NoSQL and Review
Big Data (some old numbers)

- **Facebook:**
  - 130TB/day: user logs
  - 200-400TB/day: 83 million pictures

- **Google:** > 25 PB/day processed data

- **Gene sequencing:** 100M kilobases per day per machine
  - Sequence 1 human cell costs Illumina $1k
  - Sequence 1 cell for every infant by 2015?
  - 10 trillion cells / human body

- **Total data created in 2010:** 1 ZettaByte (1,000,000 PB)/year
  - ~60% increase every year
Big data is not only databases

• Big data is more about data analytics and online querying

Many components:
• Storage systems
• Database systems
• Data mining and statistical algorithms
• Visualization
What is NoSQL?

from “Geek and Poke”

HOW TO WRITE A CV

DO YOU HAVE ANY EXPERTISE IN SQL?

NO

gene & poke

DOESN’T MATTER. WRITE: "EXPERT IN NO SQL"

Leverage the NoSQL boom
What is NoSQL?

• An emerging “movement” around non-relational software for Big Data
• Roots are in the Google and Amazon homegrown software stacks

Wikipedia: “A NoSQL database provides a mechanism for storage and retrieval of data that use looser consistency models than traditional relational databases in order to achieve horizontal scaling and higher availability. Some authors refer to them as "Not only SQL" to emphasize that some NoSQL systems do allow SQL-like query language to be used.”
## Some NoSQL Components

<table>
<thead>
<tr>
<th>Analytics Interface (Pig, Hive, ...)</th>
<th>Imperative Lang (RoR, Java, Scala, ...)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Parallel Processing (MapReduce/Hadoop)</td>
<td></td>
</tr>
<tr>
<td>Distributed Key/Value or Column Store (Cassandra, Hbase, Voldemort, ...)</td>
<td></td>
</tr>
<tr>
<td>Scalable File System (GFS, HDFS, ...)</td>
<td></td>
</tr>
</tbody>
</table>

- **Query Optimization and Execution**
- **Relational Operators**
- **Access Methods**
- **Buffer Management**
- **Disk Space Management**
The Issue: Scalability

The Million-Server Data Center

Today's most advanced data centers house tens of thousands of servers. What would it take to house 1 million?
Scalability

Parallel Database (circa 1990)

Map Reduce (circa 2005)
Scalability (continued)

• Often cited as the main reason for moving from DB technology to NoSQL
• DB Position: there is no reason a parallel DBMS cannot scale to 1000’s of nodes
• NoSQL Position: a) Prove it; b) it will cost too much anyway
The Structure Spectrum

Structured (schema-first)
- Relational Database
- Formatted Messages

Semi-Structured (schema-later)
- Documents
- XML
- Tagged Text/Media

Unstructured (schema-never)
- Plain Text
- Media
CouchDB Data Model (JSON)

• “With CouchDB, no schema is enforced, so new document types with new meaning can be safely added alongside the old.”

• A CouchDB document is an object that consists of named fields. Field values may be:
  ▪ strings, numbers, dates,
  ▪ ordered lists, associative maps

"Subject": "I like Plankton"
"Author": "Rusty"
"PostedDate": "5/23/2006"
"Tags": ["plankton", "baseball", "decisions"]
"Body": "I decided today that I don't like baseball. I like plankton."
MongoDB

- Data are organized in **collections**. A collection stores a set of **documents**.
  - Collection like table and document like record
    - but: each document can have a different set of attributes even in the same collection
    - Semi-structured schema!
  - Only requirement: every document should have an “**_id**” field
Example mongodb

```json
{
    "_id": ObjectId("4efa8d2b7d284dad101e4bc9"),
    "Last Name": "Cousteau",
    "First Name": "Jacques-Yves",
    "Date of Birth": "06-1-1910"
}

{
    "_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
    "Last Name": "PELLERIN",
    "First Name": "Franck",
    "Date of Birth": "09-19-1983",
    "Address": "1 chemin des Loges",
    "City": "VERSAILLES"
}
```
“With Mongo, you do less "normalization" than you would perform designing a relational schema because there are no server-side joins.

Generally, you will want one database collection for each of your top level objects.”

from the MongoDB manual

DBRef to define links (like foreign keys)
Whither Schemas?

DB: Schemas are necessary for correctness, robustness, and evolvability

NoSQL: a) Schemas keep me from getting *my* job done. b) messy data is reality
Other Structure Issues

• NoSQL: a) Tables are unnatural, b) “joins” are evil, c) need to be able to “grep” my data

• DB: a) Tables are a natural/neutral structure, b) data independence lets you precompute joins under the covers, c) this is a price of all the DBMS goodness you get

This is an Old Debate – Object-oriented databases, XML DBs, Hierarchical, …
Fault Tolerance

- **DBs: coarse-grained FT – if trouble, restart transaction**
  - Fewer, Better nodes, so failures are rare
  - Transactions allow you to kill a job and easily restart it

- **NoSQL: Massive amounts of cheap HW, failures are the norm and massive data means long running jobs**
  - So must be able to do mini-recoveries
  - This causes some overhead (file writes)
Other Issues

• **Consistency**
  ▪ Application-dependent and religious

• **Declarative vs. Imperative**
  ▪ Issues well-known
  ▪ Bordering on religious at this point

• **Cost**
  ▪ “Making Free Software Affordable”
  ▪ Need to look at TCO not just initial cost

• **Coolness**
  ▪ Clearly in the eye of the beholder...
• Back to

Relational Database Systems
(the topic of the course)
Recall the Fundamentals...

- **Why Use a DBMS?**
  - Data independence and efficient access.
  - Reduced application development time.
  - Data integrity and security.
  - Uniform data administration.
  - Concurrent access, recovery from crashes.

- **Remind me again why we learned this stuff?**
  - Shift from computation to information
  - data sets get bigger and bigger
  - CS microcosm
Query in:
e.g. “Select min(account balance)”

Data out:
e.g. 2000

Customer accounts stored on disk

Database app

Query Optimization and Execution

Relational Operators

Access Methods

Buffer Management

Disk Space Management
Final

- Based on the second part of the course + SQL
- Disks and Storage
- Indexing - B+-trees and Hashing
- Relational Operations and Cost estimation
- Query Optimization
- Transaction processing – Concurrency Control
- Recovery
Book chapters

- 5, 5.1-5.6
- 8, 8.1-8.4
- 9, 9.1, 93.-9.7
- 10, 10.1-10.7
- 11, 11.1-11.4
- 12, 12.1-12.6
- 13, 13.1-13.3
- 14, 14.1-14.6
- 15, 15.1-15.4
- 16, 16.1-16.7
- 17, 17.1-17.5 except 17.5.2
- 18, 18.1-18.6