cloudera®

Apache Hadoop Fundamentals

Chapter 2

Course Chapters

- Introduction
- **Apache Hadoop Fundamentals**
- Introduction to Apache Pig
- Basic Data Analysis with Apache Pig
- Processing Complex Data with Apache Pig
- Multi-Dataset Operations with Apache Pig
- Apache Pig Troubleshooting and Optimization
- Introduction to Apache Hive and Impala
- Querying with Apache Hive and Impala
- Apache Hive and Impala Data Management
- Data Storage and Performance
- Relational Data Analysis with Apache Hive and Impala
- Complex Data with Apache Hive and Impala
- Analyzing Text with Apache Hive and Impala
- Apache Hive Optimization
- Apache Impala Optimization
- Extending Apache Hive and Impala
- Choosing the Best Tool for the Job
- Conclusion

Apache Hadoop Fundamentals

In this chapter, you will learn

- **Which factors led to the era of Big Data**
- **What Hadoop is and what significant features it offers**
- **How Hadoop offers reliable storage for massive amounts of data with HDFS**
- **How Hadoop supports large-scale data processing through MapReduce and Spark**
- **How Hadoop ecosystem tools can boost an analyst's productivity**
- **Several ways to integrate Hadoop into the modern data center**

Chapter Topics

Apache Hadoop Fundamentals

- **The Motivation for Hadoop**
- Hadoop Overview
- Data Storage: HDFS
- Distributed Data Processing: YARN, MapReduce, and Spark
- Data Processing and Analysis: Pig, Hive, and Impala
- Database Integration: Sqoop
- Other Hadoop Data Tools
- Exercise Scenario Explanation
- Essential Points
- Hands-On Exercise: Data Ingest with Hadoop Tools

Volume

- **Every day…**
  - Over 2.25 billion shares are traded on the New York Stock Exchange
  - Facebook stores 4.5 billion “Likes”
  - Google processes about 24 petabytes of data
- **Every minute…**
  - Facebook users share nearly 2.5 million pieces of content
  - Email users send 204,000,000 messages
- **Every second…**
  - Financial institutions process more than 10,000 credit card transactions
  - Amazon Web Services fields more than 650,000 requests
- **And it’s not stopping…**
  - They can’t collect it all yet, but the Large Hadron Collider produces 572 terabytes of data per second

Velocity

- **We are generating data faster than ever**
  - Automating many processes
  - Interconnecting numerous systems
  - Interacting with each other online

Variety

- **We are producing a wide variety of data**
  - Social network connections
  - Server and application log files
  - Electronic medical records
  - Images, audio, and video
  - RFID and wireless sensor network events
  - Product ratings on shopping and review websites
  - And much more…
- **Not all of this maps cleanly to the relational model**

Value

- **This data has many valuable applications**
  - Product recommendations
  - Predicting demand
  - Marketing analysis
  - Fraud detection
  - And many, many more…
- **We must process it to extract that value**
  - And processing *all the data* can yield more accurate results

We Need a System that Scales

- **We're generating too much data to process with traditional tools**
- **Two key problems to address**
  - How can we reliably store large amounts of data at a reasonable cost?
  - How can we analyze all the data we have stored?

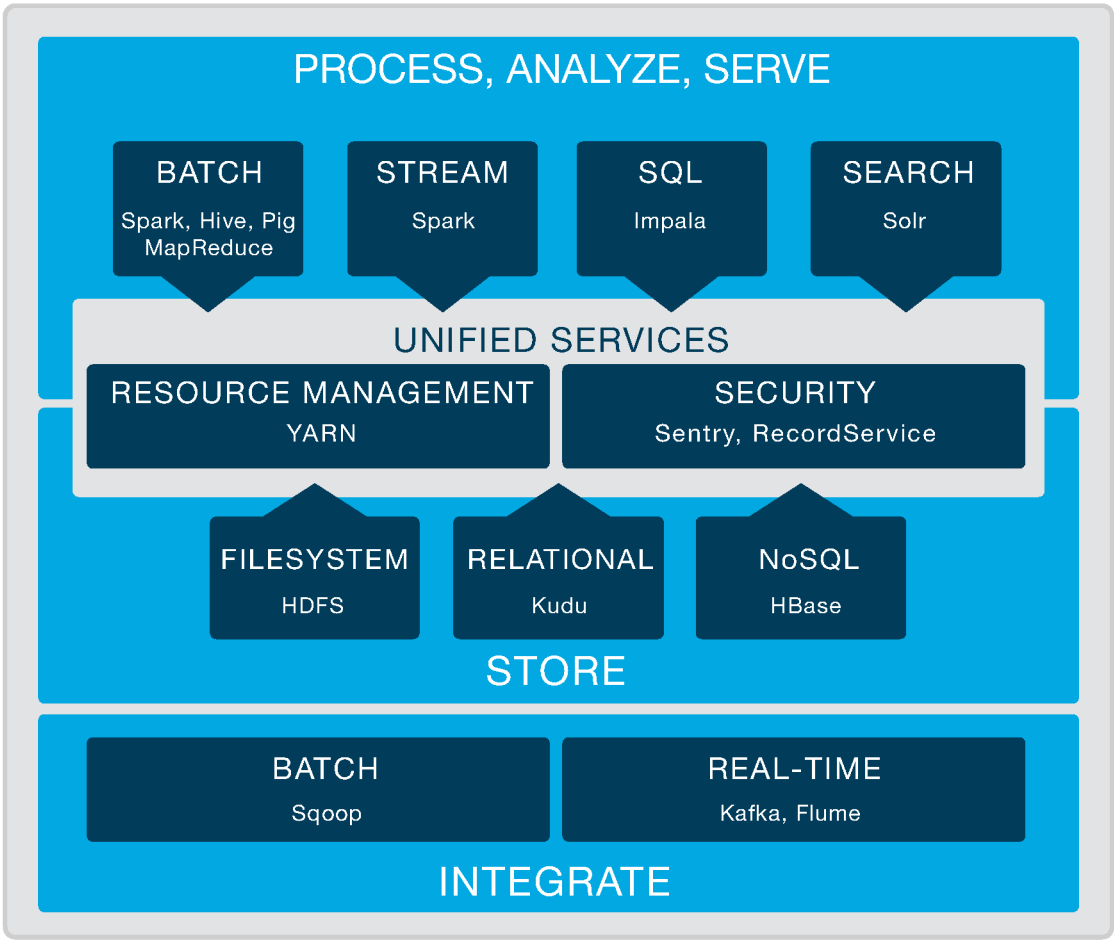
Chapter Topics

Apache Hadoop Fundamentals

- The Motivation for Hadoop
- **Hadoop Overview**
- Data Storage: HDFS
- Distributed Data Processing: YARN, MapReduce, and Spark
- Data Processing and Analysis: Pig, Hive, and Impala
- Database Integration: Sqoop
- Other Hadoop Data Tools
- Exercise Scenario Explanation
- Essential Points
- Hands-On Exercise: Data Ingest with Hadoop Tools

What is Apache Hadoop?

