Course Chapters

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

Introduction to Impala

In this chapter, you will learn

- **What Impala is and how it compares to Hive, Pig, and RDBMSs**
- **How Impala executes queries**
- **Where Impala fits into the data center**
- **What notable differences exist between Impala and Hive**
- **How to run queries from the shell or browser**

Chapter Topics

Introduction to Impala

- **What is Impala?**
- How Impala Differs from Hive and Pig
- How Impala Differs from Relational Databases
- Limitations and Future Directions
- Using the Impala Shell
- Conclusion

What is Impala?

- **High-performance SQL engine for vast amounts of data**
  - Massively-parallel processing (MPP)
  - Inspired by Google's Dremel project
  - Query latency measured in milliseconds
- **Impala runs on Hadoop clusters**
  - Can query data stored in HDFS or HBase tables
  - Reads and writes data in common Hadoop file formats
- **Developed by Cloudera**
  - 100% open source, released under the Apache software license

Interacting with Impala

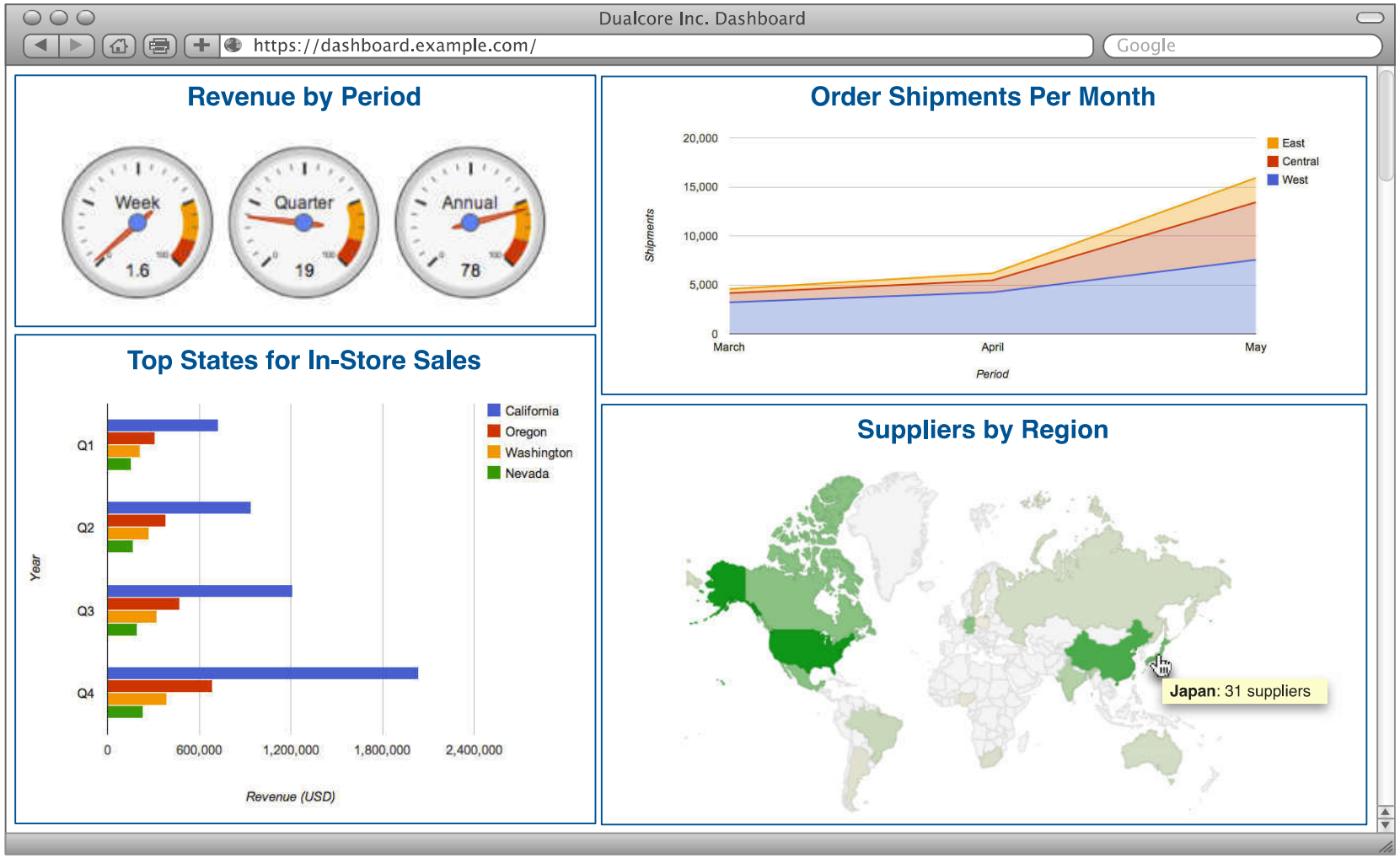
- **Impala supports a subset of SQL-92**
  - Plus a few extensions found in MySQL and Oracle SQL dialects
  - Almost identical to HiveQL
- **Impala offers many interfaces for running queries**
  - Command-line shell
  - Hue Web application
  - ODBC / JDBC

Why Use Impala?

