CS 4604: Introduction to Database Management Systems

B. Aditya Prakash

Lecture #12: NoSQL and MapReduce
(some slides from Xiao Yu)

NO SQL
Why No SQL?

**HOW TO WRITE A CV**

1. Do you have any expertise in SQL?
2. No
4. Leverage the NoSQL boom
RDBMS

- The predominant choice in storing data
  - Not so true for data miners since we much in txt files.
- First formulated in 1969 by Codd
  - We are using RDBMS everywhere
Aside: RDBMS performance

Performance

Salary List

Majority of Webapps

Social network

Semantic Trading

custom

Data complexity

Relational database

I DON'T ALWAYS USE RDBMS

BUT WHEN I DO, I DUMP EVERYTHING IN IT

Prakash 2018
When RDBMS met Web 2.0

Big data
Connectivity
P2P Knowledge

Concurrency
Diversity
Cloud-Grid

Slide from Lorenzo Alberton, "NoSQL Databases: Why, what and when"
What to do if data is really large?

- Peta-bytes (exabytes, zettabytes.....)

- Google processed 24 PB of data per day (2009)

- FB adds 0.5 PB per day
BIG data

These are just some of the more common ways that Internet users add to the big data pool. In truth, depending on the niche of business you’re in, there are virtually countless other sources of relevant data to pay attention to. Consider the following:

The global Internet population grew 0.55 percent from 2010 to 2011 and now represents 2.1 Billion people.

These users are real, and they are out there leaving data trails everywhere they go. The team at Domo can help you make sense of this seemingly insurmountable heap of data, with solutions that help executives and managers bring all of their critical information together in one intuitive interface, and then use that insight to transform the way they run their business. To learn more, visit www.romo.com.

What’s Wrong with Relational DB?

- Nothing is wrong. You just need to use the right tool.
- Relational is hard to scale.
  - Easy to scale reads
  - Hard to scale writes
What’s NoSQL?

- The misleading term “NoSQL” is short for “Not Only SQL”.
- non-relational, schema-free, non-(quite)-ACID
  - More on ACID transactions later in class
- horizontally scalable, distributed, easy replication support
- simple API
Four (emerging) NoSQL Categories

- **Key-value (K-V) stores**
  - Based on Distributed Hash Tables/ Amazon’s Dynamo paper *
  - Data model: (global) collection of K-V pairs
  - Example: Voldemort

- **Column Families**
  - BigTable clones **
  - Data model: big table, column families
  - Example: HBase, Cassandra, Hypertable

---

*G DeCandia et al, Dynamo: Amazon's Highly Available Key-value Store, SOSP 07
** F Chang et al, Bigtable: A Distributed Storage System for Structured Data, OSDI 06
Four (emerging) NoSQL Categories

- **Document databases**
  - Inspired by Lotus Notes
  - Data model: collections of K-V Collections
  - Example: CouchDB, MongoDB

- **Graph databases**
  - Inspired by Euler & graph theory
  - Data model: nodes, relations, K-V on both
  - Example: AllegroGraph, VertexDB, Neo4j
Focus of Different Data Models

Slide from neo technology, “A NoSQL Overview and the Benefits of Graph Databases"
C-A-P “theorem"
When to use NoSQL?

- Bigness
- Massive write performance
  - Twitter generates 7TB / per day (2010)
- Fast key-value access
- Flexible schema or data types
- Schema migration
- Write availability
  - Writes need to succeed no matter what (CAP, partitioning)
- Easier maintainability, administration and operations
- No single point of failure
- Generally available parallel computing
- Programmer ease of use
- Use the right data model for the right problem
- Avoid hitting the wall
- Distributed systems support
- Tunable CAP tradeoffs

from http://highscalability.com/
Key-Value Stores

<table>
<thead>
<tr>
<th>id</th>
<th>hair_color</th>
<th>age</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1923</td>
<td>Red</td>
<td>18</td>
<td>6’0”</td>
</tr>
<tr>
<td>3371</td>
<td>Blue</td>
<td>34</td>
<td>NA</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Table in relational db

Store/Domain in Key-Value db

user1923_color  Red
user1923_age    18
user3371_color  Blue
user4344_color  Brackish
user1923_height 6' 0"
user3371_age    34

Find users whose age is above 18?
Find all attributes of user 1923?
Find users whose hair color is Red and age is 19?
(Join operation) Calculate average age of all grad students?
Voldemort in LinkedIn

People You May Know

Viewers of this profile also viewed

Related Searches

Related searches for hadoop
mapreduce java
big data hbase
machine learning lucene
data mining data warehouse

