cloudera®

Developer Training for Spark and Hadoop I

cloudera®

Introduction

Chapter 1

Course Chapters

▪ **Introduction**	**Course Introduction**
▪ Introduction to Hadoop and the Hadoop Ecosystem ▪ Hadoop Architecture and HDFS	Introduction to Hadoop
▪ Importing Relational Data with Apache Sqoop ▪ Introduction to Impala and Hive ▪ Modeling and Managing Data with Impala and Hive ▪ Data Formats ▪ Data Partitioning	Importing and Modeling Structured Data
▪ Capturing Data with Apache Flume	Ingesting Streaming Data
▪ Spark Basics ▪ Working with RDDs in Spark ▪ Aggregating Data with Pair RDDs ▪ Writing and Deploying Spark Applications ▪ Parallel Processing in Spark ▪ Spark RDD Persistence ▪ Common Patterns in Spark Data Processing ▪ Spark SQL and DataFrames	Distributed Data Processing with Spark
▪ Conclusion	Course Conclusion

Chapter Topics

Introduction | **Course Introduction**

- **About This Course**
- About Cloudera
- Course Logistics
- Introductions

Course Objectives

During this course, you will learn

- **How the Hadoop Ecosystem fits in with the data processing lifecycle**
- **How data is distributed, stored and processed in a Hadoop cluster**
- **How to use Sqoop and Flume to ingest data**
- **How to process distributed data with Spark**
- **Best practices for data storage**
- **How to model structured data as tables in Impala and Hive**
- **How to choose a data storage format for your data usage patterns**

Chapter Topics

Introduction | **Course Introduction**

- About This Course
- **About Cloudera**
- Course Logistics
- Introductions

About Cloudera (1)

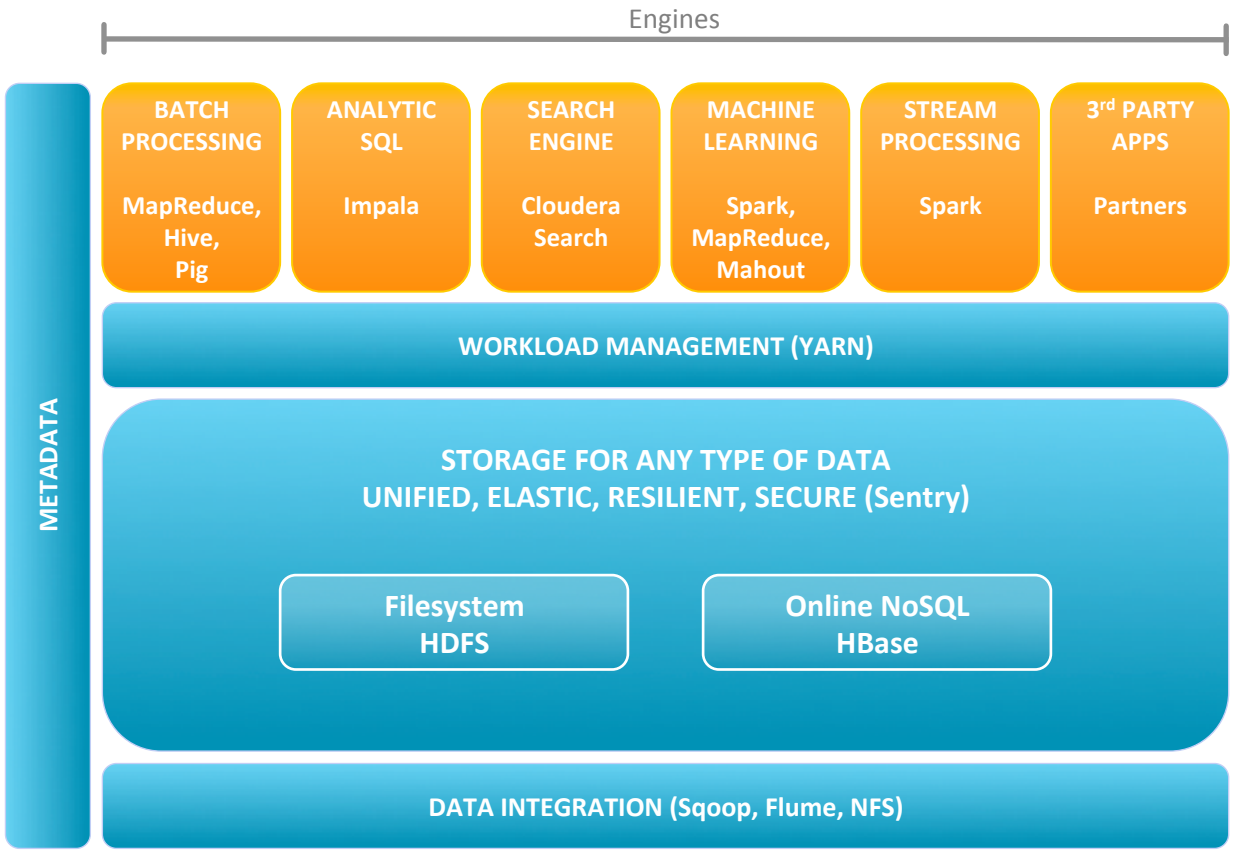
- **The leader in Apache Hadoop-based software and services**
- **Founded by leading experts on Hadoop from Facebook, Yahoo, Google, and Oracle**
- **Provides support, consulting, training, and certification for Hadoop users**
- **Staff includes committers to virtually all Hadoop projects**
- **Many authors of industry standard books on Apache Hadoop projects**
  - Tom White, Lars George, Kathleen Ting, etc.

About Cloudera (2)

- **Customers include many key users of Hadoop**
  - Allstate, AOL Advertising, Box, CBS Interactive, eBay, Experian, Groupon, National Cancer Institute, Orbitz, Social Security Administration, Trend Micro, Trulia, US Army, …
- **Cloudera public training including**
  - Cloudera Developer Training for Apache Spark
  - Developer Training for Spark and Hadoop II: Advanced Techniques
  - Cloudera Data Analyst Training: Using Pig, Hive, and Impala with Hadoop
  - Cloudera Training for Apache HBase
  - Introduction to Data Science: Building Recommender Systems
  - Cloudera Essentials for Apache Hadoop
- **Onsite and custom training is also available**

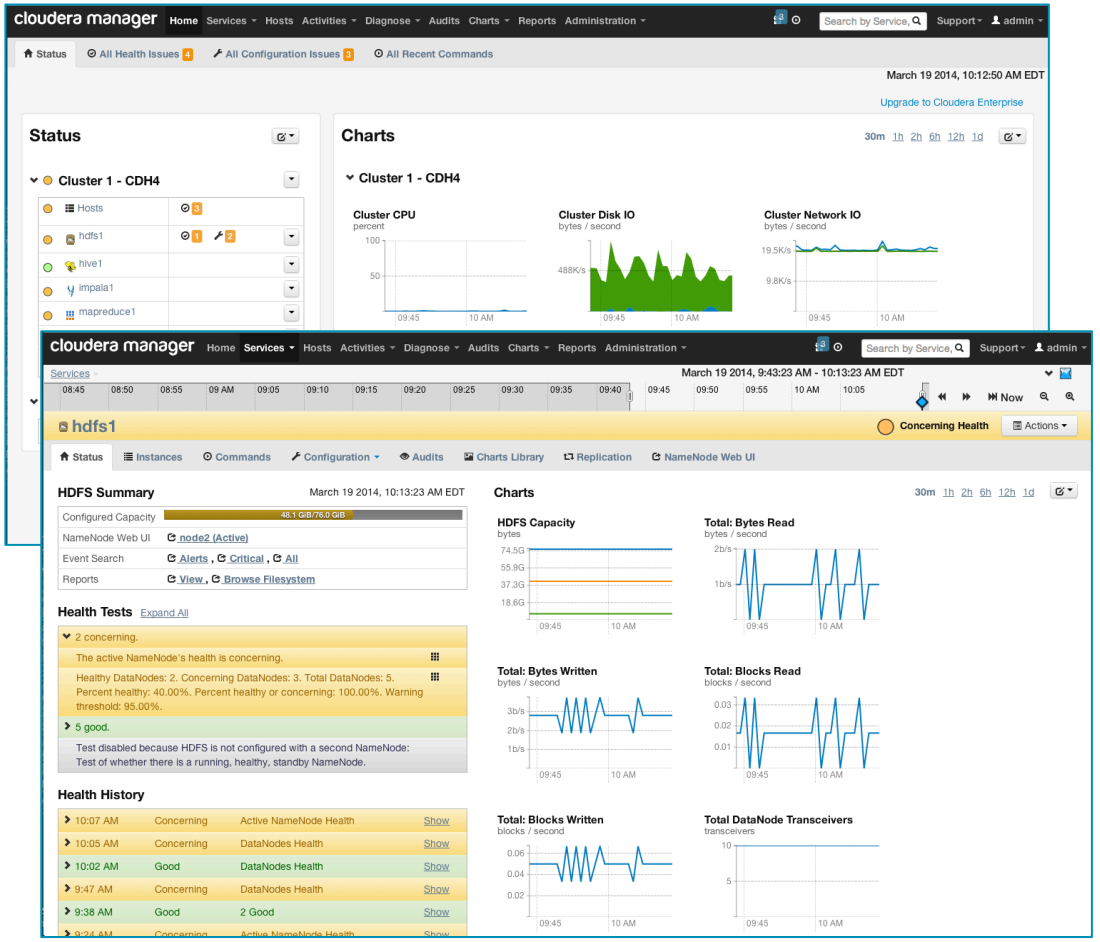
CDH (Cloudera's Distribution including Apache Hadoop)

- 100% open source, enterprise-ready distribution of Hadoop and related projects
- The most complete, tested, and widely-deployed distribution of Hadoop
- Integrates all the key Hadoop ecosystem projects
- Available as RPMs and Ubuntu, Debian, or SuSE packages, or as a tarball



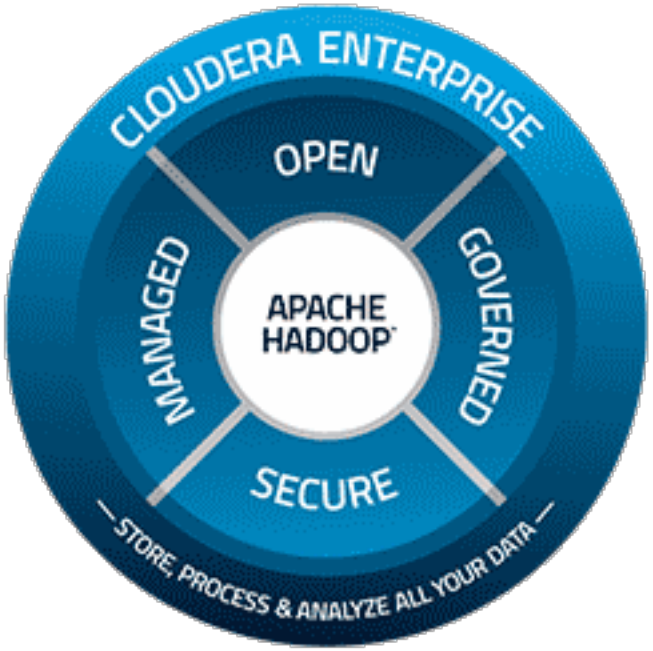
Cloudera Express

- **Cloudera Express**
  - Completely free to download and use
- **The best way to get started with Hadoop**
- **Includes CDH**
- **Includes Cloudera Manager**
  - End-to-end administration for Hadoop
  - Deploy, manage, and monitor your cluster



Cloudera Enterprise

- **Cloudera Enterprise**
  - Subscription product including CDH and Cloudera Manager
- **Includes support**
- **Includes extra Cloudera Manager features**
  - Configuration history and rollbacks
  - Rolling updates
  - LDAP integration
  - SNMP support
  - Automated disaster recovery



- **Extend capabilities with Cloudera Navigator subscription**
  - Event auditing, metadata tagging capabilities, lineage exploration
  - Available in both the Cloudera Enterprise Flex and Data Hub editions

Chapter Topics

Introduction | **Course Introduction**

- About This Course
- About Cloudera
- **Course Logistics**
- Introductions

Logistics

- **Course start and end times**
- **Lunch**
- **Breaks**
- **Restrooms**

Chapter Topics

Introduction | **Course Introduction**

- About This Course
- About Cloudera
- Course Logistics
- **Introductions**

Introductions

- **About your instructor**
- **About you**
  - Company and role?
  - Experience with Hadoop and Spark?
  - Language preference: Python or Scala?
  - Expectations from the course?

cloudera®

Introduction to Hadoop and the Hadoop Ecosystem

Chapter 2

Course Chapters

▪ Introduction	Course Introduction
▪ **Introduction to Hadoop and the Hadoop Ecosystem** ▪ Hadoop Architecture and HDFS	**Introduction to Hadoop**
▪ Importing Relational Data with Apache Sqoop ▪ Introduction to Impala and Hive ▪ Modeling and Managing Data with Impala and Hive ▪ Data Formats ▪ Data File Partitioning	Importing and Modeling Structured Data
▪ Capturing Data with Apache Flume	Ingesting Streaming Data
▪ Spark Basics ▪ Working with RDDs in Spark ▪ Aggregating Data with Pair RDDs ▪ Writing and Deploying Spark Applications ▪ Parallel Processing in Spark ▪ Spark RDD Persistence ▪ Common Patterns in Spark Data Processing ▪ Spark SQL and DataFrames	Distributed Data Processing with Spark
▪ Conclusion	Course Conclusion

Introduction to Hadoop and the Hadoop Ecosystem

In this chapter you will learn

- **What Hadoop is and how it addresses big data challenges**
- **The guiding principles behind Hadoop**
- **The major components of the Hadoop Ecosystem**
- **What tools you will be using in the Hands-On Exercises in this course**

Chapter Topics

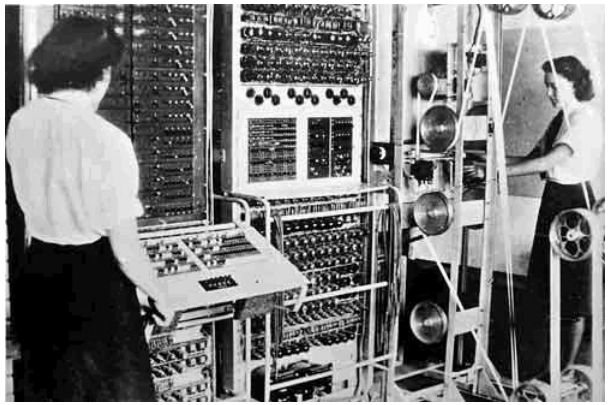
Introduction to Hadoop and the Hadoop Ecosystem

Introduction to Hadoop

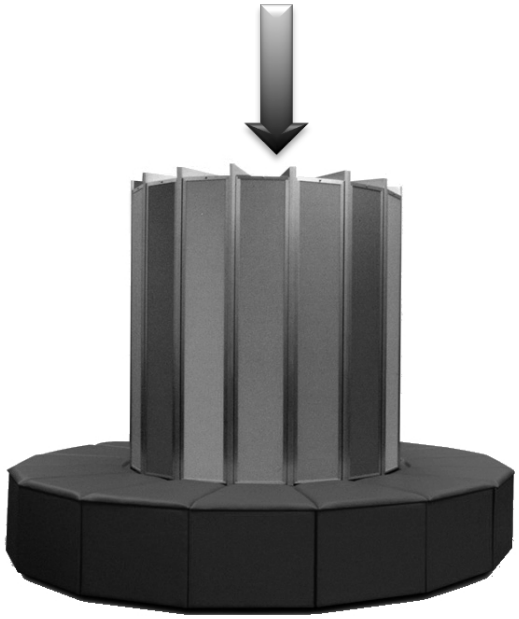
- **Problems with Traditional Large-scale Systems**
- Hadoop!
- Data Storage and Ingest
- Data Processing
- Data Analysis and Exploration
- Other Ecosystem Tools
- Introduction to the Hands-On Exercises
- Conclusion

Traditional Large-Scale Computation

- **Traditionally, computation has been processor-bound**
  - Relatively small amounts of data
  - Lots of complex processing



- **The early solution: bigger computers**
  - Faster processor, more memory
  - But even this couldn't keep up

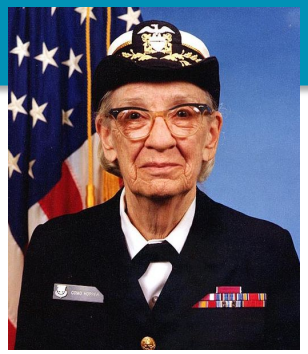


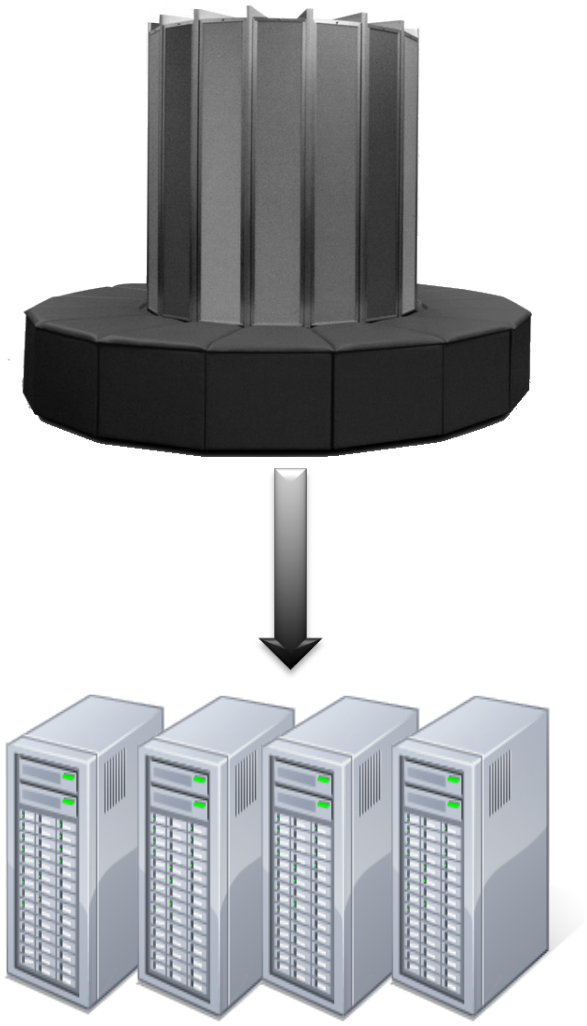
Distributed Systems

- **The better solution: more computers**
  - Distributed systems – use multiple machines for a single job

"In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, we didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for *more systems* of computers."

– Grace Hopper





Challenges with Distributed Systems

- **Challenges with distributed systems**
  - Programming complexity
    - Keeping data and processes in sync
  - Finite bandwidth
  - Partial failures

- **The solution?**
  - Hadoop!

Chapter Topics

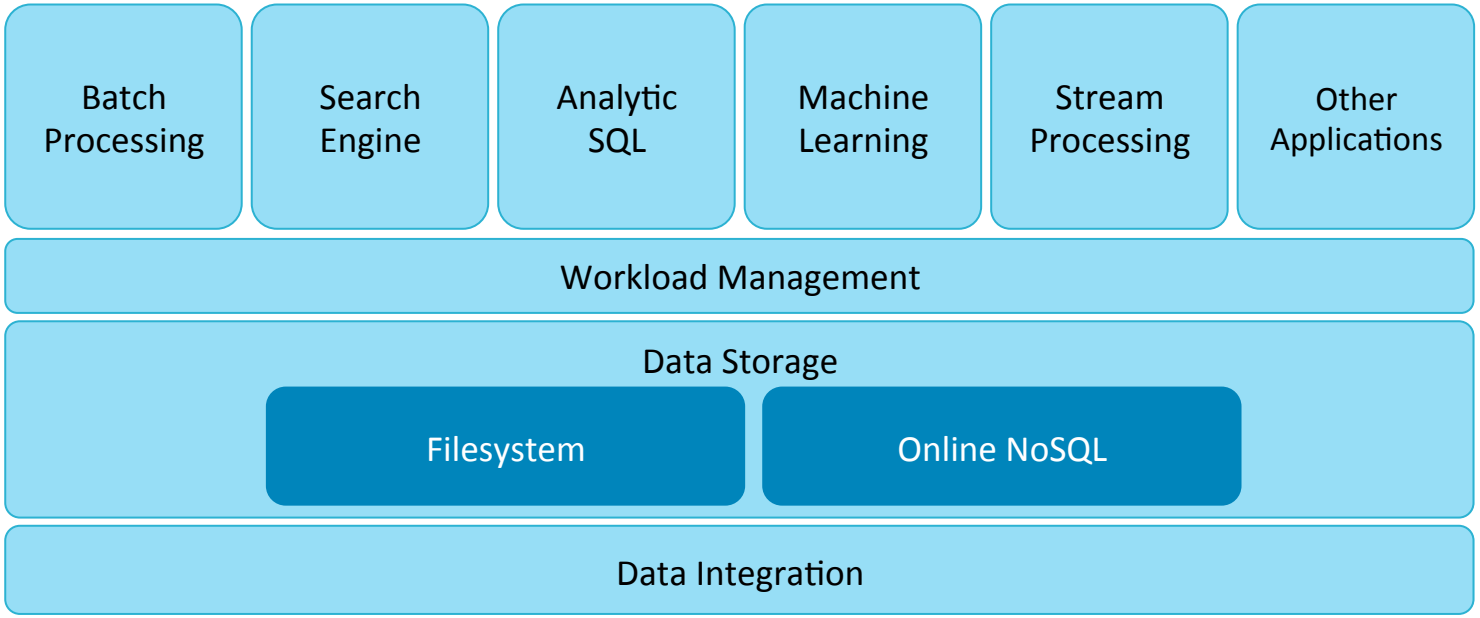
Introduction to Hadoop and the Hadoop Ecosystem

Introduction to Hadoop

- Problems with Traditional Large-scale Systems
- **Hadoop!**
- Data Storage and Ingest
- Data Processing
- Data Analysis and Exploration
- Other Ecosystem Tools
- Introduction to the Hands-On Exercises
- Conclusion

What is Apache Hadoop?