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



Scalability

- **Hadoop is a distributed system**
  - A collection of servers running Hadoop software is called a *cluster*
- **Individual servers within a cluster are called *nodes***
  - Typically industry-standard servers running Linux
  - Each node both stores and processes data
- **Add more nodes to the cluster to increase scalability**
  - A cluster may contain up to several thousand nodes
  - You can scale out incrementally as required

Fault Tolerance

- **Paradox: Adding nodes increases the chance that any one of them will fail**
  - Solution: Build redundancy into the system and handle it automatically
- **Files loaded into HDFS are replicated across nodes in the cluster**
  - If a node fails, its data is replicated using one of the other copies
- **Data processing jobs are broken into individual tasks**
  - Each task takes a small amount of data as input
  - Thousands of tasks (or more) often run in parallel
  - If a node fails during processing, its tasks are rescheduled elsewhere
- **Routine failures are handled automatically without any loss of data**

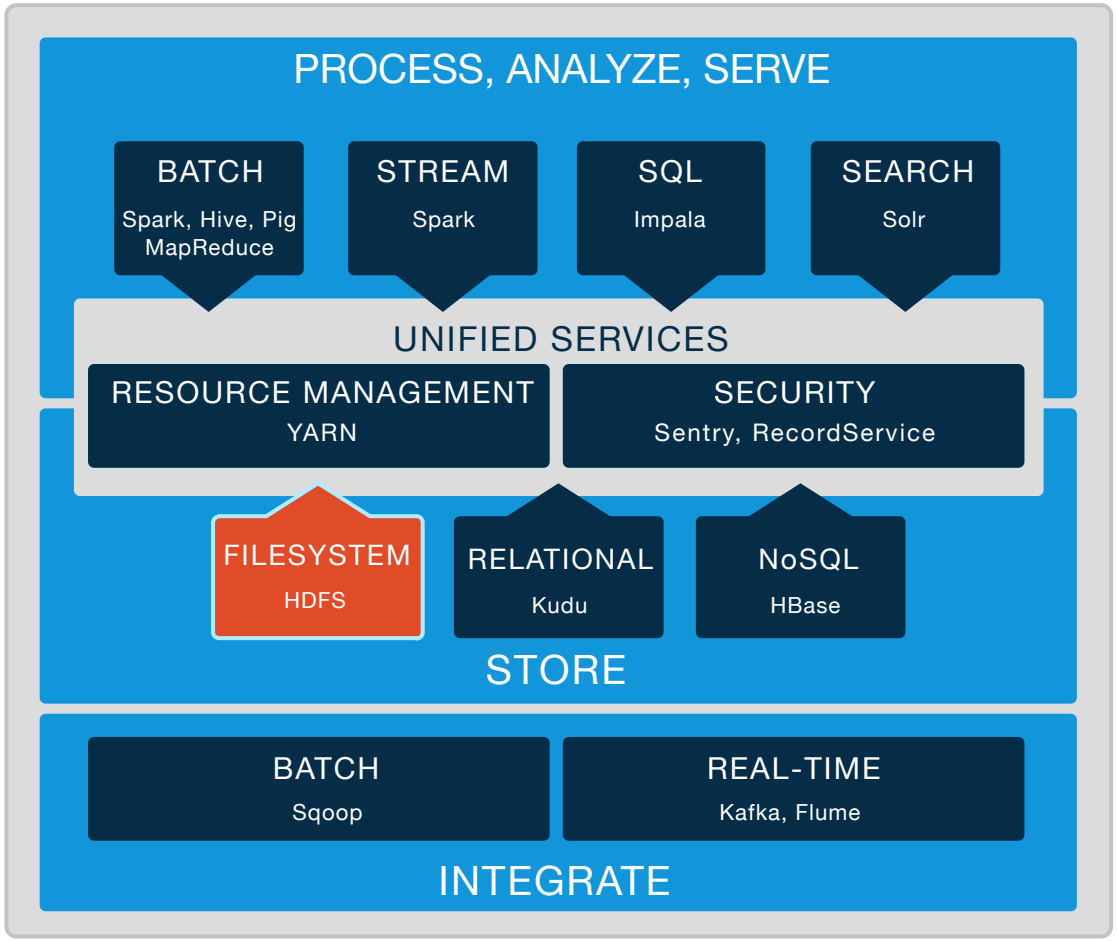
Chapter Topics

Apache Hadoop Fundamentals

- The Motivation for Hadoop
- Hadoop Overview
- **Data Storage: HDFS**
- Distributed Data Processing: YARN, MapReduce, and Spark
- Data Processing and Analysis: Pig, Hive, and Impala
- Database Integration: Sqoop
- Other Hadoop Data Tools
- Exercise Scenario Explanation
- Essential Points
- Hands-On Exercise: Data Ingest with Hadoop Tools

HDFS: Hadoop Distributed File System

- **HDFS provides the storage layer for Hadoop data processing**
- **Provides inexpensive and reliable storage for massive amounts of data**
- **Other Hadoop components work with data in HDFS**
  - Such as MapReduce, Hive, Impala, Pig, and Spark



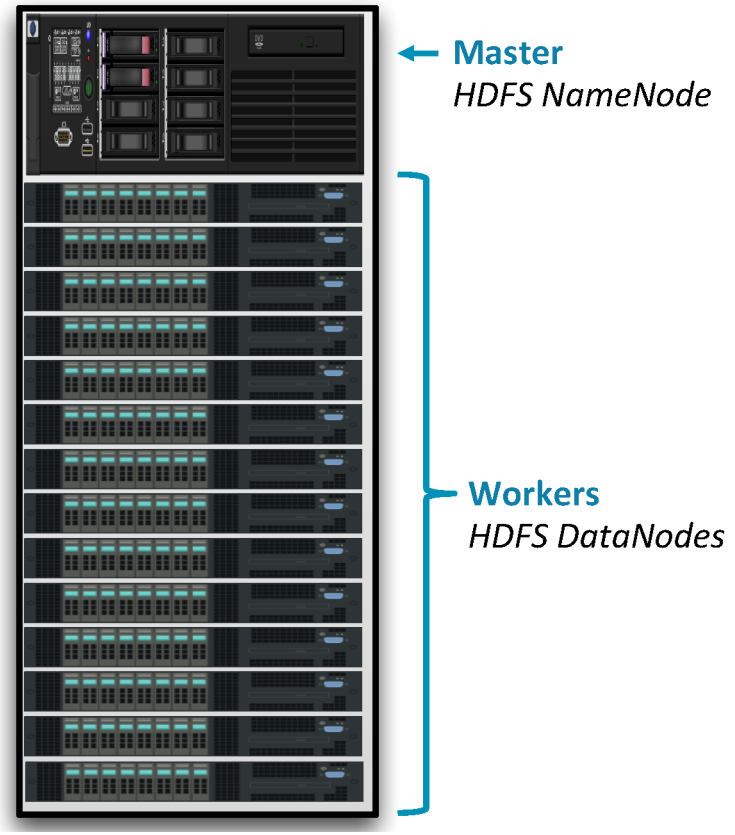
HDFS Characteristics

- **Optimized for sequential access to a relatively small number of large files**
  - Each file is likely to be 100MB or larger
  - Multi-gigabyte files are typical
- **In some ways, HDFS is similar to a UNIX filesystem**
  - Hierarchical: Everything descends from a root directory
  - Uses UNIX-style paths (such as **`/sales/rpt/asia.txt`**)
  - UNIX-style file ownership and permissions
- **There are also some major deviations from UNIX**
  - No concept of a current directory
  - Cannot modify files once written
  - Must use Hadoop-specific utilities or custom code to access HDFS