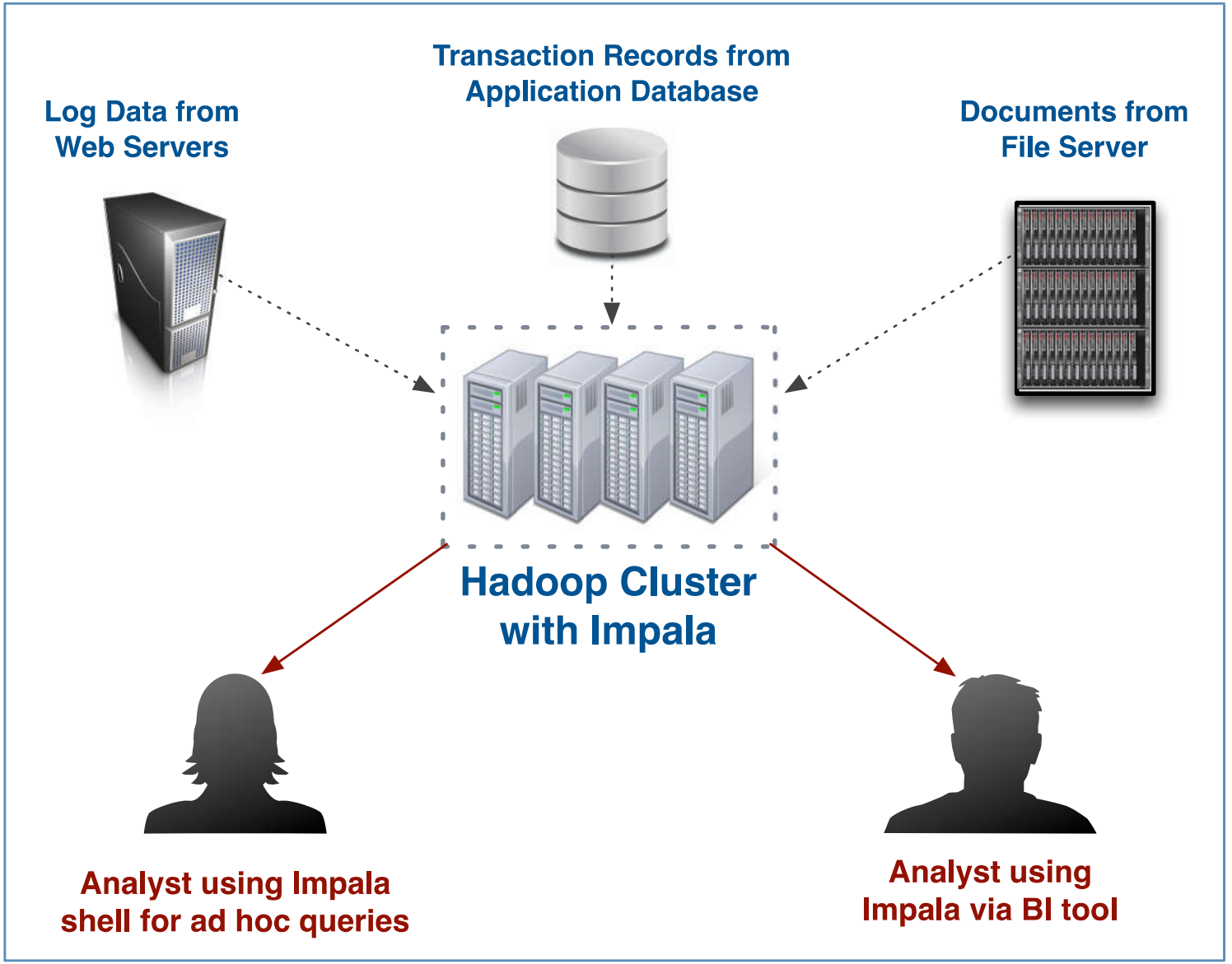
- **Many benefits are the same as with Hive or Pig**
  - More productive than writing MapReduce code
  - No software development experience required
  - Leverage existing knowledge of SQL
- **One benefit exclusive to Impala is speed**
  - Highly-optimized for queries
  - Almost always at least five times faster than either Hive or Pig
    - Often 20 times faster or more

Use Case: Business Intelligence

- **Many leading business intelligence tools support Impala**



Where Impala Fits Into the Data Center



Where to Get Impala

- **Download Impala from `http://www.cloudera.com/`**
- **We strongly recommend running Impala on CDH 4.2 or higher**
  - Requires a 64-bit Linux platform
- **Installation and configuration are outside the scope of this course**
  - Your virtual machine includes a working installation of Impala

Chapter Topics

Introduction to Impala

- What is Impala?
- **How Impala Differs from Hive and Pig**
- How Impala Differs from Relational Databases
- Limitations and Future Directions
- Using the Impala Shell
- Conclusion

Comparing Impala to Hive and Pig

- **Let's first look at *similarities* between Hive, Pig, and Impala**
  - Queries expressed in high-level languages
  - Alternatives to writing MapReduce code
  - Used to analyze data stored on Hadoop clusters
- **Impala shares the metastore with Hive**
  - Tables created in Hive are visible in Impala (and vice versa)

Contrasting Impala to Hive and Pig (1)

- **Hive and Pig answer queries by running MapReduce jobs**
  - MapReduce is a general-purpose computation framework
  - Not optimized for executing interactive SQL queries
- **MapReduce overhead results in high latency**
  - Even a trivial query takes 10 seconds or more
- **Impala does not use MapReduce**
  - Uses a custom execution engine built specifically for Impala
  - Queries can complete in a fraction of a second

Contrasting Impala to Hive and Pig (2)

- **Hive, Pig, and Impala also support**
  - Execute queries via interactive shell or command line
  - Grouping, joining, and filtering data
  - Read and write data in multiple formats
- **Impala currently lacks some Hive and Pig features**
  - More details later in this chapter
- **Hive and Pig are best suited to long-running batch processes**
  - Particularly data transformation tasks
- **Impala is best for interactive/ad hoc queries**

How Impala Executes a Query

- **Each 'slave' node in the cluster runs an Impala daemon**
  - Co-located with the HDFS Data Node
  - Client issues query to an Impala daemon
- **Impala daemon plans the query**
  - Checks the local metastore cache
  - Distributes the query across other Impala daemons in the cluster
  - Daemons read data from HDFS or HBase
  - Streams results to client
- **Two other daemons running on master nodes support query execution**
  - The State Store daemon
    - Provides lookup service for Impala daemons
    - Periodically checks status of Impala daemons
  - The Catalog daemon relays metadata changes to all the Impala daemons in a cluster

Chapter Topics

Introduction to Impala

- What is Impala?
- How Impala Differs from Hive and Pig
- **How Impala Differs from Relational Databases**
- Limitations and Future Directions
- Using the Impala Shell
- Conclusion

Comparing Impala To A Relational Database

	Relational Database	Impala
Query language	SQL	SQL-92 subset
Update individual records	Yes	No
Delete individual records	Yes	No
Transactions	Yes	No
Indexing	Yes	No
Latency	Very low	Low
Data size	Terabytes	Petabytes
ODBC / JDBC support	Yes	Yes

Chapter Topics

Introduction to Impala

- What is Impala?
- How Impala Differs from Hive and Pig
- How Impala Differs from Relational Databases
- **Limitations and Future Directions**
- Using the Impala Shell
- Conclusion

Hive Features Currently Unsupported in Impala

- **Impala does not currently support some features in Hive**
  - Many of these are being considered for future releases
- **Complex data types (`ARRAY`, `MAP`, or `STRUCT`)**
- **`BINARY` data type**
- **External transformations**
- **Custom SerDes**
- **Indexing**
- **Bucketing and table sampling**

