NoSQL Databases

Around 2008 people started propagating for NoSQL databases

Traditional RDBMS were seen as:

- slow
- complex
- inflexible
- not suited for web usage
- expensive

They argued for something radically simpler, faster, cheaper.

What does NoSQL mean?

No consensus on the meaning, a huge variety between systems.
However, most systems have at least some of the following:

- distributed
- weak consistency guarantees
- no explicit schema
- optimized for point access
- limited query functionality
- aim for high throughput / high availability
- open source

But nearly any combination exists. We will look at individual systems later.

CAP Theorem

In a distributed compute system, one can get only two out of the following three:

- Consistency
- Availability
- Partition Tolerance

Used as argument for weaker consistency. However, theorem is somewhat misleading.

Why Consistency?

Consider these examples:

- People you do not want seeing your pictures
  - removes mom from list of people who can view photos
  - Alice posts embarrassing pictures from Spring Break
  - Can mom see Alice's photo?
- Why am I still getting messages?
  - Bob unsubscribes from mailing list
  - Message sent to mailing list right after
  - Does Bob receive the message?

Consistency Problems

Fixing consistency in the application is difficult!

Example: Facebook architecture

- Read path
  - Look in memcached
  - Look in MySQL
  - Populate in memcached
- Write path
  - Write in MySQL
  - Remove in memcached
- Subsequent read
  - Look in MySQL
  - Populate in memcached

Consistency Problems - Lag

Replication between data centers takes time. Example: CA and VA

1. User updates first name from “Jason” to “Monkey”
2. Write “Monkey” in master DB in CA, delete memcached entry in CA and VA
3. Someone goes to profile in Virginia, read VA slave DB, get “Jason”
4. Update VA memcache with first name as “Jason”
5. Replication catches up. “Jason” stuck in memcached until another write!

Unit of Consistency

- Single record
  - Relatively straightforward
  - Complex application logic to handle multi record transactions
- Arbitrary transactions
  - Requires 2PC/Paxos
- Middle ground: entity groups
  - Groups of entities that share affinity
  - Co-locate entity groups
  - Provide transaction support within entity groups
  - Example: user + user's photos + user's posts etc.

Key-Value Stores

Stores associations between keys and values

- Keys are usually primitives
  - For example, ints, strings, raw bytes, etc.
- Values can be primitive or complex: usually opaque to store
  - Primitives: ints, strings, etc.
  - Complex: JSON, HTML fragments, etc.

Limited functionality, but easy to implement and easy to scale

Operations

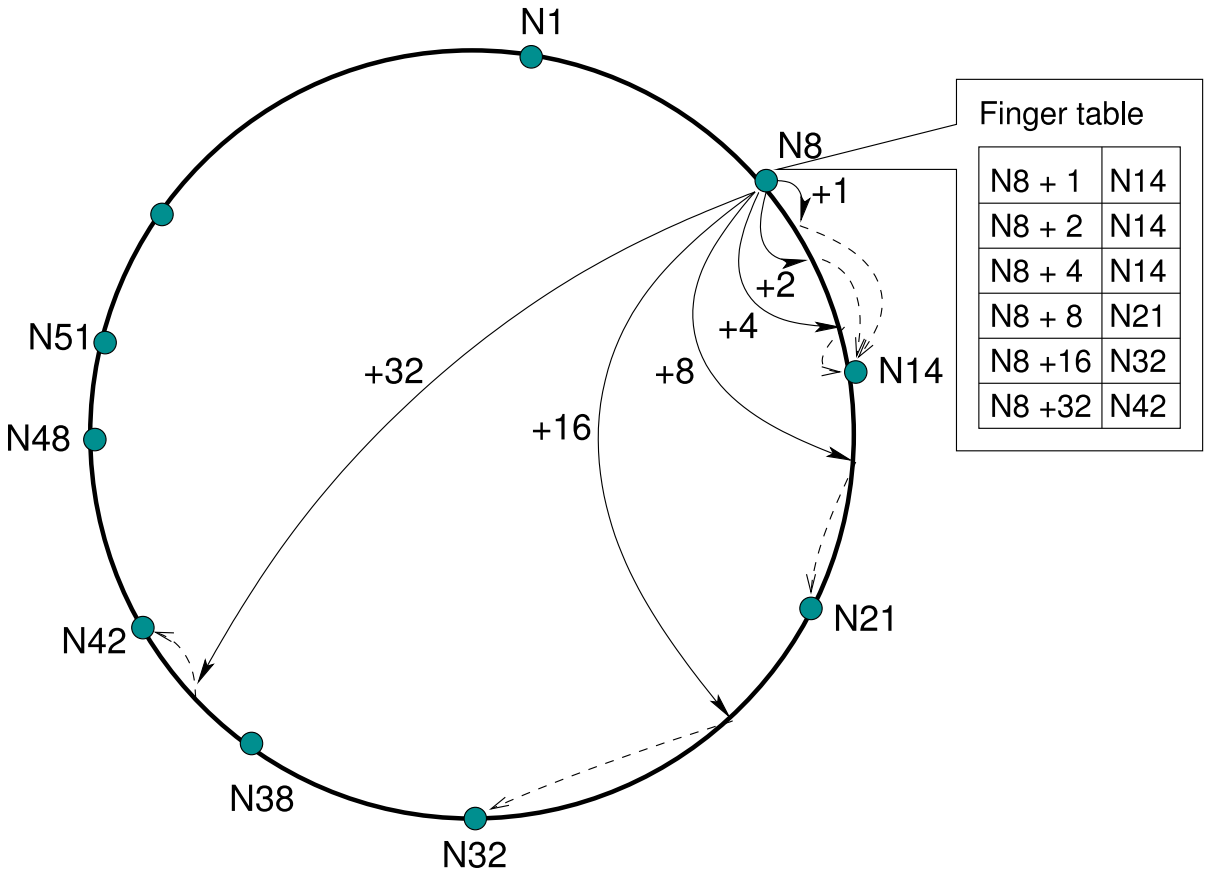
- Very simple API:
  - Get – fetch value associated with key
  - Put – set value associated with key
- Optional operations:
  - Multi-get
  - Multi-put
  - Range queries
- Consistency model:
  - Atomic puts (usually)
  - Cross-key operations: who knows?

Dealing with scale

- Partition the key space across multiple machines
  - Let's say, hash partitioning
  - For n machines, store key k at machine h(k) mod n
- Okay... but
  - How do we know which physical machine to contact?
  - How do we add a new machine to the cluster?
  - What happens if a machine fails?
- We need something better
  - Hash the keys
  - Hash the machines
  - Distributed hash tables

Distributed Hash Tables - Chord

Hash both data and nodes into a ring structure



- similar to a skip list
- handles replication, churn, locating nodes, etc.

MongoDB

Many NoSQL systems exists, MongoDB is quite popular

- has database/collection as contains
- stores JSON documents within a collection
- indexed by `_id` attribute, additional indexes possible
- queries are JSON documents, too
- we will look at example code

The Scary World of NoSQL systems

`http://jepsen.io/analyses.html`