- **Scalable and economical data storage, processing and analysis**
  - Distributed and fault-tolerant
  - Harnesses the power of industry standard hardware
- **Heavily inspired by technical documents published by Google**



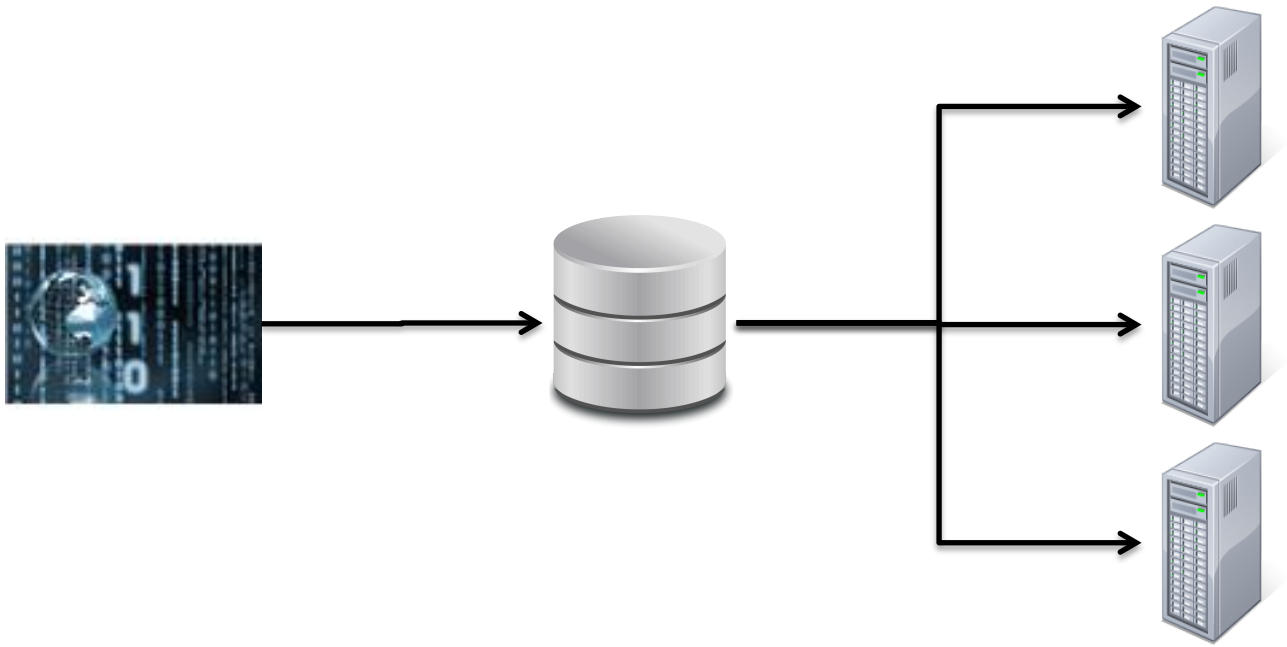
Common Hadoop Use Cases

- **Extract/Transform/Load (ETL)**
- **Text mining**
- **Index building**
- **Graph creation and analysis**
- **Pattern recognition**
- **Collaborative filtering**
- **Prediction models**
- **Sentiment analysis**
- **Risk assessment**

- **What do these workloads have in common? Nature of the data…**
  - Volume
  - Velocity
  - Variety

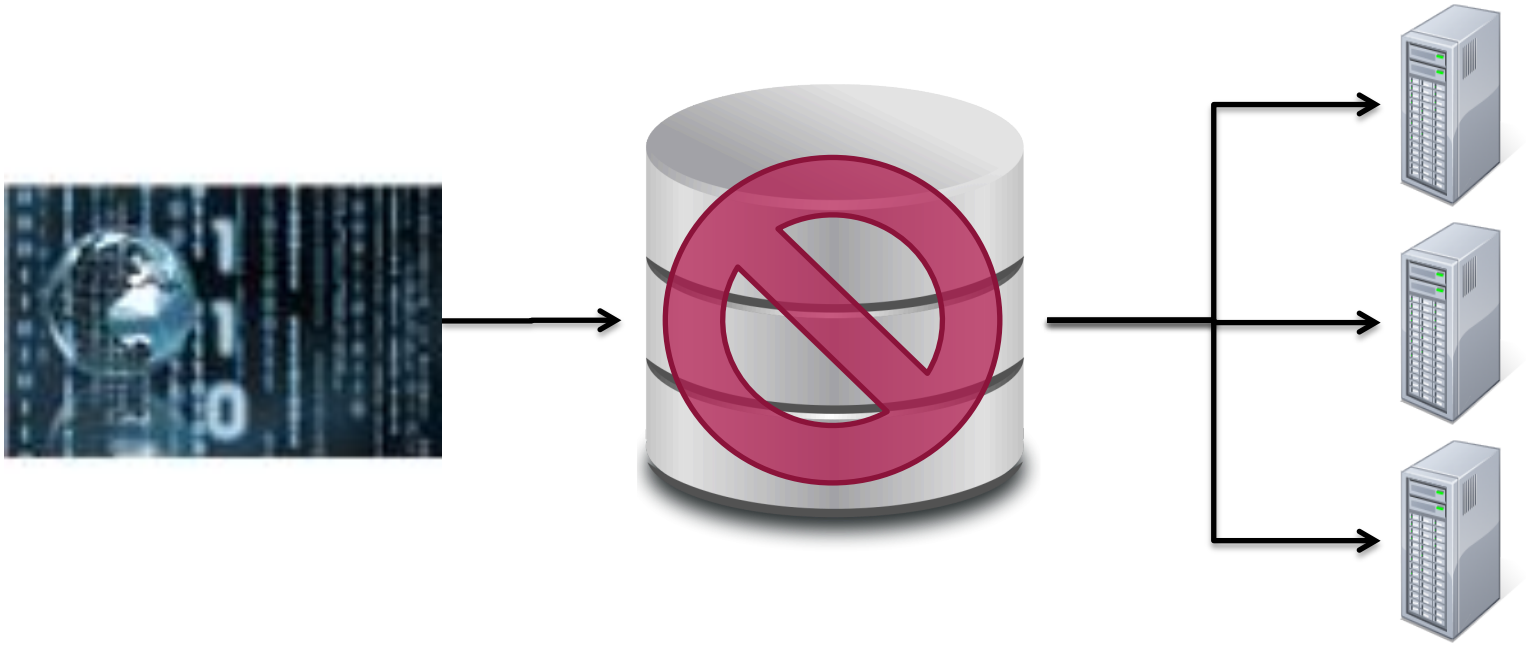
Distributed Systems: The Data Bottleneck (1)

- **Traditionally, data is stored in a central location**
- **Data is copied to processors at runtime**
- **Fine for limited amounts of data**



Distributed Systems: The Data Bottleneck (2)

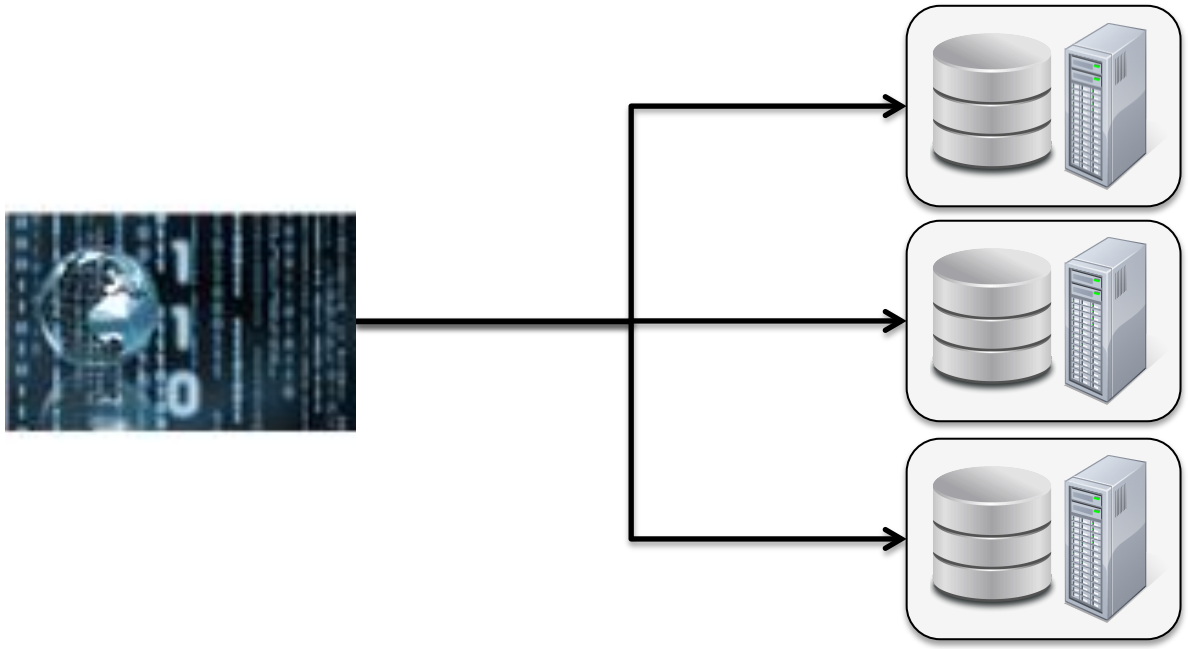
- **Modern systems have much more data**
  - terabytes+ a day
  - petabytes+ total
- **We need a new approach…**



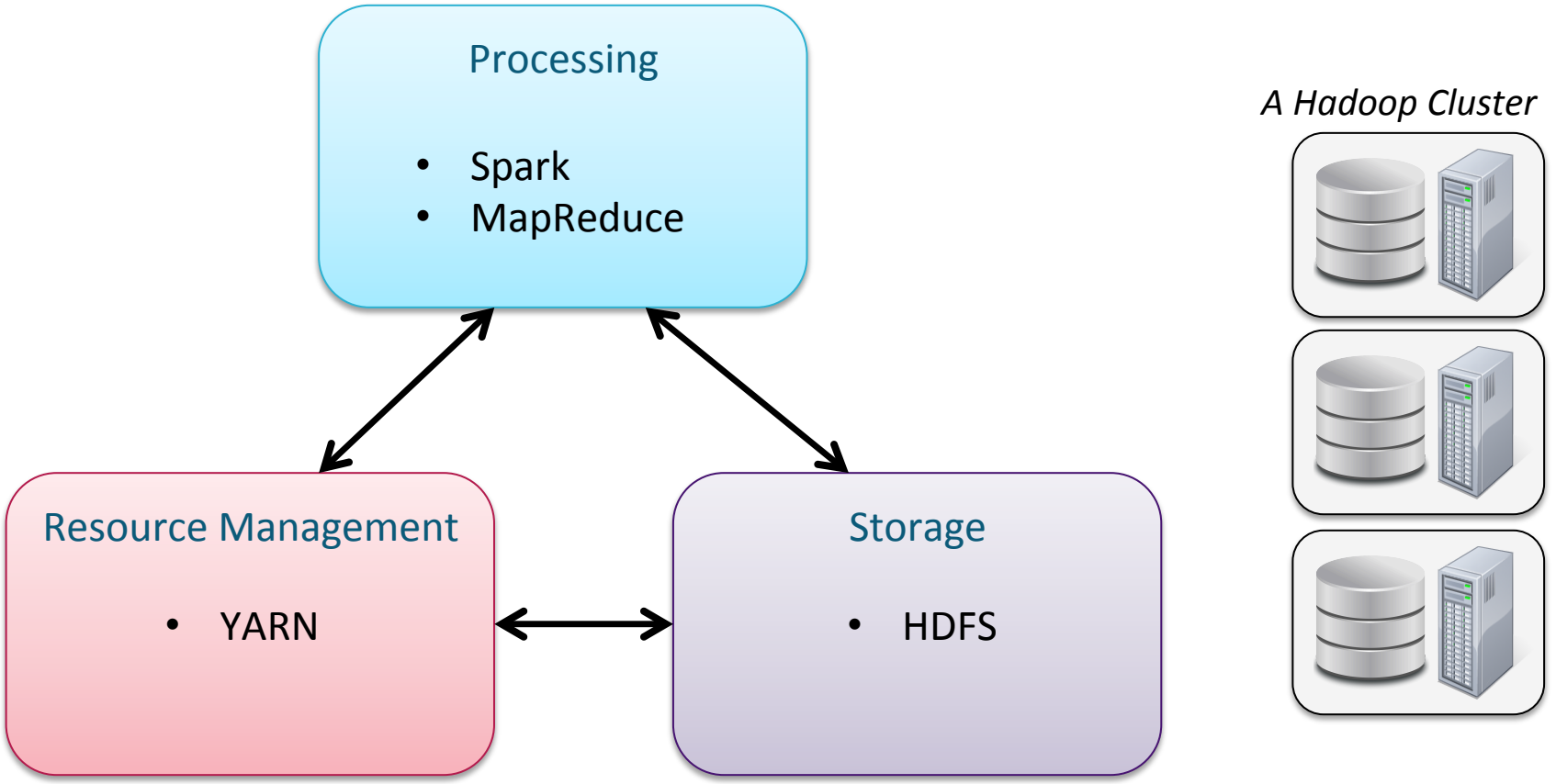
Big Data Processing with Hadoop

- **Hadoop introduced a radical new approach:**
  - Bring the program to the data rather than the data to the program
- **Based on two key concepts**
  - Distribute data when the data is stored
  - Run computation where the data is

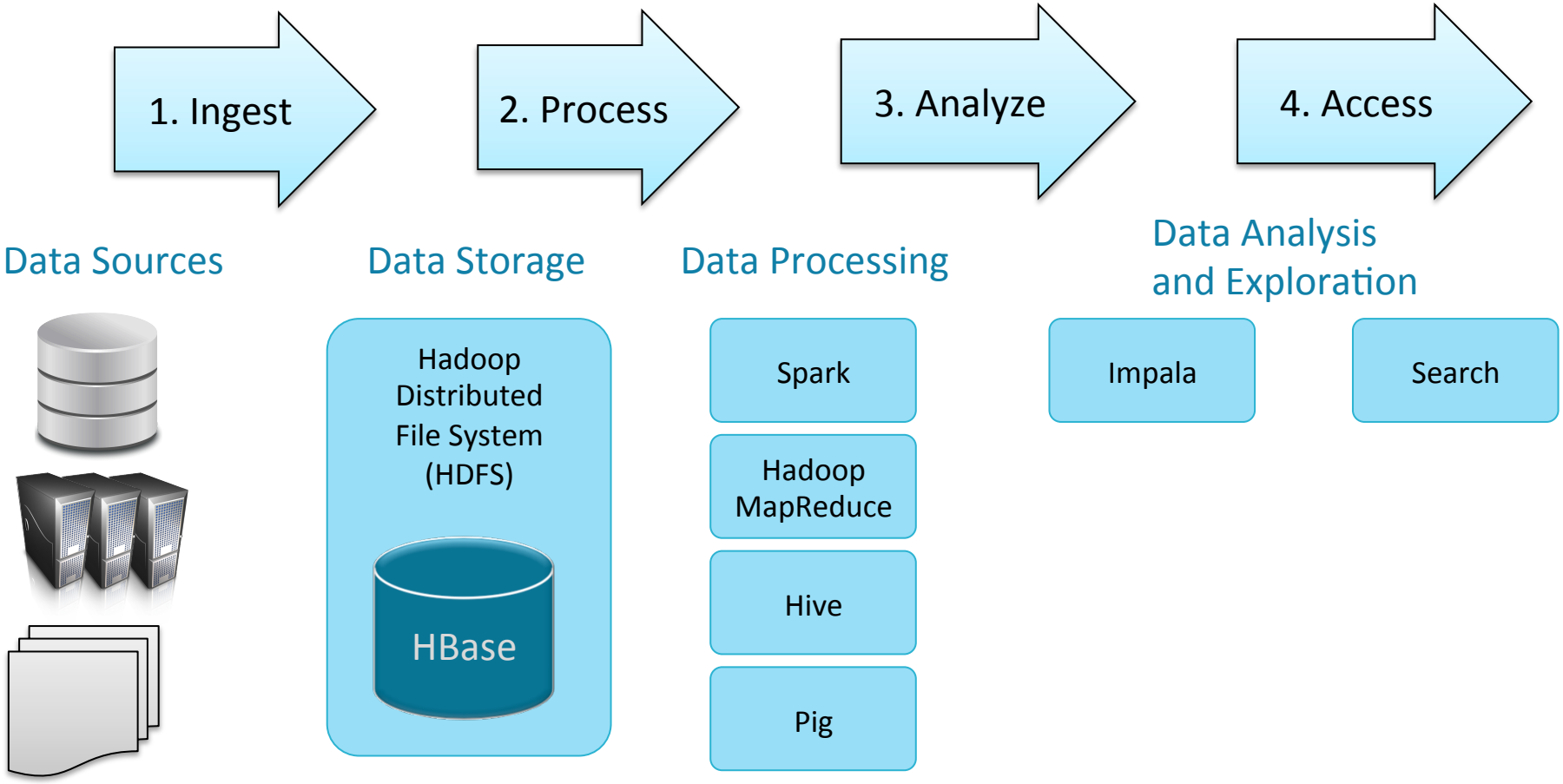
A Hadoop Cluster



Core Hadoop



Big Data Processing



Chapter Topics

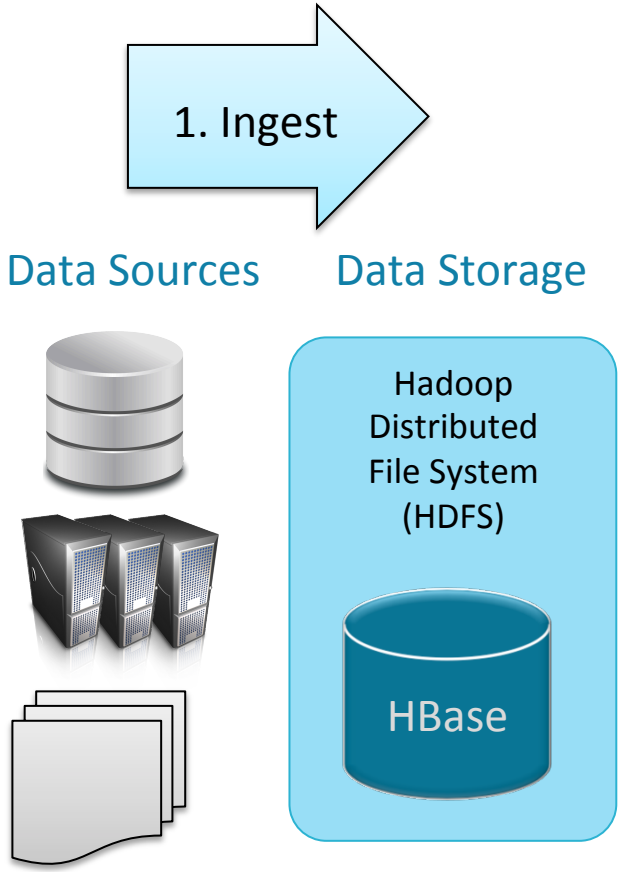
Introduction to Hadoop and the Hadoop Ecosystem

Introduction to Hadoop

- Problems with Traditional Large-scale Systems
- Hadoop!
- **Data Storage and Ingest**
- Data Processing
- Data Analysis and Exploration
- Other Ecosystem Tools
- Introduction to the Hands-On Exercises
- Conclusion

Data Ingest and Storage

- **Hadoop typically ingests data from many sources and in many formats**
  - Traditional data management systems, e.g. databases
  - Logs and other machine generated data (event data)
  - Imported files



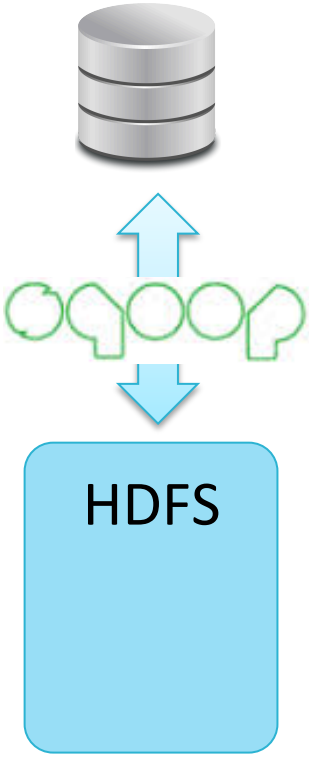
Data Storage

- **Hadoop Distributed File System (HDFS)**
  - HDFS is the storage layer for Hadoop
  - Provides inexpensive reliable storage for massive amounts of data on industry-standard hardware
  - Data is distributed when stored
  - Covered later in this course