Other Notable Differences Between Impala and Hive

- **Only one `DISTINCT` clause allowed per query in Impala**
  - Typical workaround is to use subselect and `UNION`
- **Impala requires that queries with `ORDER BY` also specify a `LIMIT`**
  - This sets an outer bound on the result set
  - The size of the `LIMIT` can be arbitrarily large
- **Impala and Hive handle out-of-range values differently**
  - Hive returns `NULL`
  - Impala returns the maximum value for that type

Query Fault Tolerance in Impala

- **Queries in both Hive and Impala are distributed across nodes**
- **Hive answers queries by running MapReduce jobs**
  - Takes advantage of Hadoop's fault tolerance
  - If a node fails during query, MapReduce runs the task elsewhere
- **Impala has its own execution engine**
  - Currently lacks fault tolerance
  - If a node fails during a query, the query will fail
  - Workaround: just re-run the query

Chapter Topics

Introduction to Impala

- What is Impala?
- How Impala Differs from Hive and Pig
- How Impala Differs from Relational Databases
- Limitations and Future Directions
- **Using the Impala Shell**
- Conclusion

Starting the Impala Shell

- **You can execute statements in the Impala's shell**
  - This interactive tool is similar to the shell in MySQL or Hive
- **Execute the `impala-shell` command to start the shell**
  - Some log messages truncated to better fit the slide

```
$ impala-shell
Connected to localhost.localdomain:21000
Server version: impalad version 1.0.1
Welcome to the Impala shell.
[localhost.localdomain:21000] >
```

- **Use `-i hostname:port` option to connect to another server**

```
$ impala-shell -i myserver.example.com:21000
[myserver.example.com:21000] >
```

Using the Impala Shell

- **Enter semicolon-terminated statements at the prompt**
  - Hit enter to execute the query
  - Impala pretty-prints the output
  - Use the `quit` command to exit the Impala shell

```
$ impala-shell
> SELECT cust_id, fname, lname FROM customers
  WHERE zipcode='20525';
+---------+--------+-----------+
| cust_id | fname  | lname     |
+---------+--------+-----------+
| 1133567 | Steven | Robertson |
| 1171826 | Robert | Gillis    |
+---------+--------+-----------+
> quit;
$
```

Note: shell prompt abbreviated as **>**

Running Queries from the Command Line

- You can execute a file containing queries using the `-f` option

```
$ impala-shell -f myquery.hql
```

- Run queries directly from the command line with the `-q` option

```
$ impala-shell -q 'SELECT * FROM users'
```

- Use `-o` (and optionally specify delimiter) to capture output to file

```
$ impala-shell -f myquery.hql \
      -o results.txt \
      --output_file_field_delim='\t'
```

Interacting with the Operating System

- **Use `shell` to execute system commands from within Impala shell**

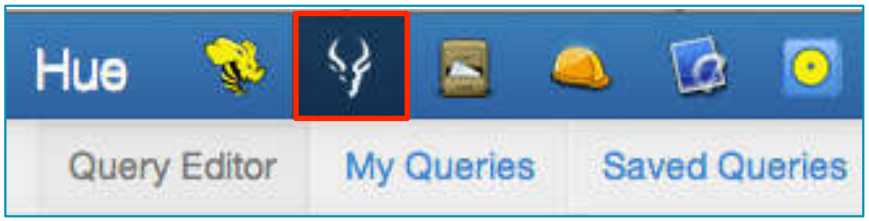
```
> shell date;
Mon May 20 16:44:35 PDT 2013
```

- **No direct support for HDFS commands**
  - But could run `hadoop fs` using `shell`

```
> shell hadoop fs -mkdir /reports/sales/2013;
```

Accessing Impala with Hue (1)

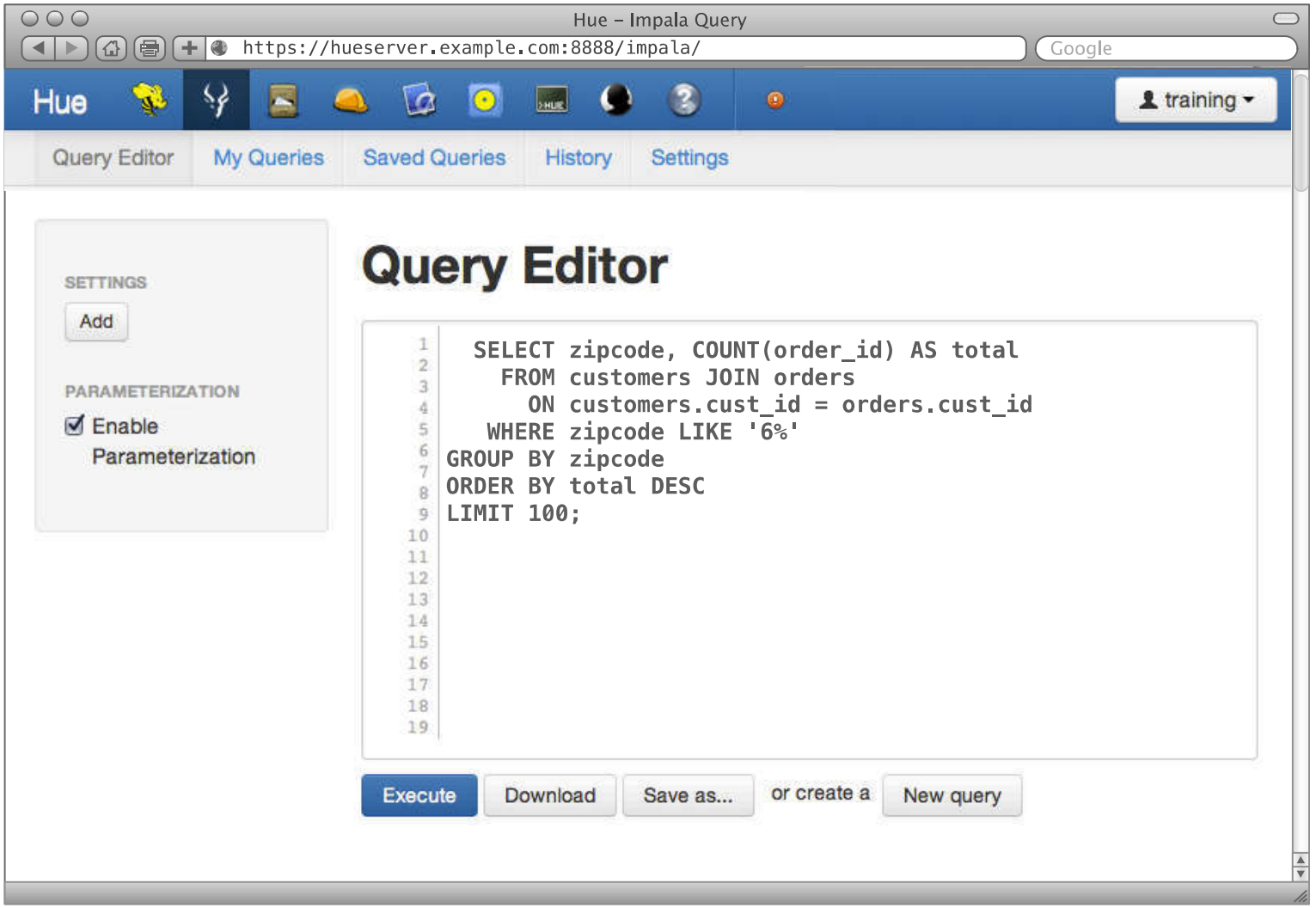
- **Alternatively, you can access Impala through Hue**
- **To use Hue, browse to `http://hue_server:8888/`**
  - May need to start Hue service first (`sudo service hue start`)
- **Launch Hue's Impala interface by clicking its icon**



