PART A. BIG DATA TECHNOLOGY
2. DATA PROCESSING PARADIGMS
FOR BIG DATA

Sangmi Lee Pallickara
Computer Science, Colorado State University
http://www.cs.colostate.edu/~cs535

FAQs

- Slides are available on the course web
- Canvas Discussion Board is available: Find your teammates!
- PA1
  - Hadoop and Spark installation guides are posted
  - Questions/need helps? Send an email to cs535@cs.colostate.edu or post your question on Piazza!
Overview of Part A

• Duration: Week 1 ~ Week 4

1. Introduction to Big Data (W1)
2. Data Process Paradigms for Big Data (W2)
3. Distributed Computing Models for Scalable Batch Computing
   Part 1. MapReduce (W2)
   Part 2. In-Memory Cluster Computing Model: Apache Spark (W3, W4)
4. Real-time Streaming Computing Models (W4)
   Apache Storm and Twitter Heron

2. Data Processing Paradigms For Big data
   Lambda Architecture
This material is built based on


Why do we need Big Data Technologies?

- To perform large-scale analytics over voluminous data, we need a high-level architecture that provides,
  - Robustness
    - Fault-tolerant: Both against hardware failures and human mistakes
  - Support for a wide range of workloads and use cases
    - Low-latency reads and updates
    - Batch analytics jobs
  - Scalability
    - Scale-out capabilities with minimal maintenance
Typical problems for scaling traditional databases

- Suppose that the application should track the **number of page views for any URL a customer wishes to track**
  - The customer’s web page pings the application’s web server with its URL every time a page view is received
  - Application tells you top 100 URLs by number of page views

<table>
<thead>
<tr>
<th>Customer-ID (varchar(255))</th>
<th>url (varchar(255))</th>
<th>Pageviews (bigint)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coloradoan</td>
<td><a href="https://www.coloradoan.com/life">https://www.coloradoan.com/life</a></td>
<td>13,483,401</td>
</tr>
</tbody>
</table>

Direct access

- **Direct access from Web server to the backend DB** cannot handle the large amount of frequent write requests
- Timeout errors

<table>
<thead>
<tr>
<th>Customer-ID (varchar(255))</th>
<th>url (varchar(255))</th>
<th>Pageviews (bigint)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coloradoan</td>
<td><a href="https://www.coloradoan.com/life">https://www.coloradoan.com/life</a></td>
<td>13,483,401</td>
</tr>
</tbody>
</table>
Scaling with a **queue**

- Batch many increments in a single request
- **What if your data amount increases even more?**
  - Your worker cannot keep up with the writes
- **What if you add more workers?**
  - Again, the Database will be overloaded

![Queue Diagram](image)

Scaling by **sharding the database**

- **Horizontal partitioning or sharding of database**
  - Uses multiple database servers and spreads the table across all the servers
  - Chooses the shard for each key by taking the hash of the key modded by the number of shards
- **What if your current number of shards cannot handle your data?**
  - Your mapping script should cope with new set of shards
  - Application and data should be re-organized
Other issues

- Fault-tolerance issues
  - What if one of the database machines is down?
  - A portion of the data is unavailable

- Corruption issues
  - What if your worker code accidentally generated a bug and stored the wrong number for some of the data portions

Reactive Solution

How will Big Data techniques help?

- The databases and computation systems used in Big Data applications are aware of their distributed nature
  - Sharding and replications will be considered as a fundamental component in the design of Big Data systems

- E.g. Data is dealt as immutable
  - Users will mutate data continuously, however
    - The raw pageview information is not modified

- Applications will be designed in different ways
Lambda Architecture

- Big Data data processing architecture as a series of layers
  1. **Batch layer**
  2. **Serving layer**
  3. **Speed layer**

1. **Batch layer**
   - **Precomputes results using distributed processing system**
     - The component that performs the batch view processing
     - batch view= function(e.g. Sum of values)
     - batch view= function(e.g. Training a Predictive Model)
   - After the computation → Stores an immutable, constantly growing master dataset
     - E.g. values, model, or distribution
   - Computes arbitrary functions on that dataset
     - Batch-processing systems
     - e.g. Hadoop, Spark, TensorFlow
Generating batch views

Data → Batch layer → Batch view

Batch layer is often a high-latency operation

Generating batch views: E.g. PageRank

Data → Batch layer → Batch view

Crawling the Web every day

Generate the PageRank values every 24 hours

PageRank values of the web on 01/25/2020
PageRank values of the web on 01/26/2020
PageRank values of the web on 01/27/2020
2. Serving layer

- The batch layer emits **batch view** as the **result of its functions**
  - These views should be loaded somewhere and queried

- Specialized distributed database that loads in a batch view and makes it possible to do **random reads** on it

- **Batch update and random reads should be supported**
  - e.g. BigQuery, ElephantDB, Dynamo, MongoDB, Cassandra

3. Speed layer

- Q: Is there **any data not represented** in the batch view?