- **Apache HBase: The Hadoop Database**
  - A NoSQL distributed database built on HDFS
  - Scales to support very large amounts of data and high throughput
  - A table can have thousands of columns
  - Covered in depth in *Cloudera Training for Apache HBase*

Data Ingest Tools (1)

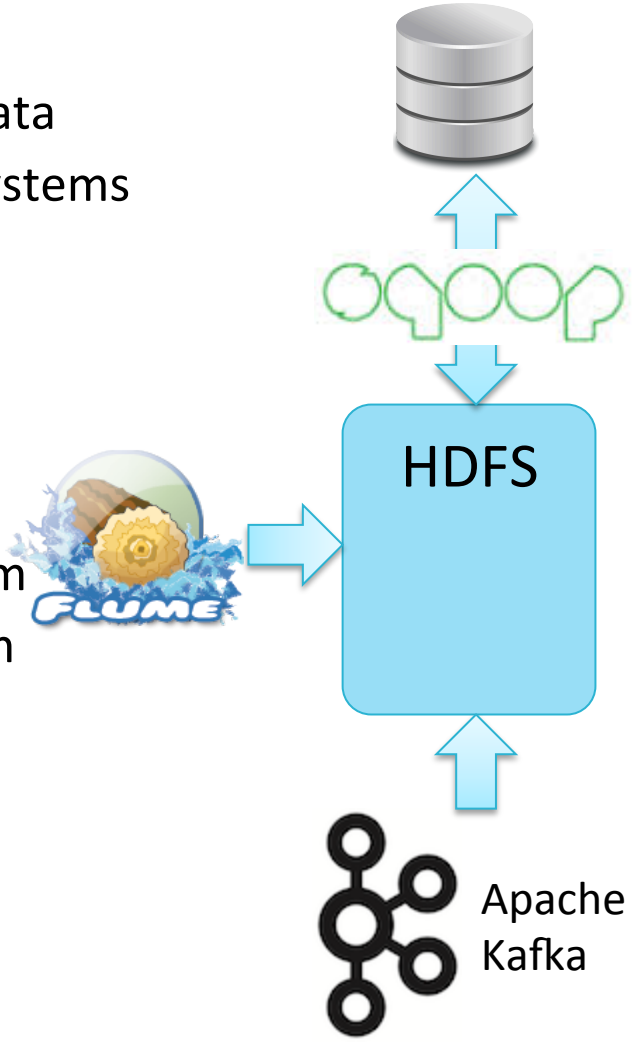
- **HDFS**
  - Direct file transfer
- **Apache Sqoop**
  - High speed import to HDFS from Relationship Database (and vice versa)
  - Supports many data storage systems
    - e.g. Netezza, Mongo, MySQL, Teradata, Oracle
  - Covered later in this course



Data Ingest Tools (2)

- **Apache Flume**
  - Distributed service for ingesting streaming data
  - Ideally suited for event data from multiple systems
    - For example, log files
  - Covered later in this course

- **Kafka**
  - A high throughput, scalable messaging system
  - Distributed, reliable publish-subscribe system
  - Integrates with Flume and Spark Streaming
  - Covered in *Developer Training for Spark and Hadoop II: Advanced Techniques*



Chapter Topics

Introduction to Hadoop and the Hadoop Ecosystem

Introduction to Hadoop

- Problems with Traditional Large-scale Systems
- Hadoop!
- Data Storage and Ingest
- **Data Processing**
- Data Analysis and Exploration
- Other Ecosystem Tools
- Introduction to the Hands-On Exercises
- Conclusion

Apache Spark: An Engine For Large-scale Data Processing

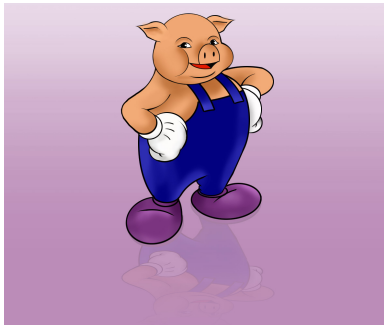
- **Spark is large-scale data processing engine**
  - General purpose
  - Runs on Hadoop clusters and data in HDFS
- **Supports a wide range of workloads**
  - Machine learning
  - Business intelligence
  - Streaming
  - Batch Processing
- **This course uses Spark for data processing**

Hadoop MapReduce: The Original Hadoop Processing Engine

- **Hadoop MapReduce is the original Hadoop framework**
  - Primarily Java based
- **Based on the MapReduce programming model**
- **The core Hadoop processing engine before Spark was introduced**
- **Still the dominant technology**
  - But losing ground to Spark fast
- **Many existing tools are still built using MapReduce code**
- **Has extensive and mature fault tolerance built into the framework**

Apache Pig: Scripting for MapReduce

- **Apache Pig builds on Hadoop to offer high-level data processing**
  - This is an alternative to writing low-level MapReduce code
  - Pig is especially good at joining and transforming data
- **The Pig interpreter runs on the client machine**
  - Turns Pig Latin scripts into MapReduce or Spark jobs
  - Submits those jobs to a Hadoop cluster
  - Covered in Cloudera *Data Analyst Training*



```
people = LOAD '/user/training/customers' AS (cust_id, name);
orders = LOAD '/user/training/orders' AS (ord_id, cust_id, cost);
groups = GROUP orders BY cust_id;
totals = FOREACH groups GENERATE group, SUM(orders.cost) AS t;
result = JOIN totals BY group, people BY cust_id;
DUMP result;
```

Chapter Topics

Introduction to Hadoop and the Hadoop Ecosystem

Introduction to Hadoop

- Problems with Traditional Large-scale Systems
- Hadoop!
- Data Storage and Ingest
- Data Processing
- **Data Analysis and Exploration**
- Other Ecosystem Tools
- Introduction to the Hands-On Exercises
- Conclusion

Cloudera Impala: High Performance SQL

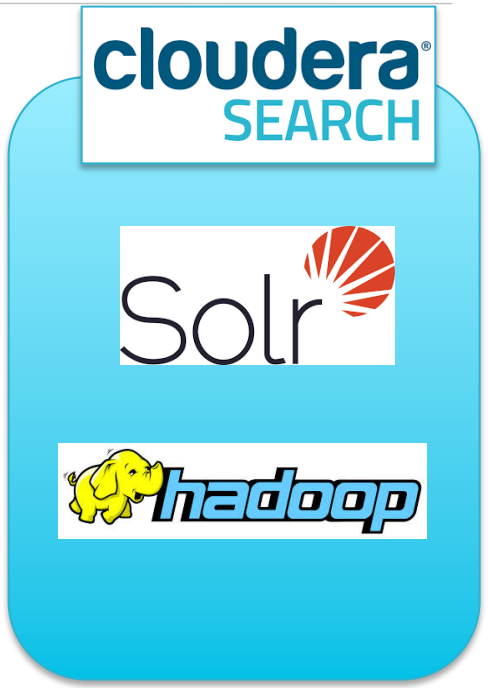
- **Impala is a high-performance SQL engine**
  - Runs on Hadoop clusters
  - Data stored in HDFS files
  - Inspired by Google's Dremel project
  - Very low latency – measured in milliseconds
  - Ideal for interactive analysis
- **Impala supports a dialect of SQL (Impala SQL)**
  - Data in HDFS modeled as database tables
- **Impala was developed by Cloudera**
  - 100% open source, released under the Apache software license
- **Impala is used for data analysis in this course**

Apache Hive: SQL on MapReduce

- **Hive is an abstraction layer on top of Hadoop**
  - Hive uses a SQL-like language called HiveQL
  - Similar to Impala SQL
  - Useful for data processing and ETL
    - Impala is preferred for ad hoc analytics
- **Hive executes queries using MapReduce**
  - Hive on Spark is available for early adopters; not yet recommended for production
- **Hive can optionally be used for data analysis in this course**
  - Covered in more depth in *Cloudera Data Analyst Training*

Cloudera Search: A Platform For Data Exploration

- **Interactive full-text search for data in a Hadoop cluster**
- **Allows non-technical users to access your data**
  - Nearly everyone can use a search engine
- **Cloudera Search enhances Apache Solr**
  - Integrates Solr with HDFS, MapReduce, HBase, and Flume
  - Supports file formats widely used with Hadoop
  - Dynamic Web-based dashboard interface with Hue
  - Apache Sentry based security
- **Cloudera Search is 100% open source**
- **Covered in depth in *Cloudera Search Training***



Chapter Topics

Introduction to Hadoop and the Hadoop Ecosystem

Introduction to Hadoop

- Problems with Traditional Large-scale Systems
- Hadoop!
- Data Storage and Ingest
- Data Processing
- Data Analysis and Exploration
- **Other Ecosystem Tools**
- Introduction to the Hands-On Exercises
- Conclusion

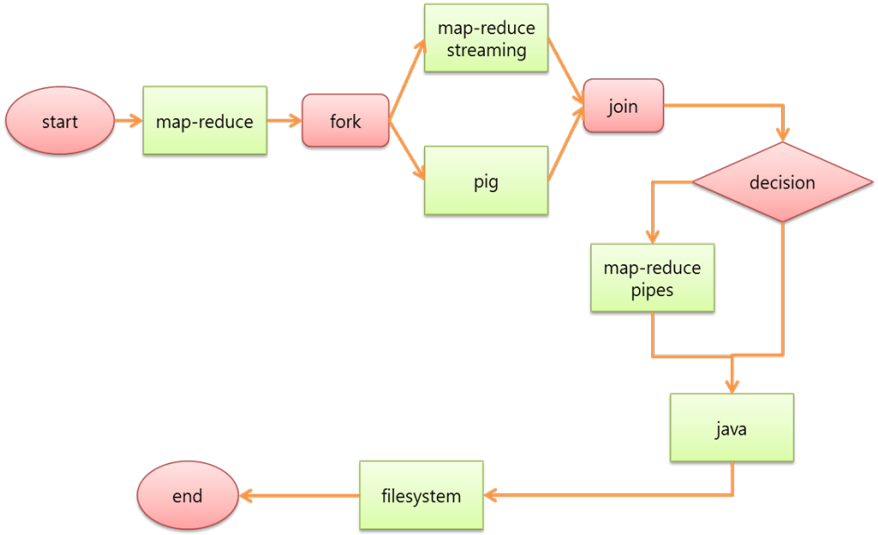
Hue: The UI for Hadoop

- **Hue = Hadoop User Experience**
- **Hue provides a Web front-end to a Hadoop**
  - Upload and browse data
  - Query tables in Impala and Hive
  - Run Spark and Pig jobs and workflows
  - Search
  - And much more
- **Makes Hadoop easier to use**
- **Hue is 100% open-source**
- **Created by Cloudera**
  - Open source, released under Apache license
- **Hue is used throughout this course**

Apache Oozie: Workflow Management

- **Oozie**
  - Workflow engine for Hadoop jobs
  - Defines dependencies between jobs

- **The Oozie server submits the jobs to the server in the correct sequence**



Apache Sentry: Hadoop Security

- **Sentry provides fine-grained access control (authorization) to various Hadoop ecosystem components**
  - Impala
  - Hive
  - Cloudera Search
  - HDFS
- **In conjunction with Kerberos authentication, Sentry authorization provides a complete cluster security solution**
- **Created by Cloudera**
  - Now an open-source Apache project

Chapter Topics

Introduction to Hadoop and the Hadoop Ecosystem

Introduction to Hadoop

- Problems with Traditional Large-scale Systems
- Hadoop!
- Data Storage and Ingest
- Data Processing
- Data Analysis and Exploration
- Other Ecosystem Tools
- **Introduction to the Hands-On Exercises**
- Conclusion

Introduction to the Hands-On Exercises

- **The best way to learn is to *do!***
- **Most topics in this course have Hands-On Exercises to practice the skills you have learned in the course**

Scenario Explanation (1)

- **The exercises are based on a hypothetical scenario**
  - However, the concepts apply to nearly any organization
- **Loudacre Mobile is a (fictional) fast-growing wireless carrier**
  - Provides mobile service to customers throughout western USA

Scenario Explanation (2)

- **Loudacre needs to migrate their existing infrastructure to Hadoop**
  - The size and velocity and their data has exceeded their ability to processing and analyze their data
- **Loudacre data sources**
  - MySQL database – customer account data (name, address, phone numbers, devices)
  - Apache web server logs from Customer Service site
  - HTML files – Knowledge base articles
  - XML files – Device activation records
  - Real-time device status logs
  - Base stations – cell tower locations

Introduction to Exercises: Getting Started

- **Instructions are in the Hands-On Exercise Manual**
- **Start with**
  - General Notes
  - Setting Up
    - Run setup script for the course

Introduction to Exercises: Classroom Virtual Machine

- **Your virtual machine**
  - Log in as user **`training`** (password **`training`**)
  - Pre-installed and configured with
    - Spark and CDH (Cloudera's Distribution, including Apache Hadoop)
    - Various tools including Firefox, gedit, Emacs, Eclipse, and Maven
- **Training materials: `~/training_materials/dev1` folder on the VM**
  - **`examples`** – all the example code in this course
  - **`exercises`** – starter files, scripts and solutions for the Hands-On Exercises
  - **`scripts`** – course setup scripts
- **Course data: `~/training_materials/data`**

Chapter Topics

Introduction to Hadoop and the Hadoop Ecosystem

Introduction to Hadoop

- Problems with Traditional Large-scale Systems
- Hadoop!
- Data Storage and Ingest
- Data Processing
- Data Analysis and Exploration
- Other Ecosystem Tools
- Introduction to the Hands-On Exercises
- **Conclusion**

Essential Points

- **Hadoop is a framework for distributed storage and processing**
- **Core Hadoop includes HDFS for storage and YARN for cluster resource management**
- **The Hadoop ecosystem includes many components for**
  - Ingesting data (Flume, Sqoop, Kafka)
  - Storing data (HDFS, HBase)
  - Processing data (Spark, Hadoop MapReduce, Pig)
  - Modeling data as tables for SQL access (Impala, Hive)
  - Exploring data (Hue, Search)
  - Protecting Data (Sentry)
- **This course introduces most of the key Hadoop infrastructure**
- **Hands-On Exercises let you practice and refine your Hadoop skills!**

Bibliography

The following offer more information on topics discussed in this chapter

- ***Hadoop: The Definitive Guide*** **(published by O'Reilly)**
  - `http://tiny.cloudera.com/hadooptdg`
- ***Cloudera Essentials for Apache Hadoop*** **– free online training**
  - `http://tiny.cloudera.com/esscourse`

cloudera®

Hadoop Architecture and HDFS

Chapter 3

Course Chapters

▪ Introduction	Course Introduction
▪ Introduction to Hadoop and the Hadoop Ecosystem ▪ **Hadoop Architecture and HDFS**	**Introduction to Hadoop**
▪ Importing Relational Data with Apache Sqoop ▪ Introduction to Impala and Hive ▪ Modeling and Managing Data with Impala and Hive ▪ Data Formats ▪ Data File Partitioning	Importing and Modeling Structured Data
▪ Capturing Data with Apache Flume	Ingesting Streaming Data
▪ Spark Basics ▪ Working with RDDs in Spark ▪ Aggregating Data with Pair RDDs ▪ Writing and Deploying Spark Applications ▪ Parallel Processing in Spark ▪ Spark RDD Persistence ▪ Common Patterns in Spark Data Processing ▪ Spark SQL and DataFrames	Distributed Data Processing with Spark
▪ Conclusion	Course Conclusion

Hadoop Architecture and HDFS

In this chapter you will learn

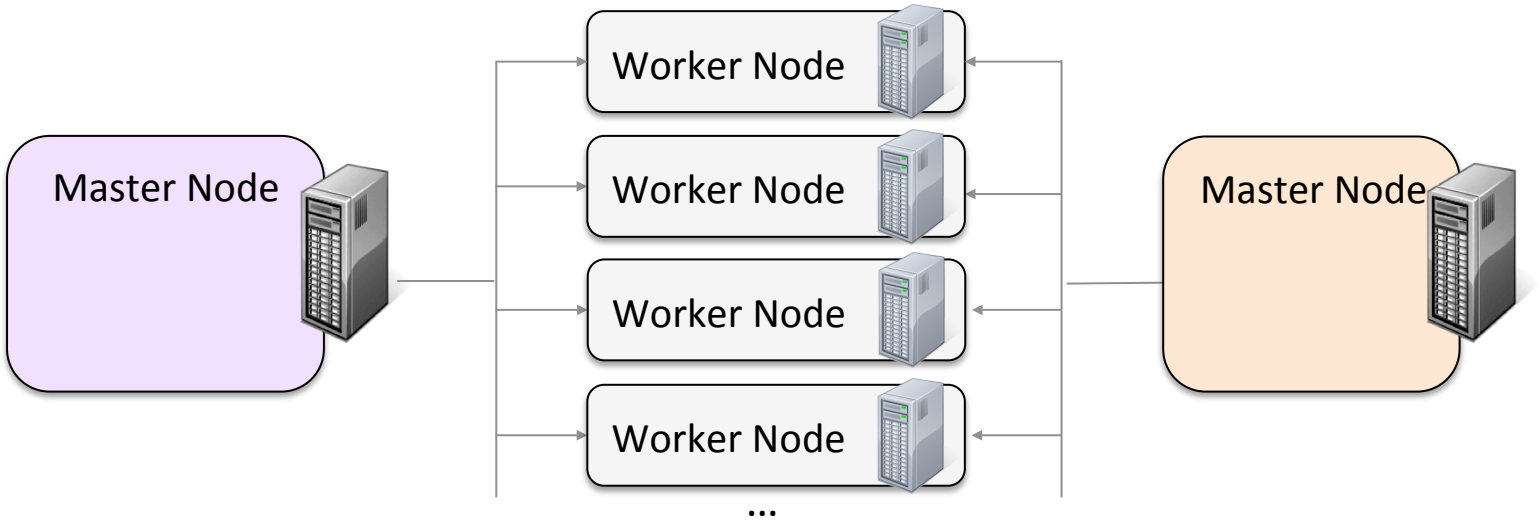
- **How Hadoop Distributed File System stores data across a cluster**
- **How to use HDFS using the Hue File Browser or the `hdfs` command**
- **How Hadoop YARN provides cluster resource management for distributed data processing**
- **How to use Hue, the YARN Web UI or the yarn command to monitor your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

Chapter Topics

- **Distributed Processing on a Cluster**
- Storage: HDFS Architecture
- Storage: Using HDFS
- Hands-on Exercises: Access HDFS with Command Line and Hue
- Resource Management: YARN Architecture
- Resource Management: Working with YARN
- Conclusion
- Hands-On Exercises: Run a YARN Job

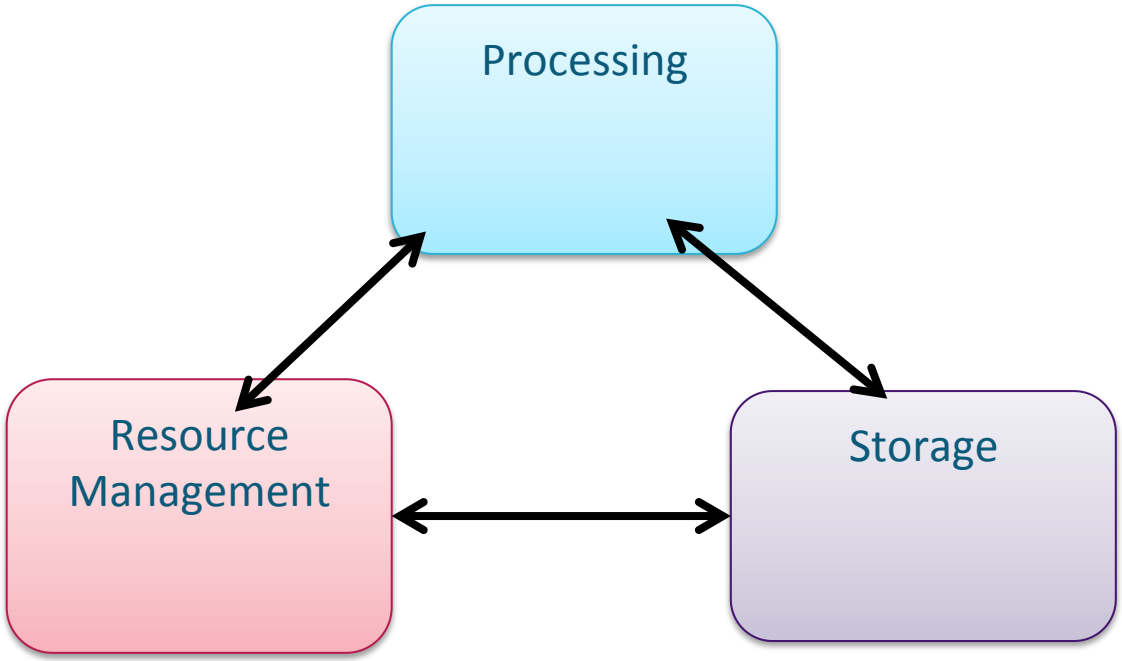
Hadoop Cluster Terminology

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



Cluster Components

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

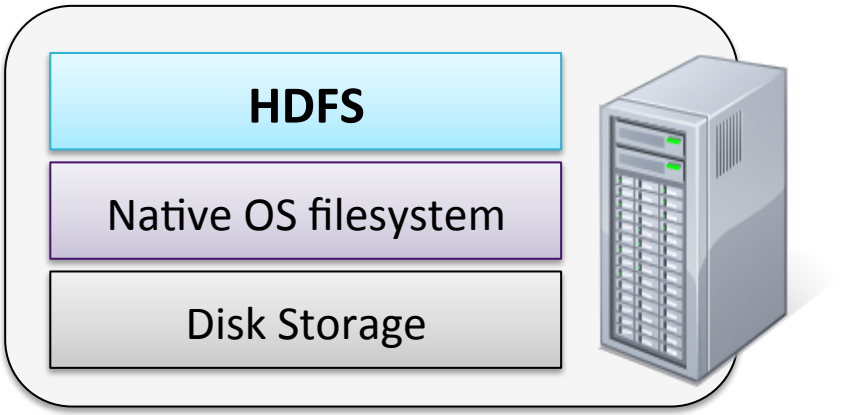


Chapter Topics

- Distributed Processing on a Cluster
- **Storage: HDFS Architecture**
- Storage: Using HDFS
- Hands-on Exercises: Access HDFS with Command Line and Hue
- Resource Management: YARN Architecture
- Resource Management: Working with YARN
- Conclusion
- Hands-On Exercises: Run a YARN Job