Events you may be interested in

LinkedIn Skills

Jobs you may be interested in

Sid Anand, LinkedIn Data Infrastructure (QCon London 2012)
Voldemort vs MySQL

Sid Anand, LinkedIn Data Infrastructure (QCon London 2012)
Sparse, distributed, persistent multi-dimensional sorted map indexed by \((row\_key, column\_key, timestamp)\)
The row name is a reversed URL. The contents column family contains the page contents, and the anchor column family contains the text of any anchors that reference the page.
BigTable Performance

Values read/written per second vs. Number of tablet servers

- Scans
- Random reads (mem)
- Random writes
- Sequential reads
- Sequential writes
- Random reads
Document Database - mongoDB

Table in relational db

<table>
<thead>
<tr>
<th>Last Name</th>
<th>First Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUMONT</td>
<td>Jean</td>
<td>43</td>
</tr>
<tr>
<td>PELLERIN</td>
<td>Franck</td>
<td>29</td>
</tr>
<tr>
<td>MATTHIEU</td>
<td>Nicolas</td>
<td>51</td>
</tr>
</tbody>
</table>

Documents in a collection

Open source, document db
Json-like document with dynamic schema

Initial release 2009
mongoDB Product Deployment

And much more...
Graph Database

Data Model Abstraction:
- Nodes
- Relations
- Properties
// Get factory

NeoService neo = ... // Get factory

// Create Thomas 'Neo' Anderson
Node mrAnderson = neo.createNode();
mrAnderson.setProperty( "name", "Thomas Anderson" );
mrAnderson.setProperty( "age", 29 );

// Create Morpheus
Node morpheus = neo.createNode();
morpheus.setProperty( "name", "Morpheus" );
morpheus.setProperty( "rank", "Captain" );
morpheus.setProperty( "occupation", "Total bad ass" );

// Create a relationship representing that they know each other
mrAnderson.createRelationshipTo( morpheus, RelTypes.KNOWS );
// ...create Trinity, Cypher, Agent Smith, Architect similarly
A Debatable Performance Evaluation

Got neo4j to do a do a lookup in 2 seconds, that sql server did in 45 minutes. neo4j rocks!

6:28 AM Jun 30th from web

o_O turboCodr
John Conwell
Conclusion

- Use the right data model for the right problem
THE HADOOP ECOSYSTEM
Single vs Cluster

- 4TB HDDs are coming out
- Cluster?
  - How many machines?
  - Handle machine and drive failure
  - Need redundancy, backup..

3% of 100K HDDs fail in <= 3 months

Hadoop

- Open source software
  - Reliable, scalable, distributed computing
- Can handle thousands of machines
- Written in JAVA
- A simple programming model
- HDFS (Hadoop Distributed File System)
  - Fault tolerant (can recover from failures)
Idea and Solution

- **Issue:** Copying data over a network takes time
- **Idea:**
  - Bring computation close to the data
  - Store files multiple times for reliability
- **Map-reduce addresses these problems**
  - Google’s computational/data manipulation model
  - Elegant way to work with big data
  - **Storage Infrastructure – File system**
    - Google: GFS. Hadoop: HDFS
  - **Programming model**
    - Map-Reduce

Prakash 2018
Map-Reduce [Dean and Ghemawat 2004]

- Abstraction for simple computing
  - Hides details of parallelization, fault-tolerance, data-balancing

- MUST Read!

Hadoop VS NoSQL

- Hadoop: computing framework
  - Supports data-intensive applications
  - Includes MapReduce, HDFS etc.
    (we will study MR mainly next)

- NoSQL: Not only SQL databases
  - Can be built ON hadoop. E.g. HBase.
Storage Infrastructure

- **Problem:**
  - If nodes fail, how to store data persistently?

- **Answer:**
  - **Distributed File System:**
    - Provides global file namespace
    - Google GFS; Hadoop HDFS;

- **Typical usage pattern**
  - Huge files (100s of GB to TB)
  - Data is rarely updated in place
  - Reads and appends are common
Distributed File System

- **Chunk servers**
  - File is split into contiguous chunks
  - Typically each chunk is 16-64MB
  - Each chunk replicated (usually 2x or 3x)
  - Try to keep replicas in different racks

- **Master node**
  - a.k.a. Name Node in Hadoop’s HDFS
  - Stores metadata about where files are stored
  - Might be replicated

- **Client library for file access**
  - Talks to master to find chunk servers
  - Connects directly to chunk servers to access data
Programming Model: MapReduce

Warm-up task:

- We have a huge text document
- Count the number of times each distinct word appears in the file

Sample application:

- Analyze web server logs to find popular URLs
Task: Word Count

Case 1:
  – File too large for memory, but all <word, count> pairs fit in memory

Case 2:
  ▪ Count occurrences of words:
    – `words(doc.txt) | sort | uniq -c`
      • where `words` takes a file and outputs the words in it, one per a line
  ▪ Case 2 captures the essence of **MapReduce**
    – Great thing is that it is naturally parallelizable
MapReduce: Overview

