CS122D:
Beyond SQL Data Management
—Lecture #19 —

Mike Carey
UC Irvine
mjcarey@ics.uci.edu
Announcements

• You are within view of the finish line...!

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>W 5/19</td>
<td>Big Data Analytics: Google, MapReduce, HDFS</td>
<td>Big Data Platforms paper (skim)</td>
</tr>
<tr>
<td>M 5/24</td>
<td>Big Data Analytics: Spark &amp; SparkSQL</td>
<td>Spark Overview paper (skim)</td>
</tr>
<tr>
<td>W 5/26</td>
<td>Big Data Analytics: Spark &amp; DataFrames</td>
<td>Databricks and Spark materials (as needed)</td>
</tr>
<tr>
<td>W 6/02</td>
<td>Data Stream Systems: Spark Structured Streaming</td>
<td>Databricks and Spark materials (as needed)</td>
</tr>
<tr>
<td></td>
<td>Overflow</td>
<td>Column stores, search, message streams, timeseries/IoT, ...</td>
</tr>
<tr>
<td>M 6/07</td>
<td><strong>Final Exam (Cumulative)</strong></td>
<td>4:00-6:00 PM -- be there!!!</td>
</tr>
</tbody>
</table>

• HW #6 is due on Monday

| HW6   | Th 5/27 | Mo 6/07 (11:59 PM) | Spark | HW6 Setup | HW6 Details Template | HW6 Solution |

• Two more quizzes! (Spark, Course Feedback)

• Final exam is **Monday 4-6pm**! (Online, but I would highly recommend preparing a cheat sheet, as it will be a timed exam.)

• **Today:** *What we covered (and not 😊)!*
A Typical Big Data Software Stack

Data Lakes

A collection of multi-modal data stored in their raw formats

Data Lake vs. Data Warehouse

**Data Lake**
- Shorter development process
- Schema-on-read
- Multiworkload processing
- Cost-effective architecture

**Data Warehouse**
- Long development process
- Schema-on-write
- OLAP workloads
- Complex development with ETL

© 2020, M.T. Özsu & P. Valduriez
Column Store DB Systems

• Another flavor of RDBMS
  • Focused on analytical use cases
  • Traditionally found in Data Warehouses today
  • Often coupled with in-memory assumption

Why Use a Column Store?

**Row-store**

<table>
<thead>
<tr>
<th>Date</th>
<th>Store</th>
<th>Product</th>
<th>Customer</th>
<th>Price</th>
</tr>
</thead>
</table>

+ Easy to insert/update records
+ Easy to deliver records to apps
- May read in unneeded data
- Not cache-friendly

**Column-store**

<table>
<thead>
<tr>
<th>Date</th>
<th>Store</th>
<th>Product</th>
<th>Customer</th>
<th>Price</th>
</tr>
</thead>
</table>

- An insert requires many writes
- Expensive to access whole records
+ Only read in necessary data
+ Cache-friendly and compressible

→ Suitable for read-mostly, read-intensive, large data repositories
Search Systems

• Focus is on indexing large collections of documents that include significant *textual* content

• **Elasticsearch** is the big gorilla in this space
Elasticsearch

• Elasticsearch index
  • Collection of related *documents* (JSON)
  • Document is collection of keys (field names) and values (strings, numbers, Booleans, dates, arrays, geolocations)

• Based on *inverted index* data structure
  • Indexes all words appearing in any document and identifies documents that contain them (inverted list)
  • Designed to support fast full-text searches

• Scales out across a cluster (with Apache Lucene inside)
  • Documents hash-partitioned to nodes of cluster
  • Searches routed to all nodes (to search in parallel)

• REST-based API for searching and other operations
  • Searches (boolean, relevance-ranked, parametric, mix)
Distributed Streaming

• Three key capabilities
  • Enable apps to publish and subscribe to streams of records
  • Store streams of records in a fault-tolerant durable way
  • Process streams of records as they occur

• Core Kafka concepts
  • Cluster of one or more servers that can span datacenters
  • Stores streams of records in categories called topics
  • A record consists of a key, a value, and a timestamp
Kafka (cont.)

