Australian National University

NoSQL Databases – Part 2

Key-value Data Stores

Key-value Data Stores

- Inspired by Amazon's Dynamo (2007)
- The simplest type of NoSQL databases to use from an API perspective (the implementation may be complex)
- Look like a simple hash table (i.e., **a unique key and a value**), but not – it is a **big, distributed, fault-tolerant, persistent hash table**!
- Other examples:
  - MemcacheDB
  - Redis
  - Voldemort (LinkedIn)

Key-value Data Stores - Data Model

- The schema of a key-value data store is:

  Key and **Value**

  (**Key** is a string and **Value** is a blob).

- The user determines how to understand the values and how to parse them.
- This data model is particularly good for looking up things by keys.

Australian National University

Key-value Data Stores - Data Model

- Consider the following relation:

USER					
UserID	Name	Gender	DoB	Address	Hobbies
1	Peter	M	03-07-1990	34 Wattle St	fishing
2	Tom	M	01-09-1995	3 Arnold St	swimming

- **Question:**
  - *How can we express the relation* USER *in a key-value data store?*

Key-value Data Stores - Data Model

- Relational databases

User					
UserID	Name	Gender	DoB	Address	Hobbies
1	Peter	M	03-07-1990	34 Wattle St	fishing
2	Tom	M	01-09-1995	3 Arnold St	swimming

- Key-value data stores

Key	**Value**
1	Peter, M, 03-07-1990, 34 Wattle St, fishing
2	Tom, M, 01-09-1995, 3 Arnold St, swimming

Australian National University

Key-value Data Stores - Data Operations

- A typical API looks like:

```
void Put(string key, byte[] data);
```

```
byte[] Get(string key);
```

```
void Remove(string key);
```

Example:

```
user-peter = session.Get(1);
```

```
session.Put(4, 'Jane, F, 5 McCaughey Street');
```

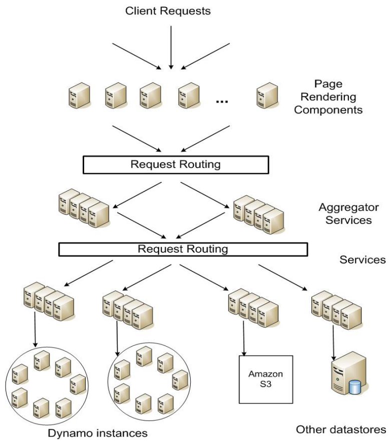
- Simple queries can be performed based on a key (but range queries on keys are usually not possible).

Amazon's Dynamo[1]

[1] G. DeCandia, D. Hastorun, et al., Dynamo: Amazons highly available key-value store, SOSP, 2007.

Amazon’s Infrastructure

- Amazon uses a highly decentralized, service oriented architecture.



Amazon’s Dynamo - Problem Analysis

- **Technological context**
  - Tens of thousands of servers and network components are located in many datacenters around the world.
  - Commodity hardware is used, and failure of a component is the “standard mode of operation”.
- **One of main design considerations**

“To give services control over their system properties, such as durability and consistency, and to let services make their own tradeoffs between functionality, performance and cost effectiveness.”

Australian National University

Amazon’s Dynamo - Key Features

- Key features of Dynamo:
  1. **Incremental scalability**
  2. **Eventual consistency**
  3. **High availability for writes**
  4. **Handling failures**

Amazon’s Dynamo - Partitioning

- **Problem:** Partition data over a set of servers to achieve incremental scalability.
  - Given a key, it is to figure out which machine stores the value.
- **A possible solution:** Use modulo hashing approach:

  hash(key) mod N

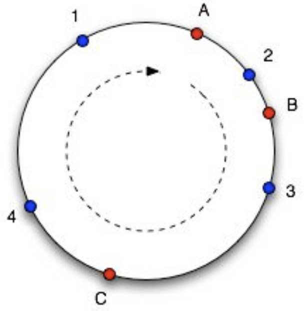
- **Question:**

  How do you distribute data when one or more servers become unavailable, or when more servers become available?

Amazon's Dynamo - Partitioning

- **Solution:** **Consistent Hashing**
- Both objects and servers are hashed to a number range in a circle, and an object is stored on the server that is closest in the clockwise direction.

  Example: four objects (1, 2, 3, 4) and three severs (A, B, C)[1]



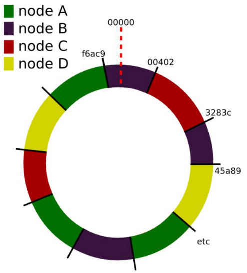
[1] The figure is taken from White, Tom, Consistent hashing, 2007.

Amazon’s Dynamo - Partitioning

- **Advantages** of consistent hashing
  - **Servers unavailable:** Adjacent servers can take over objects in the segments of these servers.
  - **Servers available again:** Adjacent servers can give away some objects in their own segments.
- But ... consistent hashing also has **drawbacks**:
  1. Servers and objects are randomly hashed onto the circle which may lead to an unbalanced distribution of objects on the servers.
  2. Consistent hashing treats each server equally and does not take into account its hardware resources.

Amazon's Dynamo - Partitioning

- **Better solution: Consistent Hashing + Virtual Nodes**
  - A number of replicas - called virtual nodes - for each physical server get hashed onto the circle.



  - The number of virtual nodes per physical server can be defined individually according to its hardware capacity (cpu, memory, disk).

Amazon's Dynamo - Limitations

While Dynamo gave them a system that met their reliability, performance, and scalability needs, it did nothing to reduce the operational complexity of running large database systems ... they had to become experts on the various components running in multiple data centers. Also, they needed to make complex tradeoff decisions between consistency, performance, and reliability....

Source: from Werner Vogels, CTO - Amazon.com

Key-value Data Stores - Summary

- **Highly scalable**, and there are two major options for scaling
  - **Partitioning**
    data is partitioned so that each database has **a subset of the data** stored on local disks.
  - **Replication**
    data is copied so that more than one database has **the same data** stored on local disks.
- Concurrency is only applicable on a single key, and concurrency conflict is thus easy to handle.
- Can **gain significant performance benefit** when structuring data access along keys for right applications, e.g. Amazons shopping cart runs on a key-value store (Amazon Dynamo).
- If you need complex operations on values, you should look at other solutions, such as document-oriented data stores.