Carnegie Mellon Univ.
Dept. of Computer Science
15-415/615 - DB Applications

C. Faloutsos – A. Pavlo
Lecture#28: Modern Database Systems

Administrivia – Final Exam

- **Who:** You
- **What:** R&G Chapters 15-22
- **When:** Tuesday May 6th 5:30pm-8:30pm
- **Where:** WEH 7500
- **Why:** Databases will help your love life.

Handwritten Notes
Printed Notes
Today’s Class

• Distributed OLAP
• OldSQL vs. NoSQL vs. NewSQL
• How to scale a database system

OLTP vs. OLAP

• On-line Transaction Processing:
  – Short-lived txns.
  – Small footprint.
  – Repetitive operations.
• On-line Analytical Processing:
  – Long running queries.
  – Complex joins.
  – Exploratory queries.

Workload Characterization

Michael Stonebraker — “Ten Rules For Scalable Performance In Simple Operation" DeIntersect
http://cacm.acm.org/magazines/2011/6/108651

Social Networks

OLAP

OLTP

Simple

Complex

Reads

Writes
Relational Database Backlash

- New Internet start-ups hit the limits of single-node DBMSs.
- Early companies used custom middleware to shard databases across multiple DBMSs.
- Google was a pioneer in developing non-relational DBMS architectures.

MapReduce

- Simplified parallel computing paradigm for large-scale data analysis.
- Originally proposed by Google in 2004.
- Hadoop is the current leading open-source implementation.

MapReduce Example

Calculate total order amount per day after Jan 1st.

```
MAP (key, value) {
  if (key >= "2009-01-01") {
    output(key, value);
  }
}

REDUCE (key, values) {
  sum = 0;
  while (values.hasNext()) {
    sum += values.next();
  }
  output(key, sum);
}
```
What MapReduce Does Right

- Since all intermediate results are written to HDFS, if one node crashes the entire query does not need to be restarted.
- Easy to load data and start running queries.
- Great for semi-structured data sets.

What MapReduce Did Wrong

- Have to parse/cast values every time:
  - Multi-attribute values handled by user code.
  - If data format changes, code must change.
- Expensive execution:
  - Have to send data to executors.
  - A simple join requires multiple MR jobs.

Join Example

- Find sourceIP that generated most adRevenue along with its average pageRank.
Join Example – SQL

```
SELECT INTO Temp sourceIP,
        AVG(pageRank) AS avgPageRank,
        SUM(adRevenue) AS totalRevenue
FROM Rankings AS R,
     UserVisits AS UV
WHERE R.pageURL = UV.destURL
    AND UV.visitDate BETWEEN "2000-01-15" AND "2000-01-22"
GROUP BY UV.sourceIP;

SELECT sourceIP, totalRevenue, avgPageRank
FROM Temp
ORDER BY totalRevenue DESC
LIMIT 1;
```

Join Example – MapReduce

<table>
<thead>
<tr>
<th>Phase 1: Filter</th>
<th>Phase 2: Aggregation</th>
<th>Phase 3: Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map: Emit all records for Rankings. Filter UserVisits data.</td>
<td>Map: Emit all tuples (i.e., passthrough) Reduce: Compute avg pageRank for each sourceIP.</td>
<td>Map: Emit all tuples (i.e., passthrough) Reduce: Scan entire input and emit the record with greatest adRevenue sum.</td>
</tr>
</tbody>
</table>

Join Example – Results

- Find sourceIP that generated most adRevenue along with its average pageRank.
Distributed Joins Are Hard

SELECT * FROM table1, table2
WHERE table1.val = table2.val

- Assume tables are horizontally partitioned:
  - Table1 Partition Key → table1.key
  - Table2 Partition Key → table2.key
- Q: How to execute?
- Naïve solution is to send all partitions to a single node and compute join.

Semi-Joins

- First distribute the join attributes between nodes and then recreate the full tuples in the final output.
  - Send just enough data from each table to compute which rows to include in output.
- Lots of choices make this problem hard:
  - What to materialize?
  - Which table to send?

MapReduce in 2014

- SQL/Declarative Query Support
- Table Schemas
  - Column-oriented storage.
Column Stores

- Store tables as sections of columns of data rather than as rows of data.

---

\[
\text{SELECT sex, AVG(GPA) FROM student GROUP BY sex}
\]

---

Column-oriented Storage

---

Row-oriented Storage

---

Column-oriented Storage
Column Stores

- Only scan the columns that a query needs.
- Allows for amazing compression ratios:
  - Values for the same query are usually similar.
- Main goal is delay materializing a record back to its row-oriented format for as long as possible inside of the DBMS.
- Inserts/Updates/Deletes are harder…

Column Store Systems

- Many column-store DBMSs
  - Examples: Vertica, Sybase IQ, MonetDB
- Hadoop storage library:
  - Example: Parquet, RCFile

NoSQL

- In addition to MapReduce, Google created a distributed DBMS called BigTable.
  - It used a GET/PUT API instead of SQL.
  - No support for txns.
- Newer systems have been created that follow BigTable’s anti-relational spirit.
NoSQL Systems

Key/Value
- redis
- riak

Column-Family
- cassandra
- HBase

Documents
- MongoDB
- CouchDB

NoSQL Drawbacks

- Developers write code to handle eventually consistent data, lack of transactions, and joins.
- Not all applications can give up strong transactional semantics.

NewSQL

- Next generation of relational DBMSs that can scale like a NoSQL system but without giving up SQL or txns.
Aslett White Paper

[Systems that] deliver the scalability and flexibility promised by NoSQL while retaining the support for SQL queries and/or ACID, or to improve performance for appropriate workloads.

Wikipedia Article

A class of modern relational database systems that provide the same scalable performance of NoSQL systems for OLTP workloads while still maintaining the ACID guarantees of a traditional database system.

NewSQL Systems

- New Design
- MySQL Engines
- Middleware
NewSQL Implementations

- Distributed Concurrency Control
- Main Memory Storage
- Hybrid Architectures
  - Support OLTP and OLAP in single DBMS.
- Query Code Compilation