- Sequentially read a lot of data
- **Map:**
  - Extract something you care about
- **Group by key:** Sort and Shuffle
- **Reduce:**
  - Aggregate, summarize, filter or transform
- Write the result

Outline stays the same, Map and Reduce change to fit the problem
MapReduce: The Map Step

**Input key-value pairs**

- \( k \) → \( v \) → map
- \( k \) → \( v \) → map
- ... 
- \( k \) → \( v \)

**Intermediate key-value pairs**

- \( k \) → \( v \)
- \( k \) → \( v \)
- ... 
- \( k \) → \( v \)
MapReduce: The Reduce Step

Intermediate key-value pairs

Key-value groups

Output key-value pairs

Group by key

reduce

reduce

reduce

Prakash 2018

VT CS 4604
More Specifically

- **Input:** a set of key-value pairs

- Programmer specifies two methods:
  - **Map** ($k, v$) $\rightarrow <k’, v’>^*$
    - Takes a key-value pair and outputs a set of key-value pairs
      - E.g., key is the filename, value is a single line in the file
    - There is one Map call for every $(k, v)$ pair
  - **Reduce** ($k’, <v’>^*$) $\rightarrow <k’, v”>^*$
    - All values $v’$ with same key $k’$ are reduced together and processed in $v’$ order
    - There is one Reduce function call per unique key $k’$
The crew of the space shuttle Endeavor recently returned to Earth as ambassadors, harbingers of a new era of space exploration. Scientists at NASA are saying that the recent assembly of the Dextre bot is the first step in a long-term space-based man/machine partnership. “The work we’re doing now -- the robotics we’re doing -- is what we’re going to need ….”
Word Count Using MapReduce

map(key, value):
   // key: document name; value: text of the document
   for each word w in value:
       emit(w, 1)

reduce(key, values):
   // key: a word; value: an iterator over counts
   result = 0
   for each count v in values:
       result += v
   emit(key, result)
Map-Reduce (MR) as SQL

- select `count(*)` Reducer
  from DOCUMENT
  group by word

Mapper
Map-Reduce: Environment

Map-Reduce environment takes care of:

- Partitioning the input data
- Scheduling the program’s execution across a set of machines
- Performing the group by key step
- Handling machine failures
- Managing required inter-machine communication
Map-Reduce: A diagram

**Map:**
Read input and produces a set of key-value pairs

**Intermediate:**
- k1:v, k1:v, k2:v
- k1:v
- k3:v, k4:v
- k4:v, k5:v
- k4:v
- k1:v, k3:v

**Group by key:**
Collect all pairs with same key
(Hash merge, Shuffle, Sort, Partition)

**Grouped:**
- k1:v, v, v, v
- k2:v
- k3:v, v
- k4:v, v, v
- k5:v

**Reduce:**
Collect all values belonging to the key and output

**Output:**
Map-Reduce: In Parallel

All phases are distributed with many tasks doing the work

Map Task 1

Map Task 2

Map Task 3

Sort and Group

Reduce Task 1

Sort and Group

Reduce Task 2

Prakash 2018
Map-Reduce

- **Programmer specifies:**
  - Map and Reduce and input files

- **Workflow:**
  - Read inputs as a set of key-value-pairs
  - **Map** transforms input kv-pairs into a new set of k'v'-pairs
  - Sorts & Shuffles the k'v'-pairs to output nodes
  - All k’v’-pairs with a given k’ are sent to the same **reduce**
  - **Reduce** processes all k'v'-pairs grouped by key into new k''v''-pairs
  - Write the resulting pairs to files

- All phases are distributed with many tasks doing the work
Data Flow

- **Input and final output** are stored on a **distributed file system (FS)**:
  - Scheduler tries to schedule map tasks “close” to physical storage location of input data

- **Intermediate results** are stored on **local FS of Map and Reduce workers**

- **Output is often input to another MapReduce task**
Coordination: Master

- **Master node takes care of coordination:**
  - **Task status:** (idle, in-progress, completed)
  - **Idle tasks** get scheduled as workers become available
  - When a map task completes, it sends the master the location and sizes of its \( R \) intermediate files, one for each reducer
  - Master pushes this info to reducers

- **Master pings workers periodically to detect failures**
Dealing with Failures

- **Map worker failure**
  - Map tasks completed or in-progress at worker are reset to idle
  - Reduce workers are notified when task is rescheduled on another worker

- **Reduce worker failure**
  - Only in-progress tasks are reset to idle
  - Reduce task is restarted

- **Master failure**
  - MapReduce task is aborted and client is notified
PROBLEMS SUITED FOR MAP-REDUCE
Example: Host size

- Suppose we have a large web corpus
- Look at the metadata file
  - Lines of the form: (URL, size, date, ...)
- For each host, find the total number of bytes
  - That is, the sum of the page sizes for all URLs from that particular host