Impala icon in Hue

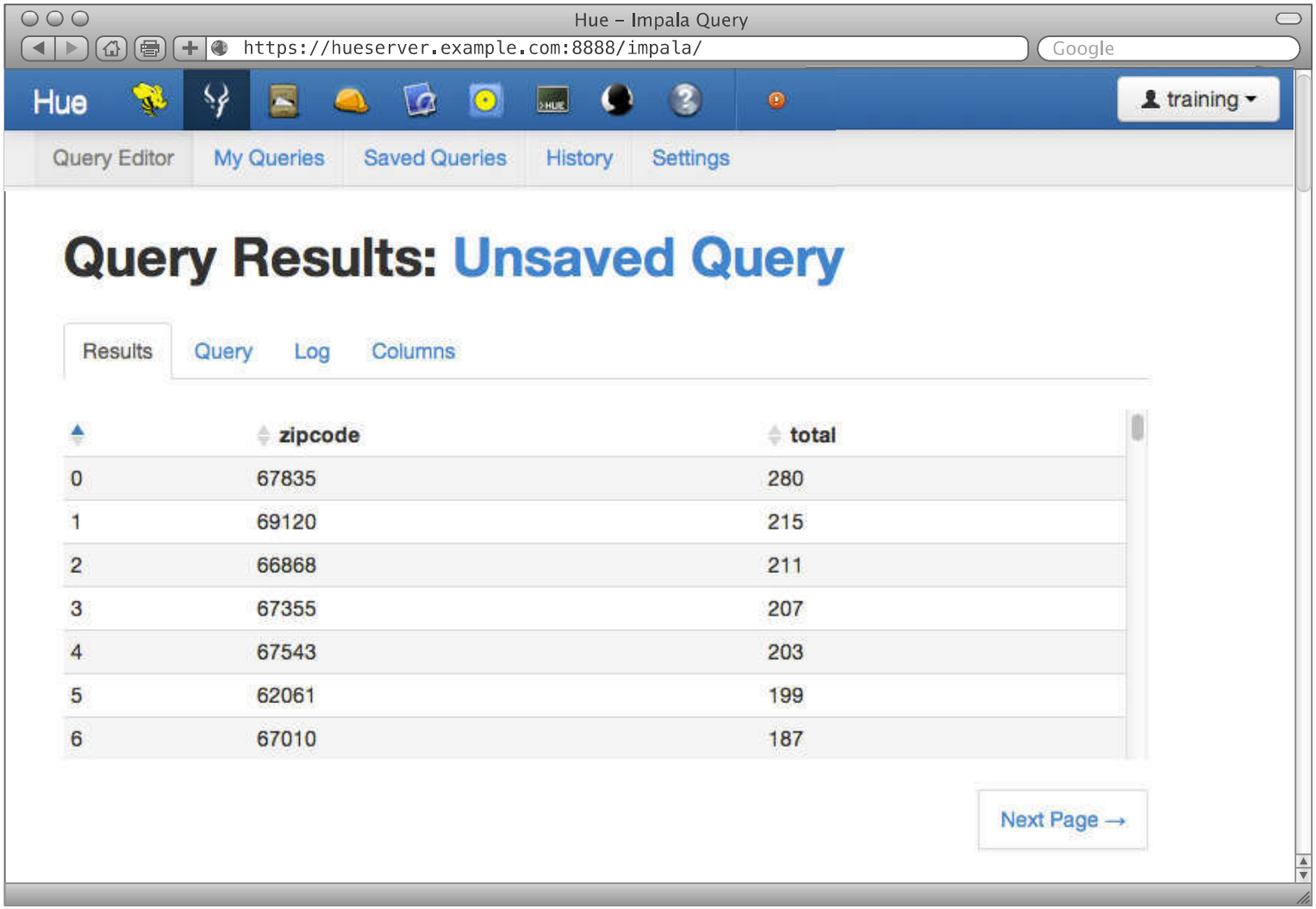
Accessing Impala with Hue (2)

- **Hue allows you to run Impala queries from your Web browser**



Accessing Impala with Hue (3)

- **Hue displays the results in a sortable table**



Chapter Topics

Introduction to Impala

- What is Impala?
- How Impala Differs from Hive and Pig
- How Impala Differs from Relational Databases
- Limitations and Future Directions
- Using the Impala Shell
- **Conclusion**

Essential Points

- **Impala is a high-performance SQL engine**
  - Runs on Hadoop clusters
  - Reads and writes data in HDFS or HBase tables
- **Queries are expressed in SQL dialect similar to HiveQL**
- **Primary difference compared to Hive/Pig is speed**
  - Impala avoids MapReduce latency and overhead
- **Impala is best suited to ad hoc/interactive queries**
  - Hive and Pig are better for long-running batch processes
  - Impala does not currently support all features of Hive

Bibliography (1)