HDFS Architecture

- **Hadoop has a master/worker architecture**
- **HDFS master daemon: NameNode**
  - Manages namespace and metadata
  - Monitors worker nodes
- **HDFS worker daemon: DataNode**
  - Reads and writes the actual data

A Small Hadoop Cluster



Accessing HDFS Using the Command Line

- **HDFS is not a general-purpose filesystem**
  - Not built into the OS, so only specialized tools can access it
  - End users typically access HDFS using the **`hdfs dfs`** command
- **Example: Display the contents of the `/user/fred/sales.txt` file**

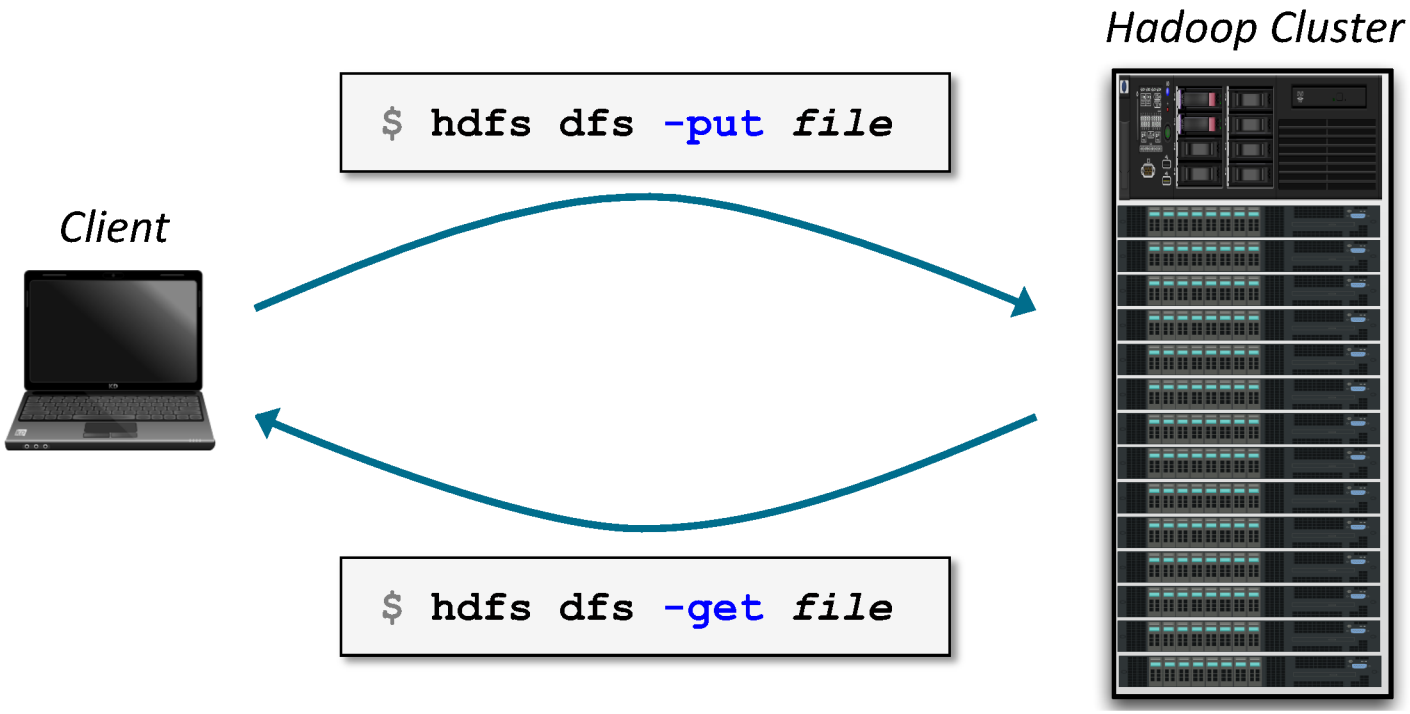
```
$ hdfs dfs -cat /user/fred/sales.txt
```

- **Example: Create a directory (below the root) called `reports`**

```
$ hdfs dfs -mkdir /reports
```

Copying Local Data to and from HDFS

- **Remember that HDFS is distinct from your local filesystem**
  - Use **`hdfs dfs -put`** to copy local files *to* HDFS
  - Use **`hdfs dfs -get`** to fetch a local copy of a file *from* HDFS



More `hdfs dfs` Command Examples

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

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

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

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

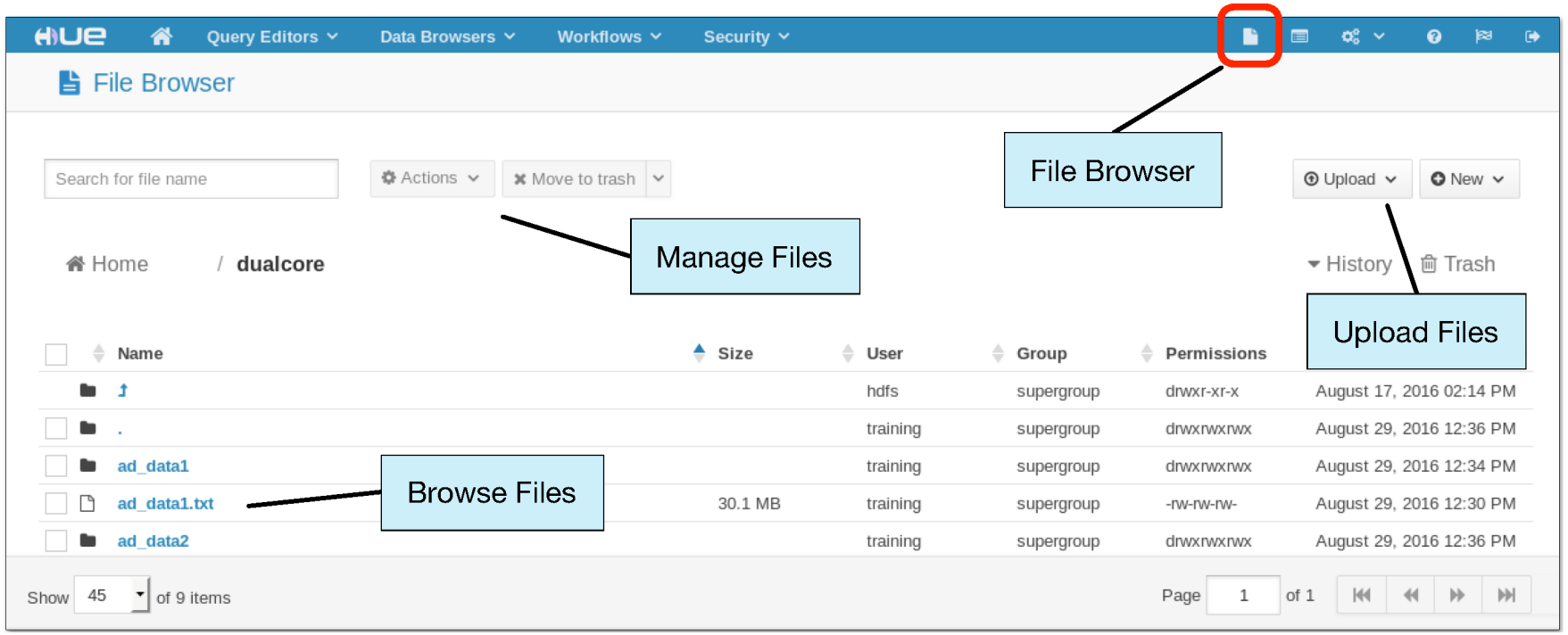
```
$ hdfs dfs -ls /
```

- **Delete the file `/reports/sales.txt`**

```
$ hdfs dfs -rm /reports/sales.txt
```

