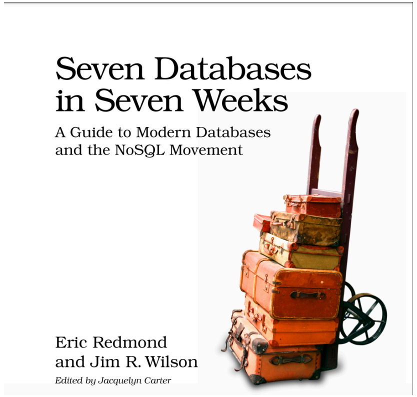
NoSQL DB properties and its data models

Topic 5 Lesson 1
An alternative to the relational data model

Adapted from:



Appendix 2

Content from the MongoDB website: https://www.mongodb.com/
https://www.mongodb.com/nosql-explained

Vendors for the NoSQL website: https://nosql-database.org/

What spurred the NoSQL revolution?

- Relational databases' inability to scale to meet the growing demands for high volumes of read and write operations
- Customers were dissatisfied with what their RDMS could provide and were technically savvy - so they could identify that the problem was the data model and guaranteed properties provided by the RDMS (ACID)
  - Companies were able to quantify the amount of money they were losing due to how long it took them to complete a transaction.
  - Big impetus was the web and web data or online transaction processing (OLTP)

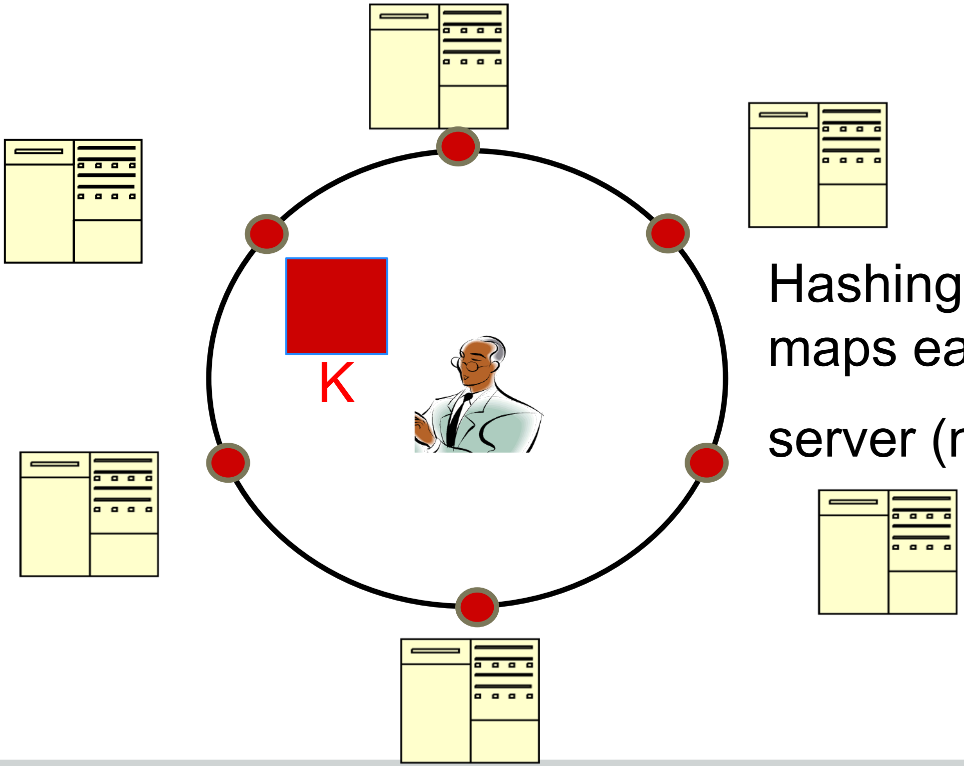
Taxonomy of NoSQL data models

- Key-value
- Graph database
- Document-oriented
- Column family

Current NoSQL data models

- **Key-value :** associate a data value with a specific key (data structure is not specified)
- **Document-oriented :** associate a structured data value with a specific key. The data structure is embedded in the object, objects can contain other objects
- **Graph database :** consists of nodes and edges. Typically, the nodes represent entities, and the edges represent relationships.
- **Columnar database:** stores data by columns as oppose to rows. Columns are grouped into families. Typically, a family corresponds to a real-world object