The following offer more information on topics discussed in this chapter

- **Free O'Reilly *Cloudera Impala* book**
  - http://tiny.cloudera.com/dac15f
- **Cloudera Impala: Real-Time Queries in Apache Hadoop**
  - http://tiny.cloudera.com/dac15a
- ***Wired* Article on Impala**
  - http://tiny.cloudera.com/dac15b
- **Cloudera Blog Detailing Impala Features and Performance**
  - http://tiny.cloudera.com/dac15c
- **Impala Documentation at Cloudera Web site**
  - http://tiny.cloudera.com/dac15e

Bibliography (2)

- **37Signals Blog Comparing Performance of Impala, Hive, and MySQL**
  - `http://tiny.cloudera.com/dac15d`

cloudera
Ask Bigger Questions

Analyzing Data with Impala

Chapter 16

Course Chapters

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

Analyzing Data with Impala

In this chapter, you will learn

- **How Impala's query syntax compares to HiveQL**
- **How to create databases and tables in Impala**
- **How and when to refresh the metadata cache**
- **Which data types Impala supports**
- **How to add support for a User-Defined Function (UDF)**
- **How to structure your query for better performance**
- **What other factors influence Impala performance**

Chapter Topics

Analyzing Data with Impala

- **Basic Syntax**
- Data Types
- Filtering, Sorting, and Limiting Results
- Joining and Grouping Data
- User-Defined Functions
- Improving Impala Performance
- Hands-On Exercise: Interactive Analysis with Impala
- Conclusion

Overview of Impala Query Syntax

- **Impala's query language is a subset of SQL-92**
  - Plus a few extensions from Oracle and MySQL dialects
- **Syntax almost identical to HiveQL**
  - Differences mainly related to features unsupported in Impala
  - Most Hive queries can be executed verbatim in Impala
- **Impala may also support statements that Hive does not**
  - Such as the ability to insert individual rows *

```
> INSERT INTO customers VALUES (1234567, 'Abe',
  'Froman', '123 Oak St.', 'Chicago', 'IL', '60601');
```

* Not intended for bulk loads

Case-Sensitivity and Comments

- **Keywords are not case-sensitive, but often capitalized by convention**
- **Supports single- and multi-line comments**
  - These are allowed in scripts, shell, and command line

```
$ cat find_customers.sql

/* This script will query the customers table
 * and find all customers in a given ZIP code
 */
SELECT cust_id, fname, lname
   FROM customers
   WHERE zipcode='60601';    -- downtown Chicago
```

Databases and Tables in Impala

- **Every Impala table belongs to a database**
  - Impala and Hive share a metastore
  - The same databases and tables are visible in both Hive and Impala
- **The `default` database is selected at startup**
  - The `USE` command switches to another database
  - List tables in a database with the `SHOW TABLES` command

```
> USE accounting;
> SHOW TABLES;
+----------+
| name     |
+----------+
| accounts |
+----------+
```

Creating Databases and Tables in Impala

- **Data definition is generally identical to Hive**
  - Custom SerDes and bucketing are unsupported

```
> CREATE DATABASE sales;
> USE sales;
> CREATE EXTERNAL TABLE prospects
    (id INT,
     name STRING COMMENT 'Include surname',
     email STRING,
     active BOOLEAN COMMENT 'True, if on mailing list',
     last_contact TIMESTAMP)
  ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '\t'
    LOCATION '/dept/sales/prospects';
```

Displaying Table Structure

- **Use `DESCRIBE` to display a table's structure**
  - `DESCRIBE EXTENDED` is unsupported

```
> DESCRIBE prospects;
+--------------+-----------+--------------------------+
| name         | type      | comment                  |
+--------------+-----------+--------------------------+
| id           | int       |                          |
| name         | string    | Include surname          |
| email        | string    |                          |
| active       | boolean   | True, if on mailing list |
| last_contact | timestamp |                          |
+--------------+-----------+--------------------------+
```

Metadata Caching in Impala

- **Impala shares the metastore with Hive**
  - Tables created in Hive are visible in Impala (and vice versa)
- **Impala caches metadata**
  - The tables' schema definitions
  - The locations of tables' HDFS blocks
- **Metadata updates made *from within Impala* are broadcast throughout the cluster**
  - The Impala metadata cache is updated automatically throughout the Impala cluster
  - No additional actions are required
- **Metadata updates made *from outside of Impala* are not known to Impala**
  - For example, changes made in Hive, or changes to the tables in HDFS
  - Additional actions are required to update the metadata cache

Updating the Impala Metadata Cache

Metadata Change	Required Action	Effect on Cache
New table added to the metastore in Hive	`INVALIDATE METADATA` (with no table name)	**Marks the entire cache as stale; metadata cache is reloaded as needed**
Table schema modified in Hive	`REFRESH <table>`	**Reloads the metadata for one table immediately. Reloads HDFS block locations for new data files only.**
New data added to a table	`REFRESH <table>`	**(Same as above)**
Data in a table has been extensively altered, such as by HDFS balancing	`INVALIDATE METADATA <table>`	**Marks the metadata for a single table as stale. When the metadata is needed, all HDFS block locations are retrieved.**

Selecting Data in Impala

- **Use `SELECT` to retrieve data from tables**
  - Results are formatted for display

```
> SELECT fname,lname,city,state FROM customers
    WHERE cust_id = 1234567;
+-------+--------+---------+-------+
| fname | lname  | city    | state |
+-------+--------+---------+-------+
| Abe   | Froman | Chicago | IL    |
+-------+--------+---------+-------+
```

  - Impala does not require a `FROM` clause

```
> SELECT SQRT(64) AS square_root;
```

Using Impala Built-in Functions

- **Invoke built-in functions as you would in SQL or HiveQL**

```
> SELECT CONCAT_WS(', ', lname, fname) AS fullname
    FROM customers WHERE cust_id=1234567;

+--------------+
| fullname     |
+--------------+
| Froman, Abe  |
+--------------+
```

- **Impala supports many of the same built-in functions as Hive**
  - Lacks some others, including many formatting and text processing functions
  - `SHOW FUNCTIONS` command is not implemented

Chapter Topics

Analyzing Data with Impala

- Basic Syntax
- **Data Types**
- Filtering, Sorting, and Limiting Results
- Joining and Grouping Data
- User-Defined Functions
- Improving Impala Performance
- Hands-On Exercise: Interactive Analysis with Impala
- Conclusion

