Integrating Big Data into the Computing Curricula

Yasin Silva, Suzanne Dietrich, Jason Reed, Lisa Tsosie
Arizona State University

http://www.public.asu.edu/~ynsilva/iBigData/
Overview

• Motivation
• Big Data Learning Units
• Data Processing with MapReduce
• Big Data Analysis with NoSQL
• Big Data Processing with NewSQL
• Discussion and Conclusions
The development of highly distributed and scalable systems to process Big Data is one of the recent key technological developments in CS.

Big Data Management Systems (BDMSs), have been successfully used in many application scenarios:
- Scientific data analysis
- Social media data mining
- Recommendation systems
- Correlation analysis on web service logs, etc.

The importance of BDMSs in the current landscape of DB systems creates the need to integrate their study as part of the computing curricula.

This paper presents multiple guidelines to perform this integration.
Our Contribution

• The identification of three important types of BDMSs and the description of their:
  – Core concepts
  – Taxonomy
  – Sample systems
  – Guidelines about when to use them

• Guidelines to integrate each type of BDMS as a course unit and the description of the learning objectives

• Detailed description of class resources for each unit to enable a hands-on experience with BDMSs
  – Virtual machines
  – Data generators
  – Sample programs, projects, and in-class exercises

• Files of all the resources made available to instructors. The goal is to enable instructors to use and extend these resources based on their specific needs
Big Data Learning Units

- BDMSs are relatively recent but there are many BDMS-related open-source and commercial products
- How to organize these different technologies/products into well-defined learning units?
- We propose the following units:
  - MapReduce
  - NoSQL
  - NewSQL

Learning Outcomes
LO1 Recognize the key properties, strengths and limitations of each type of BDMS
LO2 Build applications using specific instances of each type of BDMS
LO3 Understand when to use each type of BDMS
LU1: The MapReduce Framework

- One of the core programming models for Big Data processing
- Started the wave of BDMS systems
- Enables building distributed programs that run on failure-tolerant and scalable clusters of commodity machines
- Examples: Apache Hadoop, Google MapReduce, and Microsoft's Dryad
- Divides the processing task into two main phases: map and reduce.
- Map and reduce tasks are executed in a distributed and parallel fashion

\[
\text{map: (k1,v1) } \rightarrow \text{ list(k2,v2)} \\
\text{reduce: (k2,list(v2)) } \rightarrow \text{ list(k3,v3)}
\]
MapReduce Learning Resources

- **Hadoop**
  - **Fully distributed mode**: Requires a computer cluster
  - **Pseudo-distributed mode**: Single machine, significant installation/conf. effort
  - **Our approach**: Virtual Machine (VM) with packages installed and configured
    - Linux + Hadoop + Eclipse + Hadoop plug-in + sample programs + datasets

- **Data Generators and Datasets**
  - **MobyShakespeare**: complete works of Shakespeare
  - **MStation**: weather station data (Station ID, temp, year, etc.)

- **MapReduce Program Template**
  - Facilitates the process of writing MapReduce programs in Hadoop
  - Imports required libraries, provides empty definitions of the Map and Reduce methods, and includes a generic version of the run method

- **In-class Exercises**
  - Word Count
  - Average Temp By Station
  - Average Temp By Zip Code
  - Sort By Station ID

- **Class assignment ideas**
Achieving learning objectives- MapReduce

- We found that the use of diagrams is key to help students to understand MapReduce (LO1)
- The use of the described VM provides students with hands-on experience with Hadoop MapReduce (LO2)
  - Students focus on design and implementation rather than installation and tedious configuration
  - Data generator can be modified for additional exercises or projects
- It is important to contrast MapReduce with other data processing systems (LO3)
  - In a database class, MapReduce can be compared with traditional relational databases
  - MapReduce doesn’t make older data processing technologies obsolete. Different systems might be more suitable for different tasks.
    - Database systems could be the right tool to process selling transactions (consistent and durable)
    - MapReduce may be the right tool to analyze and get useful insights from data that contains selling operations of the last 5 years
LU2: NoSQL Data Stores

• Provide higher scalability and availability than conventional relational databases
• Have a distributed, fault-tolerant architecture
• Data is partitioned (performance) and replicated (fault tolerance)
• In general, do not support relational data model and SQL
• Implement a simplified transaction/consistency model (e.g., eventual consistency)
• Types of NoSQL systems:
  – Document stores
    • Store documents that contain data in some format (XML, JSON, binary, etc.)
  – Key-Value stores
    • Store the data in a schema-less way (commonly key-value pairs)
  – Tabular data stores
    • Stores tabular data with potentially many columns and rows
  – Graph databases
    • Stores graph data: social relations, public transport links, road maps, etc.
NoSQL Learning Resources

- **Apache HBase**
  - Open-source NoSQL system included in the class VM
  - Runs on top of HDFS and provides a fault-tolerant way of storing and processing large amounts of sparse tabular data
  - Provides efficient random reads and writes on top of HDFS

- **In-class activity: Maintaining Blog Posts**

  **Using shell commands**
  - Students use the HBase shell interface to run the `create` and `put` commands to create an HBase table and insert the blog posts data
  - Next, students use commands to print all the content of the table, get the content of individual records, and delete records

  **Using Java applications**
  - Students programmatically access the HBase table created in the first part
  - Students establish a connection with the blog posts table, and use instances of the `Put`, `Scan`, and `Get` classes to insert additional rows, list all rows, and retrieve the content of single rows
Achieving Learning Objectives - NoSQL

