Week 10: NoSQL

Lecturer: Dr. Thomas Roelleke

Room CS423

Outline

- The NoSQL Movement
- Motivation
- Types of Datastores
- Applications
- Important Considerations & Summary

The NoSQL Movement

- An emerging "movement" around non-relational software for Big Data
  - A "Tea Party" for data management
- Roots are in the Google (BigTable) and Amazon homegrown software stacks (AWS)
- Vibrant Open Source community
  - plus some recent converts
- Currently defined as what it's not

NoSQL [1]

- Not Only SQL or "Not Relational"
- Schema-less Datastore (e.g. Key-Value based)
- Key features:
  - Scale horizontally "simple operations" throughput over many servers
    - Replicate/distribute data over many servers
  - Simple call level interface (contrast w/ SQL)
  - Weaker concurrency model than ACID
  - Efficient use of distributed indexes and RAM
  - Flexible schema

NoSQL [2]

- The NoSQL systems described here generally do not provide ACID transactional properties.
- Updates are eventually propagated, but there are limited guarantees on the consistency of reads.
- Some authors suggest a "BASE" acronym in contrast to the "ACID" acronym:
  - BASE = Basically Available, Soft state, Eventually consistent
- The idea is that by giving up ACID constraints, one can achieve much higher performance and scalability.

Key-Value Stores

- Allow the application to store its data in a schema-less way.
- The data could be stored in a datatype of a programming language or an object.
- There is no need for a fixed data model.
- Only primary index: lookup by key.
- No secondary indexes.

Document Stores

- A "document" = a pointerless object = e.g. JSON = schema-less
- In addition to KV stores, may have secondary indexes
- Examples: SimpleDB, CouchDB, MongoDB
- Simple query mechanism is provided.

Extensible Record Stores

- Typical Access: Row ID, Column ID, Timestamp
- Rows: sharding by primary key
  - BigTable: split table into tablets = units of distribution
- Columns: "column groups" = indication for which columns to be stored together (e.g. customer name/address group, financial info group, login info group)
- Examples: HBase, HyperTable, Cassandra, PNUT, BigTable

Application 1

- Web application that needs to display lots of customer information; the users data is rarely updated, and when it is, you know when it changes because updates go through the same interface.
- Store this information persistently using a KV store

Application 2

- Department of Motor Vehicle: lookup objects by multiple fields (driver's name, license number, birth date, etc); "eventual consistency" is ok, since updates are usually performed at a single location.

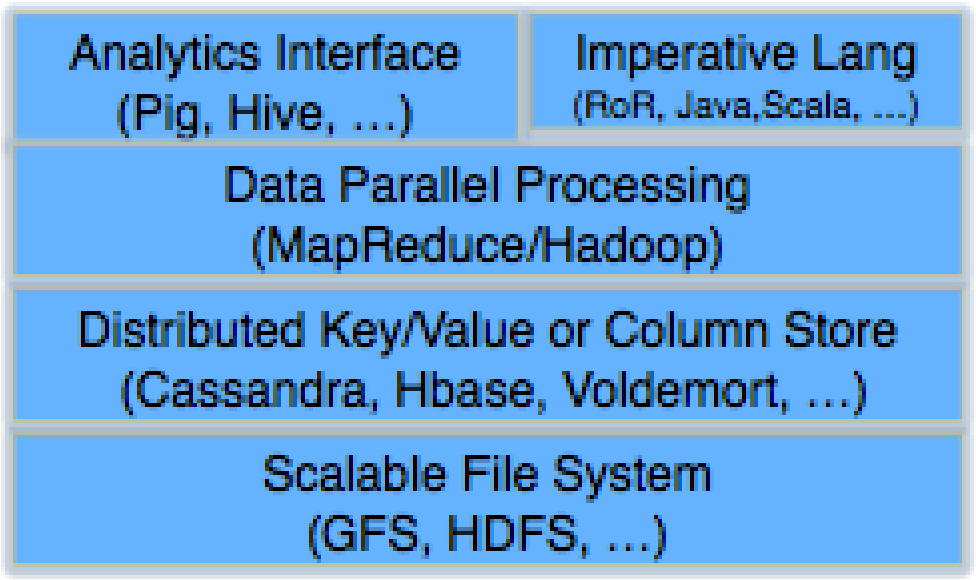
Document Store

Application 3

- eBay-like application.
- Cluster customers by country; separate the rarely changed "core" customer information (address, email) from frequently-updated info (current bids).

Extensible Record Store

Some NoSQL Components



Why such (semi-structured) data stores are needed?

- Semi-structured or flat files based data stores are best for massive data that is read, possibly frequently, but with minimal updates
- There is much less overhead to process data in this format, why?
- We also have the flexibility to process data that doesn't have a completely fixed structure

What NoSQL should NOT be used for

- Anything that requires frequent updates as well as reads, or that requires high integrity and atomicity (ACID properties)
- Examples are things like transaction databases for inventory and financial records
- Note that this is not just a question of massive data or distributed processing
- There are large, distributed relational databases like Visa or Amazon that need more structured data with transaction semantics
- These applications are better suited to relational databases even at large scale

Summary

- NoSQL Movement: “light-weight” DB systems
- NoSQL: alternative, non-traditional DB technology to be used in large scale environments where (ACID) transactions are not a priority
- Avoid overhead of transaction-oriented systems (ACID)
- Simple, extensible record stores; key-value stores; schema-less