Data Types in Impala

- **Each column in a table is associated with a data type**
  - Impala supports most types available in Hive
  - Most are similar to those found in relational databases

```
> DESCRIBE prospects;
+--------------+-----------+--------------------------+
| name         | type      | comment                  |
+--------------+-----------+--------------------------+
| id           | int       |                          |
| name         | string    | Include surname          |
| email        | string    |                          |
| active       | boolean   | True, if on mailing list |
| last_contact | timestamp |                          |
+--------------+-----------+--------------------------+
```

Impala's Integer Types

- **Integer types are appropriate for whole numbers**
  - Both positive and negative values allowed

Name	Description	Example Value
`TINYINT`	Range: -128 to 127	`17`
`SMALLINT`	Range: -32,768 to 32,767	`5842`
`INT`	Range: -2,147,483,648 to 2,147,483,647	`84127213`
`BIGINT`	Range: ~ -9.2 quintillion to ~ 9.2 quintillion	`632197432180964`

Impala's Decimal Types

- **Decimal types are appropriate for floating point numbers**
  - Both positive and negative values allowed
  - **Caution**: avoid using when exact values are required!

Name	Description	Example Value
`FLOAT`	Decimals	`3.14159`
`DOUBLE`	Very precise decimals	`3.141592653589793238 46`

Other Types in Impala

- **Impala can also store a few other types of information**
  - Only one character type (variable length)

Name	Description	Example Value
`STRING`	Character sequence	`Betty F. Smith`
`BOOLEAN`	True or False	`TRUE`
`TIMESTAMP`	Instant in time	`2013-06-14 16:51:05`

- **Impala does not support `BINARY` or complex types**

Data Type Conversion

- **Hive auto-converts a `STRING` column used in numeric context**

```
hive> SELECT zipcode FROM customers LIMIT 1;
60601
hive> SELECT zipcode + 1.5 FROM customers LIMIT 1;
60602.5
```

- **Impala requires an explicit `CAST` operation for this**

```
> SELECT zipcode + 1.5 FROM customers LIMIT 1;
ERROR: AnalysisException: Arithmetic operation ...
> SELECT CAST(zipcode AS float) + 1.5
  FROM customers LIMIT 1;
60602.5
```

Chapter Topics

Analyzing Data with Impala

- Basic Syntax
- Data Types
- **Filtering, Sorting, and Limiting Results**
- Joining and Grouping Data
- User-Defined Functions
- Improving Impala Performance
- Hands-On Exercise: Interactive Analysis with Impala
- Conclusion

Limiting and Sorting Query Results

- **The `LIMIT` clause sets the maximum number of rows returned**

```
> SELECT fname, lname FROM customers LIMIT 10;
```

- **Caution: no guarantee regarding *which* 10 results are returned**
  - Use `ORDER BY` for top-N queries
  - The field(s) you `ORDER BY` must be selected
- **When using `ORDER BY`, the `LIMIT` clause is mandatory in Impala**

```
> SELECT cust_id, fname, lname FROM customers
    ORDER BY cust_id DESC LIMIT 10;
```

Chapter Topics

Analyzing Data with Impala

- Basic Syntax
- Data Types
- Filtering, Sorting, and Limiting Results
- **Joining and Grouping Data**
- User-Defined Functions
- Improving Impala Performance
- Hands-On Exercise: Interactive Analysis with Impala
- Conclusion

Joins in Impala

- **Like Hive, Impala can join multiple data sets**
- **Impala supports the same types of joins that Hive does**
  - Inner joins
  - Outer joins (left, right, and full)
  - Left semi joins
  - Cross joins

Record Grouping and Aggregate Functions

- **`GROUP BY` groups selected data by one or more columns**
  - **Caution**: Columns not part of aggregation must be listed in `GROUP BY`

stores table

id	city	state	region
a	Albany	NY	EAST
b	Boston	MA	EAST
c	Chicago	IL	NORTH
d	Detroit	MI	NORTH
e	Elgin	IL	NORTH

```
SELECT region, state,
       COUNT(id) AS num
  FROM stores
  GROUP BY region, state;
```

Result of query

region	state	num
EAST	MA	1
EAST	NY	1
NORTH	IL	2
NORTH	MI	1

Chapter Topics

Analyzing Data with Impala

- Basic Syntax
- Data Types
- Filtering, Sorting, and Limiting Results
- Joining and Grouping Data
- **User-Defined Functions**
- Improving Impala Performance
- Hands-On Exercise: Interactive analysis with Impala
- Conclusion

Overview of Impala User-Defined Functions (UDFs)

- **Like Hive, Impala supports User-Defined Functions (UDFs)**
- **Hive UDFs can be used in Impala with no changes**
  - With a few exceptions
- **There are two types of UDFs in Impala**
  - Standard UDFs
  - User-Defined Aggregate Functions (UDAFs)
- **Impala UDFs can be written in Java or C++**
  - C++ UDFs are implemented as shared objects
- **Impala C++ UDFs cannot be used in Hive**

Using a Java UDF in Impala (1)

- **Register the function with Impala**
  - Specify data types that correspond to the method signature of the UDF class' `evaluate` method after the function name
  - Specify data types that correspond to the return type of the UDF class' `evaluate` method in the `RETURNS` clause
  - Identify the jar file containing the UDF class in the `LOCATION` clause
  - Specify the UDF class name in the `SYMBOL` clause
  - You do not need to run a a separate `ADD JAR` step

```
CREATE FUNCTION STRIP(STRING) RETURNS STRING
  LOCATION '/user/hive/udfs/MyUDFs.jar'
  SYMBOL='com.example.hive.ql.udf.UDFStrip';
```

Using a Java UDF in Impala (2)

- **You may then use the function in Impala queries**

```
SELECT STRIP(email_address) FROM employees;
```

Using a C++ UDF in Impala

- **Register the function with Impala**

```
CREATE FUNCTION COUNT_VOWELS(STRING)
  RETURNS INT
  LOCATION '/user/hive/udfs/sampleudfs.so'
  SYMBOL='CountVowels';
```

- **You may then use the function in your query**

```
SELECT COUNT_VOWELS(email_address) FROM employees;
```

Cluster Hardware

- **CPU: Impala benefits from newer processors**
  - Takes advantage of additional optimization when available
- **Memory: 32GB minimum, 64GB is better**
- **Disks: more is better**
  - Impala tries to maximize throughput across disks
  - Servers with 12 or more disks are ideal