HDFS Basic Concepts (1)

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

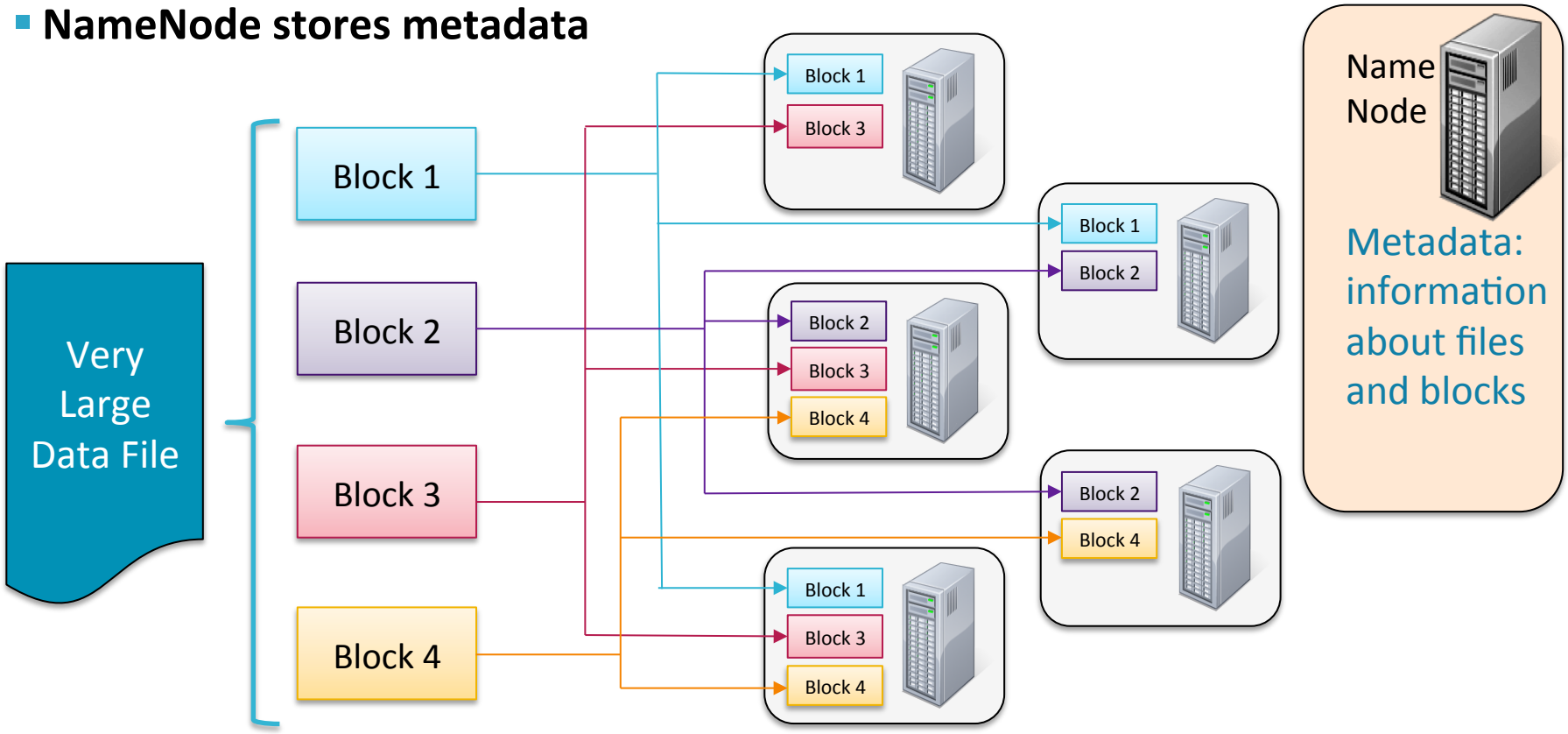


HDFS Basic Concepts (2)

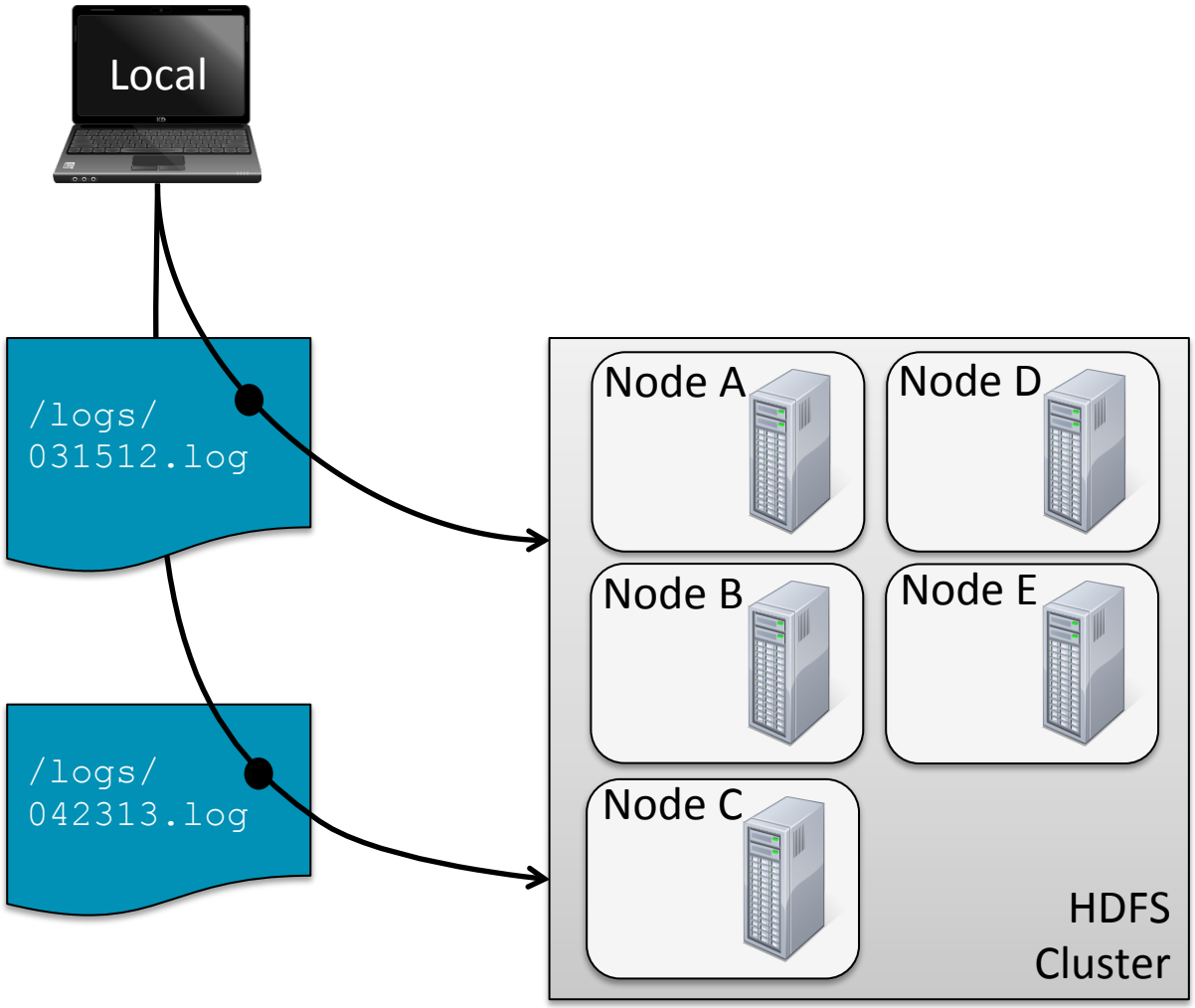
- **HDFS performs best with a 'modest' number of large files**
  - Millions, rather than billions, of files
  - Each file typically 100MB or more
- **Files in HDFS are 'write once'**
  - No random writes to files are allowed
- **HDFS is optimized for large, streaming reads of files**
  - Rather than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

How Files Are Stored

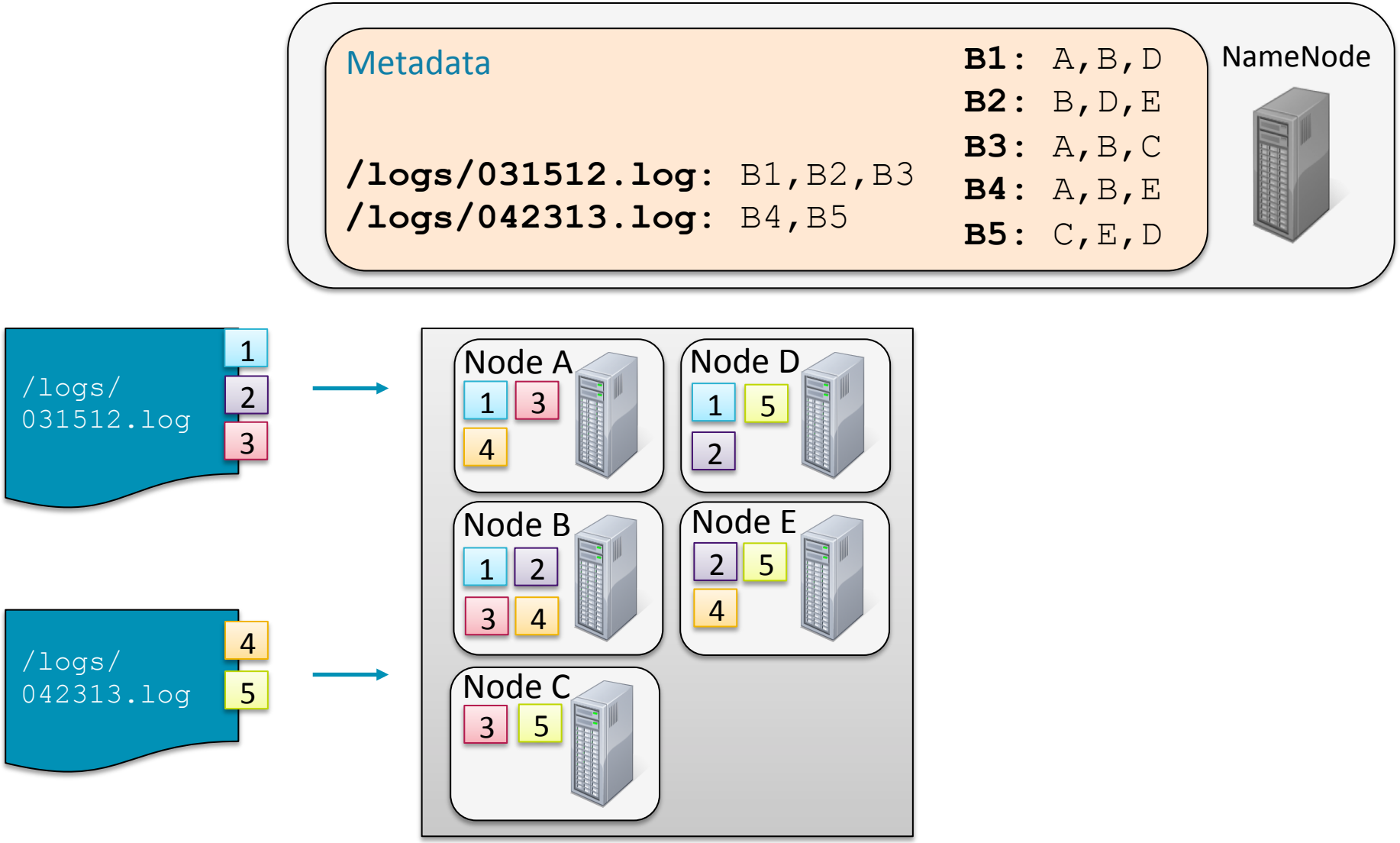
- **Data files are split into 128MB blocks which are distributed at load time**
- **Each block is replicated on multiple data nodes (default 3x)**
- **NameNode stores metadata**



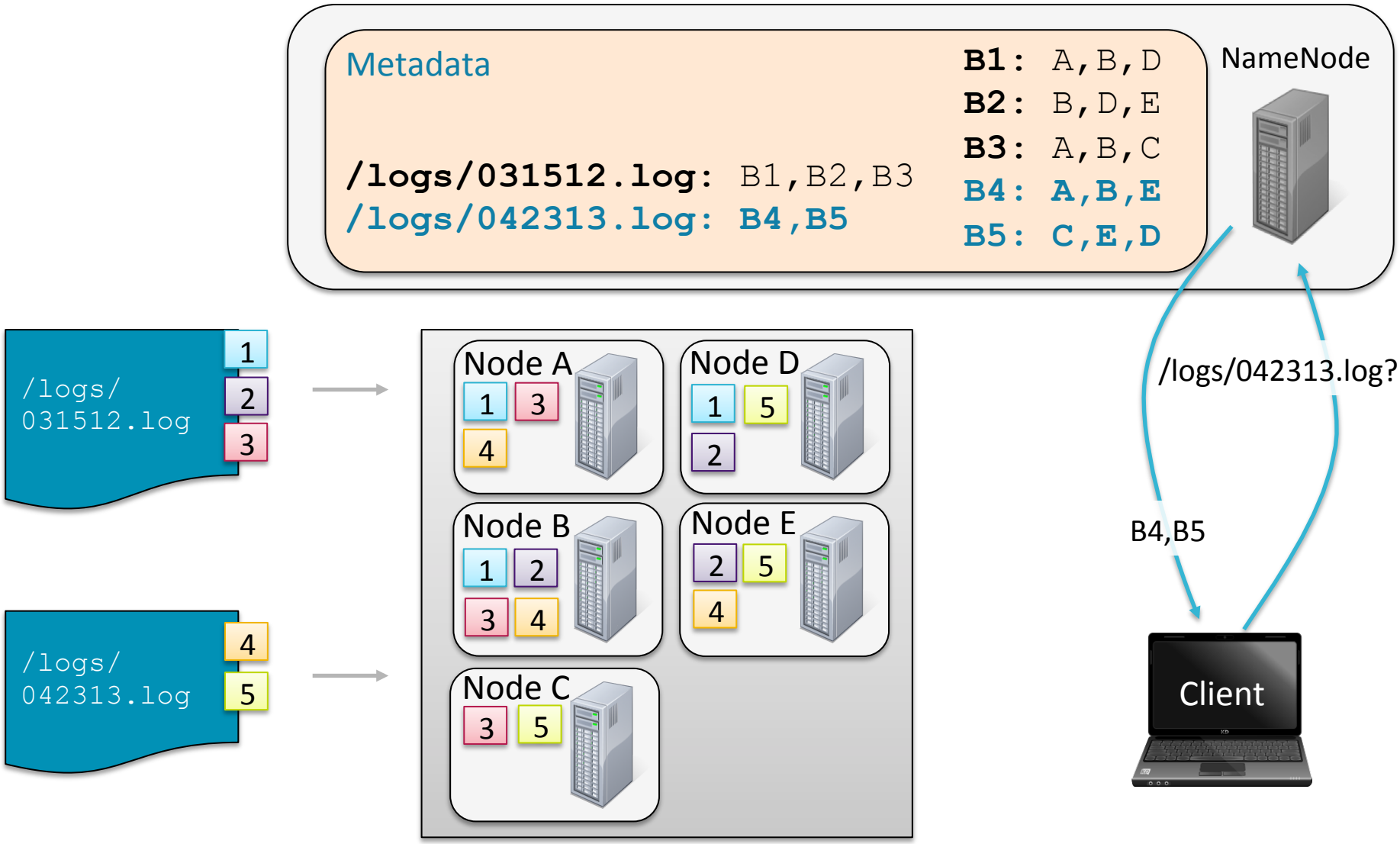
Example: Storing and Retrieving Files (1)



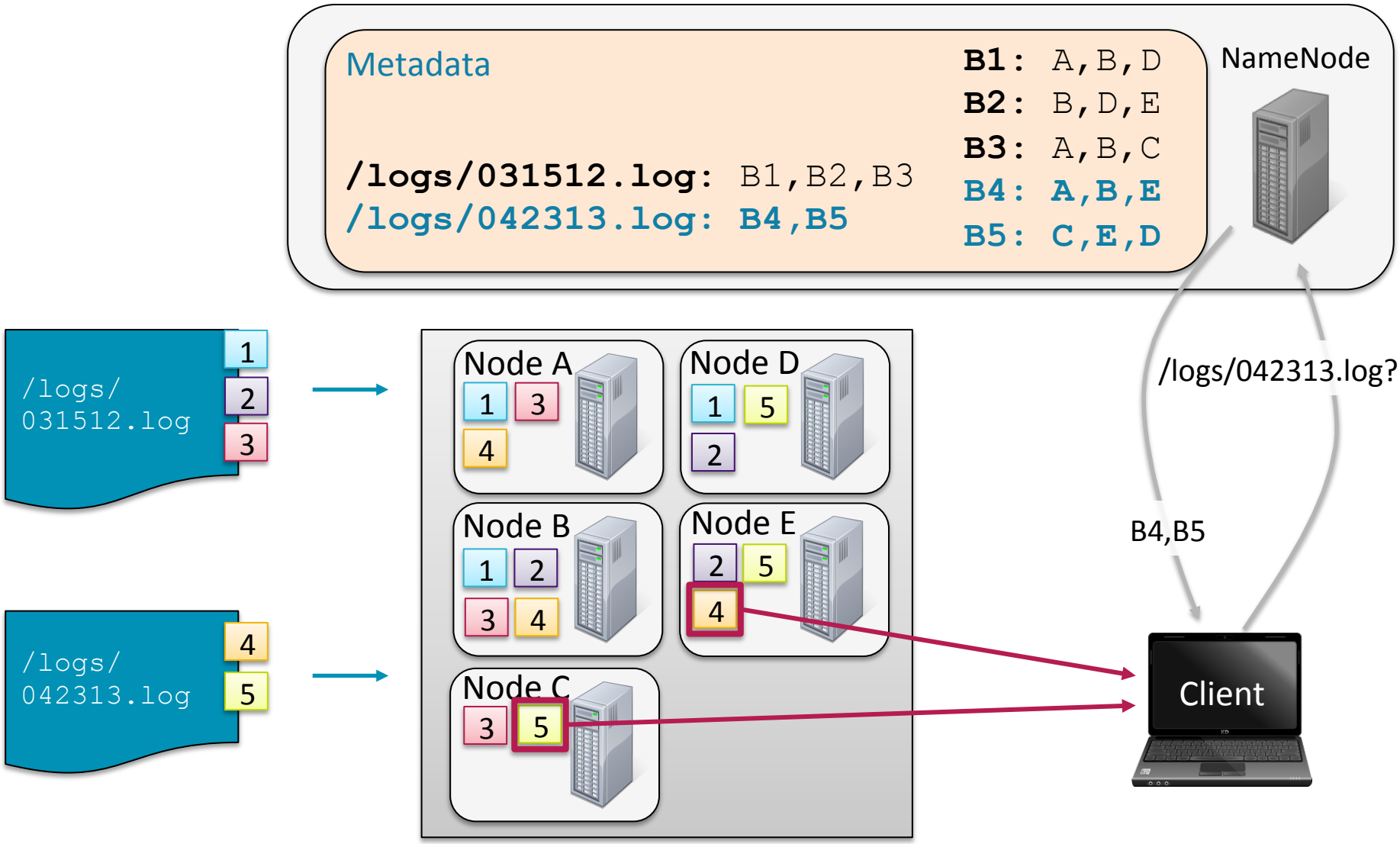
Example: Storing and Retrieving Files (2)



Example: Storing and Retrieving Files (3)



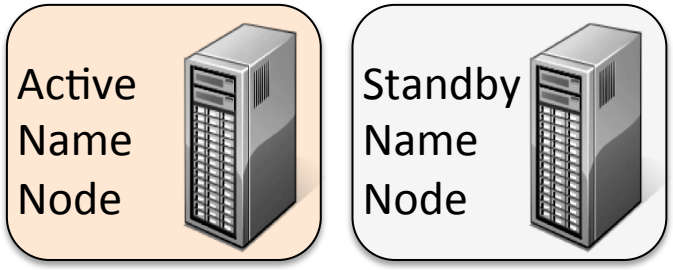
Example: Storing and Retrieving Files (4)



