

NoSQL Introduction

Lecture 5



Motivation

- Even today, RDBMSs are ubiquitous for small/medium-sized business applications
 - Features: data models, SQL+APIs, ACID, reasonable performance, security, backup, ...
- But much of an RDBMS is all-or-nothing, which is to say that many applications (e.g. web) need a smaller/different feature set, with a much different performance profile



NoSQL

- Originally saying “no” to RDBMS...
 - “NoSQLers came to share how they had overthrown the tyranny of slow, expensive relational databases in favor of more efficient and cheaper ways of managing data.”
- Now more like “Not Only SQL”
 - Recognize the right tool for the right job
- NoSQL databases focus on semi-structured data, high performance/availability/scalability
 - The movement has also had a positive impact on RDBMSs, motivating them to adapt



NoSQL Characteristics: Performance

- Focus on **horizontal** scaling
- Emphasize availability and fault tolerance, settling for **eventual consistency**
 - Replication: Master-Slave, Master-Master
 - **Sharding** (horizontal partitioning) of data
- High performance data access
 - Hashing and range partitioning



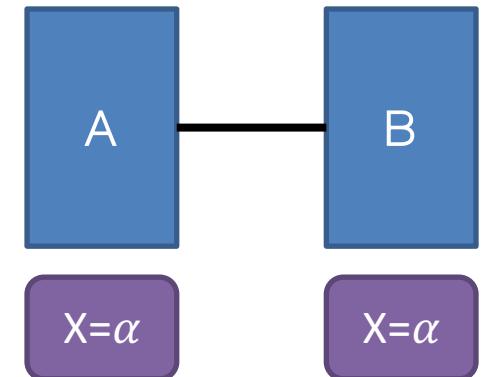
Aside: CAP Theorem

- For any shared-data system...
 - **Consistency**: all nodes have the same copies of a replicated data item visible for various transactions (note: different from ACID)
 - **Availability**: every request (to a non-failed node) will result in a response
 - **Partition tolerance**: system still operates despite inability of nodes to communicate
- Commonly stated as “choose 2 of 3”
 - Really: when there is a partition, choose either consistency or availability



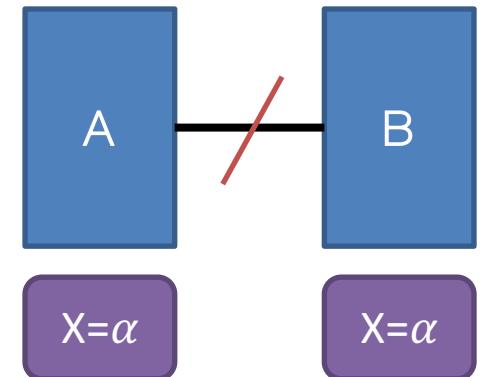
CAP: Illustrative Example

1. Two nodes: A, B
 - Share data X, initially α



CAP: Illustrative Example

1. Two nodes: A, B
 - Share data X, initially α
2. Network outage!
 - A can't communicate with B :(

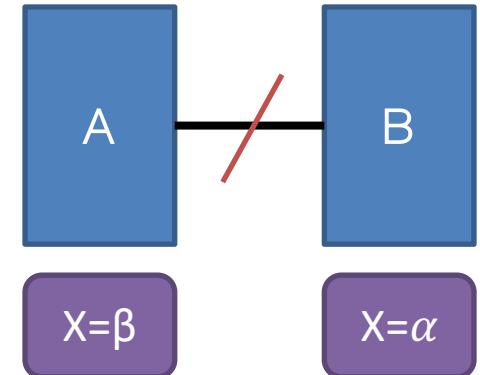


t



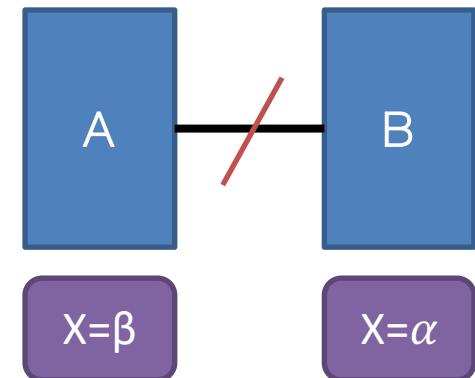
CAP: Illustrative Example

1. Two nodes: A, B
 - Share data X, initially α
2. Network outage!
 - A can't communicate with B :(
3. User 1 writes $X=\beta$ on A



CAP: Illustrative Example

1. Two nodes: A, B
 - Share data X, initially α
2. Network outage!
 - A can't communicate with B :(
3. User 1 writes $X=\beta$ on A
4. User 2 reads X from B



Question: what should User 2 receive?

- Consistent: β
 - Must wait for A, and so not available
- Available: α
 - Not consistent



NoSQL Characteristics: Data/Query

- Schema not necessarily required
- Less powerful query languages
 - **(S)CRUD**: Search Create Read Update Delete
 - Often programmers need to manually join
- Built-in versioning of data



Common NoSQL Categories (1)

- Document Stores
 - Commonly JSON, no schema
 - Access by document id or content
 - Example: MongoDB
- Key-Value Stores
 - Associative array: access data via id
 - Basically fast, distributed hash tables
 - Example: DynamoDB, Redis



Common NoSQL Categories (2)

- Graph
 - Stores nodes, columns; graph ops optimized
 - Example: Neo4j
- Others
 - Column-based (horizontal partitioning)
 - Hybrid
 - XML
 - Objects

...



Former DBMS Student :)

3 Database SQL walked into
a NoSQL bar.

A little while later...
they walked out

Because they couldn't find a table.

