SQL: From Traditional Databases to Big Data

Yasin N. Silva, Isadora Almeida, Michell Queiroz

Arizona State University
1. Key Big Data Technologies

Key Types of Big Data Systems
- MapReduce (Hadoop)
- NoSQL
- NewSQL

Contributions
- Identification of broad types of Big Data systems where SQL can be effectively used
- Guidelines to prepare course units on the use of SQL with Big Data systems
- Class resources for each unit
- Files of all the resources made available to instructors

2. SQL in MapReduce Systems

MapReduce

- Core programming model for distributed processing
- Distributed programs that run on failure-tolerant and scalable clusters of commodity machines
- Divides the processing task into two main phases: map and reduce
- Could be complex and difficult to learn
- Systems have been proposed to:
  1. Enable the use of SQL on top of MapReduce (Apache Hive and Apache Pig)
  2. Integrate SQL with MapReduce-based computations (Spark SQL)
• Supports the processing and analysis of data stored in Hadoop
• Allows **projecting structure** onto this data and querying it using HiveQL (~SQL)
• Transparentsly **converts** queries specified in HiveQL to MapReduce programs
• Built for data warehouse applications
• Many SQL features: DDL and DML
• It support transactions with full ACID semantics at the row (record) level

2.1 Hive: SQL Queries on Hadoop

C1: sudo hive
C2: CREATE database MStation;
C3: CREATE TABLE stationData (stationed int, zipcode int, latitude double, longitude double, stationname string);
C4: CREATE TABLE weatherReport (stationid int, temp double, humi double, precip double, year int, month string);
C5: LOAD DATA LOCAL INPATH '/home/cloudera/datasetation/stationData.txt' into table stationData;
C6: LOAD DATA LOCAL INPATH '/home/cloudera/datasetation/weatherReport.txt' into table weatherReport;
Q1: SELECT S.zipcode, AVG(W.precip) FROM stationData S JOIN weatherReport W ON S.stationid = W.stationid GROUP BY S.zipcode;
Q2: SELECT stationid, AVG(humi) FROM weatherReport GROUP BY stationid;
Q3: SELECT stationid, MIN(temp), MAX(temp) FROM weatherReport WHERE year > 2000 GROUP BY stationid;
Q4: SELECT S.zipcode, W.temp, W.month, W.year FROM stationData S JOIN weatherReport W ON S.stationid = W.stationid ORDER BY W.temp DESC LIMIT 10;
• Spark is a highly distributed data processing framework
• Spark loads the data into memory and uses a wider set of processing primitives
• Spark SQL is the Spark module for structured data processing
• DataFrames can be created from existing Spark datasets, Hive tables, JSON
• Spark SQL supports many of the features of SQL
• The output of SQL queries can be used as the input of MapReduce computations

C11: applestock.registerTempTable("applestock")
C12: applestock.show
C13: applestock.count
C14: output.map(t => "Record: " + t.toString()).collect().foreach(println)

Q1: val output = sql("SELECT * FROM applestock WHERE close >= open")
Q2: val output = sql("SELECT MAX(close-open) FROM applestock")
Q3: val output = sql("SELECT date, high FROM applestock ORDER BY high DESC LIMIT 10")
Q4: val output = sql("SELECT year, AVG(olume) FROM applestock WHERE year > 1999 GROUP BY year")

Spark SQL | Spark Streaming | MLlib (machine learning) | GraphX (graph)

Apache Spark
3. SQL In NoSQL Systems

- Provide higher scalability and availability than conventional relational databases
- Have a distributed, fault-tolerant architecture
- Data is partitioned and replicated
- Implement a simplified transaction/consistency model (e.g., eventual consistency)
- Types: Document/Key-Value/Tabular data stores
- Originally NoSQL systems did not support the relational database model and SQL
- Recent recognition of the value of SQL-like languages
- Systems that enable SQL queries on NoSQL stores:
  - Impala → HBase
  - SlamData → MongoDB
  - Presto → Cassandra
• MongoDB is a document-oriented and open-source NoSQL database
• Documents (JSON) are equivalent to rows in relational DBs
• JSON: key-value pairs (values can be documents or arrays)
• Embedded documents and arrays reduce the need for joins

```json
{
    name: "sue",
    age: 26,
    status: "A",
    groups: ["news", "sports"]
}
```