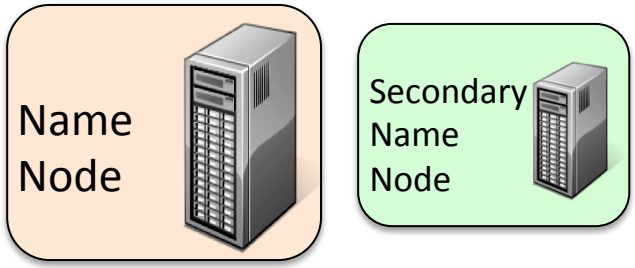
HDFS NameNode Availability

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

- **HDFS is typically set up for High Availability**
  - Two NameNodes: Active and Standby



- **Small clusters may use 'Classic mode'**
  - One NameNode
  - One "helper" node called the Secondary NameNode
    - Bookkeeping, not backup



Chapter Topics

Hadoop Architecture and HDFS | **Introduction to Hadoop**

- Distributed Processing on a Cluster
- Storage: HDFS Architecture
- **Storage: Using HDFS**
- Hands-on Exercises: Access HDFS with Command Line and Hue
- Resource Management: YARN Architecture
- Resource Management: Working with YARN
- Conclusion
- Hands-On Exercises: Run a YARN Job

Options for Accessing HDFS

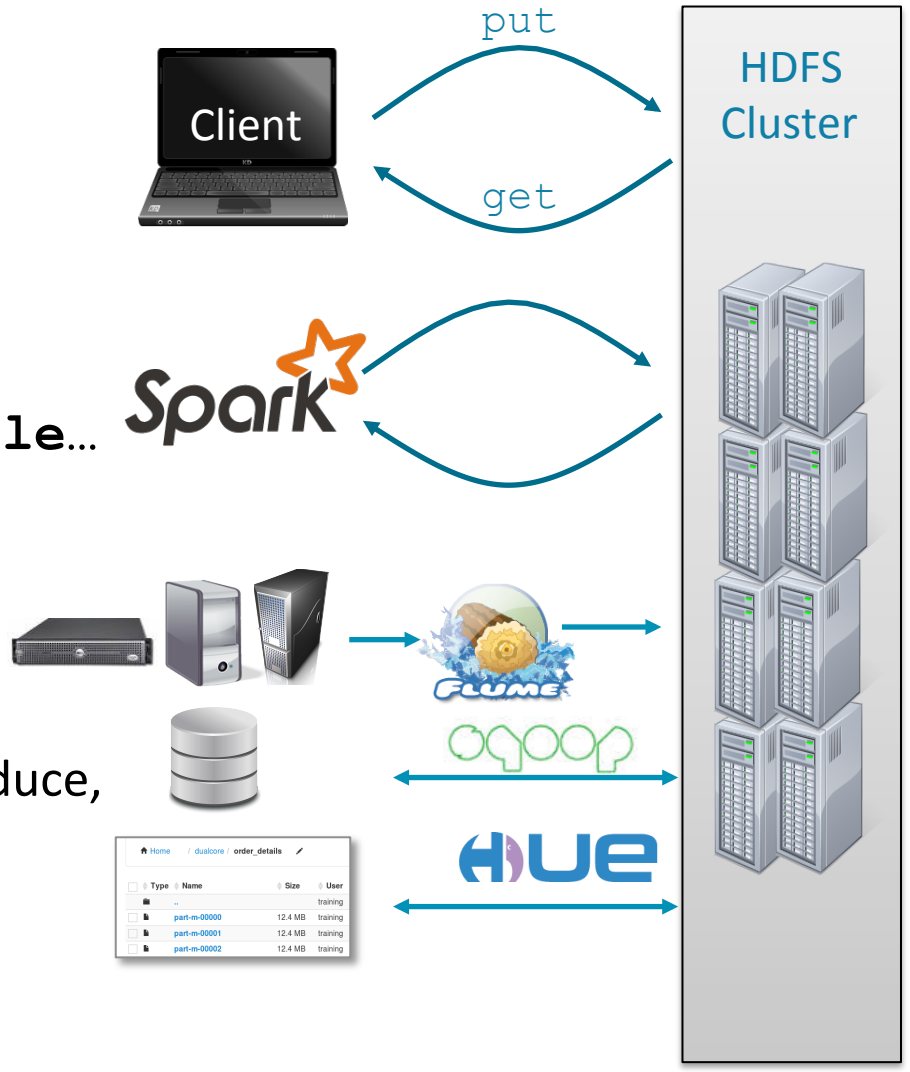
- **From the command line**
  - FsShell:

```
$ hdfs dfs
```

- **In Spark**
  - By URI, e.g.

```
hdfs://nnhost:port/file…
```

- **Other programs**
  - Java API
    - Used by Hadoop MapReduce, Impala, Hue, Sqoop, Flume, etc.
  - RESTful interface



HDFS Command Line Examples (1)

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

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

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

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

```
$ hdfs dfs -ls
```

- **Get a directory listing of the HDFS root directory**

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

HDFS Command Line Examples (2)

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

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

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

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

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

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

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

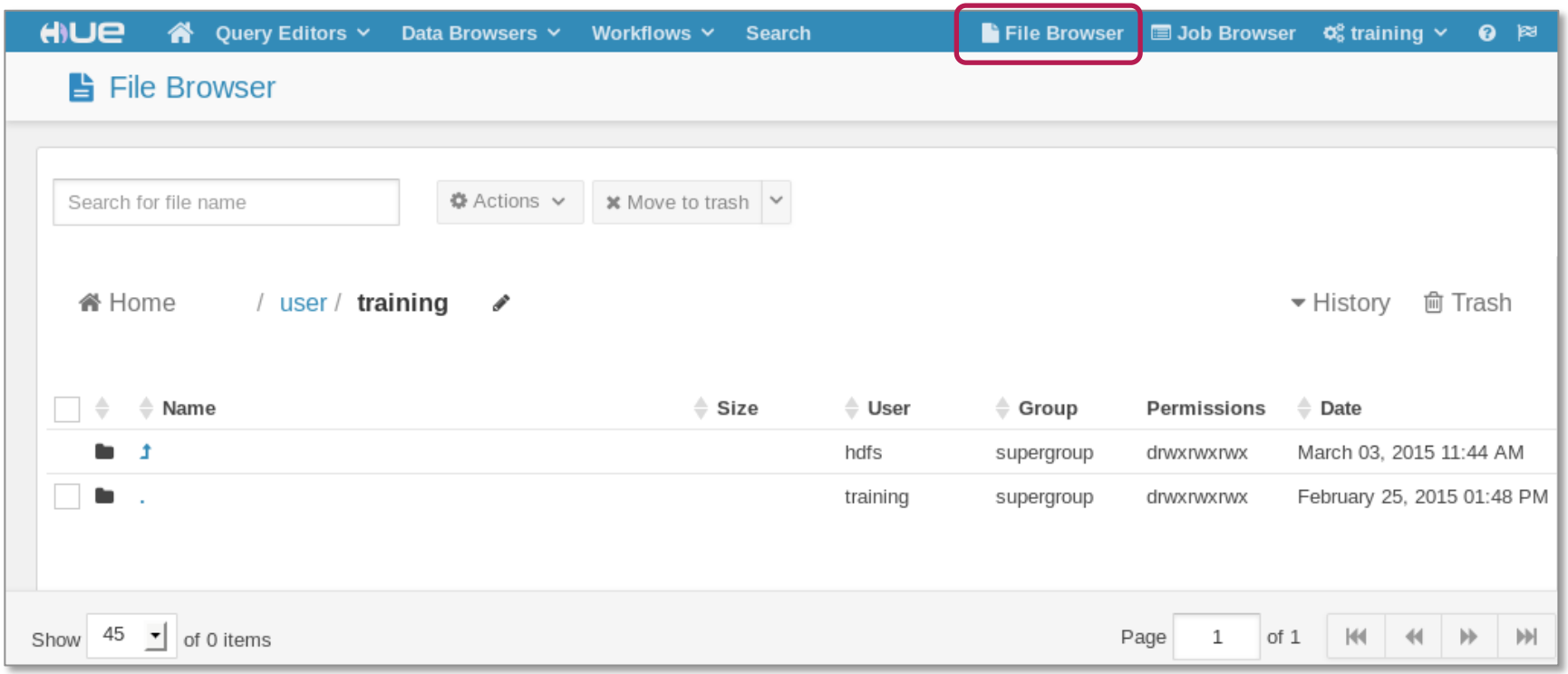
HDFS Command Line Examples (3)

- **Delete the directory `input_old` and all its contents**

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

The Hue HDFS File Browser

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



HDFS Recommendations

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

Chapter Topics

Hadoop Architecture and HDFS | **Introduction to Hadoop**

- Distributed Processing on a Cluster
- Storage: HDFS Architecture
- Storage: Using HDFS
- **Hands-on Exercises: Access HDFS with Command Line and Hue**
- Resource Management: YARN Architecture
- Resource Management: Working With YARN
- Conclusion
- Hands-On Exercises: Run a YARN Job

Hands-On Exercise: Access HDFS with Command Line and Hue

- **In this exercise you will**
  - Create a `/loudacre` base directory for course exercises
  - Practice uploading and viewing data files
- **Please refer to the Hands-On Exercise Manual**

Chapter Topics

- Distributed Processing on a Cluster
- Storage: HDFS Architecture
- Storage: Using HDFS
- Hands-on Exercises: Access HDFS with Command Line and Hue
- **Resource Management: YARN Architecture**
- Resource Management: Working With YARN
- Conclusion
- Hands-On Exercises: Run a YARN Job

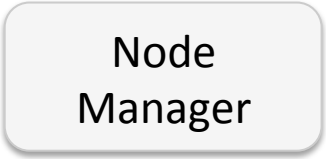
What is YARN?

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

YARN Daemons

- **Resource Manager (RM)**
  - Runs on master node
  - Global resource scheduler
  - Arbitrates system resources between competing applications
  - Has a pluggable scheduler to support different algorithms (capacity, fair scheduler, etc.)

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

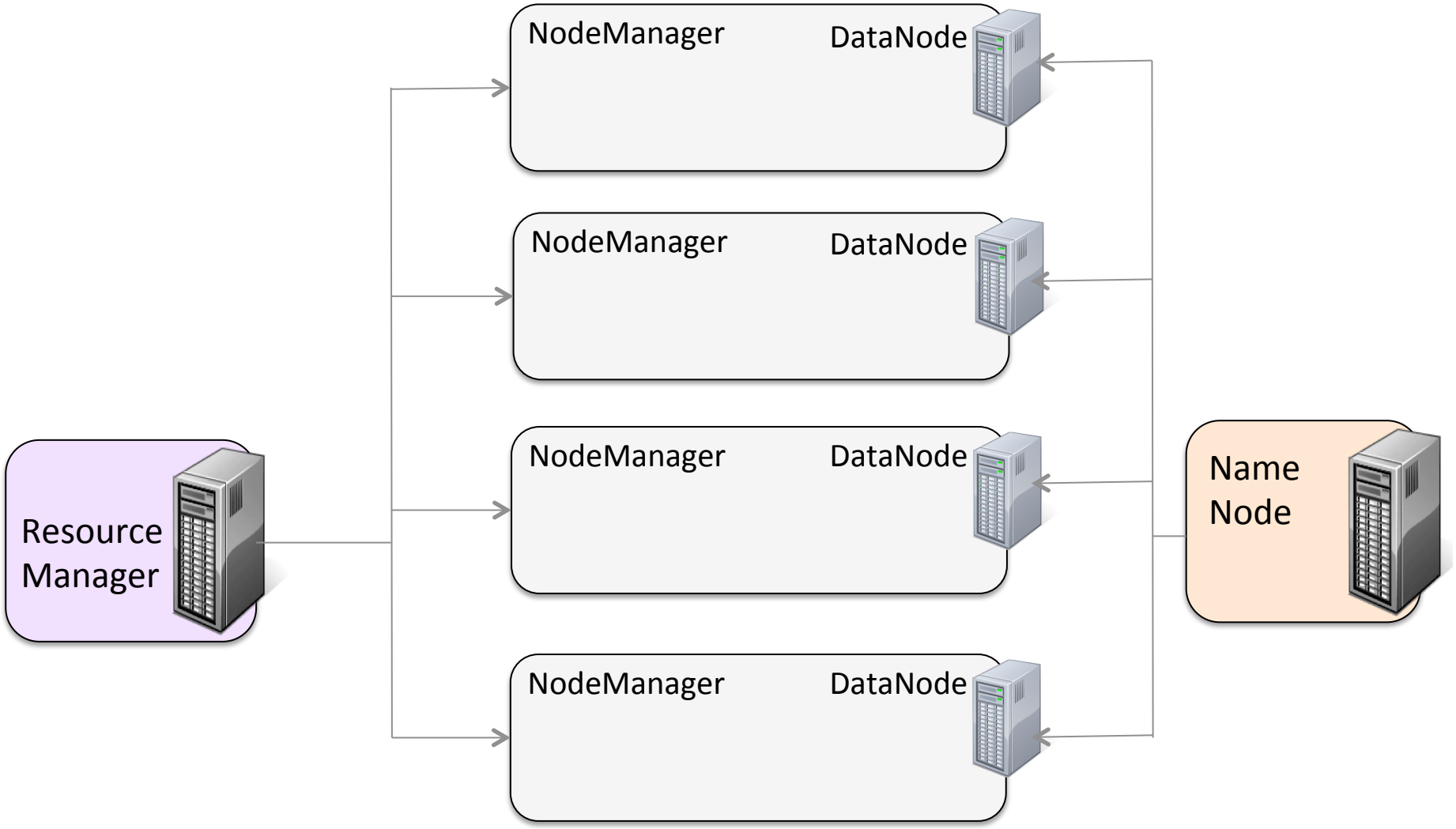


Running an Application in YARN

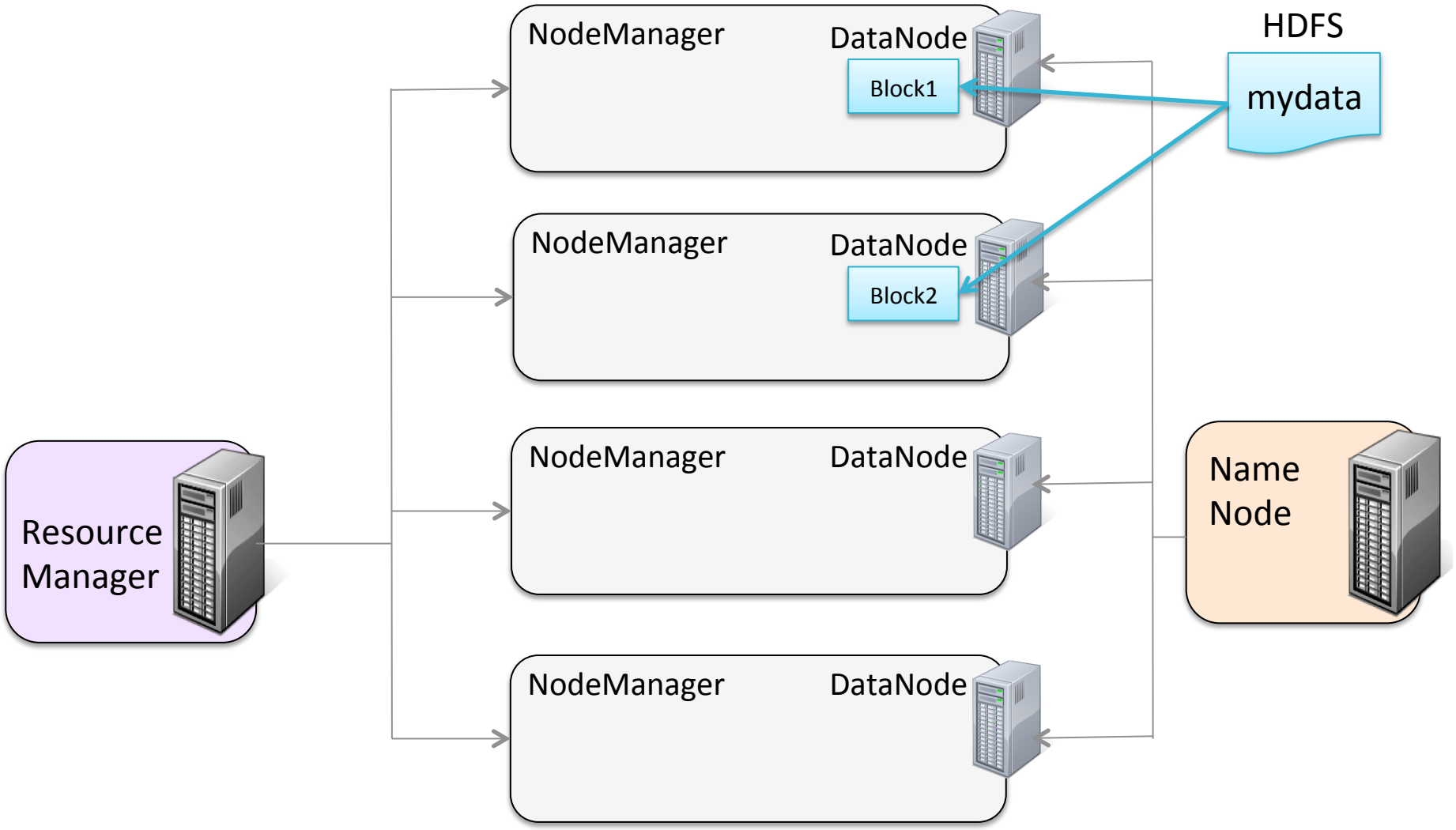
- **Containers**
  - Created by the RM upon request
  - Allocate a certain amount of resources (memory, CPU) on a slave node
  - Applications run in one or more containers

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

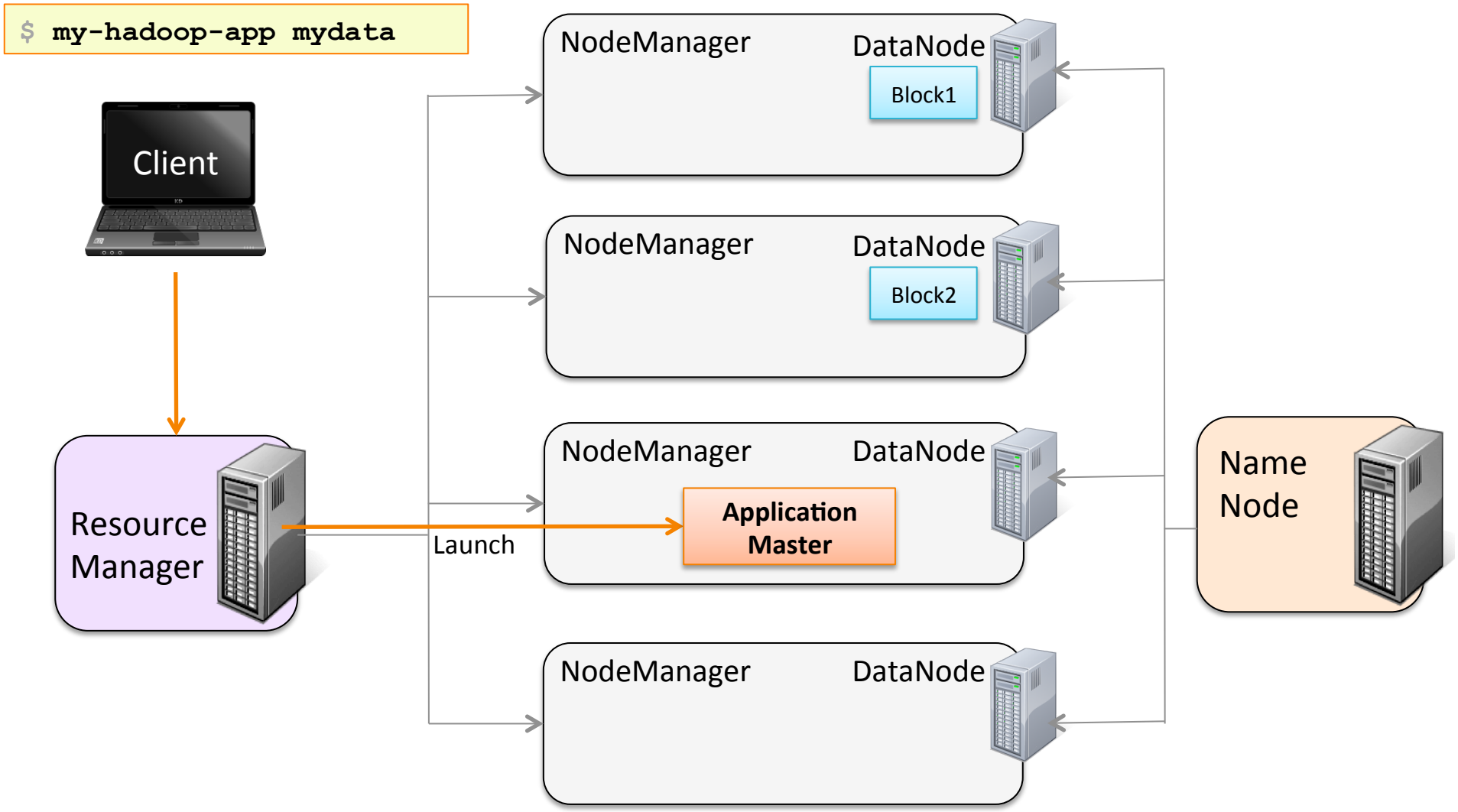
Running an Application on YARN (1)



