Summary of NoSQL databases and big data storage systems

By S. Xiao Fernández Marín

1 Summary

The idea of NoSQL is to store large amount of data and be efficient with working woth that data

• Availability more important than consistency: Inspired by the CAP theorem

• It's hard to guarantee ACID: Move implementation cost from server (DB system) to application (client)

• It's complex to implement SQL: Move complex operations from server to application and Imperative languages avoids implementing optimizer

• Schemaful forces application developers to plan ahead: Move schema enforcement from server to application, logic in the app side

Typical characteristics

- Horizontal scalability
- High availability
- Weak consistency
  - Eventual consistency
  - Applications must handle inconsistencies
- Data distribution
  - Horizontal sharding (each row of a table in different server) bc horizontal scalability, the app handle the app, so the app combines the sharding // dist db partitioning: do query independently where the data is located and db will locate
  - Limited cross-shard functionality
- Simple data model
  - Key/value, like hash
  - Limited indexing
- Schemaless, no specific requirements in schema
  - Dynamic typing
  - Schema encoded in application logic
- Data model tied closer to programming language
  - Object model, JSON from Java script eg
- Denormalized data models,, //NOT SURE duplicated data
- Less advanced query languages
  - Create, read, update, delete (CRUD)
  - Syntax often close to source code
  - Imperative instead of declarative
  - Join often on application side
- No transactions
  - No ACID properties
  - Little isolation

Simpler system, data models:
• Key-value databases: Value is a binary string or slightly more advanced
• Document databases: Values are objects (often compatible with JSON)
• Column databases: Similar to RDBMSs, but weaker schema
• Graph databases: Nodes and links with attributes (similar to ER models)

Key-value databases Simples of them all
A key points to a value,, hash
- Unique key
- Value is string of bytes
- Values is an object (often JSON compatible)

Very simple data model

No complex operations
- CRUD
- CRD (no updates)

Examples: Amazon DynamoDB, Voldemort, Oracle NoSQL Database, Redis

Document databases
Really object databases
- Structured objects like JSON, not documents like HTML

Similar to key-value stores
- Values are objects, not random byte strings
- Richer query capabilities

Usually (almost) JSON compatible

Key can be separate or embedded in value object

Indexing on non-key members, not like key-value

Examples: MongoDB, CouchDB, RethinkDB, Couchbase

Column databases
More similar to RDBMSs
- Tabular data model
- More SQL like query languages

Weak schema
- Columns not predefined, may vary between rows
- Predifined "column families"

Often versioned cells

Sometimes called "wide column stores"

Examples: Google Bigtable, Apache Hbase

Graph databases
Stores vertexes and edges
- Often called nodes and relationships
- Both vertexes and edges may have attributes

Close to (E)ER model

Searched/navigated with path expressions

Special query languages

Examples: Neo4j, RedisGraph, Amazon Neptune

NoSQL in RDBMSs

• All major RDBMSs have some NoSQL functionality
– JSON object stores
– Graph functionality

• RDBMS replication
– High availability
– Sharding

• Implement superset of NoSQL functionality
– Optimized for different use cases

NoSQL vs. SQL

• **NoSQL** systems are becoming more advanced: query languages, Joins and Transactions

• **SQL** systems are adding more schemaless features: - JSON: Similar to object-relational and XML (in SQL since year 2000)
– SQL/JSON standard
– Graph query languages