Typical NoSQL architecture



Hashing function maps each key to a server (node)

CAP theorem

What the CAP theorem really says:

- If you cannot limit the number of faults and requests can be directed to any server and you insist on serving every request you receive, then you cannot possibly be consistent

Eric Brewer

How it is interpreted:

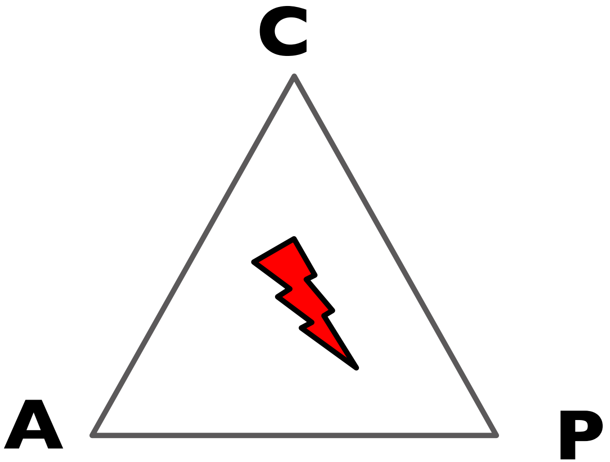
- You must always give something up: consistency, availability or tolerance to failure and reconfiguration

Theory of NoSQL: CAP

GIVEN:

- Many nodes (distributed DB)
- Nodes contain ***replicas of partitions*** of the data

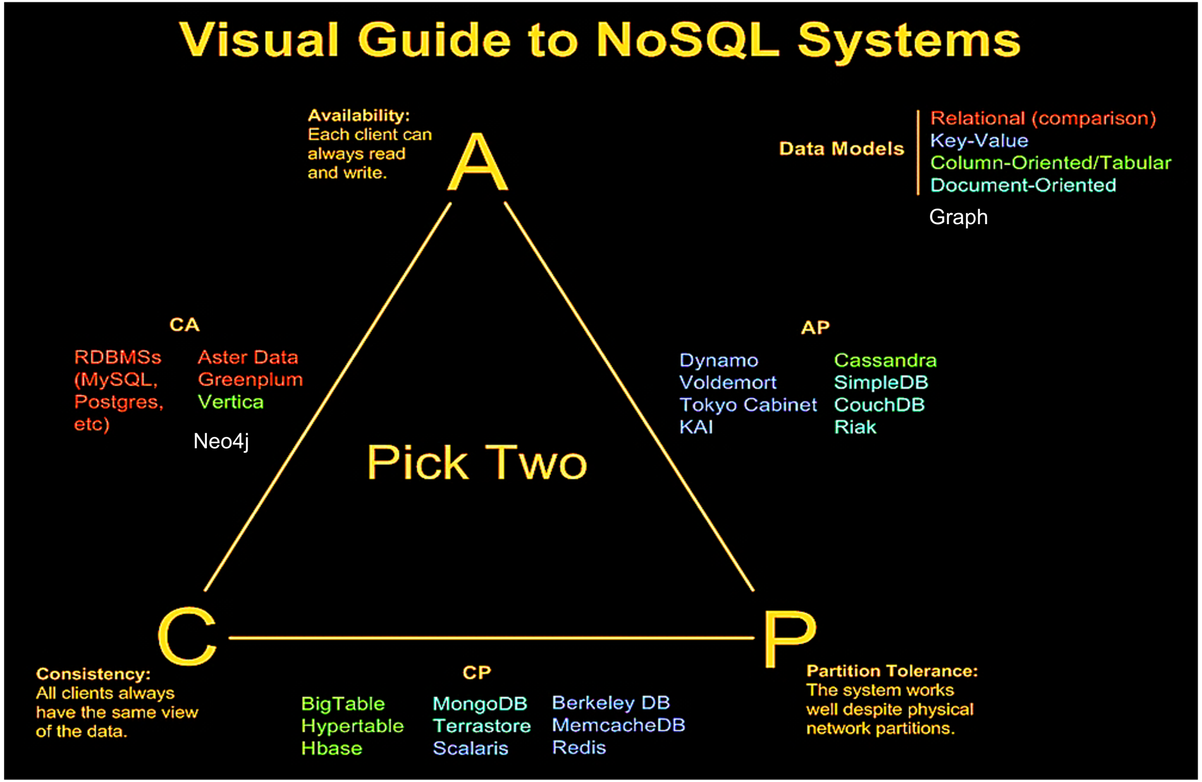
- Consistency
  - All replicas contain the same version of data
  - Client always has the same view of the data (no matter what node)
- Availability
  - System remains operational
  - All clients can always read and write
- Partition tolerance
  - multiple entry points to DB network
  - System remains operational on system split (communication malfunction)
  - System works well across physical network partitions