- Core APIs
  - *Producer* – Allows an app to publish a stream of records to one or more Kafka topics
  - *Consumer* – Allows an app to subscribe to one or more topics and process the stream of records produced to them
  - *Streams* – Allows an app to act as a stream processor (transforming data from input topics to output topics)
  - *Connector* – Supports creation of reusable producers or consumers for existing apps or data systems

- Supports both *queuing* and *pub-sub* messaging models
  - *Queuing* – A pool of consumers read from a queue, with each record going to one of the consumers
  - *Pub-Sub* – Records are broadcast to all subscribing consumers for a given topic
  - The Kafka model is a generalization, and scales to a cluster by supporting partitions within topics
Time Series Data Management
InfluxDB

• What is time series data?
  • Measurements or events that are tracked, monitored, downsampled, and aggregated over time
  • *Ex*: server metrics, app perf metrics, network data, sensor data, events, clicks, trades, ...
  • Always asking questions about it *over time*

• Comes in two forms
  • Regular – Measurements come at regular time intervals
  • Irregular – Events driven by users or other external events

• InfluxDB “line protocol”
  • `<measurement name>,<tag set> <field set> <timestamp>`
  • *Ex*: `cpu,host=serverA,region=uswest idle=23, user=42,system=12 1549063516`

• On disk, InfluxDB’s data org is columnar
  • Contiguous blocks of time set for measurement, tagset, fieldset
  • Makes single-field, time-based aggregation very fast
Last But Not Least: AsterixDB

**Semistructured Data Management**

**BDMS Desiderata:**
- Flexible data model
- Efficient runtime
- Full query capability
- Cost proportional to task at hand (!)
- Designed for continuous data ingestion
- Support today’s “Big Data data types”

---

M. Carey, Spring 2021: CS122D
AsterixDB’s Open Data Model

```
create dataverse TinySocial;
use dataverse TinySocial;

create type MugshotUserType as {
  id: int32,
  alias: string,
  name: string,
  userSince: datetime,
  address: {
    street: string,
    city: string,
    state: string,
    zip: string,
    country: string
  },
  friendIds: {{ int32 }},
  employment: [EmploymentType]
};
create type EmploymentType as open {
  organizationName: string,
  startDate: date,
  endDate: date?
};
create dataset MugshotUsers(MugshotUserType)
  primary key id;
```

**Highlights include:**
- JSON++ based data model
- Rich type support (spatial, temporal, ...)
- Objects, arrays, multisets
- Open vs. closed types
AsterixDB’s Open Data Model (cont.)

```sql
create dataverse TinySocial;
use dataverse TinySocial;

create type MugshotUserType as {
    id: int32
};

create type EmploymentType as open {
    organizationName: string,
    startDate: date,
    endDate: date?
};

create dataset MugshotUsers(MugshotUserType)
    primary key id;
```

**Highlights include:**
- JSON++ based data model
- Rich type support (spatial, temporal, …)
- Objects, arrays, multisets
- **Open vs. closed types**
AsterixDB’s Open Data Model (cont.)

create dataverse TinySocial;
use dataverse TinySocial;

create type MugshotUserType as {
    id: int32
};

create type MugshotMessageType as closed {
    messageId: int32,
    authorId: int32,
    timestamp: datetime,
    inResponseTo: int32?,
    senderLocation: point?,
    tags: {{ string }},
    message: string
};

create dataset MugshotUsers(MugshotUserType) primary key id;
create dataset
    MugshotMessages(MugshotMessageType) primary key messageId;

Highlights include:
• JSON++ based data model
• Rich type support (spatial, temporal, ...)
• Objects, arrays, multisets
• Open vs. closed types
Enough Already!
## Beyond SQL Data Management