Chapter Topics

Analyzing Data with Impala

- Basic Syntax
- Data Types
- Filtering, Sorting, and Limiting Results
- Joining and Grouping Data
- User-Defined Functions
- Improving Impala Performance
- **Hands-On Exercise: Interactive Analysis with Impala**
- Conclusion

Hands-on Exercise: Interactive Analysis with Impala

- **In this Hands-On Exercise, you will run ad hoc queries with Impala**
- **Please refer to the Hands-On Exercise Manual for instructions**

Chapter Topics

Analyzing Data with Impala

- Basic Syntax
- Data Types
- Filtering, Sorting, and Limiting Results
- Joining and Grouping Data
- User-Defined Functions
- Improving Impala Performance
- Hands-On Exercise: Interactive analysis with Impala
- **Conclusion**

Essential Points

- **Impala's query syntax is nearly identical to HiveQL**
  - Most Hive queries can be executed verbatim in Impala
- **Impala caches metadata from the metastore it shares with Hive**
  - Use `INVALIDATE METADATA` or `REFRESH` to update the cache following external changes
- **Impala supports most simple data types from Hive**
- **Query structure and file format can affect performance**
- **Your cluster's hardware also affects performance**

Bibliography

The following offer more information on topics discussed in this chapter

- **Introducing Parquet: Efficient Columnar Storage for Apache Hadoop**
  - `http://tiny.cloudera.com/dac16a`
- **Impala Language Reference**
  - `http://tiny.cloudera.com/dac16b`

cloudera
Ask Bigger Questions

Choosing the Best Tool for the Job

Chapter 17

Course Chapters

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

Choosing the Best Tool for the Job

In this chapter, you will learn

- **How MapReduce, Pig, Hive, Impala, and RDBMSs compare to one another**
- **Why a workflow might involve several different tools**
- **How to select the best tool for a given job**

Chapter Topics

Choosing the Best Tool for the Job

- **Comparing MapReduce, Pig, Hive, Impala, and Relational Databases**
- Which to Choose?
- Conclusion

Recap of Data Analysis/Processing Tools

- **MapReduce**
  - Low-level processing and analysis
- **Pig**
  - Procedural data flow language executed using MapReduce
- **Hive**
  - SQL-based queries executed using MapReduce
- **Impala**
  - High-performance SQL-based queries using a custom execution engine

Comparing Pig, Hive, and Impala

Description of Feature	Pig	Hive	Impala
SQL-based query language	No	Yes	Yes
Optional schema and metastore	Yes	No	No
User-defined functions (UDFs)	Yes	Yes	Yes
Process data with external scripts	Yes	Yes	No
Extensible file format support	Yes	Yes	No
Complex data types	Yes	Yes	No
Query latency	High	High	Low
Built-in data partitioning	No	Yes	Yes
Accessible via ODBC / JDBC	No	Yes	Yes

Do These Replace an RDBMS?

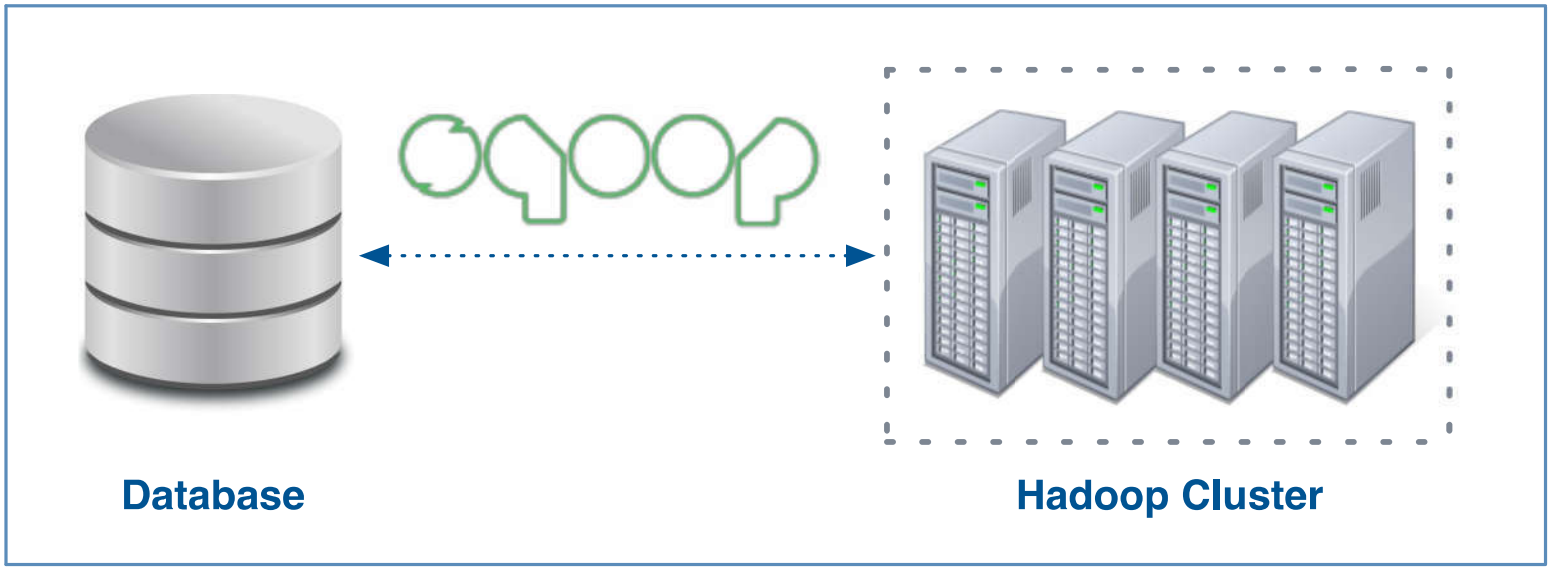
- **Probably not if RDBMS is used for its intended purpose**
- **Relational databases are optimized for**
  - Relatively small amounts of data
  - Immediate results
  - In-place modification of data (`UPDATE` and `DELETE`)
- **Pig, Hive, and Impala are optimized for**
  - Large amounts of read-only data
  - Extensive scalability at low cost
- **Pig and Hive are better suited for batch processing**
  - Impala and RDBMSs are better for interactive use