Using the Hue HDFS File Manager

- **Hue is a web interface for Hadoop**
  - **H**adoop **U**ser **E**xperience
- **Hue includes an application for browsing and managing files in HDFS**
  - To use Hue, browse to **`http://hue_server:8888/`**



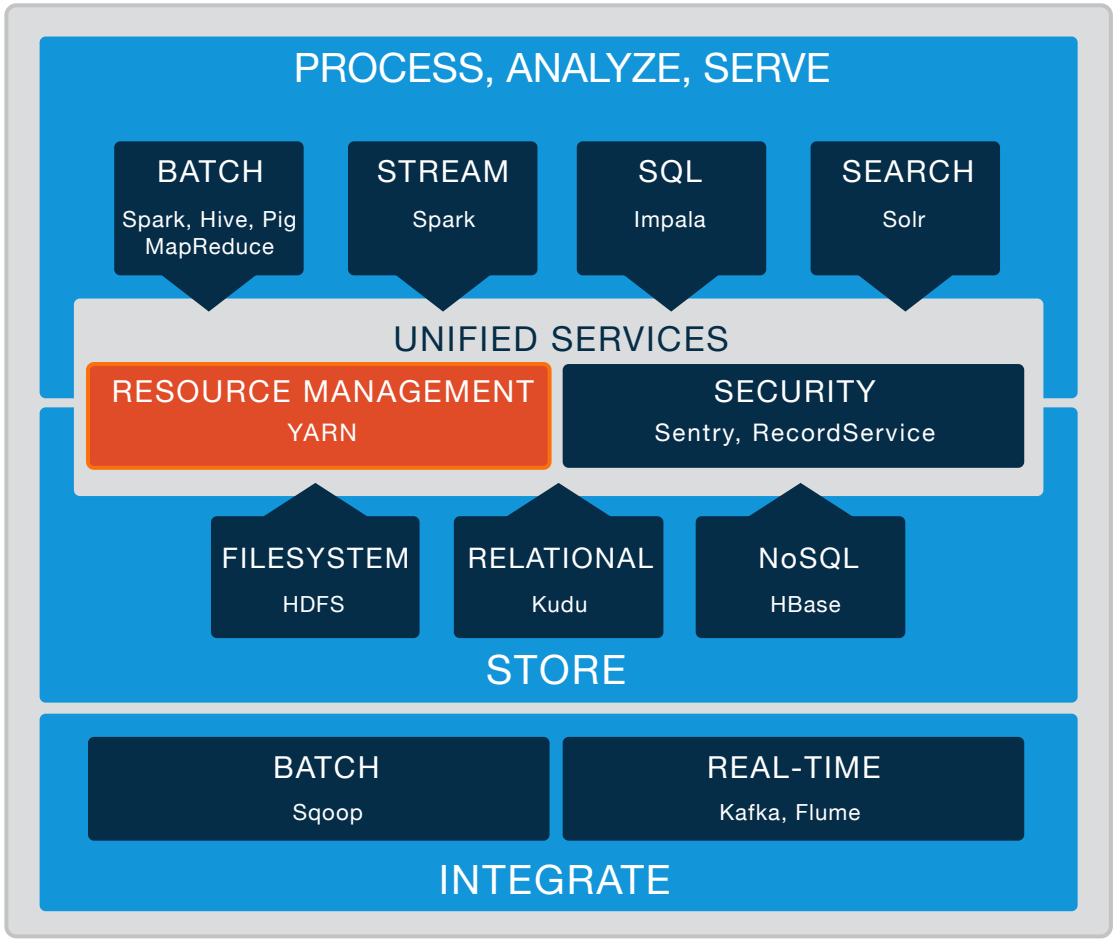
Chapter Topics

Apache Hadoop Fundamentals

- The Motivation for Hadoop
- Hadoop Overview
- Data Storage: HDFS
- **Distributed Data Processing: YARN, MapReduce, and Spark**
- Data Processing and Analysis: Pig, Hive, and Impala
- Database Integration: Sqoop
- Other Hadoop Data Tools
- Exercise Scenario Explanation
- Essential Points
- Hands-On Exercise: Data Ingest with Hadoop Tools

Workload Management: YARN

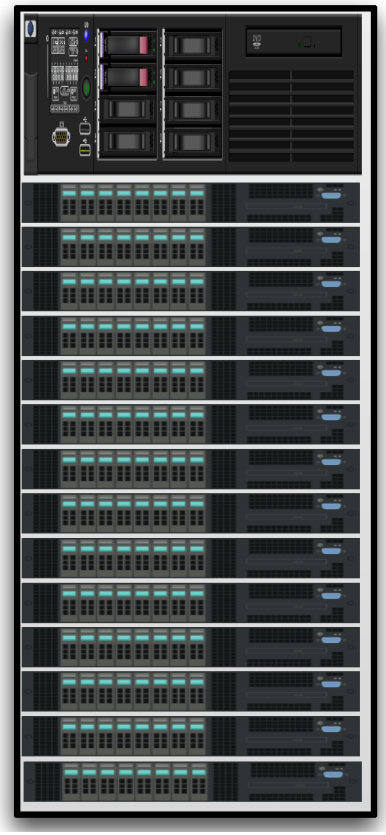
- **Many Hadoop tools work with data in a Hadoop cluster**
- **Distributing and monitoring work across the cluster requires workload management**
- **YARN (Yet Another Resource Negotiator) allocates cluster resources to workloads and monitors resource usage**



Hadoop Cluster Architecture

- **Master/Worker Architecture**
  - Like HDFS, YARN runs on a master node and worker nodes
- **YARN master daemon: ResourceManager**
  - Accepts jobs, tracks resources, monitors and distributes work
- **YARN worker daemon: NodeManager**
  - Launches tasks to run on worker node, reports status back to master
- **HDFS and YARN are colocated**
  - Worker nodes run both HDFS and YARN on the same machines