• The VM/HBase and in-class activities enabled students to:
  – Get concrete understanding of the features of a NoSQL system (LO1)
  – Gain experience about the different ways to interact with HBase (LO2)

• It is important to highlight that NoSQL databases should be considered when:
  – Data is very large
  – No fixed schema
  – Performance more important than consistency

• NoSQL should be contrasted against other types of DB systems (LO3)
  – Traditional relational databases may be the right tool for a banking application
  – HBase (NoSQL) is a proper tool for large messaging systems
  – MapReduce could be the best tool to aggregate the entire dataset

• NoSQL systems differ in how they supports the various levels of consistency (C), availability (A), and network partition tolerance (P)

• CAP Theorem: a system has to pick two of these three properties
LU3: NewSQL Databases

- 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
    - New platforms designed to operate in a distributed cluster of shared-nothing nodes
  - MySQL Engines
    - Highly optimized storage engines for MySQL.
    - Use the same programming interface as MySQL but scale better
  - Transparent Sharding
    - Middleware layer to automatically split databases across multiple nodes
NewSQL Learning Resources

- **VoltDB**
  - Open-source, in-memory and ACID-compliant NewSQL database
  - Uses a shared nothing architecture
  - Supported on Linux and Mac OS X. Client libraries for Java, C#, PHP, etc.

- **VM (available at VoltDB's web site)**
  - VoltDB software and application development tools
  - Sample applications
    - Voting process simulator
    - Key-Value store implementation
    - Use of flexible schema and JSON

- **In-class activity: Voting process simulator**
  - Students execute SQL commands to **create** the database tables
  - Students run an election call center simulator and **monitor** the results
  - Students **write** and run SQL queries
    - **Q1**: compute the total number of votes obtained by each candidate
    - **Q3**: the total number votes in each state
  - Highlight: **high transaction throughput and low latency**

```sql
C1: CREATE TABLE votes (phone_number bigint NOT NULL, state varchar(2) NOT NULL, contestant_number integer NOT NULL);
C2: CREATE VIEW v_votes_by_phone_number (phone_number, num_votes) AS SELECT phone_number, COUNT(*) FROM votes GROUP BY phone_number;
Q1: SELECT contestant_number, COUNT(*) FROM votes GROUP BY contestant_number ORDER BY contestant_number;
Q2: SELECT count(*) FROM votes;
Q3: SELECT phone_number FROM v_votes_by_phone_number WHERE num_votes > 1;
Q4: SELECT state, COUNT(*) cnt FROM votes GROUP BY state ORDER BY cnt DESC;
Q5: SELECT contestant_number, state, COUNT(*) FROM votes GROUP BY contestant_number, state;
```
Similarly to the previous units, the use of the VM enables students to get:

- Hands-on experience with VoltDB (LO2)
- A better understanding of key features of NewSQL systems (LO1), particularly the support of the relational model and SQL

It is also important to describe scenarios where NewSQL is the right type of system to use (LO3)

- NoSQL and NewSQL systems can handle datasets with massive size and high velocity
- NewSQL is particularly suitable for applications that require support of the relational model, ACID guarantees, and SQL
- HBase (NoSQL) can be the right data store for a web-scale blogging system
- VoltDB (NewSQL) can be the right tool to process a rapid stream of stock exchange operations ensuring transactional consistency
Integration into the Computing Curricula

- Big Data modules can be integrated into several CS courses
  - An intro to MapReduce could be integrated into introductory databases or distributed systems courses
    - Incorporate presented in-class exercises and activities
  - A high level exposure to NoSQL/NewSQL could be integrated into an introductory database course
    - Assignment focused on exploring database technologies beyond relational databases
    - In-class discussion about key BDMS’s properties and when they should be used
  - Extensive exposure of the three types of BDMSs can be incorporated in an advanced database class
    - Incorporate presented hands-on activities and projects
  - MapReduce could be integrated into a programming course
    - Focus on presenting the MapReduce programming model as an effective way to distribute computations
Assessment (end-of-class survey)

LO1  *I am able to recognize the key properties, strengths and limitations of MapReduce/NoSQL/NewSQL*

- 50% of students strongly agree and 50% agree

LO3  *I understand when to use MapReduce/NoSQL/NewSQL systems*

- 67% of students strongly agree and 33% agree

LO2  *I am familiar with Hadoop/HBase/VoltDB and can build applications that interact with these systems*

- 42% of students strongly agree, 42% agree, and 16% gave lower scores

Student comments:

- “using the VMs really captured my attention when using hands-on exercises and working directly with the systems”
- “the diagrams really helped visualize relationships and comparisons between different technologies“
- “the hands-on exercises were great”
Conclusions

- Many application scenarios require processing massive datasets in a highly scalable and distributed fashion.
- Different types of BDMSs have been designed to address this challenge.
- Graduating students gain employment in companies that use BDMSs or are eager to use them to get better insights from their data.
- Thus, it is important to integrate the study of these systems as part of the computing curricula.

This paper presents a set of guidelines and a wide array of class resources to integrate the study of three core types of BDMSs: MapReduce, NoSQL, and NewSQL.

- The paper also reports results of integrating the proposed units into a database course.
Questions?

Interest over time

http://www.google.com/trends