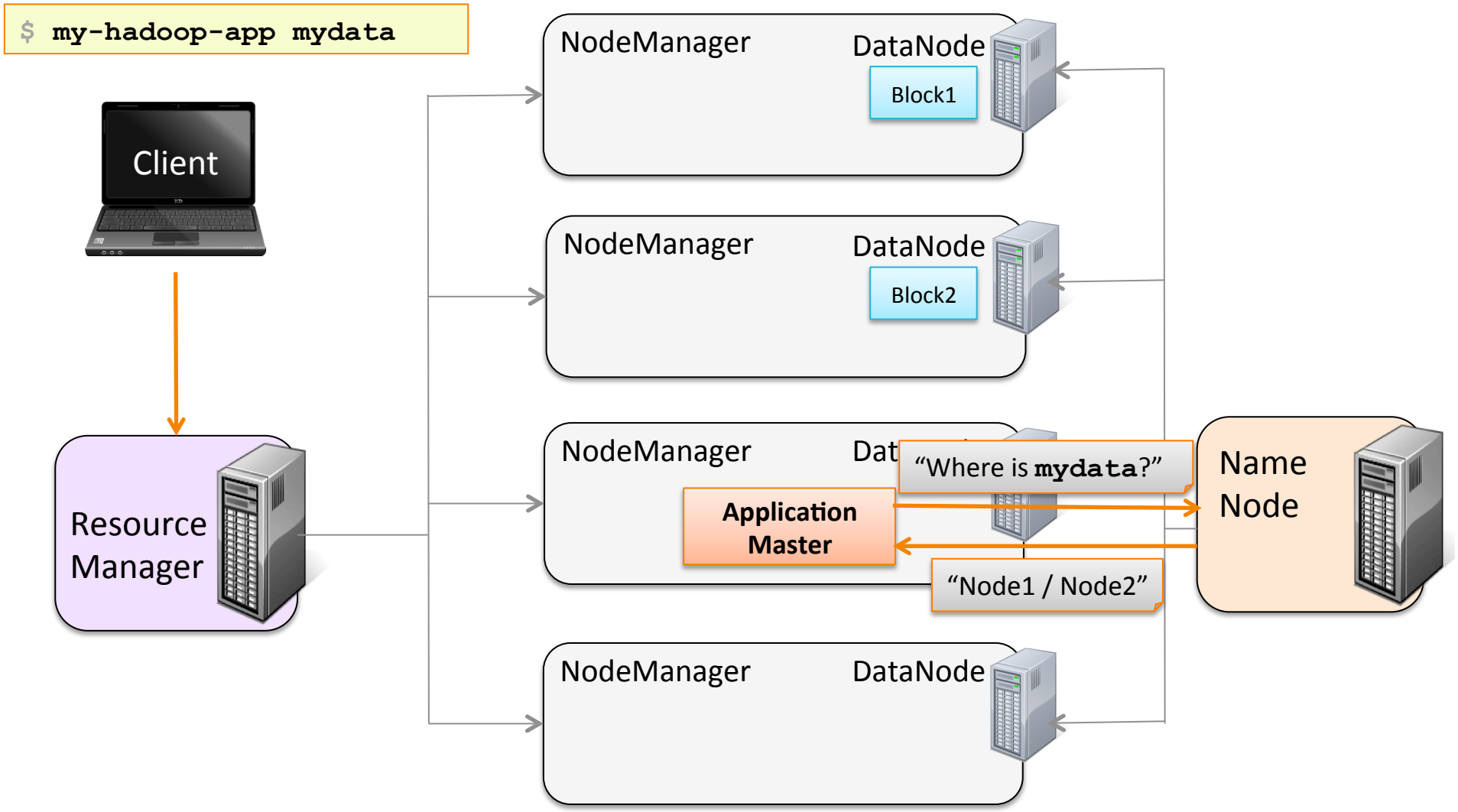
Running an Application on YARN (2)



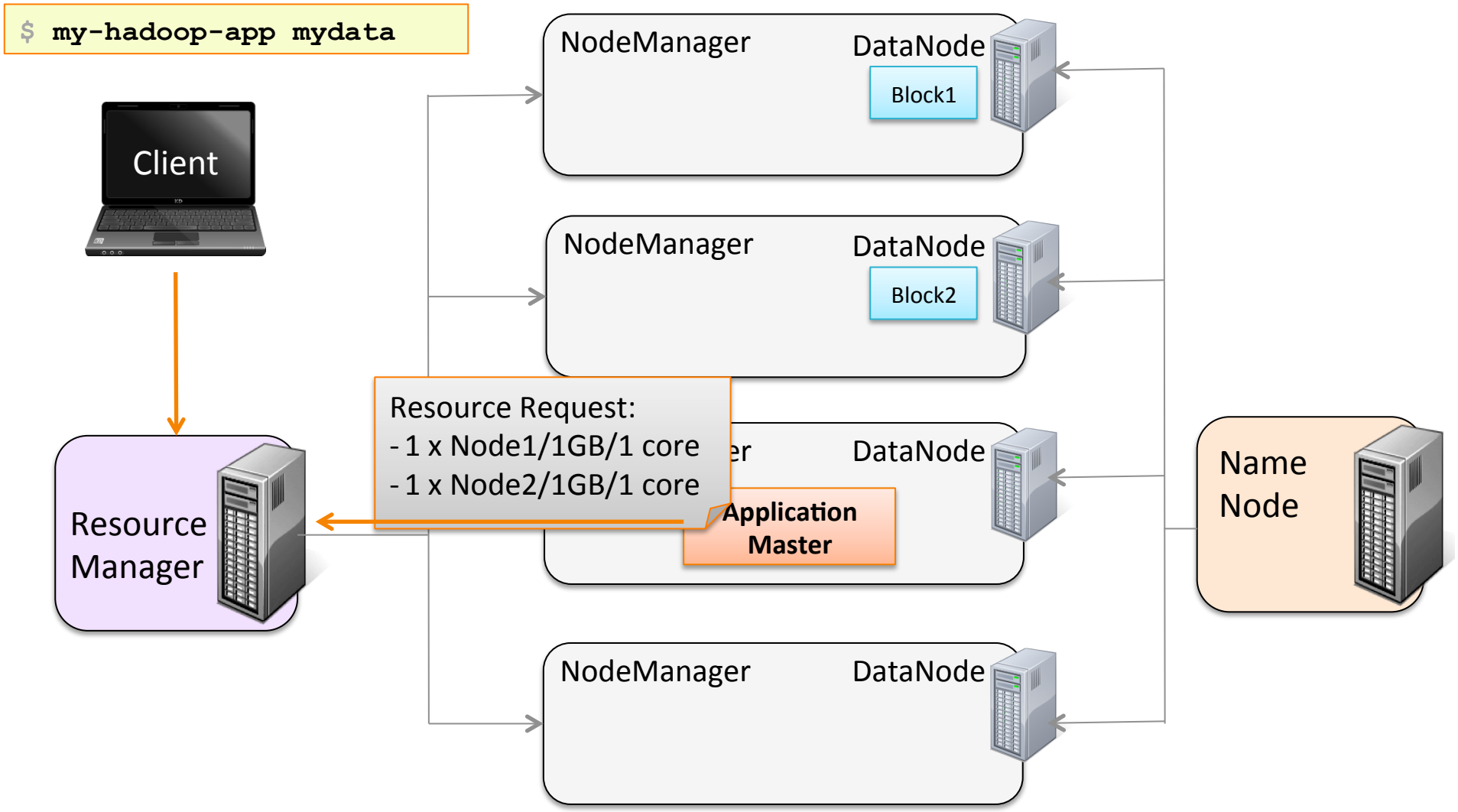
Running an Application on YARN (3)



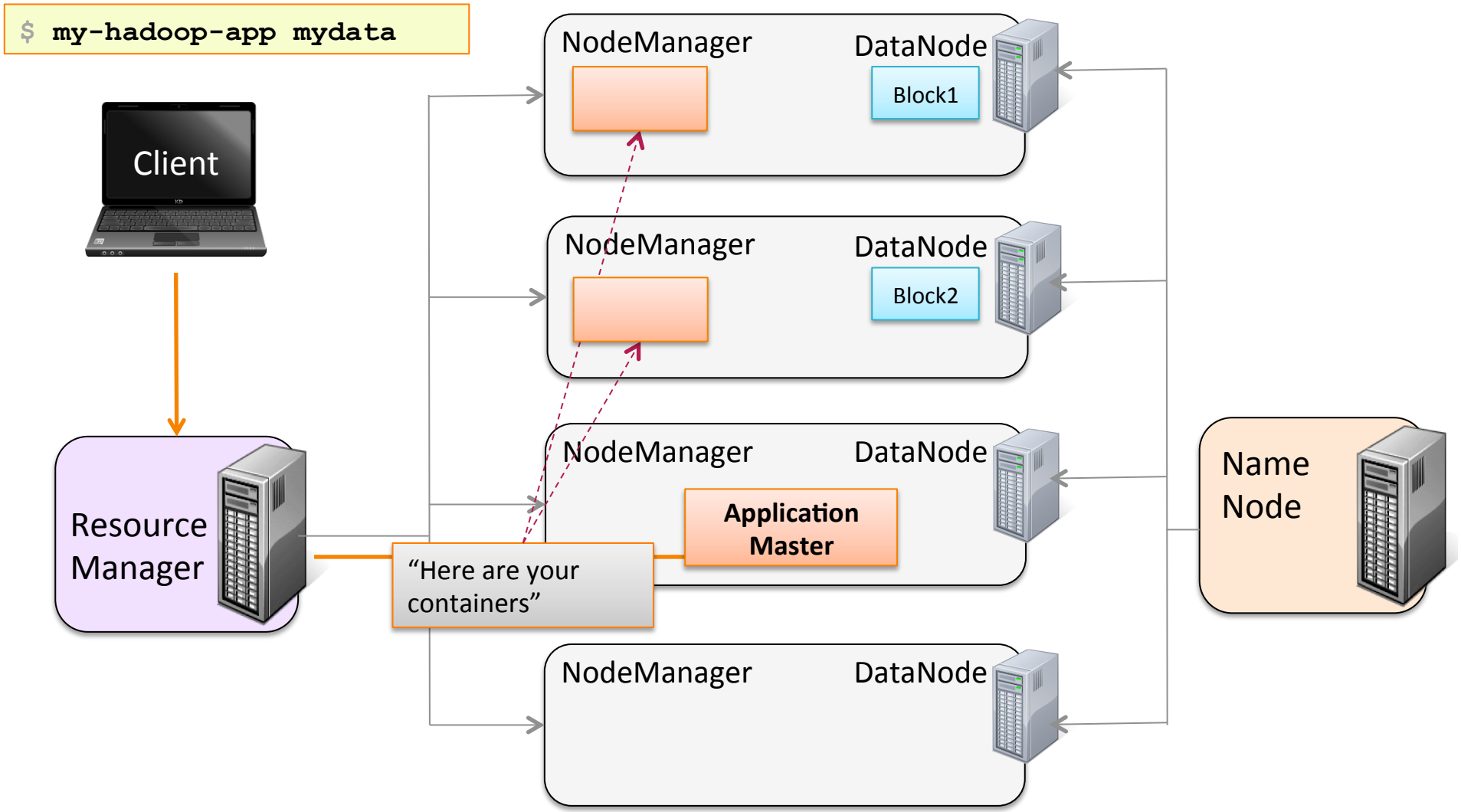
Running an Application on YARN (4)



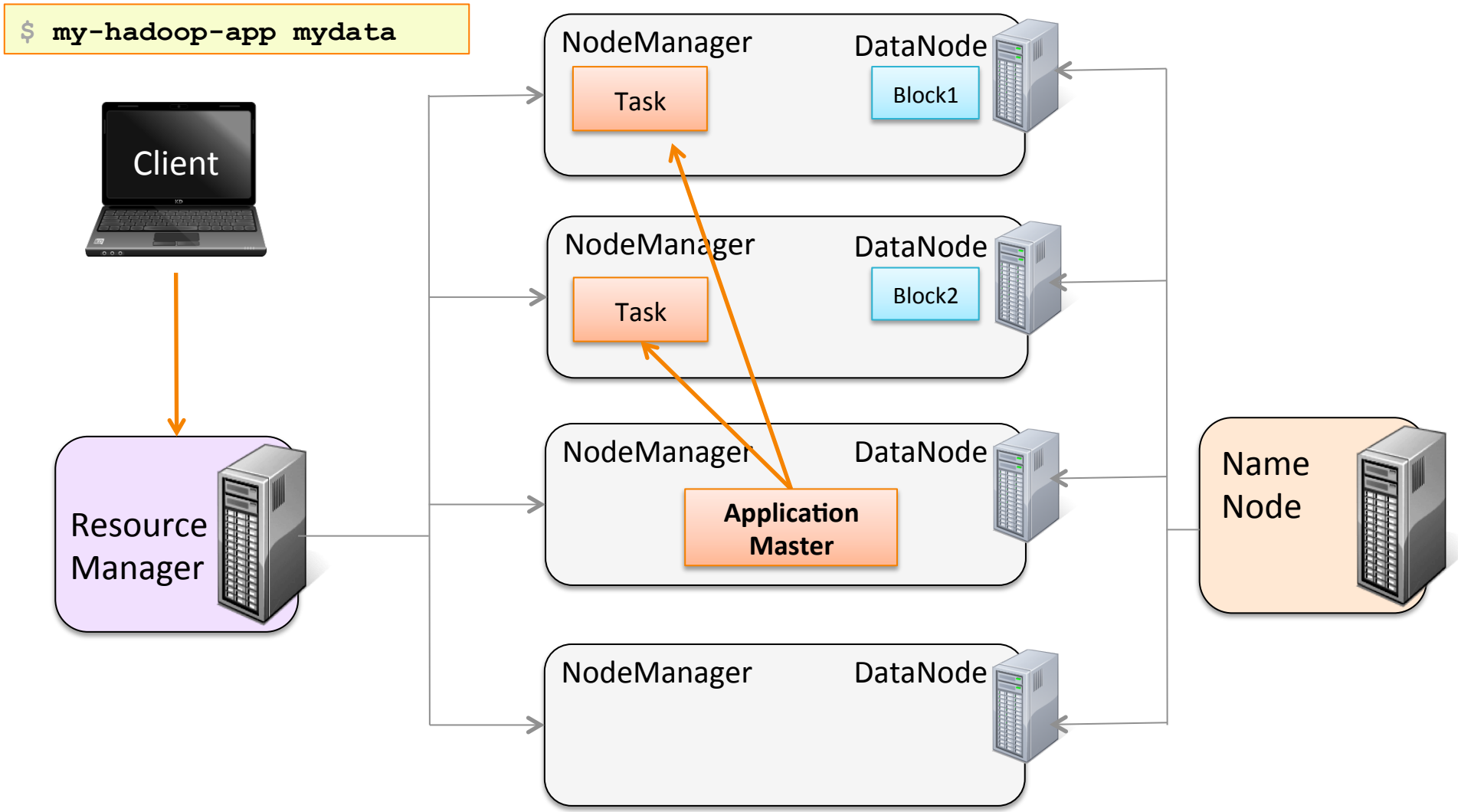
Running an Application on YARN (5)



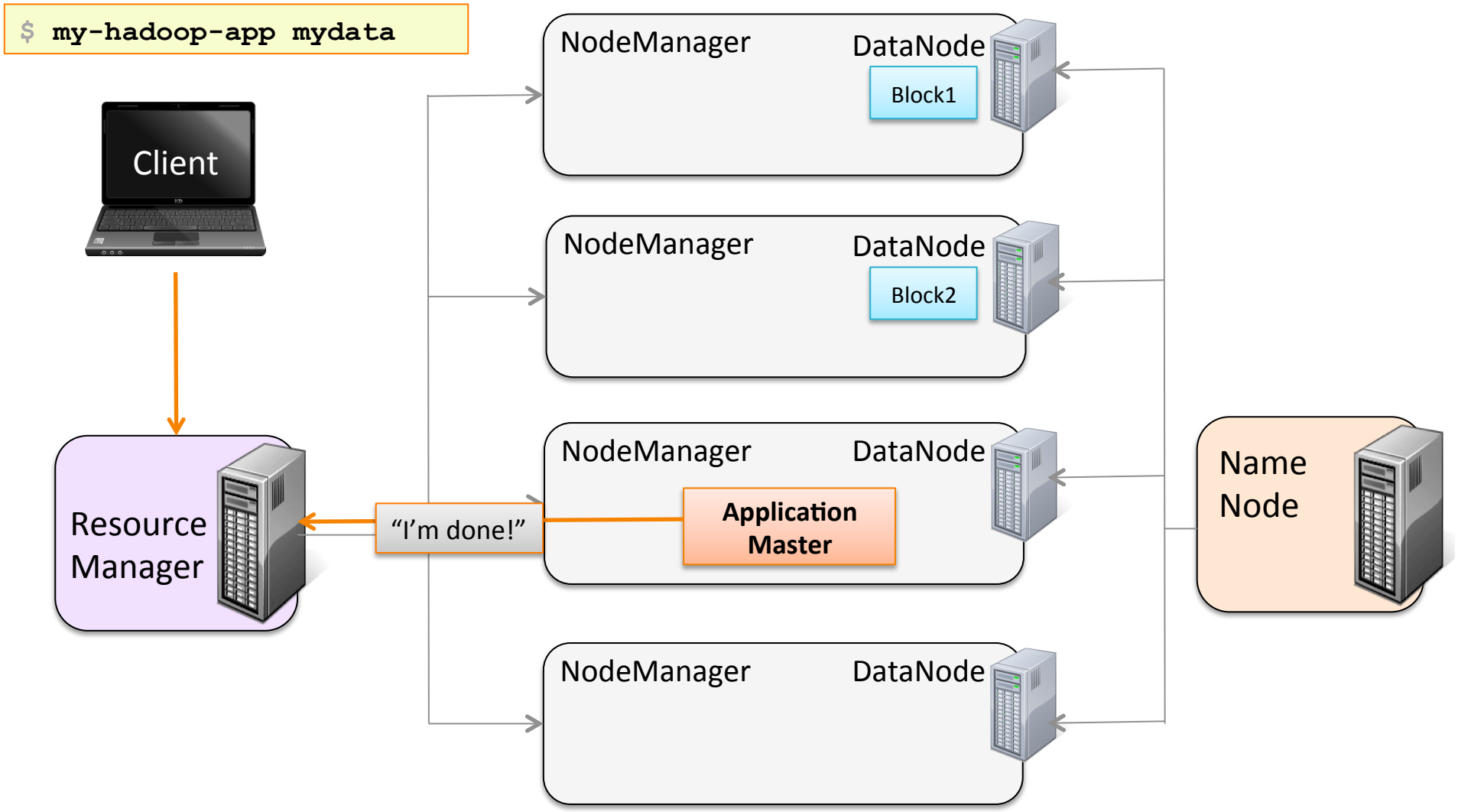
Running an Application on YARN (6)



Running an Application on YARN (7)



Running an Application on YARN (8)



Chapter Topics

Hadoop Architecture and HDFS | **Introduction to Hadoop**

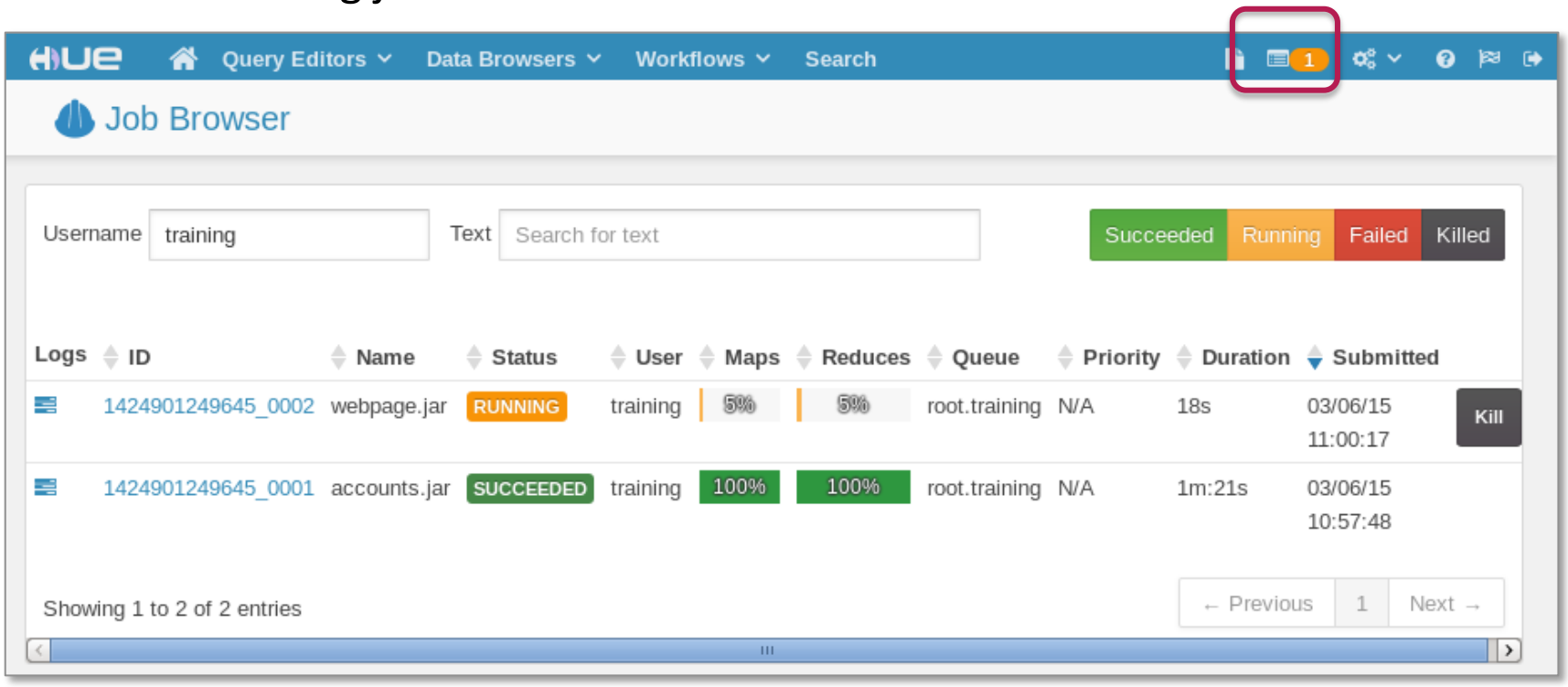
- Distributed Processing on a Cluster
- Storage: HDFS Architecture
- Storage: Using HDFS
- Hands-on Exercises: Access HDFS with Command Line and Hue
- Resource Management: YARN Architecture
- **Resource Management: Working With YARN**
- Conclusion
- Hands-On Exercises: Run a YARN Job