A Small Hadoop Cluster

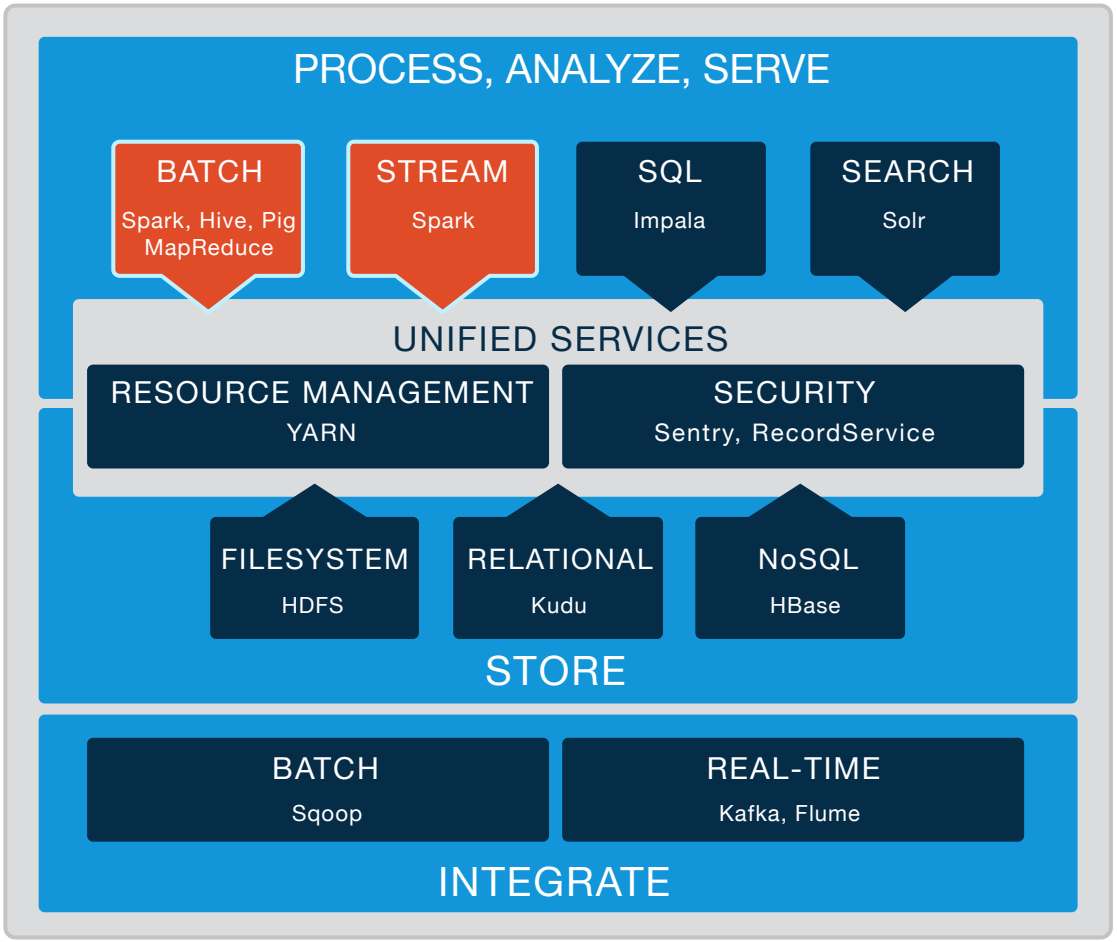


Master
YARN ResourceManager
HDFS NameNode

Workers
YARN NodeManagers
HDFS DataNodes

General Data Processing

- **Hadoop includes two general data processing engines**
  - MapReduce
  - Spark
- **Higher-level tools like Hive and Pig use these engines to process data**
- **Developers can use the MapReduce or Spark APIs to write custom data processing code**



Hadoop MapReduce

- **MapReduce is the original processing engine for Hadoop**
  - Still a commonly used general data processing engine
- **Based on the the "map-reduce" programming model**
  - A style of processing data popularized by Google
- **Provides a set of programming libraries**
  - Primarily supports Java
  - Streaming MapReduce provides (limited) support for scripting languages such as Python
- **Benefits of MapReduce**
  - Simplicity
  - Flexibility
  - Scalability

Apache Spark

- **Spark is the next-generation general data processing engine**
- **Builds on the same "map-reduce" programming model as MapReduce**
- **Originally developed at AMPLab at the University of California, Berkeley**
- **Supports Scala, Java, Python, and R**
- **Has the same benefits as MapReduce, plus…**
  - Improved performance using in-memory processing
  - Higher-level programming model to speed development

Chapter Topics

Apache Hadoop Fundamentals

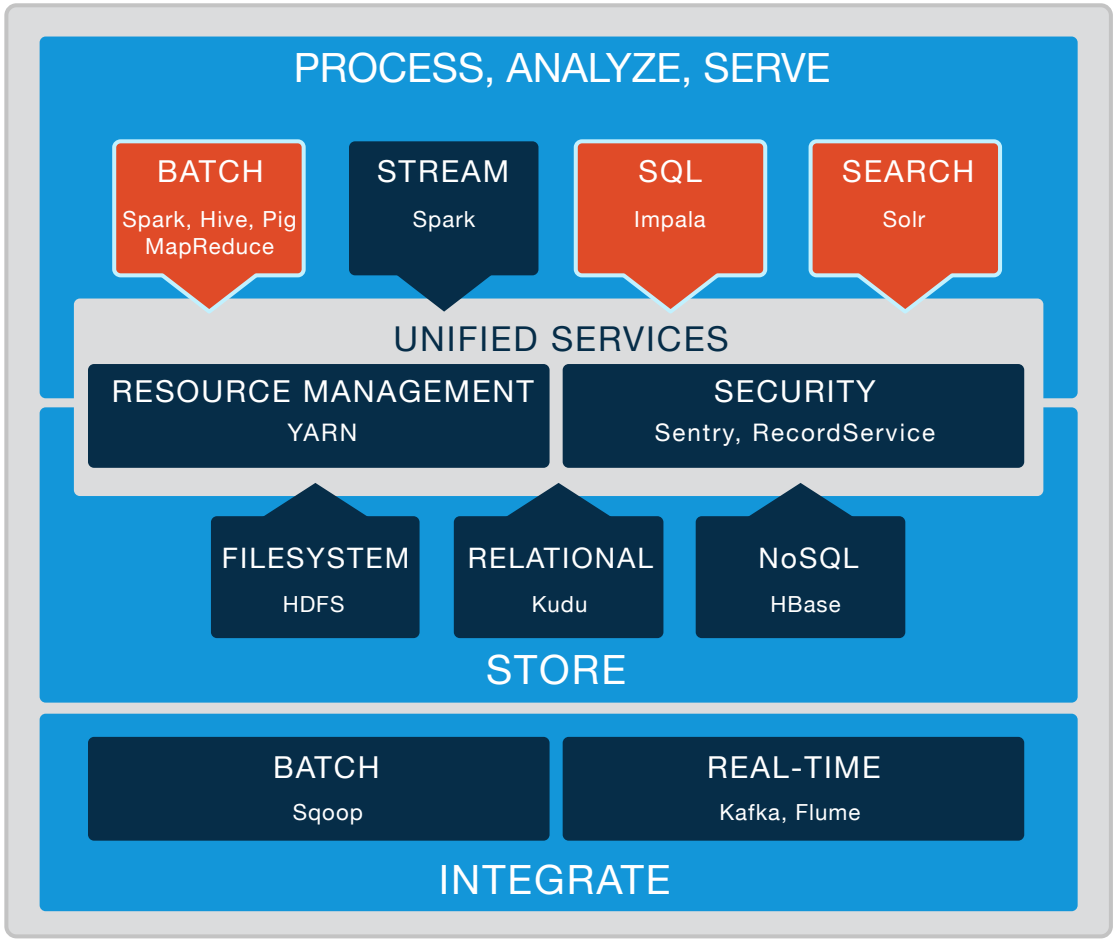
- The Motivation for Hadoop
- Hadoop Overview
- Data Storage: HDFS
- Distributed Data Processing: YARN, MapReduce, and Spark
- **Data Analysis and Processing: Pig, Hive, and Impala**
- Database Integration: Sqoop
- Other Hadoop Data Tools
- Exercise Scenario Explanation
- Essential Points
- Hands-On Exercise: Data Ingest with Hadoop Tools