- Other examples:
  - Link analysis and graph processing
  - Machine Learning algorithms
Example: Language Model

- **Statistical machine translation:**
  - Need to count number of times every 5-word sequence occurs in a large corpus of documents

- **Very easy with MapReduce:**
  - **Map:**
    - Extract (5-word sequence, count) from document
  - **Reduce:**
    - Combine the counts
In HW5

- You’ll deal with n-grams
  - n-gram is a contiguous sequence of n items from a given sequence of text or speech

- Example

- Sentence: “the rain in Spain falls mainly on the plain”
  - 2 grams: the rain, rain in, in Spain, Spain falls, etc.
  - 3 grams: the rain in, rain in Spain, in Spain falls, etc.
In HW5

- You will work with the Google 4-gram corpus. Example:
  - analysis is often described 1991 10 1 1
  - analysis is often described 1992 30 2 1

- We will ask you to
  - Find total occurrence counts (this will be similar to just word count)
    • in the example above “analysis is often described” occurs total of 10+30 = 40 times.
  - Convert 4-grams to 2-grams (think what should be the mapper and reducer for this)
    • Example: “analysis is often described” will give rise to the following 2 grams: analysis is, is often, often described
Degree of graph Example

- Find degree of every node in a graph

Example: In a friendship graph, what is the number of friends of every person:

Node 6 = 1  Node 2 = 3
Node 4 = 3  Node 1 = 2
Node 3 = 2  Node 5 = 3
Degree of each node in a graph

- Suppose you have the edge list

```
6 4
4 6
4 3
3 4
4 5
5 4
...  
```

== a table!

Schema?

Edges(from, to)
Degree of each node in a graph

- Suppose you have the edge list

```plaintext
6 4
4 6
4 3
3 4
4 5
5 4
...
```

== a table!

Schema?

Edges(from, to)

SQL for degree list?

```sql
SELECT from, count(*)
FROM Edges
GROUP BY from
```
Degree of each node in a graph

- So in SQL:
  
  ```sql
  SELECT from, count(*)
  FROM Edges
  GROUP BY from
  ```

- MapReduce?
  
  **Mapper:**
  emit (from, 1)

  **Reducer:**
  emit (from, count())

  - Remember

  ![Map-Reduce (MR) as SQL](image)

  **I.E. essentially equivalent to the ‘word-count’ example 😊**
Conclusions

- Hadoop is a distributed data-intensive computing framework
- MapReduce
  - Simple programming paradigm
  - Surprisingly powerful (may not be suitable for all tasks though)
- Hadoop has specialized FileSystem, Master-Slave Architecture to scale-up
NoSQL and Hadoop

- Hot area with several new problems
  - Good for academic research
  - Good for industry

= Fun AND Profit 😊
POINTERS AND FURTHER READING
Implementations

- **Google**
  - Not available outside Google

- **Hadoop**
  - An open-source implementation in Java
  - Uses HDFS for stable storage

- **Aster Data**
  - Cluster-optimized SQL Database that also implements MapReduce
Cloud Computing

- Ability to rent computing by the hour
  - Additional services e.g., persistent storage

- Amazon’s “Elastic Compute Cloud” (EC2)

- Aster Data and Hadoop can both be run on EC2
Reading

- Jeffrey Dean and Sanjay Ghemawat: MapReduce: Simplified Data Processing on Large Clusters

- Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung: The Google File System
Resources

- **Hadoop Wiki**
  - Introduction
  - Getting Started
  - Map/Reduce Overview
    - [http://wiki.apache.org/lucene-hadoop/HadoopMapReduce](http://wiki.apache.org/lucene-hadoop/HadoopMapReduce)
    - [http://wiki.apache.org/lucene-hadoop/HadoopMapRedClasses](http://wiki.apache.org/lucene-hadoop/HadoopMapRedClasses)
  - Eclipse Environment

- **Javadoc**
  - [http://lucene.apache.org/hadoop/docs/api/](http://lucene.apache.org/hadoop/docs/api/)
Resources

- Releases from Apache download mirrors

- Nightly builds of source

- Source code from subversion
  - [http://lucene.apache.org/hadoop/version_control.html](http://lucene.apache.org/hadoop/version_control.html)
Further Reading

- Programming model inspired by functional language primitives
- Partitioning/shuffling similar to many large-scale sorting systems
  - NOW-Sort ['97]
- Re-execution for fault tolerance
  - BAD-FS ['04] and TACC ['97]
- Locality optimization has parallels with Active Disks/Diamond work
  - Active Disks ['01], Diamond ['04]
- Backup tasks similar to Eager Scheduling in Charlotte system
  - Charlotte ['96]
- Dynamic load balancing solves similar problem as River's distributed queues
  - River ['99]