Working With YARN

- **Developers need to be able to**
  - Submit jobs (applications) to run on the YARN cluster
  - Monitor and manage jobs
- **Hadoop includes three major YARN tools for developers**
  - The Hue Job Browser
  - The YARN Web UI
  - The YARN command line
- **YARN administrators can use Cloudera Manager**
  - May also be helpful for developers
  - Included in Cloudera Express and Cloudera Enterprise
  - Not covered in this course

The Hue Job Browser

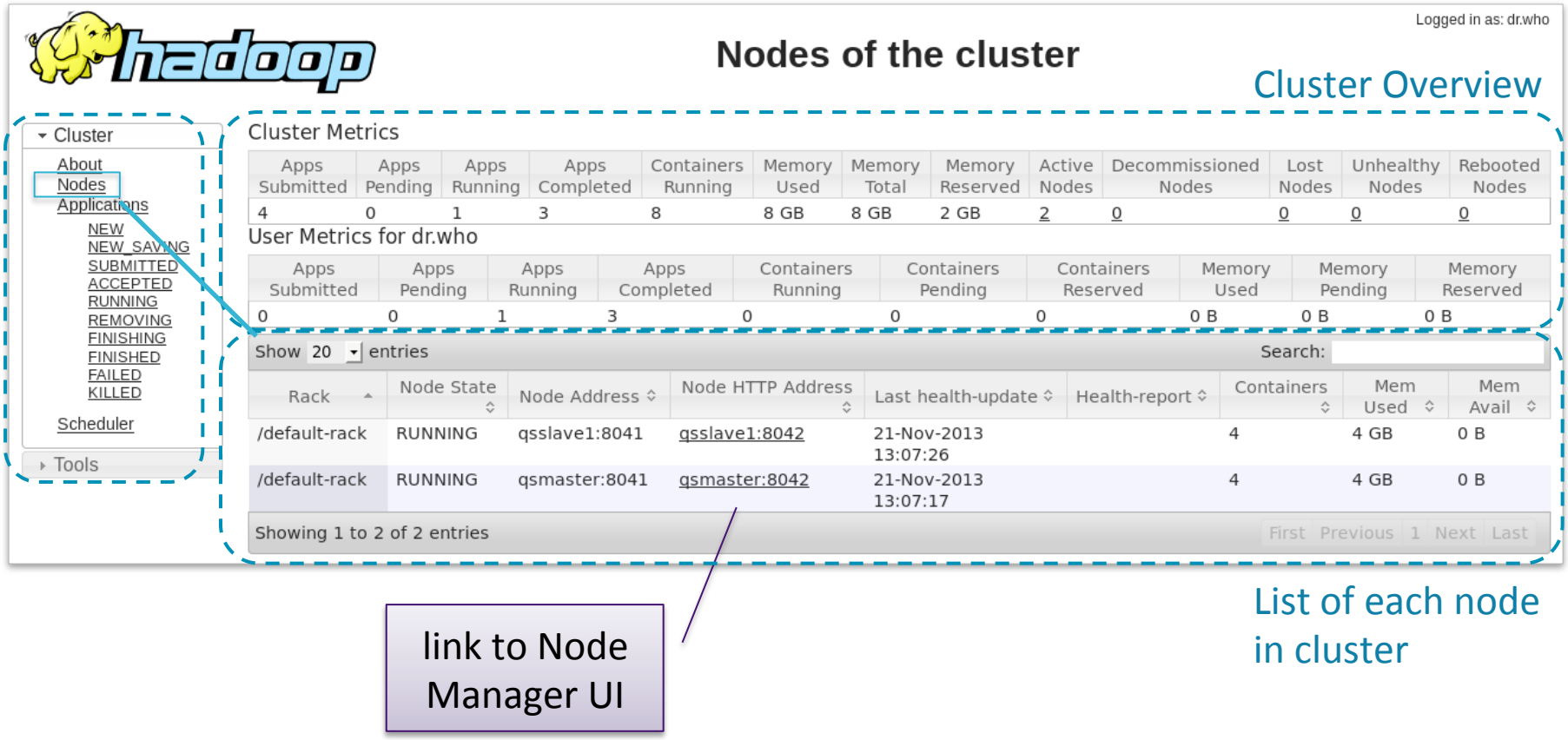
- **The Hue Job Browser allows you to**
  - Monitor the status of a job
  - View the logs
  - Kill a running job



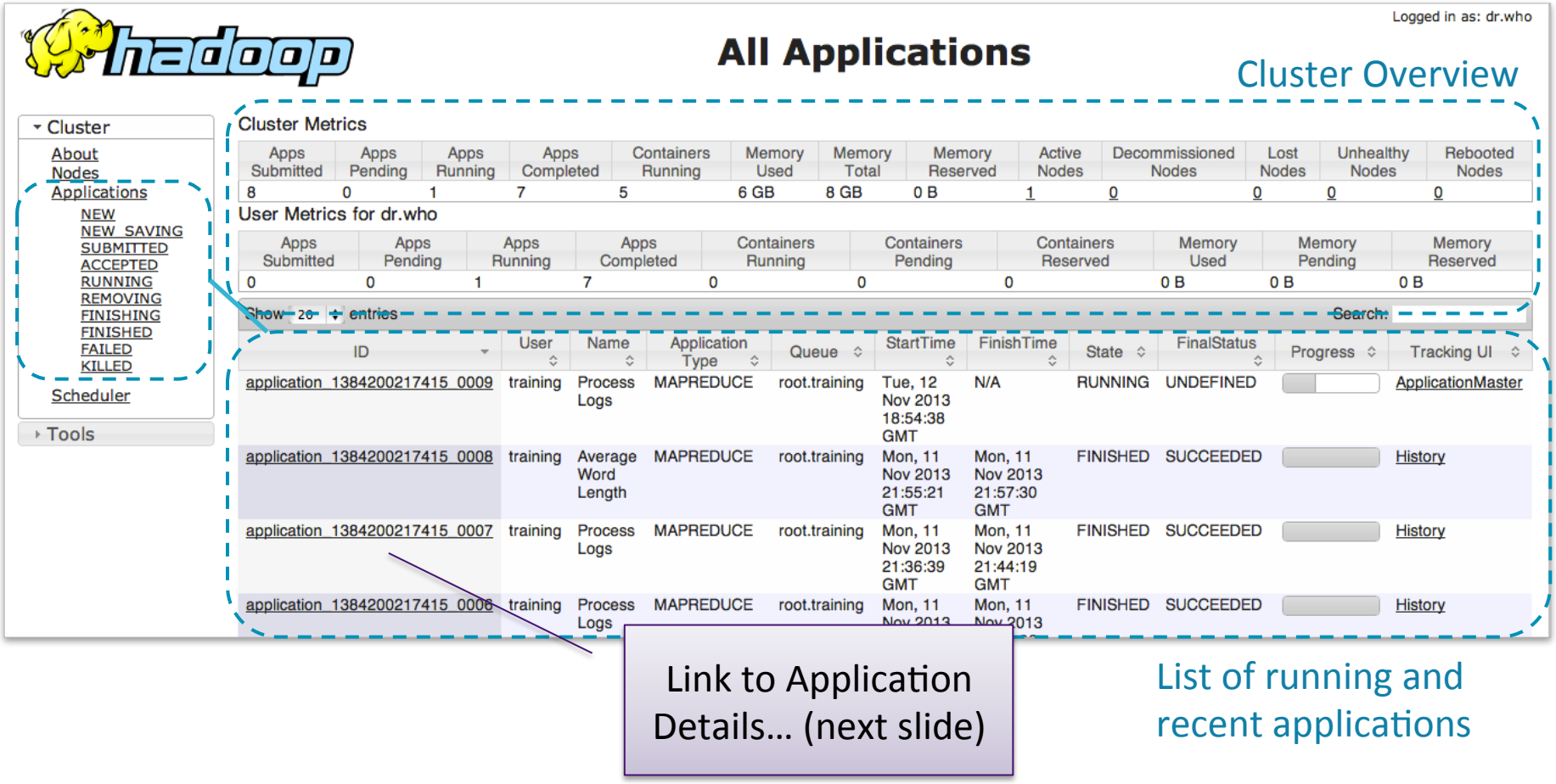
The YARN Web UI

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

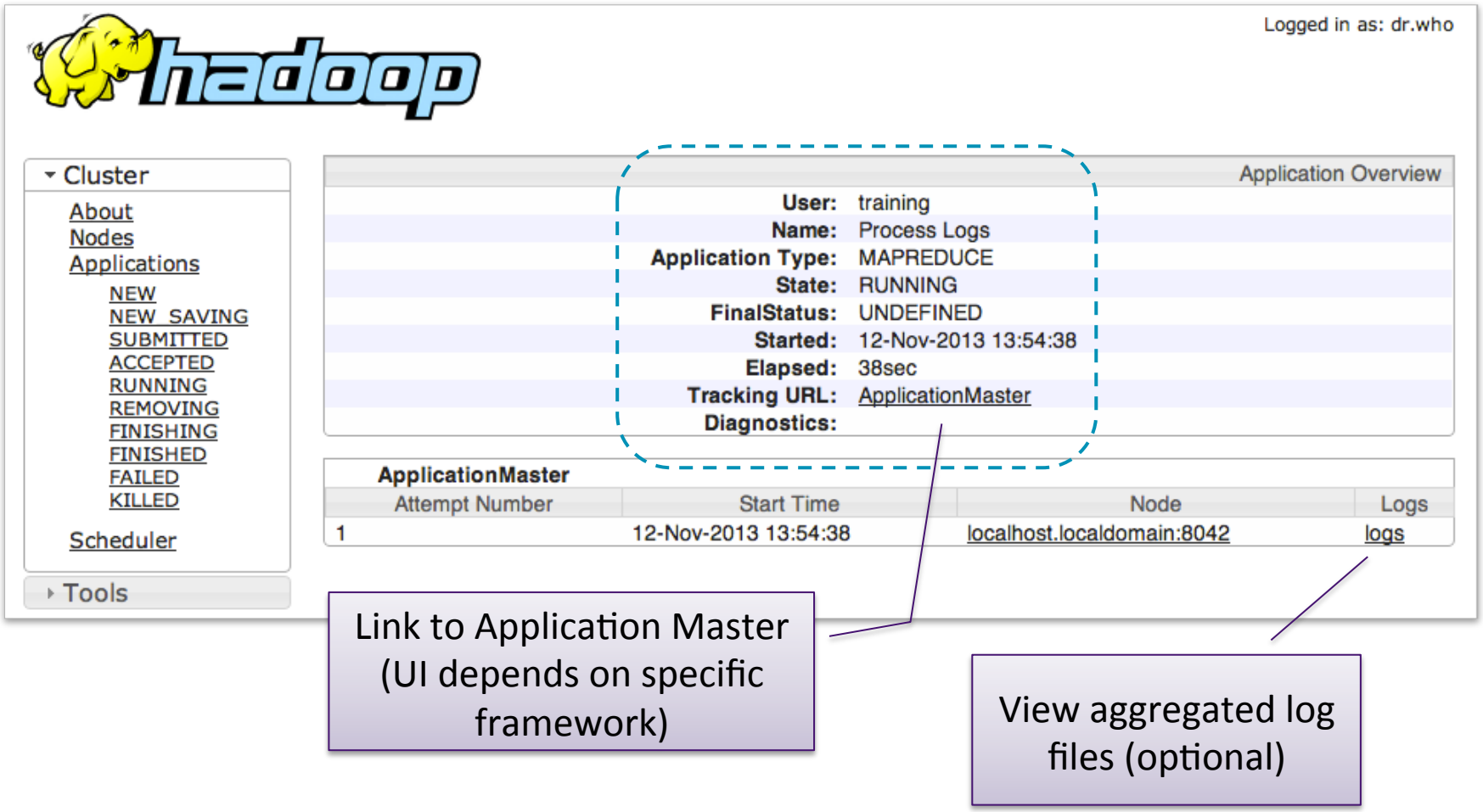
Resource Manager UI: Nodes



Resource Manager UI: Applications

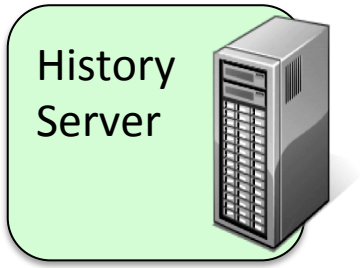


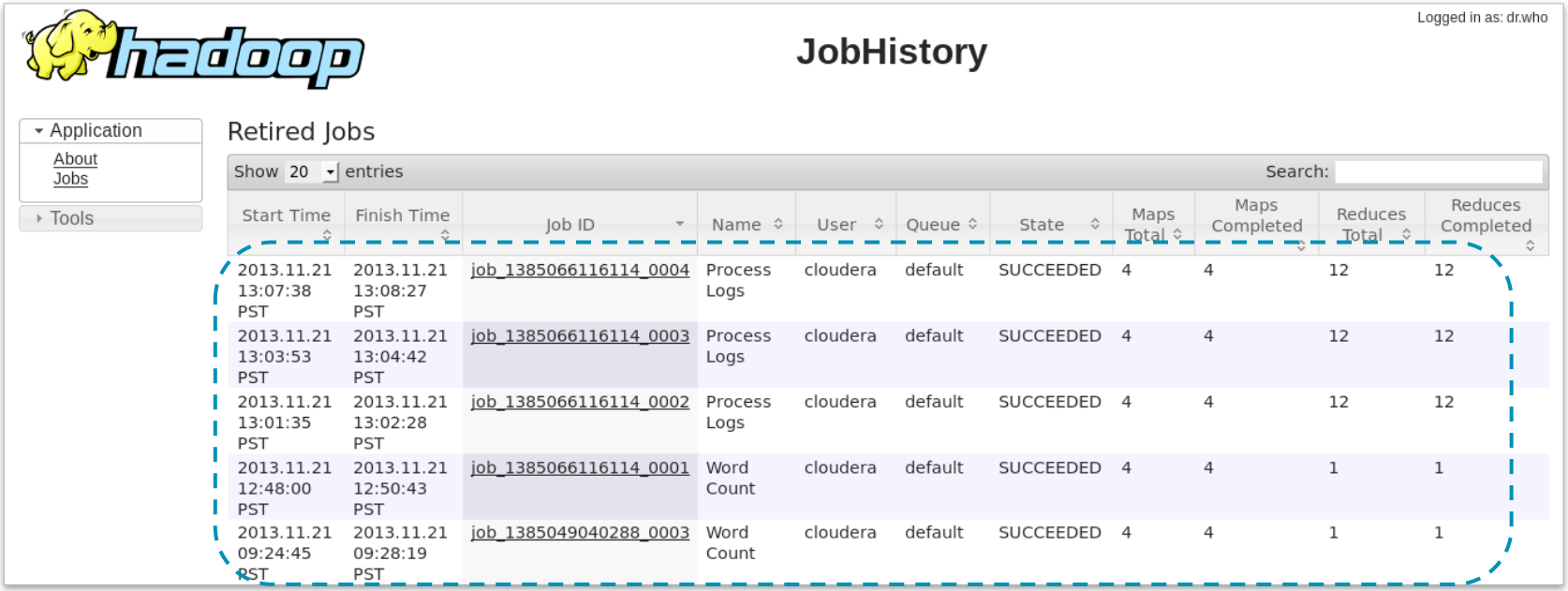
Resource Manager UI: Application Detail



Job History Server

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



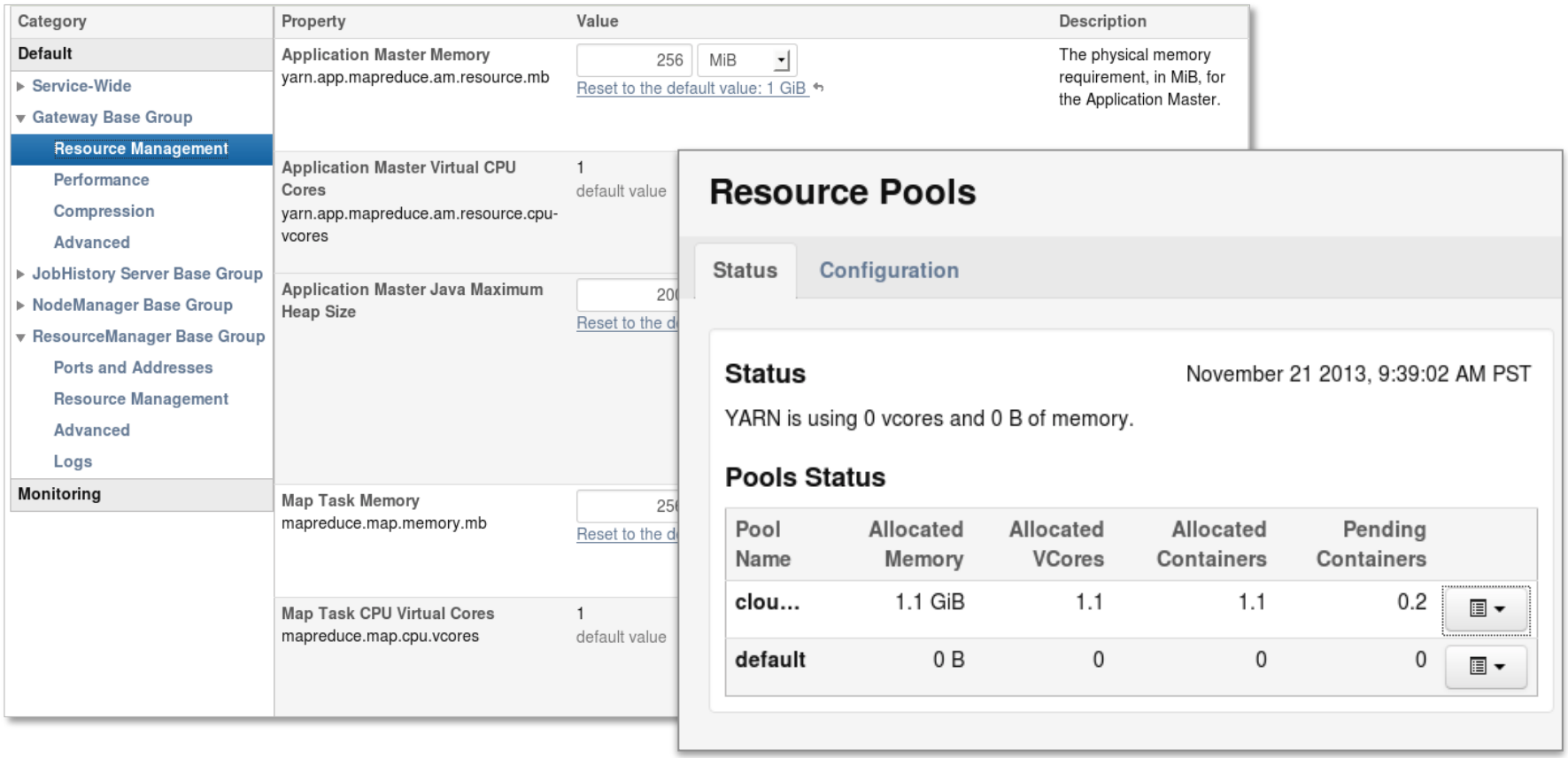


YARN Command Line

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

Cloudera Manager

- **Cloudera Manager providers a greater ability to monitor and configure a cluster from a single location**
  - Covered in *Cloudera Administrator Training for Apache Hadoop*



Chapter Topics

Hadoop Architecture and HDFS | **Introduction to Hadoop**

- Distributed Processing on a Cluster
- Storage: HDFS Architecture
- Storage: Using HDFS
- Hands-on Exercises: Access HDFS with Command Line and Hue
- Resource Management: YARN Architecture
- Resource Management: Working With YARN
- **Conclusion**
- Hands-On Exercises: Run a YARN Job

Essential Points

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

Bibliography

The following offer more information on topics discussed in this chapter

- ***Hadoop Application Architectures: Designing Real-World Big Data Applications*** **(published by O'Reilly)**
  - `http://tiny.cloudera.com/archbook`
- **HDFS User Guide**
  - `http://tiny.cloudera.com/hdfsuser`
- **YARN documentation**
  - `http://tiny.cloudera.com/yarndocs`
- **Cloudera Engineering Blog YARN articles**
  - `http://tiny.cloudera.com/yarnblog`