Data Processing and Analysis with Hadoop (1)

- **Hadoop MapReduce and Spark are powerful data processing engines but…**
  - Hard to master
  - Require programming skills
  - Slow to develop, hard to maintain
- **Hadoop includes several other tools for data processing and analysis**
  - Tools for data analysts, not programmers

Data Processing and Analysis with Hadoop (2)

- **Higher-level abstractions for general data processing**
  - Such as Pig and Hive
- **Specialized processing engines for interactive analysis**
  - Such as Impala and Cloudera Search (based on Apache Solr)



Apache Pig

- **Pig uses 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

```
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;
```

- **The Pig interpreter runs on the client machine**
  - Turns Pig Latin scripts into MapReduce jobs
  - Submits those jobs to the cluster

Apache Hive

- **Hive is another abstraction on top of Hadoop**
  - Like Pig, it also reduces development time
  - Hive uses a SQL-like language called HiveQL


```
SELECT customers.cust_id, SUM(cost) AS total
  FROM customers
    JOIN orders
      ON (customers.cust_id = orders.cust_id)
  GROUP BY customers.cust_id
  ORDER BY total DESC;
```

- **A Hive Server runs on a master node**
  - Turns HiveQL queries into MapReduce or Spark jobs
  - Submits those jobs to the cluster

Apache Impala (Incubating)

- **Impala is a massively parallel SQL engine that runs on a Hadoop cluster**
  - Inspired by Google's Dremel project
  - Can query data stored in HDFS or HBase tables
- **Uses Impala SQL**
  - Very similar to HiveQL
- **High performance**
  - Typically at least 10 times faster than Hive on MapReduce
  - High-level query language (subset of SQL-92)
- **Developed by Cloudera**
  - Donated to the Apache Software Foundation
  - 100% Apache-licensed open source

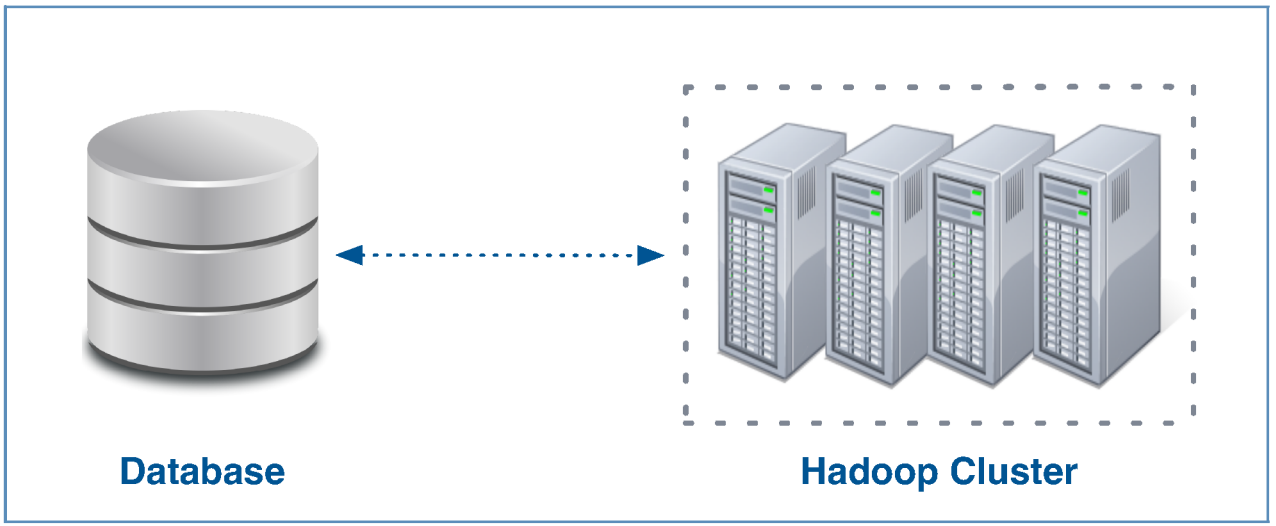
Chapter Topics

Apache Hadoop Fundamentals

- The Motivation for Hadoop
- Hadoop Overview
- Data Storage: HDFS
- Distributed Data Processing: YARN, MapReduce, and Spark
- Data Processing and Analysis: Pig, Hive, and Impala
- **Database Integration: Sqoop**
- Other Hadoop Data Tools
- Exercise Scenario Explanation
- Essential Points
- Hands-On Exercise: Data Ingest with Hadoop Tools

Apache Sqoop

- **Sqoop exchanges data between an RDBMS and Hadoop**
- **It can import all tables, a single table, or a portion of a table into HDFS**
  - Does this very efficiently using a map-only MapReduce job
  - Result is a directory in HDFS containing comma-delimited text files
- **Sqoop can also export data from HDFS back to the database**



Importing Tables with Sqoop

- **This example imports the `customers` table from a MySQL database**
  - Will create directory **`customers`** in the user's home directory in HDFS
  - Directory will contain comma-delimited text files

```
$ sqoop import \
    --connect jdbc:mysql://localhost/company \
    --username jdoe --password bigsecret \
    --table customers
```

- **Adding the `--direct` parameter may offer better performance**
  - Uses a database-specific direct connector instead of generic JDBC
  - This option is not compatible with all databases

Specifying Import Directory with Sqoop

- Use `--target-dir` to specify the import directory

```
$ sqoop import \
    --connect jdbc:mysql://localhost/company \
    --username jdoe --password bigsecret \
    --table customers \
    --target-dir /mydata/customers
```

- Or use `--warehouse-dir` to specify the parent directory

```
$ sqoop import \
    --connect jdbc:mysql://localhost/company \
    --username jdoe --password bigsecret \
    --table customers \
    --warehouse-dir /mydata
```

  - Both commands create **`/mydata/customers`** directory in HDFS

Importing an Entire Database with Sqoop

- **Import all tables from the database (fields will be tab-delimited)**

```
$ sqoop import-all-tables \
    --connect jdbc:mysql://localhost/company \
    --username jdoe --password bigsecret \
    --warehouse-dir /mydata \
    --fields-terminated-by '\t'
```

