```
oracle@bigdatalite:~/exercises/spark
File  Edit  View  Search  Terminal  Help

scala> counts.collect().foreach(println)
(is,2)
(insurance,18)
(with,4)
(disappointed,6)
(staff,2)
(very,6)
(customer,6)
(bank,2)
(worthless,4)
(unreliable,6)
(terrible,4)
(protocols,2)
(expensive,12)
(service,12)
(cover,2)
(company,4)
(professional,2)
(and,12)
(awful,2)
(worst,16)
(the,2)

scala>
```

Monitoring Spark Jobs by Using YARN's ResourceManager Web UI



Scala Program: Word Count Example

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf
 object SparkWordCount {
  def main(args: Array[String]) {
    val sc = new SparkContext(new SparkConf().setAppName("Spark Count"))
    val threshold = args(1).toInt
     // split each document into words
    val tokenized = sc.textFile(args(0)).flatMap(_.split(" "))
     // count the occurrence of each word
    val wordCounts = tokenized.map((_, 1)).reduceByKey(_ + _)
     // filter out words with less than threshold occurrences
    val filtered = wordCounts.filter(_._2 &gt;= threshold)
     // count characters
    val charCounts = filtered.flatMap(_._1.toCharArray).map((_, 1)).reduceByKey(_ + _)
     System.out.println(charCounts.collect().mkString(", "))
  }
}
```

Definitions

Term	Description
Application	User program built on Spark. Consists of a ***driver*** program and ***executors*** on the cluster.
Application JAR	A JAR containing the user's Spark application. The user's JAR should never include Hadoop or Spark libraries; however, these are added at run time.
Driver program	The process running the `main()` function of the application and creating the `SparkContext`
Cluster Manager	An external service for acquiring resources on the cluster (e.g. stand-alone manager, Mesos, YARN)
Deploy mode	Distinguishes where the driver process runs. In "cluster" mode, the framework launches the driver inside of the cluster. In "client" mode, the submitter launches the driver outside of the cluster.
Worker node	Any slave node that can run application code in the cluster
Executor	A process launched for an application on a worker node, that runs tasks and keeps data in memory or disk storage across them. Each application has its own executors.
Task	A unit of work that will be sent to one executor.
Job	A parallel computation consisting of multiple tasks that get spawned in response to a Spark action (e.g. save, collect); you'll see this term used in the driver's logs.
Stage	Each job gets divided into smaller sets of tasks called stages that depend on each other (similar to the map and reduce stages in MapReduce); you'll see this term used in the driver's logs.

Launching Spark Applications Using the `spark-submit` Command

```
./bin/spark-submit \
  --class <main-class> \
  --master <master-url> \
  --deploy-mode <deploy-mode> \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```

```
# Run application locally on 8 cores
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master local[8] \
  /path/to/examples.jar \
  100
```

Source: Apache Spark page on http://spark.apache.org/docs/latest/submitting-applications.html

Summary

- In this lesson, you should have learned how to:
  - Describe Spark and identify its benefits
  - Identify Spark's components and architecture
  - Interact with Spark
  - Run a Spark job and monitor the job using YARN's Resource Manager Web UI

Practice 10: Overview

- This practice covers the following topics:
  - Writing a `WordCount.java` Program by Using Scala, Which Is on a Hadoop Cluster
  - Running the `WordCount.java` Program and Viewing the Results
  - Understanding the Features of Spark Web-based Interface