Chapter Topics

Hadoop Architecture and HDFS | **Introduction to Hadoop**

- Distributed Processing on a Cluster
- Storage: HDFS Architecture
- Storage: Using HDFS
- Hands-on Exercises: Access HDFS with Command Line and Hue
- Resource Management: YARN Architecture
- Resource Management: Working With YARN
- Conclusion
- **Hands-On Exercises: Run a YARN Job**

Hands-On Exercise: Run a YARN Job

- **In this exercise, you will**
  - Use the YARN Web UI to view your YARN cluster "at rest"
  - Submit an application to run on the cluster
  - Monitor the job using both the YARN UI and Hue
- **Please refer to the Hands-On Exercise Manual**

cloudera®

Importing Relational Data with Sqoop

Chapter 4

Course Chapters

▪ Introduction	Course Introduction
▪ Introduction to Hadoop and the Hadoop Ecosystem ▪ Hadoop Architecture and HDFS	Introduction to Hadoop
▪ **Importing Relational Data with Apache Sqoop** ▪ Introduction to Impala and Hive ▪ Modeling and Managing Data with Impala and Hive ▪ Data Formats ▪ Data File Partitioning	**Importing and Modeling Structured Data**
▪ Capturing Data with Apache Flume	Ingesting Streaming Data
▪ Spark Basics ▪ Working with RDDs in Spark ▪ Aggregating Data with Pair RDDs ▪ Writing and Deploying Spark Applications ▪ Parallel Processing in Spark ▪ Spark RDD Persistence ▪ Common Patterns in Spark Data Processing ▪ Spark SQL and DataFrames	Distributed Data Processing with Spark
▪ Conclusion	Course Conclusion

Importing Relational Data with Apache Sqoop

In this chapter you will learn

- **How to import tables from an RDBMS into your Hadoop cluster**
- **How to change the delimiter and file format of imported tables**
- **How to control which columns and rows are imported**
- **What techniques you can use to improve Sqoop's performance**
- **How the next-generation version of Sqoop compares to the original**

Chapter Topics

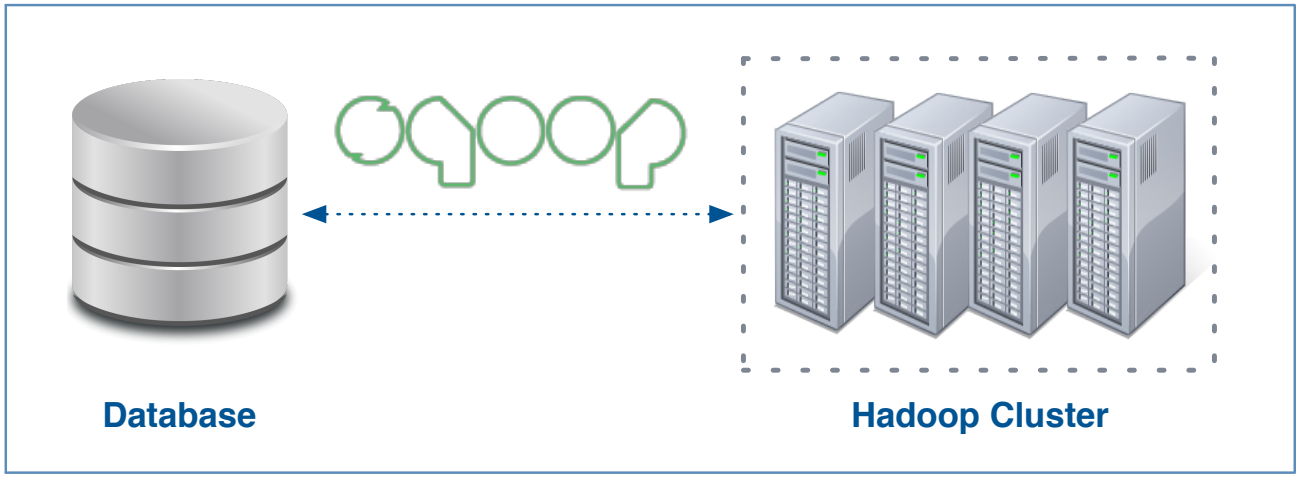
Importing Relational Data with Apache Sqoop

Importing and Modeling Structured Data

- **Sqoop Overview**
- Basic Imports and Exports
- Limiting Results
- Improving Sqoop's Performance
- Sqoop 2
- Conclusion
- Hands-On Exercise: Import Data from MySQL Using Sqoop

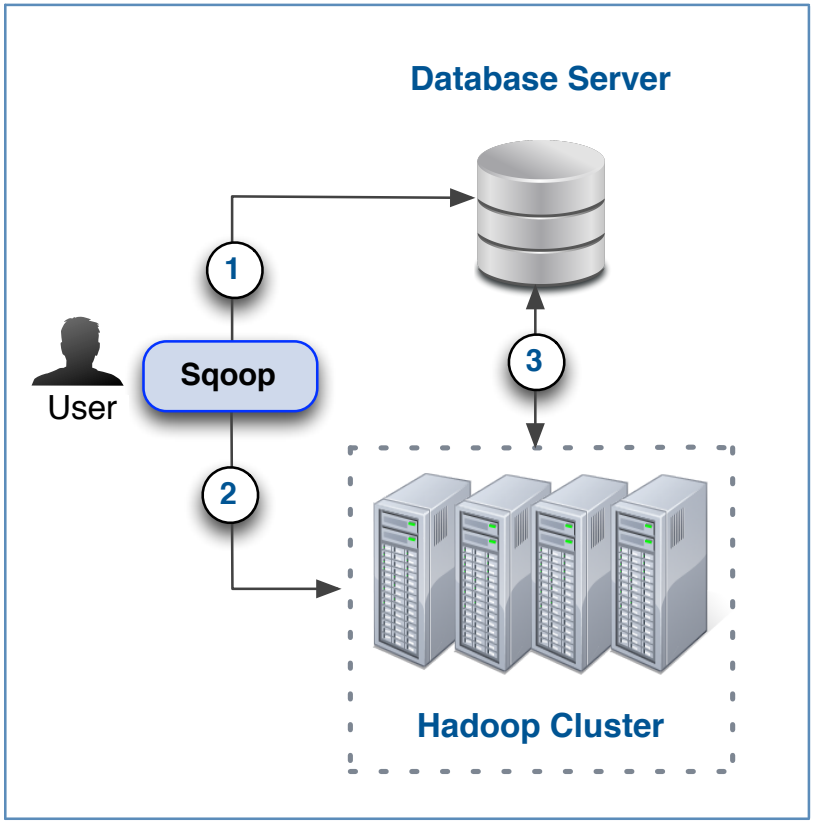
What is Apache Sqoop?

- **Open source Apache project originally developed by Cloudera**
  - The name is a contraction of "SQL-to-Hadoop"
- **Sqoop exchanges data between a database and HDFS**
  - Can import all tables, a single table, or a partial table into HDFS
  - Data can be imported a variety of formats
  - Sqoop can also export data from HDFS to a database



How Does Sqoop Work?

- **Sqoop is a client-side application that imports data using Hadoop MapReduce**
- **A basic import involves three steps orchestrated by Sqoop**
  1. Examine table details
  2. Create and submit job to cluster
  3. Fetch records from table and write this data to HDFS



Basic Syntax

- **Sqoop is a command-line utility with several subcommands, called *tools***
  - There are tools for import, export, listing database contents, and more
  - Run **`sqoop help`** to see a list of all tools
  - Run **`sqoop help tool-name`** for help on using a specific tool
- **Basic syntax of a Sqoop invocation**

```
$ sqoop tool-name [tool-options]
```

- **This command will list all tables in the `loudacre` database in MySQL**

```
$ sqoop list-tables \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser \
    --password pw
```

Chapter Topics

Importing Relational Data with Apache Sqoop | **Importing and Modeling Structured Data**

- Sqoop Overview
- **Basic Imports and Exports**
- Limiting Results
- Improving Sqoop's Performance
- Sqoop 2
- Conclusion
- Hands-On Exercise: Import Data from MySQL Using Sqoop

Overview of the Import Process

- **Imports are performed using Hadoop MapReduce jobs**
- **Sqoop begins by examining the table to be imported**
  - Determines the primary key, if possible
  - Runs a *boundary query* to see how many records will be imported
  - Divides result of boundary query by the number of tasks (mappers)
    - Uses this to configure tasks so that they will have equal loads
- **Sqoop also generates a Java source file for each table being imported**
  - It compiles and uses this during the import process
  - The file remains after import, but can be safely deleted

Importing an Entire Database with Sqoop

- **The `import-all-tables` tool imports an entire database**
  - Stored as comma-delimited files
  - Default base location is your HDFS home directory
  - Data will be in subdirectories corresponding to name of each table

```
$ sqoop import-all-tables \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw
```

- **Use the `--warehouse-dir` option to specify a different base directory**

```
$ sqoop import-all-tables \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --warehouse-dir /loudacre
```

Importing a Single Table with Sqoop

- **The `import` tool imports a single table**
- **This example imports the `accounts` table**
  - It stores the data in HDFS as comma-delimited fields

```
$ sqoop import --table accounts \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw
```

- **This variation writes tab-delimited fields instead**

```
$ sqoop import --table accounts \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --fields-terminated-by "\t"
```

Incremental Imports (1)

- **What if records have changed since last import?**
  - Could re-import all records, but this is inefficient
- **Sqoop's `incremental lastmodified` mode imports new and modified records**
  - Based on a timestamp in a specified column
  - You must ensure timestamps are updated when records are added or changed in the database

```
$ sqoop import --table invoices \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --incremental lastmodified \
    --check-column mod_dt \
    --last-value '2015-09-30 16:00:00'
```

Incremental Imports (2)

- **Or use Sqoop's `incremental append` mode to import only *new* records**
  - Based on value of last record in specified column

```
$ sqoop import --table invoices \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --incremental append \
    --check-column id \
    --last-value 9478306
```

Exporting Data from Hadoop to RDBMS with Sqoop

- **Sqoop's `import` tool pulls records from an RDBMS into HDFS**
- **It is sometimes necessary to *push* data in HDFS back to an RDBMS**
  - Good solution when you must do batch processing on large data sets
  - Export results to a relational database for access by other systems
- **Sqoop supports this via the `export` tool**
  - The RDBMS table must already exist prior to export

```
$ sqoop export \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --export-dir /loudacre/recommender_output \
    --update-mode allowinsert \
    --table product_recommendations
```

Chapter Topics

Importing Relational Data with Apache Sqoop | **Importing and Modeling Structured Data**

- Sqoop Overview
- Basic Imports and Exports
- **Limiting Results**
- Improving Sqoop's Performance
- Sqoop 2
- Conclusion
- Hands-On Exercise: Import Data from MySQL Using Sqoop

Importing Partial Tables with Sqoop

- **Import only specified columns from `accounts` table**

```
$ sqoop import --table accounts \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --columns "id,first_name,last_name,state"
```

- **Import only matching rows from `accounts` table**

```
$ sqoop import --table accounts \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --where "state='CA'"
```

Using a Free-Form Query

- **You can also import the results of a query, rather than a single table**
- **Supply a complete SQL query using the `--query` option**
  - You must add the *literal* `WHERE $CONDITIONS` token
  - Use `--split-by` to identify field used to divide work among mappers
  - The `--target-dir` option is required for free-form queries

```
$ sqoop import \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --target-dir /data/loudacre/payable \
    --split-by accounts.id \
    --query 'SELECT accounts.id, first_name,
last_name, bill_amount FROM accounts JOIN invoices ON
(accounts.id = invoices.cust_id) WHERE $CONDITIONS'
```

Using a Free-Form Query with WHERE Criteria

- **The `--where` option is ignored in a free-form query**
  - You must specify your criteria using AND following the WHERE clause

```
$ sqoop import \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    --target-dir /data/loudacre/payable \
    --split-by accounts.id \
    --query 'SELECT accounts.id, first_name,
last_name, bill_amount FROM accounts JOIN invoices ON
(accounts.id = invoices.cust_id) WHERE $CONDITIONS AND
bill_amount >= 40'
```

Chapter Topics

Importing Relational Data with Apache Sqoop | **Importing and Modeling Structured Data**

- Sqoop Overview
- Basic Imports and Exports
- Limiting Results
- **Improving Sqoop's Performance**
- Sqoop 2
- Conclusion
- Hands-On Exercise: Import Data from MySQL Using Sqoop

Options for Database Connectivity

- **Generic (JDBC)**
  - Compatible with nearly any database
  - Overhead imposed by JDBC can limit performance
- **Direct Mode**
  - Can improve performance through use of database-specific utilities
  - Currently supports MySQL and Postgres (use `--direct` option)
  - Not all Sqoop features are available in direct mode
- **Cloudera and partners offer high-performance Sqoop connectors**
  - These use native database protocols rather than JDBC
  - Connectors available for Netezza, Teradata, and Oracle
    - Download these from Cloudera's Web site
    - Not open source due to licensing issues, but free to use

Controlling Parallelism

- **By default, Sqoop typically imports data using four parallel tasks (called mappers)**
  - Increasing the number of tasks might improve import speed
  - Caution: Each task adds load to your database server
- **You can *influence* the number of tasks using the `-m` option**
  - Sqoop views this only as a hint and might not honor it

```
$ sqoop import --table accounts \
    --connect jdbc:mysql://dbhost/loudacre \
    --username dbuser --password pw \
    -m 8
```

- **Sqoop assumes all tables have an evenly-distributed numeric primary key**
  - Sqoop uses this column to divide work among the tasks
  - You can use a different column with the `--split-by` option

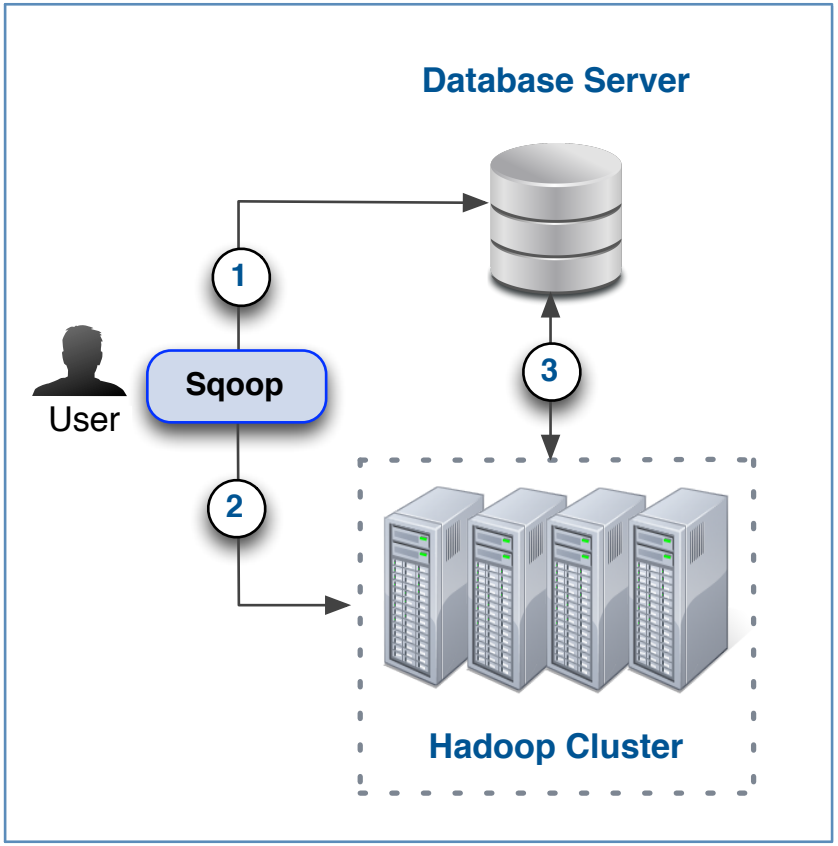
Chapter Topics

Importing Relational Data with Apache Sqoop | **Importing and Modeling Structured Data**

- Sqoop Overview
- Basic Imports and Exports
- Limiting Results
- Improving Sqoop's Performance
- **Sqoop 2**
- Conclusion
- Hands-On Exercise: Import Data from MySQL Using Sqoop

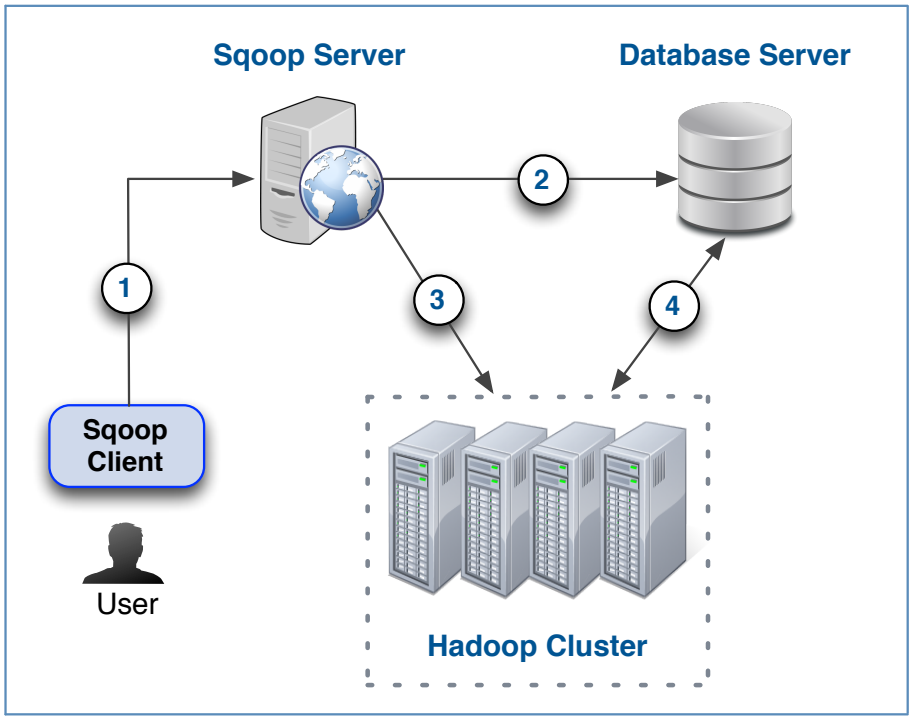
Limitations of Sqoop

- **Sqoop is stable and has been used successfully in production for years**
- **However, its client-side architecture does impose some limitations**
  - Requires connectivity to RDBMS from the client (client must have JDBC drivers installed)
  - Requires connectivity to cluster from the client
  - Requires user to specify RDBMS username and password
  - Difficult to integrate a CLI within external applications
- **Also tightly coupled to JDBC semantics**
  - A problem for NoSQL databases



Sqoop 2 Architecture

- **Sqoop 2 is the next-generation version of Sqoop**
  - Client-server design addresses limitations described earlier
  - API changes also simplify development of other Sqoop connectors
- **Client requires connectivity only to the Sqoop server**
  - DB connections are configured on the server by a system administrator
  - End users no longer need to possess database credentials
  - Centralized audit trail
  - Better resource management
  - Sqoop server is accessible via CLI, REST API, and Web UI



Sqoop 2 Status

- **Sqoop 2 is being actively developed**
  - It began shipping (alongside Sqoop) starting in CDH 4.2
- **Sqoop 2 is not yet at feature parity with Sqoop**
  - Implemented features are regarded as stable
  - Consider using Sqoop 2 unless you require a feature it lacks
- **We use Sqoop, rather than Sqoop 2, in this class**
  - Primarily due to memory constraints in the VM

Chapter Topics

Importing Relational Data with Apache Sqoop

Importing and Modeling Structured Data

- Sqoop Overview
- Basic Imports and Exports
- Limiting Results
- Improving Sqoop's Performance
- Sqoop 2
- **Conclusion**
- Hands-On Exercise: Import Data from MySQL Using Sqoop

Essential Points

- **Sqoop exchanges data between a database and the Hadoop cluster**
  - Provides subcommands (*tools*) for importing, exporting, and more
- **Tables are imported using MapReduce jobs**
  - These are written as comma-delimited text by default
  - You can specify alternate delimiters or file formats
  - Uncompressed by default, but you can specify a codec to use
- **Sqoop provides many options to control imports**
  - You can select only certain columns or limit rows
  - Supports using joins in free-form queries
- **Sqoop 2 is the next-generation version of Sqoop**
  - Client-server design improves administration and resource management

Bibliography

The following offer more information on topics discussed in this chapter

- **Sqoop User Guide**
  - `http://tiny.cloudera.com/sqoopuserguide`
- ***Apache Sqoop Cookbook* (published by O'Reilly)**
  - `http://tiny.cloudera.com/sqoopcookbook`
- **A New Generation of Data Transfer Tools for Hadoop: Sqoop 2**
  - `http://tiny.cloudera.com/adcc05c`

Chapter Topics

Importing Relational Data with Apache Sqoop | **Importing and Modeling Structured Data**

- Sqoop Overview
- Basic Imports and Exports
- Limiting Results
- Improving Sqoop's Performance
- Sqoop 2
- Conclusion
- **Hands-On Exercise: Import Data from MySQL Using Sqoop**

Hands-on Exercise: Import Data from MySQL Using Sqoop

- **In this exercise, you will**
  - Use Sqoop to import web page and customer account data from an RDBMS to HDFS
  - Perform incremental imports of new and updated account data
- **Please refer to the Hands-On Exercise Manual for instructions**