Importing Partial Tables with Sqoop

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

```
$ sqoop import \
    --connect jdbc:mysql://localhost/company \
    --username jdoe --password bigsecret \
    --table products \
    --warehouse-dir /mydata \
    --columns "prod_id,name,price"
```

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

```
$ sqoop import \
    --connect jdbc:mysql://localhost/company \
    --username jdoe --password bigsecret \
    --table products \
    --warehouse-dir /mydata \
    --where "price >= 1000"
```

Incremental Imports with Sqoop

- **What if new records are added to the database?**
  - Could re-import all records, but this is inefficient
- **Sqoop's incremental append mode imports only *new* records**
  - Based on value of last record in specified column

```
$ sqoop import \
    --connect jdbc:mysql://localhost/company \
    --username jdoe --password bigsecret \
    --table orders \
    --warehouse-dir /mydata \
    --incremental append \
    --check-column order_id \
    --last-value 6713821
```

Handling Modifications with Incremental Imports

- **What if existing records are also modified in the database?**
  - Incremental append mode doesn't handle this
- **Sqoop's `lastmodified` append mode adds *and* updates records**
  - Caveat: You must maintain a timestamp column in your table

```
$ sqoop import \
    --connect jdbc:mysql://localhost/company \
    --username jdoe --password bigsecret \
    --table shipments \
    --warehouse-dir /mydata \
    --incremental lastmodified \
    --check-column last_update_date \
    --last-value "2016-12-21 05:36:59"
```

Exporting Data from Hadoop to RDBMS with Sqoop

- **We have seen several ways to pull records from an RDBMS into Hadoop**
  - It is sometimes also helpful to *push* data in Hadoop back to an RDBMS
- **Sqoop supports this with `export`**
  - The target table must already exist in the RDBMS

```
$ sqoop export \
    --connect jdbc:mysql://localhost/company \
    --username jdoe --password bigsecret \
    --table product_recommendations \
    --export-dir /mydata/recommender_output
```

Chapter Topics

Apache Hadoop Fundamentals

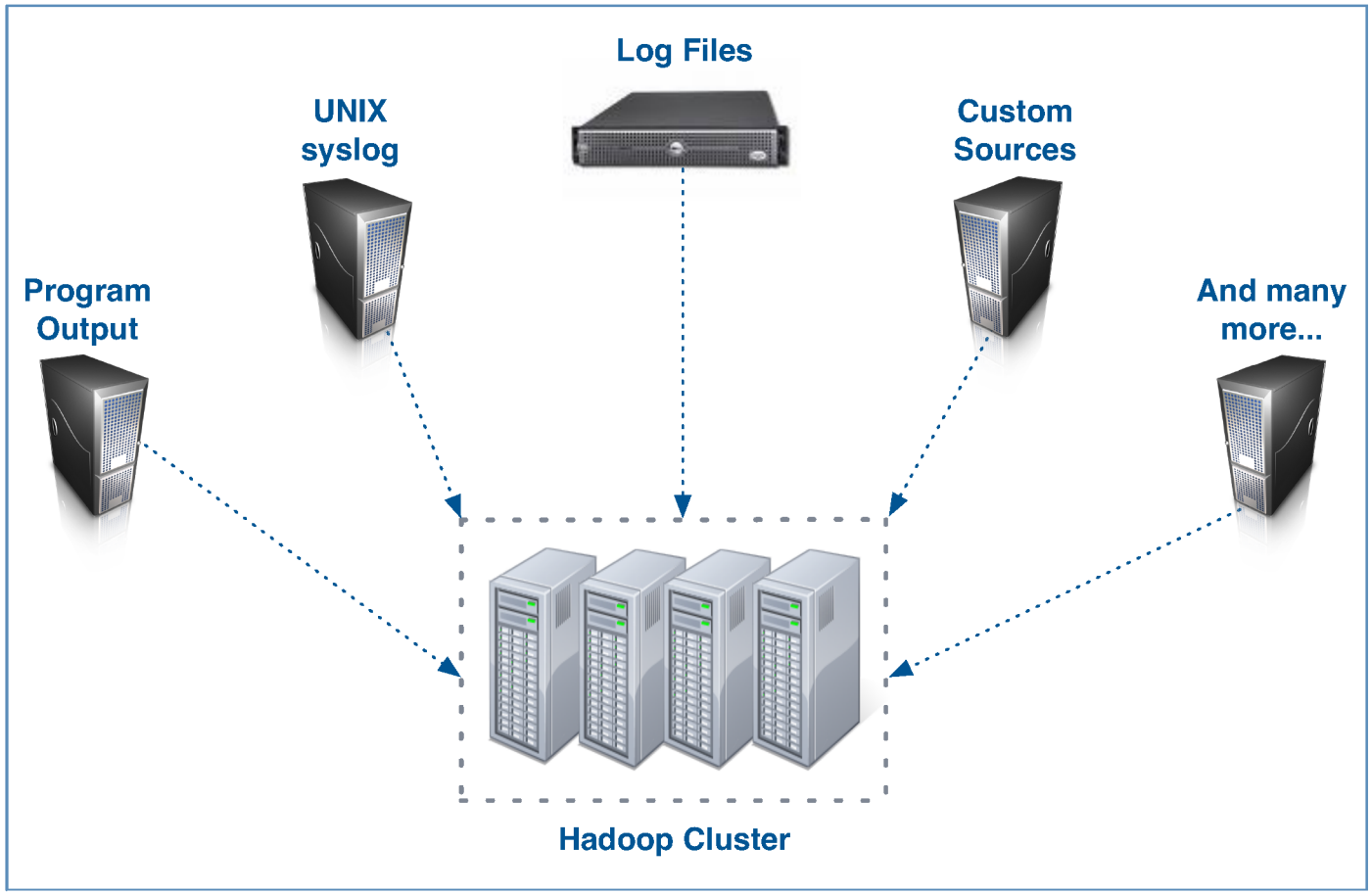
- The Motivation for Hadoop
- Hadoop Overview
- Data Storage: HDFS
- Distributed Data Processing: YARN, MapReduce, and Spark
- Data Processing and Analysis: Pig, Hive, and Impala
- Database Integration: Sqoop
- **Other Hadoop Data Tools**
- Exercise Scenario Explanation
- Essential Points
- Hands-On Exercise: Data Ingest with Hadoop Tools