### Lecture Plan

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Relevant Material</th>
</tr>
</thead>
<tbody>
<tr>
<td>M 3/29</td>
<td>Post-relational “escape attempts”</td>
<td>Ch. 1-2 NoSQL Distilled, Introducing JSON website</td>
</tr>
<tr>
<td>W 3/31</td>
<td>Relational databases and SQL (and “beyond”)</td>
<td>SQL chapters of any DB textbook, Advanced Aggregation excerpt</td>
</tr>
<tr>
<td>M 4/05</td>
<td>Logical DB design and E-R modeling</td>
<td>E-R chapter of any DB textbook</td>
</tr>
<tr>
<td>W 4/07</td>
<td>Scaling RDBMSs through parallelism</td>
<td>Parallel RDBMS paper</td>
</tr>
<tr>
<td>M 4/12</td>
<td>Key-value stores: Architecture &amp; consistency</td>
<td>Baseball paper, Ch. 4-6, 8 NoSQL Distilled, Abadi paper</td>
</tr>
<tr>
<td>W 4/14</td>
<td>Column-family stores: BigTable, Cassandra</td>
<td>Ch. 10 NoSQL Distilled (old!)</td>
</tr>
<tr>
<td>M 4/19</td>
<td>Column-family stores: Cassandra (cont.)</td>
<td>Cassandra materials (as needed)</td>
</tr>
<tr>
<td>W 4/21</td>
<td>Document stores: JSON and MongoDB</td>
<td>Ch. 9 NoSQL Distilled (old!)</td>
</tr>
<tr>
<td>M 4/26</td>
<td>Document stores: MongoDB (cont.)</td>
<td>MongoDB materials (as needed)</td>
</tr>
<tr>
<td>W 4/28</td>
<td>NoSQL DB design principles</td>
<td>Ch. 3 NoSQL Distilled</td>
</tr>
<tr>
<td>M 5/03</td>
<td><strong>Midterm Exam (Checkpoint)</strong></td>
<td>3:30–4:50 PM -- be there!!!</td>
</tr>
<tr>
<td>W 5/05</td>
<td>Document DBs: Couchbase Server &amp; N1QL</td>
<td>CB Analytics paper</td>
</tr>
<tr>
<td>M 5/10</td>
<td>Document DBs: Couchbase Server (cont.)</td>
<td>CB materials (as needed)</td>
</tr>
<tr>
<td>W 5/12</td>
<td>Graph DBs: Graph modeling &amp; Neo4J</td>
<td>Ch. 11 NoSQL Distilled (old!)</td>
</tr>
<tr>
<td>M 5/17</td>
<td>Graph DBs: Neo4J (cont.)</td>
<td>Neo4J materials (as needed)</td>
</tr>
<tr>
<td>W 5/19</td>
<td>Big Data Analytics: Google, MapReduce, HDFS</td>
<td>Big Data Platforms paper (skim)</td>
</tr>
<tr>
<td>M 5/24</td>
<td>Big Data Analytics: Spark &amp; SparkSQL</td>
<td>Spark Overview paper (skim)</td>
</tr>
<tr>
<td>W 5/26</td>
<td>Big Data Analytics: Spark &amp; DataFrames</td>
<td>Databricks and Spark materials (as needed)</td>
</tr>
<tr>
<td>W 6/02</td>
<td>Data Stream Systems: Spark Structured Streaming</td>
<td>Databricks and Spark materials (as needed)</td>
</tr>
<tr>
<td>Overflow</td>
<td>Column stores, search, message streams, timeseries/IoT, ...</td>
<td>Google to find out more...</td>
</tr>
<tr>
<td>M 6/07</td>
<td><strong>Final Exam (Cumulative)</strong></td>
<td>4:00–6:00 PM -- be there!!!</td>
</tr>
</tbody>
</table>
We Covered Two Kinds of Things

• System types and principles
  • Failed escape attempts (OODBs, ORDBs, XMLDBs)
  • Parallel databases, OLTP/OLAP, partitioned parallelism
  • Logical DB design (first SQL, then NoSQL)
  • Key-value stores, consistency, and replication
  • Column-family, document, and graph data models
  • Data streaming systems (DSS)
  • Today’s parade of other technologies

• Hands-on with real systems
  • Cassandra (scalable column-family store)
  • MongoDB (most popular document store)
  • Couchbase Server (SQL-oriented document store)
  • Neo4J (graph DBMS)
  • Spark (Big Data analytics)
Next Steps?

• CS122B: Projects in Databases and Web Applications
  • Mostly relational (SQL) but extremely valuable!

• Online tutorials and training materials from software vendors of interest
  • See “Readings and References” on the course wiki page
  • Explore your favorite vendors’ training tracks online!

• Download and play with Apache AsterixDB and join its user mailing list (to do Q&A)
Other Questions?