Amazon DynamoDB

Himani Singh
Ishan Gupta
Rajeev Suri

1

About

Why NoSQL ?

- History
- Intro to NoSQL

Overview Of DynamoDB

- Introduction
- Features

Architecture

Demo

2

History of Data Processing

Where did it happen? How about when??

- Machine readable punch cards 1880
- Relational Databases 1970
- NoSQL RDBMS 1998 (Carlo Strozzi)
- NoSQL 2009

3

Why NoSQL ?

- Low storage cost and CPU is the bottleneck
- Why optimize the least expensive resource in the data
- Suitability of a given NoSQL database depends on the problem it must solve
- No Complex queries

4

SQL vs NoSQL

Characteristic	Relational Database Management System (RDBMS)	NoSQL Database
Optimal Workloads	Ad hoc queries; data warehousing; OLAP (online analytical processing).	Web-scale applications, including social networks, gaming, media sharing, and Internet of Things (IoT).
Data Model	The relational model requires a well-defined schema.	DynamoDB is schemaless. Data can be accessed using object oriented architecture.
Data Access	SQL is the standard for storing and retrieving data.	Data is accessed through database specific APIs.

5

Motivation for NoSQL

Even the slightest outage has significant financial consequences and impacts customer trust.

The platform is implemented on top of an infrastructure of tens of thousands of servers and network components located in many data-centers around the world.

Persistent state is managed in the face of these failures - drives the reliability and scalability of the software systems

6

Motivation (Continued)

A distributed storage system:

- Scalable
- Simple data model (name, value)
- key-value store (big hash table)
- **Highly available** (Eventual consistency)
- **Guarantee Service Level Agreements (SLA)**

7

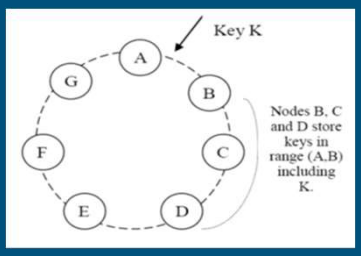
Architecture (Distributed System Concepts)

- Ring Partitioning (DHT like system)
- High Availability for writes (Vector Clocks to resolve conflicts among Replicas)
- Handling temporary failures (Quorums and replication)
- Membership and failure detection (Gossip Protocol)

8

Ring Partition

Consistent hashing: The output range of a hash function is treated as a fixed circular space or “ring”. Example: Modulus function.



Replication: An item can be stored in multiple consecutive nodes depending on the degree of replication (N). Here, N = 3.

”**Virtual Nodes**”: Each node can be responsible for more than one virtual node.

9

Core Components

- Tables, Items, Attributes.
  - Item is similar to row, Attribute is similar to column and collection of items is table.
- Primary Keys
  - Which could uniquely identify an item in a table.
- Secondary Indexes.
  - Allows you to bucket the items based on attributes other than primary keys.
- DynamoDB Streams.
  - Is an optional feature, captures a near real time data modifications.

10

DynamoDB Data Model

Pros:

DB = Tables <-- Items <-- Attributes

Don't require schema to work with the data

Have concept of primary key

Attribute = (name, value)

Value can be single or multiple (set).

Cons:

Don't support joins, developers must be aware of foreign key(s).

No concept of embedding or referencing like Mongo.

11

Data Model Example

```
{
    "PersonID": 102,
    "LastName": "Jones",
    "FirstName": "Mary",
    "Address": {
        "Street": "123 Main",
        "City": "Anytown",
        "State": "OH",
        "ZIPCode": 12345
    }
}
```

```
{
    "PersonID": 101,
    "LastName": "Smith",
    "FirstName": "Fred",
    "Phone": "555-4321"
}
```

12

Primary Keys

Must be specified during table creation.

Two types of keys:

- **Partition Key:** Single attribute defined as Primary Key is known as Partition Key. Hash value will determine the physical partition of the cluster where the record will persist.
- **Partition Key and Sort Key:** Referred as *Composite* Primary Key. Two values can have same partition, so Sort key helps to sort the items in a partition. Sort key is used to determine the sort order of 2 records having same partition.

13

Secondary Indexes

Allows us to query data from the table using indexes in addition to primary keys.

- **Global Secondary Index:** An index with a partition key and sort key different from those defined on table.
- **Local Secondary Index:** An index that has ***same*** partition key but ***different*** sort key.

DynamoDB has a limit of 20 global and 5 local secondary indexes.

14

DynamoDB Streams

Is an optional feature that captures near real-time data modification events in the table and in the order that the events occurred.

Each event generates a *stream* record.

EVENTS:

A new item is added to the table, complete item is captured as a stream record.

Item is modified in the table. “Before” and “After” image of item is recorded.

Item is deleted from the table. Entire item is captured as stream record.

15

Features




Performance at Scale

Supports key-value and document style database, 10 trillion req/day or 20 million req/sec (peak).

Microsecond latency.

Serverless

Fully managed by Amazon. No servers to mange, administer or provision. Availability and fault tolerance are built in

Automatically scales to adjust for capacity and maintains performance

Enterprise Ready

Supports ACID properties. Supports encryption by default for all the data.

Point in time recovery from last 35 days and On demand backups and restoration.

Stream

capture a time-ordered sequence of item-level modifications in any DynamoDB table and store this information in a log for up to 24 hours

16

DEMO

CREATE

READ

UPDATE

DELETE

17

Questions?

18