CAP Theorem:

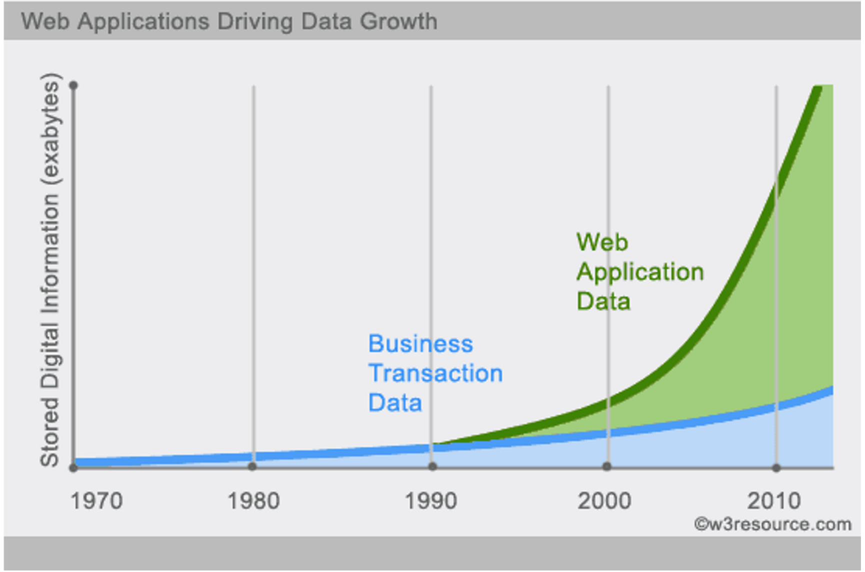
guaranteeing all three at the same time is impossible

Vendors pick 2 out of 3 properties



Changing face of data

- Big Data can be understood through the 3 V's of volume, variety, velocity:
  - **Volume**: enormous amounts of structured and unstructured data
  - **Variety**: multiple data types including documents, images, videos, and time series
  - **Velocity**: flow of data is continuous and increasing



How does NoSQL vary from RDB?

- Looser schema definition
- Applications written to deal with specific documents, objects or data from the database
  - Applications aware of the schema definition as opposed to the data
- Designed to handle distributed, large databases
- Trade offs:
  - No strong support for ad hoc queries but designed for speed and growth of database
    - Query language through an application programming interface (API)
  - Relaxation of the ACID properties

A shard of data

- Sharding distributes a single logical data collection across a cluster of machines, the data is partitioned horizontally
- A shard is the collection of data objects on a specific database server
- Shard typically uses a range-based partitioning scheme to distribute data objects based on a specific shard key
  - Can also use a natural grouping of data objects
- Algorithm automatically balances the data associated with each shard
- Can be turned on and off per data collection (table)