• SlamData allows running SQL statements on MongoDB

3.1 SlamData: SQL on MongoDB
• HBase is a NoSQL database that runs on top of HDFS
• HBase provides a fault-tolerant way of storing and processing large sparse tabular data
• HBase provides efficient random reads and writes on top of HDFS
• Impala is an open-source query engine that enables SQL queries on top of HBase tables
• Impala builds on key components of Hive, e.g., SQL syntax, metadata and schema
• Hive is best for long-running batch processing and Impala for dynamic analytic queries

3.2 Impala: SQL to Query HBase Tables
4. SQL In NewSQL Systems

- Aims to have the same levels of scalability and availability of NoSQL
- But maintaining key properties of traditional databases
  - **ACID** properties (Atomicity, Consistency, Isolation, Durability)
  - SQL
  - **Relational** data model
- Based on a distributed shared-nothing architecture that can
dynamically scale horizontally
- Types of NewSQL systems: New Architectures, Optimized SQL
  Engines, Transparent Sharding
- Can be used (in class) with:
  - **Traditional applications**: university, employee, or inventory databases
  - **Scenarios that require NewSQL capabilities**: stock market, social media
    networks and online games
4.1 Learning SQL with VoltDB

- VoltDB is an Open-source, in-memory and ACID-compliant NewSQL database
- Uses shared nothing architecture, partitioning and replication to reach high transaction throughputs
- Designed for real-time analytics
- Can be used to learn many aspects of SQL: data definition and data manipulation languages
- Resources: VoltDB VM (tools and sample apps), application gallery
- In-class activity: Ad Performance application
- This application simulates a high velocity stream of ad events that are augmented and stored

```
CREATE TABLE event_data (utc_time TIMESTAMP NOT NULL, creative_id INTEGER NOT NULL, cost DECIMAL, campaign_id INTEGER NOT NULL, advertiser_id INTEGER NOT NULL, site_id INTEGER NOT NULL, is_impression INTEGER NOT NULL, is_clickthrough INTEGER NOT NULL, is_conversion INTEGER NOT NULL);

Q1: SELECT site_id, COUNT(*) AS records, SUM(is_impression) AS impressions, SUM(is_clickthrough) AS clicks, SUM(is_conversion) AS conversions, SUM(cost) AS cost FROM event_data GROUP BY site_id;

Q2: SELECT advertiser_id, COUNT(*) AS records, SUM(is_impression) AS impressions, SUM(is_clickthrough) AS clicks, SUM(is_conversion) AS conversions, SUM(cost) AS cost FROM event_data GROUP BY advertiser_id;

Q3: SELECT advertiser_id, TRUNCATE(DAY, utc_time) AS utc_day, COUNT(*) AS records, SUM(is_impression) AS impressions, SUM(is_clickthrough) AS clicks, SUM(is_conversion) AS conversions, SUM(cost) AS cost FROM event_data GROUP BY advertiser_id, TRUNCATE(DAY, utc_time);

Q4: SELECT creative_id AS ad, SUM(cost) AS cost FROM event_data GROUP BY creative_id ORDER BY cost DESC LIMIT 10;
```
5. Final Remarks

• The proposed modules can be integrated into introductory and advanced DB courses
• Big Data systems could be used instead of traditional RBDs in intro DB courses
  ▪ NewSQL could be a good fit since it supports the relational model and SQL
• Big Data systems can also be used to complement traditional RDBs
  ▪ DB fundamentals can be taught using traditional RDBs
  ▪ Application of these principles to modern Big Data systems
• The use of SQL beyond traditional databases can also be taught as part of a course in Big Data
  ▪ Study the use of SQL in MapReduce, NoSQL and NewSQL
  ▪ Compare SQL in Big Data with native query languages
• We have integrated several modules in two DB courses
• Resources (scripts, data generators, datasets, VMs): [http://www.public.asu.edu/~ynsilva/iBigData/](http://www.public.asu.edu/~ynsilva/iBigData/)