CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

KEY-VALUE STORES NoSQL

Lecture A

WHY KEY-VALUE/NoSQL?

The Key-Value Abstraction

- (Business) Key → Value
- (twitter.com) Tweet id → information about tweet
- (amazon.com) Item number → information about it
- (kayak.com) Flight number → information about flight, e.g., availability
- (yourbank.com) Account number → information about it

The Key-Value Abstraction (2)

- It's a dictionary datastructure.
  - Insert, lookup, and delete by key
  - E.g., hash table, binary tree
- But distributed
- Sound familiar? Remember distributed hash tables (DHT) in P2P systems?
- It's not surprising that key-value stores reuse many techniques from DHTs.

Isn't that just a database?

- Yes, sort of
- Relational Database Management Systems (RDBMSs) have been around for ages
- MySQL is the most popular among them
- Data stored in tables
- Schema-based, i.e., structured tables
- Each row (data item) in a table has a primary key that is unique within that table
- Queried using SQL (Structured Query Language)
- Supports joins

Relational Database Example

users table

user_id	name	zipcode	blog_url	blog_id
101	Alice	12345	alice.net	1
422	Charlie	45783	charlie.com	3
555	Bob	99910	bob.blogspot.com	2

Primary keys

Foreign keys

blog table

id	url	last_updated	num_posts
1	alice.net	5/2/14	332
2	bob.blogspot.com	4/2/13	10003
3	charlie.com	6/15/14	7

Example SQL queries

1. SELECT zipcode
   FROM users
   WHERE name = “Bob”

2. SELECT url
   FROM blog
   WHERE id = 3

3. SELECT users.zipcode, blog.num_posts
   FROM users JOIN blog
   ON users.blog_url = blog.url

Mismatch with today's workloads

- Data: Large and unstructured
- Lots of random reads and writes
- Sometimes write-heavy
- Foreign keys rarely needed
- Joins infrequent

Needs of Today's Workloads

- Speed
- Avoid Single Point of Failure (SPOF)
- Low TCO (Total cost of operation)
- Fewer system administrators
- Incremental scalability
- Scale out, not up
  - What?

Scale out, not Scale up

- Scale up = grow your cluster capacity by replacing with more powerful machines
  - Traditional approach
  - Not cost-effective, as you're buying above the sweet spot on the price curve
  - And you need to replace machines often
- Scale out = incrementally grow your cluster capacity by adding more COTS machines (Components Off the Shelf)
  - Cheaper
  - Over a long duration, phase in a few newer (faster) machines as you phase out a few older machines
  - Used by most companies who run datacenters and clouds today

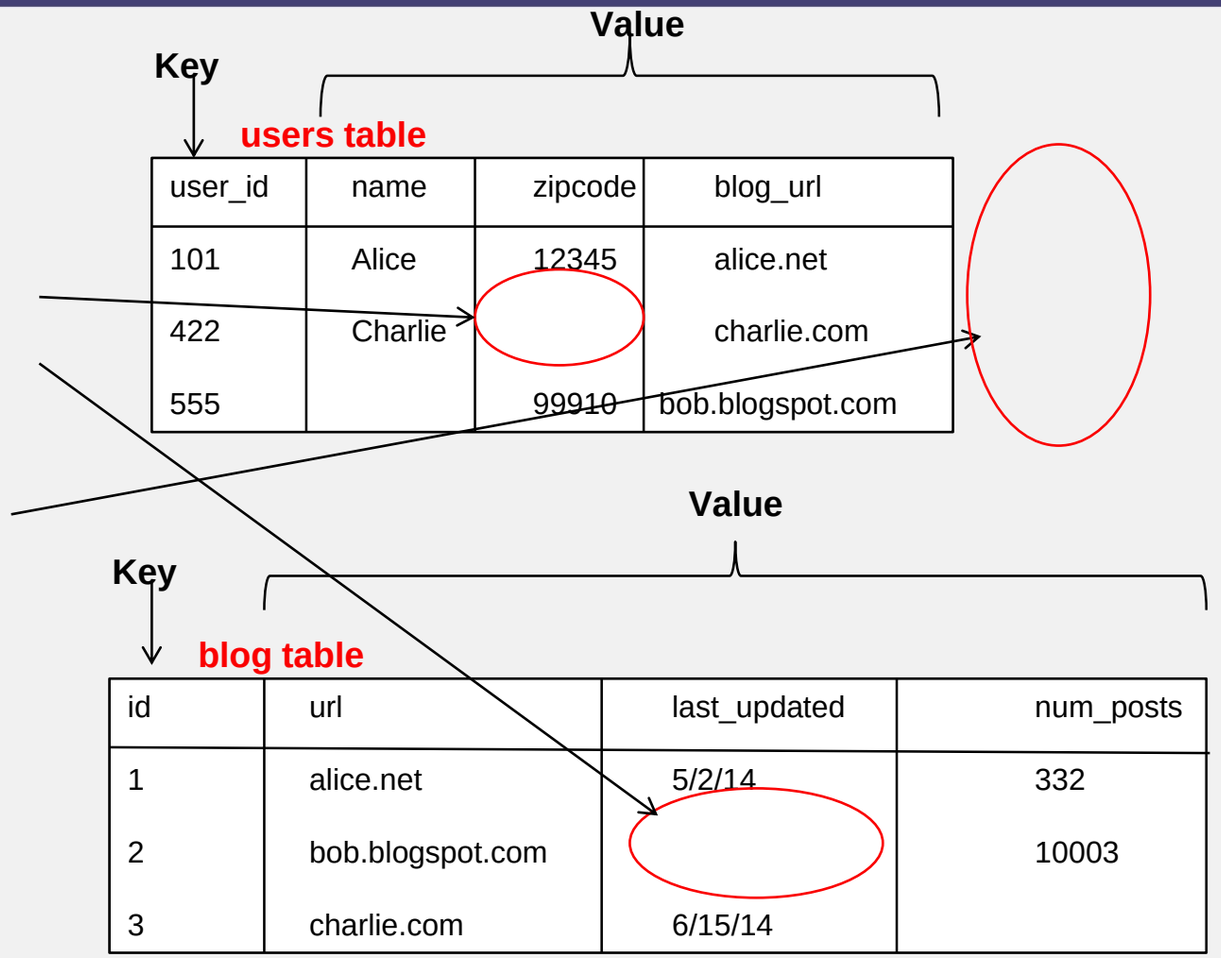
Key-Value/NoSQL Data Model

- NoSQL = “Not Only SQL”
- Necessary API operations: get(key) and put(key, value)
  - And some extended operations, e.g., “CQL” in Cassandra key-value store

- Tables
  - “Column families” in Cassandra, “Table” in HBase, “Collection” in MongoDB
  - Like RDBMS tables, but …
  - May be unstructured: May not have schemas
    - Some columns may be missing from some rows
  - Don’t always support joins or have foreign keys
  - Can have index tables, just like RDBMSs

Key-Value/NoSQL Data Model

- Unstructured
- No schema imposed
- Columns missing from some rows
- No foreign keys, joins may not be supported



Column-Oriented Storage

NoSQL systems often use column-oriented storage

- RDBMSs store an entire row together (on disk or at a server)
- NoSQL systems typically store a column together (or a group of columns).
  - Entries within a column are indexed and easy to locate, given a key (and vice-versa)
- Why useful?
  - Range searches within a column are fast since you don't need to fetch the entire database
  - E.g., get me all the blog_ids from the blog table that were updated within the past month
    - Search in the the last_updated column, fetch corresponding blog_id column
    - Don't need to fetch the other columns

Next

Design of a real key-value store, Cassandra.