Sharding can address hot spots

- One specific data object can be found in 1 or more shards
- Objects retrieved frequently can be stored on multiple servers or more multiple shards
- This solution allows the DBA to disperse the heat associated with a “hot object” across multiple servers
- Should optimize access to the objects in the “hot spot”
- Sharding also lower the number of objects stored on one server which should speed up retrieval time
- Take advantage of parallel programming

Duplicating data: replica sets

- Redundancy: multiple paths to the same object
- Failover: provides resilience in the face of a network partition
  - can switch all data requests to another node when a node fails
- Zero downtime for upgrades and maintenance
- Master-slave replication
  - Strong Consistency: all copies updated synchronously or appear to be updated synchronously.
  - Delayed Consistency: update occurs when DBMS decides it is optimal to do so

Consistency of data

- All read operations issued to the primary replica set, reads are consistent with the last write operation
  - Reads to a primary have **strict consistency**
    - Reads reflect the latest changes to the data
  - Reads to a secondary have **eventual consistency**
    - Updates propagate gradually
  - If clients permit reads from secondary sets – then client may read a previous state of the database
  - If a failure occurs before the secondary sets are updated
    - System identifies when a rollback needs to occur
    - Users are responsible for manually applying rollback changes

RDB ACID to NoSQL BASE

Atomicity

Consistency

Isolation

Durability

↔

Basically

Available (CP)

Soft-state
(State of system may change over time)

Eventually consistent
(Asynchronous propagation)

Pritchett, D.: BASE: An Acid Alternative (queue.acm.org/detail.cfm?id=1394128)

Benefits of NoSQL

Elastic Scaling

- RDBMS scale up – bigger load , bigger server
- NoSQL scale out – distribute data across multiple hosts seamlessly

DBA Specialists

- RDMS require highly trained expert to monitor DB
- NoSQL require less management, automatic repair and simpler data

Big Data

- Huge increase in data RDMS: capacity and constraints of data volumes at its limits
- NoSQL designed for big data
  - Volume
  - Variety
  - Velocity

Benefits of NoSQL (2)

Flexible data models

- RDB change management to schema must be carefully managed and is a burden
- NoSQL databases more relaxed data structure
  - Database schema changes do not have to be managed as one complicated change unit
  - Application already written to address an amorphous schema

Economics

- RDBMS rely on expensive proprietary servers to manage data
- No SQL: clusters of cheap commodity servers to manage the data and transaction volumes
- Cost per gigabyte or transaction/second for NoSQL typically lower than the cost for a RDBMS

Drawbacks of NoSQL

- Support
  - RDBMS' vendors provide a high level of support to clients
    - Stellar reputation
  - NoSQL – are open-source projects with startups supporting them
    - Reputation not yet established
- Maturity
  - RDBMS' are a mature product, means stable and dependable
    - Also means old no longer cutting edge nor interesting
  - NoSQL are still implementing their basic feature set

Drawbacks of NoSQL (2)

- **Administration**
  - RDMS administrator well defined role
  - NoSQL's goal: no administrator necessary however NO SQL still requires effort to maintain
- **Lack of Expertise**
  - Whole workforce of trained and seasoned RDMS developers
  - Still recruiting developers to the NoSQL camp
- **Analytics and Business Intelligence**
  - **RDMS designed to address this niche**
  - NoSQL designed to meet the needs of a Web 2.0 application - not designed for ad hoc query of the data
    - Tools are being developed to address this need

Summary

- NoSQL built to address a distributed database system
  - Shard: distribution of data collections across servers
  - Replica sets: duplication of data objects across servers
- CAP Theorem: consistency, availability and partition tolerant
  - Consistency: all users see the most up-to-date version of the data
  - Availability: system provides a response to a user request
  - Partition tolerance: system remains operational despite network or system failures
  - Impossible to guarantee all 3 properties all the time