NoSQL & NewSQL

Thomas Heinis

Imperial College
London

NoSQL: Overview

- Main objective: implement distributed state
  - Different objects stored on different servers
  - Same object replicated on different servers
- Main idea: give up some of the ACID constraints to improve performance
- Simple interface:
  - Write (=Put): needs to write all replicas
  - Read (=Get): may get only one
- Eventual consistency ← Strong consistency

NoSQL

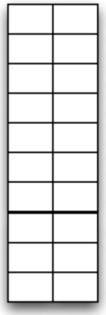
“Not Only SQL” or “Not Relational”.

Six key features:

1. Scale horizontally “simple operations”
2. Replicate/distribute data over many servers
3. Simple call level interface (contrast w/ SQL)
4. Weaker concurrency model than ACID
5. Efficient use of distributed indexes and main memory
6. Flexible schema

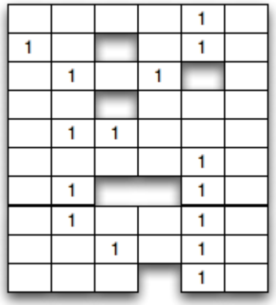
Four NoSQL Categories

Key-Value

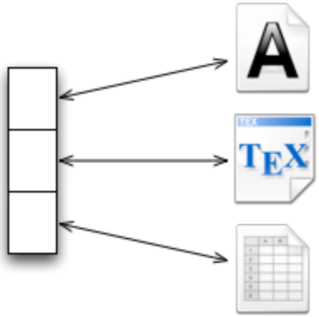


Graph DB

BigTable



Document



Data Model

- Tuple = row in a relational DB
- Document = nested values, extensible records (think XML or JSON)
- Extensible record = families of attributes have a schema, but new attributes may be added
- Object = like in a programming language, but without methods

Key-value Stores

- Think “file system” more than “database”
- Consistent hashing (DHT)
- Only primary index: lookup by key
- No secondary indexes
- Transactions: single- or multi-update

Basic Idea: Key-Value Store

Table T:

key	value
k1	v1
k2	v2
k3	v3
k4	v4

keys are sorted

- API:
  - lookup(key) → value
  - lookup(key range) → values
  - getNext → value
  - insert(key, value)
  - delete(key)
- Each row has timestemp
- Single row actions atomic
  No multi-key transactions
- No query language!

Document Stores

- A "document" = a pointerless object = e.g. JSON = nested or not = schema-less
- In addition to KV stores, may have secondary indexes
- SimpleDB, CouchDB, MongoDB, Terrastore
- Scalability:
  - Replication (e.g. SimpleDB, CounchDB – means entire db is replicated),
  - Sharding (MongoDB);
  - Both

Document Store (MongoDB)

```
> db.user.insert({
        first: "John",
        last : "Doe",
        age: 39
})
```

```
> db.user.find ({"first" : "John”})
{
        "_id" : ObjectId("51…"),
        "first" : "John",
        "last" : "Doe",
        "age" : 39
}
```

```
> db.user.update(
        {"_id" : ObjectId("51…")},
        {
                $set: {
                        age: 40,
                        salary: 7000}
        }
)
```

```
> db.user.remove({
        "first": /^J/
})
```

Column Family

- Most Based on **BigTable**: Google's Distributed Storage System for Structured Data
- Data Model:
  - A big table, with column families
  - Map Reduce for querying/processing
- Examples:
  - HBase, HyperTable, Cassandra

Column Family: Pros and Cons

- Pros:
  - Supports Semi-Structured Data
  - Naturally Indexed (columns)
  - Scalable
- Cons
  - Poor for interconnected data

Graph Databases

- Data Model:
  - Nodes and Relationships
- Examples:
  - Neo4j, OrientDB, InfiniteGraph, AllegroGraph

Graph Databases: Pros and Cons

- Pros:
  - Powerful data model, as general as RDBMS
  - Connected data locally indexed
  - Easy to query
- Cons
  - Sharding (lots of people working on this)
    - Scales UP reasonably well
  - Requires rewiring your brain

Scalable Relational Systems (NewSQL)

- Means RDBMS that offer sharding
- **Key difference**:
  - NoSQL difficult or impossible to perform large-scope operations and transactions (to ensure performance)
  - Scalable RDBMS do not **preclude** these operations, but users pay a price only when they need them
- MySQL Cluster, VoltDB, Clusterix, ScaleDB, Megastore (the new BigTable)
- Many more **NewSQL** systems becoming available…

Scalable Data Processing

- Parallel execution achieves greater efficiency
- But, parallel programming is hard
  - Parallelization
  - Fault Tolerance
  - Data Distribution
  - Load Balancing