Apache HBase

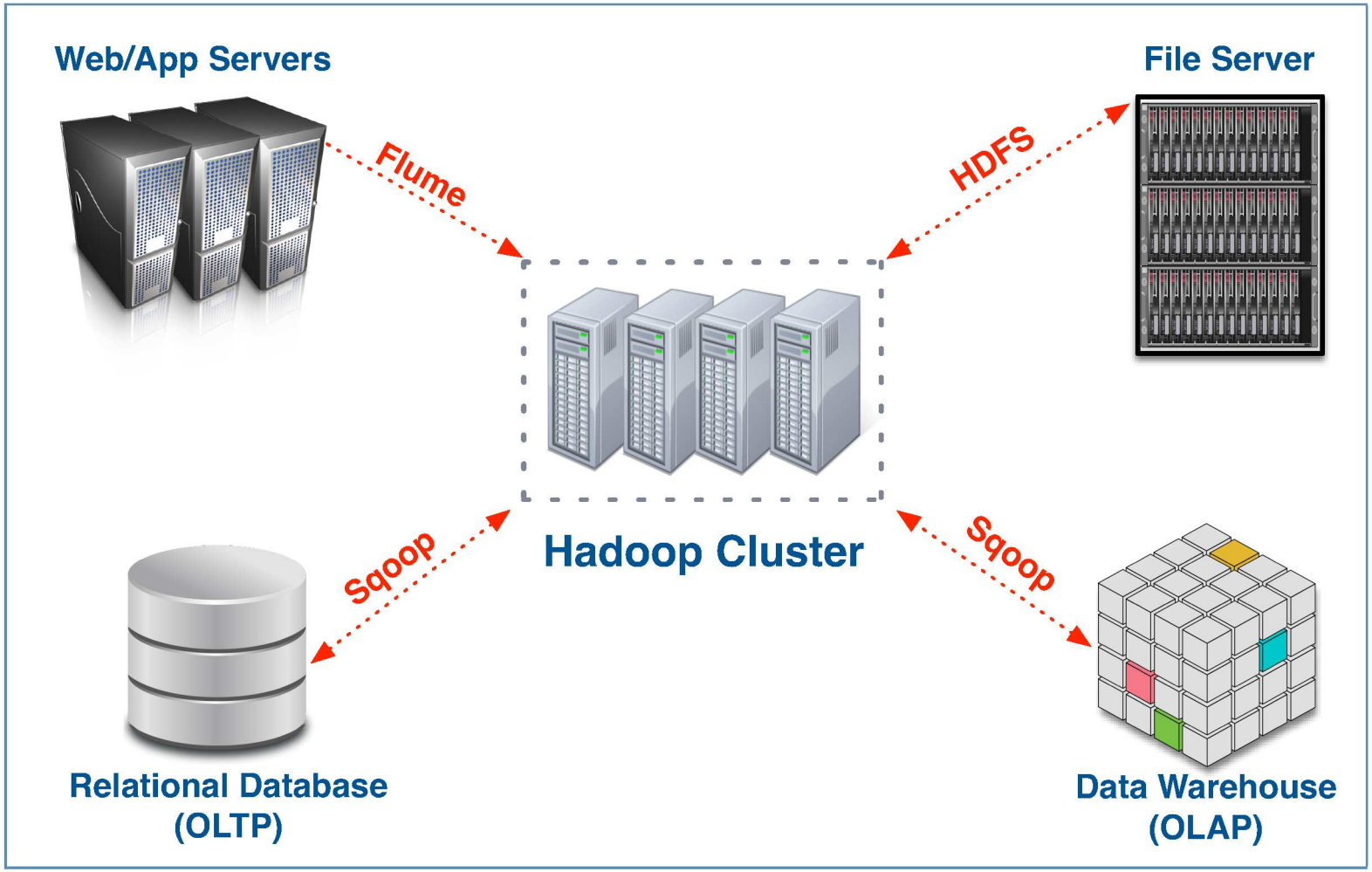
- **HBase is "the Hadoop database"**
- **Can store massive amounts of data**
  - Gigabytes, terabytes, and even petabytes of data in a table
  - Tables can have many thousands of columns
- **Scales to provide very high write throughput**
  - Hundreds of thousands of inserts per second
- **Fairly primitive when compared to an RDBMS**
  - NoSQL : There is no high-level query language
  - Use API to **`scan`** / **`get`** / **`put`** values based on keys

Apache Flume

- **Flume imports data into HDFS *as it is being generated* by various sources**



Recap: Data Center Integration



Chapter Topics

Apache Hadoop Fundamentals

- The Motivation for Hadoop
- Hadoop Overview
- Data Storage: HDFS
- Distributed Data Processing: YARN, MapReduce, and Spark
- Data Processing and Analysis: Pig, Hive, and Impala
- Database Integration: Sqoop
- Other Hadoop Data Tools
- **Exercise Scenario Explanation**
- Essential Points
- Hands-On Exercise: Data Ingest with Hadoop Tools

Hands-On Exercises: Scenario Explanation

- **Exercises throughout the course will reinforce the topics being discussed**
  - Exercises simulate the kind of tasks often performed using the tools you will learn about in class
  - Most exercises depend on data generated in earlier exercises
- **Scenario: Dualcore Inc. is a leading electronics retailer**
  - More than 1,000 brick-and-mortar stores
  - Dualcore also has a thriving e-commerce website
- **Dualcore has hired you to help find value in its data; you will**
  - Process and analyze data from internal and external sources
  - Identify opportunities to increase revenue
  - Find new ways to reduce costs
  - Help other departments achieve their goals

Chapter Topics

Apache Hadoop Fundamentals

- The Motivation for Hadoop
- Hadoop Overview
- Data Storage: HDFS
- Distributed Data Processing: YARN, MapReduce, and Spark
- Data Processing and Analysis: Pig, Hive, and Impala
- Database Integration: Sqoop
- Other Hadoop Data Tools
- Exercise Scenario Explanation
- **Essential Points**
- Hands-On Exercise: Data Ingest with Hadoop Tools

Essential Points

- **We are generating more data—and faster—than ever before**
- **Most of this data maps poorly to structured relational tables**
- **The ability to store and process this data can yield valuable insight**
- **Hadoop offers scalable data storage and processing**
- **There are lots of tools in the Hadoop ecosystem that help you to integrate Hadoop with other systems, manage complex jobs, and ease analysis**

Bibliography

The following offer more information on topics discussed in this chapter

- **10 Common Hadoop-able Problems (recorded presentation)**
  - **`http://tiny.cloudera.com/dac02a`**
- ***Apache Sqoop Cookbook*** **(O'Reilly book)**
  - **`http://tiny.cloudera.com/sqoopbook`**
- **HDFS Commands Guide**
  - **`http://tiny.cloudera.com/hdfscommands`**
- ***Hadoop: The Definitive Guide, 4th Edition*** **(O'Reilly book)**
  - **`http://tiny.cloudera.com/hadooptdg4`**
- **Sqoop User Guide**
  - **`http://tiny.cloudera.com/sqoopuser`**

Chapter Topics

Apache Hadoop Fundamentals

- The Motivation for Hadoop
- Hadoop Overview
- Data Storage: HDFS
- Distributed Data Processing: YARN, MapReduce, and Spark
- Data Processing and Analysis: Pig, Hive, and Impala
- Database Integration: Sqoop
- Other Hadoop Data Tools
- Exercise Scenario Explanation
- Essential Points
- **Hands-On Exercise: Data Ingest with Hadoop Tools**

Hands-On Exercise: Data Ingest with Hadoop Tools

- **In this Hands-On Exercise, you will gain practice adding data from the local filesystem and from a relational database server to HDFS**
  - You will analyze this data in subsequent exercises
  - Please refer to the Hands-On Exercise Manual for instructions

About the Training Virtual Machine

- **During this course, you will perform numerous hands-on exercises using the provided hands-on environment**
  - A virtual machine (VM) running Linux
- **This environment has Hadoop installed in *pseudo-distributed mode***
  - A cluster comprised of a single node
  - Typically used for testing code before deploying to a large cluster