- Data arrives **while the precomputation is running**
  - With fully real-time data system

- Speed layer looks only at recent data
  - Whereas the batch layer looks at all the data (except real-time data) at once

  - **Real-time view** = function( **real-time view**, **real-time computing**, **new data**)
How long should the real time view be maintained?

- Once the data arrives at the serving layer, the corresponding results in the real-time views are no longer needed
  - You can discard pieces of the real-time views
Relevance of Data

Data absorbed into batch view

Data absorbed into real-time view

Time

Now

Practical Example with Lambda architecture

- Web analytics application tracking the number of page views over a range of days
  - The speed layer keeps its own separate view of \([\text{url}, \text{day}]\)
    - Updates its views by incrementing the count in the view whenever it receives new data
  - The batch layer recomputes its views by counting the page views
- To resolve the query, you query both the batch and real-time views
  - With satisfying ranges
  - Sum up the results
Exercise

Suppose you are running an analytics system designed based on the Lambda Architecture. The first batch job has started without any dataset and it took 5 minutes to complete. The second batch job is scheduled 5 minutes after the first job has completed. The second batch job is performed with data that arrived in the first 10 minutes.

Question 1. After 7 minutes of running your system, what is the coverage of data stored in the Speed layer?
   a. none    b. 0 ~ 7 minutes    c. 5 ~ 7 minutes    d. 0 ~ 5 minutes

Question 2. After 7 minutes of running your system, what is the coverage of data stored in the Batch layer?
   a. none    b. 0 ~ 7 minutes    c. 5 ~ 7 minutes    d. 0 ~ 5 minutes

Exercise—Answers

Suppose you are running an analytics system designed based on the Lambda Architecture. The first batch job has started without any dataset and it took 5 minutes to complete. The second batch job is scheduled 5 minutes after the first job has completed. The second batch job is performed with data that arrived in the first 10 minutes.

Question 1. After 7 minutes of running your system, what is the coverage of data stored in the Speed layer?
   a. none    b. 0 ~ 7 minutes    c. 5 ~ 7 minutes    d. 0 ~ 5 minutes

Question 2. After 7 minutes of running your system, what is the coverage of data stored in the Batch layer?
   a. none    b. 0 ~ 7 minutes    c. 5 ~ 7 minutes    d. 0 ~ 5 minutes
Composing algorithms

- **The batch/speed layer will split your data**
  - The exact algorithm on the batch layer
  - An approximate algorithm on the speed layer

- The batch layer repeatedly overrides the speed layer
  - The approximation gets corrected
  - Eventual accuracy

Example of Cardinality Estimation

- **Cardinality estimation**
  - **Count-distinct problem**: finding number of distinct elements
  - Counting **exact unique counts** in the batch layer
  - **A Hyper-LogLog** as an approximation in the speed layer
  - Batch layer corrects what’s computed in the speed layer
    - Eventual accuracy
Recent trends in technology

1/27/2020 Week 2-A Sangmi Lee Pallickara

• Physical limits of how fast a single CPU can go
  • Parallelize computation to scale to more data
  • Scale-out solution

• Elastic clouds
  • Infrastructure as a Service (IaaS)
  • Rent hardware on demand rather than owning your hardware
  • Increase and decrease the size of your cluster nearly instantaneously
  • Simplifies system administration

Recent trends in technology (2/3)

• Open source ecosystem for Big Data
  • Batch computation systems
    • Hadoop
    • Spark
    • Flink

• Serialization frameworks
  • Serializes an object into a byte array from any language
  • Deserialize that byte array into an object in any language
  • Thrift, Protocol Buffers, and Avro
Recent trends in technology (3/3)

- Open source ecosystem for Big Data - cont.
  - Random-access NoSQL databases
    - Sacrifice the full expressiveness of SQL
    - Specializes in certain kinds of operations
    - Cassandra, Hbase, MongoDB, etc.

- Messaging/queuing systems
  - Sends and consumes messages between processes in a fault-tolerant manner
  - Apache Kafka

- Real-time computation system
  - High throughput, low latency, stream-processing systems
  - Apache Storm

Mapping recent technologies
Mapping course components

Kappa architecture

Jay Kreps, LinkedIn
Kappa architecture is a simplified version of lambda architecture
Append only immutable log
Brainstorming Quiz

- Lambda architecture vs. Kappa architecture

  Question 1. What is the difference between Lambda architecture and Kappa architecture?
  Question 2. Use case of Lambda architecture?
  Question 3. Use case of Kappa architecture?

Questions?