MapReduce (Hadoop and others)

- "*MapReduce is a programming model and an associated implementation for processing and generating large data sets*"
- Programming model
  - Abstractions to express simple computations
- Library
  - Takes care of the gory stuff: Parallelization, Fault Tolerance, Data Distribution and Load Balancing

NoSQL Systems

- Key Value Stores

Key	Value
SW3 3TB	London
B18 4BJ	Birmingham

cassandra

- Document Stores

CouchDB relax

mongoDB

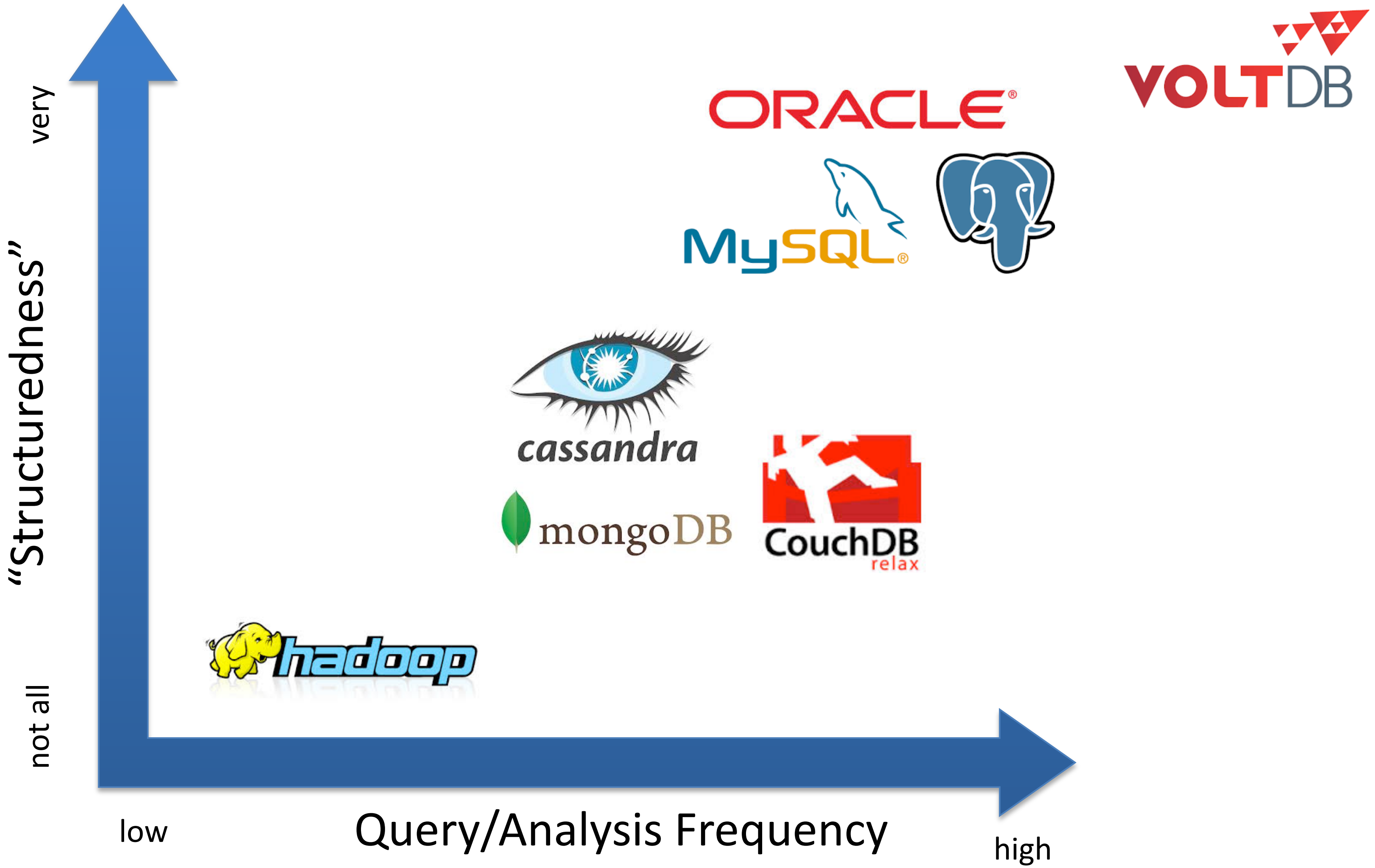
- Scalable SQL Systems

VOLTDB

- Data Processing Systems

hadoop

Data Management Landscape[*]



[*] according to me