Comparing RDBMS to Hive and Impala

	RDBMS	Hive	Impala
Insert individual records	Yes	No	Yes
Update and delete records	Yes	No	No
Transactions	Yes	No	No
Role-based authorization	Yes	Yes	No
Stored procedures	Yes	No	No
Index support	Extensive	Limited	None
Latency	Very low	High	Low
Data size	Terabytes	Petabytes	Petabytes
Complex data types	No	Yes	No
Storage cost	Very high	Very low	Very low

Recap: Apache Sqoop

- **Sqoop helps you integrate Hadoop tools with relational databases**
- **It exchanges data between RDBMS and Hadoop**
  - Can import all tables, a single table, or a portion of a table into HDFS
  - Supports incremental imports
  - Can also export data from HDFS back to the database



Chapter Topics

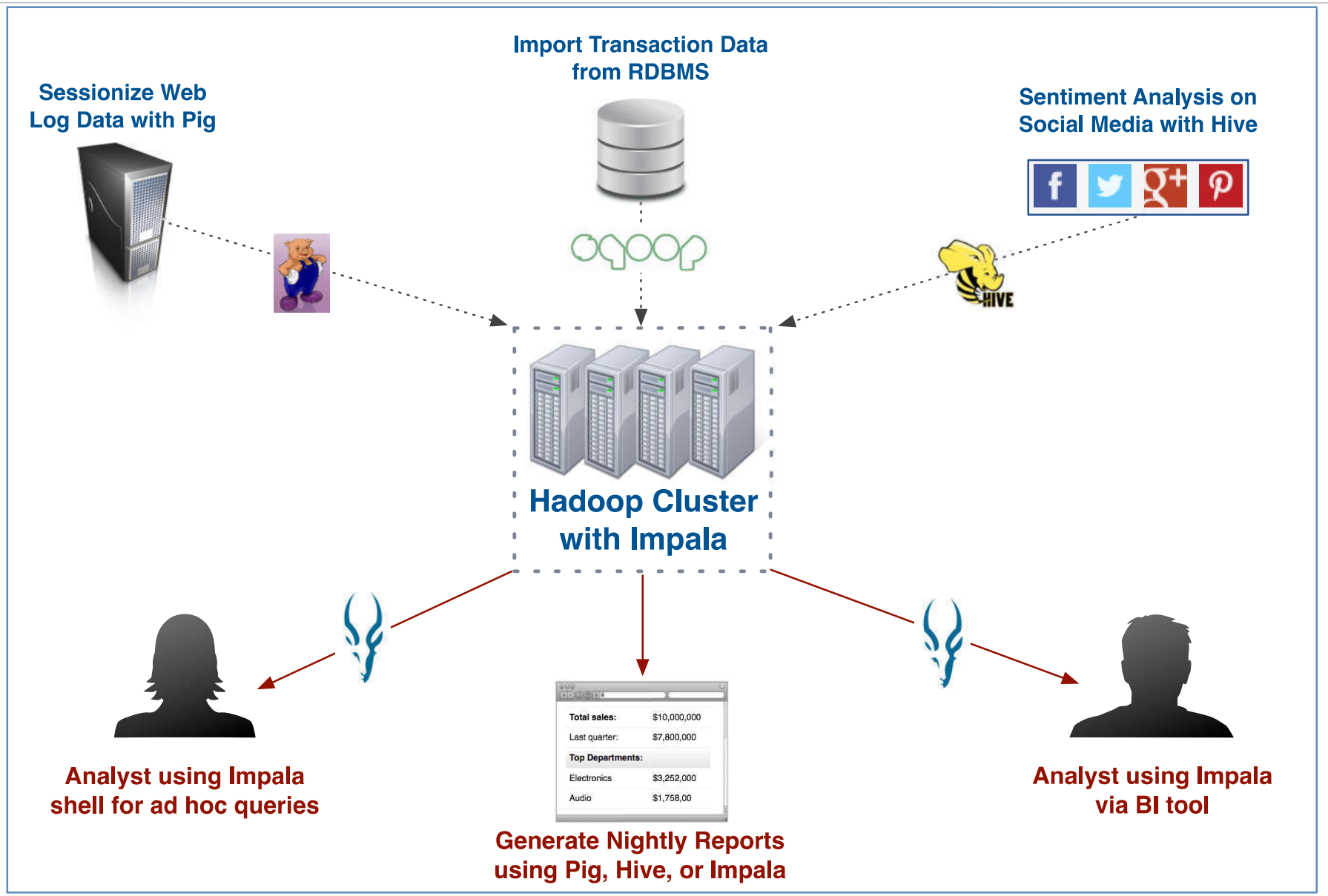
Choosing the Best Tool for the Job

- Comparing MapReduce, Pig, Hive, Impala, and Relational Databases
- **Which to Choose?**
- Conclusion

Which to Choose?

- **Choose the best one for a given task**
  - Mix and match them as needed
- **MapReduce**
  - Low-level approach offers great flexibility
  - More time-consuming and error-prone to write
  - Best when control matters more than productivity
- **Pig, Hive, and Impala offer more productivity**
  - Faster to write, test, and deploy than MapReduce
  - Better choice for most analysis and processing tasks

Analysis Workflow Example



Chapter Topics

Choosing the Best Tool for the Job

- Comparing MapReduce, Pig, Hive, Impala, and Relational Databases
- Which to Choose?
- **Conclusion**

Essential Points

- **You have learned about several tools for data analysis**
  - Each is better at some tasks than others
  - Choose the best one for a given job
  - Workflows may involve exchanging data between them
- **Selection criteria include scale, speed, control, and productivity**
  - MapReduce offers control at the cost of productivity
  - Pig and Hive offer productivity but not necessarily speed
  - Relational databases offer speed but not scalability
  - Impala offers scalability and speed but less control

cloudera
Ask Bigger Questions

Conclusion

Chapter 18

Course Chapters

- Introduction
- Hadoop Fundamentals
- Introduction to Pig
- Basic Data Analysis with Pig
- Processing Complex Data with Pig
- Multi-Dataset Operations with Pig
- Extending Pig
- Pig Troubleshooting and Optimization
- Introduction to Hive
- Relational Data Analysis with Hive
- Hive Data Management
- Text Processing With Hive
- Hive Optimization
- Extending Hive
- Introduction to Impala
- Analyzing Data with Impala
- Interoperability and Workflows
- **Conclusion**