Dynamo

S H A R I N G

Zeyuan Shang

Agenda

01 Introduction

02 System Architecture

03 Limitation & Comparison

04 DynamoDB Demo

05 Q & A

01

Introduction

- Background
- Motivation
- Assumptions & Requirements
- Design Consideration

Background

- Amazon’s e-commerce platform is getting big
- But performance is getting worse…
- A Shopping Cart

01 2. Motivation

Slight Outrage, Significant Impact

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

Scalable

The platform is implemented on top of an infrastructure of **tens of thousands of** servers and network components located in **many datacenters** around the world

Persistent

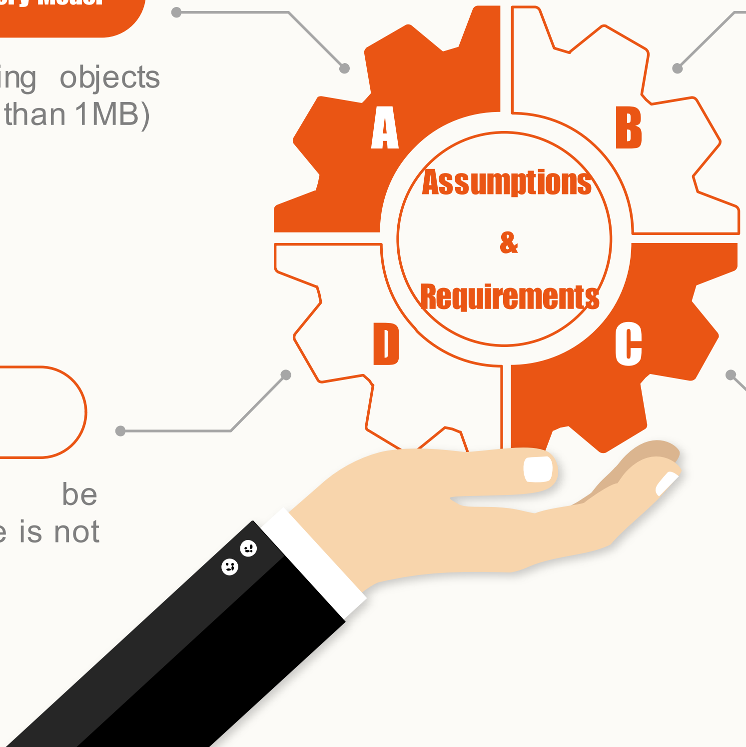
Persistent state is managed in the face of these failures - drives the **reliability** and **scalability** of the software systems

SLA

Guarantee **Service Level Agreements**(SLA): In Amazon's platform, services have stringent latency requirements which are in general measured at the 99.9th percentile of the distribution.

01

3. Assumptions & Requirements



Query Model

Key/value for storing objects with small size(less than 1MB)

Consistency

ACID is bad according to experiences at Amazon, thus Dynamo provides weaker C (eventual consistency)

SLA

99.9%Latency must be bounded, while average is not enough

Security

No security-related problems such as authentication and authorization

01

4. Design Consideration

01 Highly Available

Sacrifice strong consistency for availability

02 Always Writable

Conflict resolution is executed during ***read*** instead of ***write***

03 Scalability

able to scale out easily, with minimal impact

04 Symmetry & Heterogeneity

Same responsibilities and decentralization; adding new nodes with higher capacity without having to upgrade all hosts at once

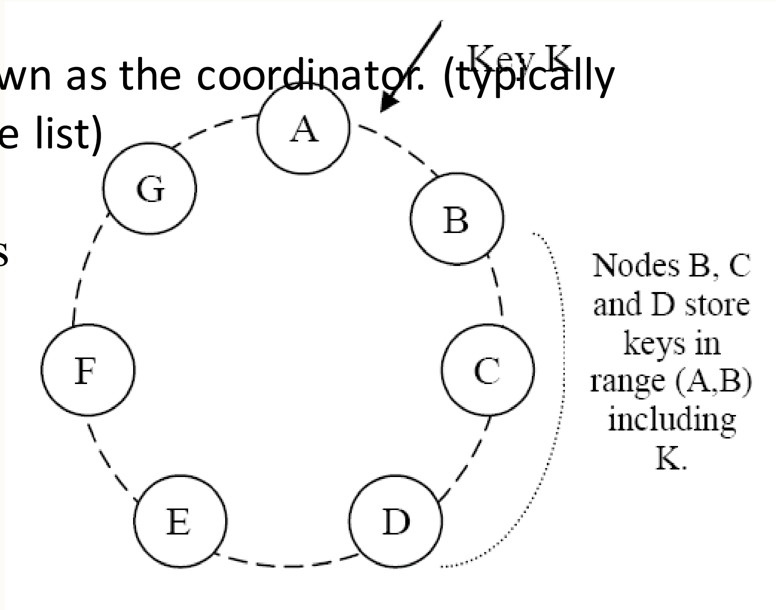
02

System Architecture

- Partitioning
- Replication
- Failure Recovery
- Membership & Failure Detection
- Execution of Put and Get
- Implementation
- Summary

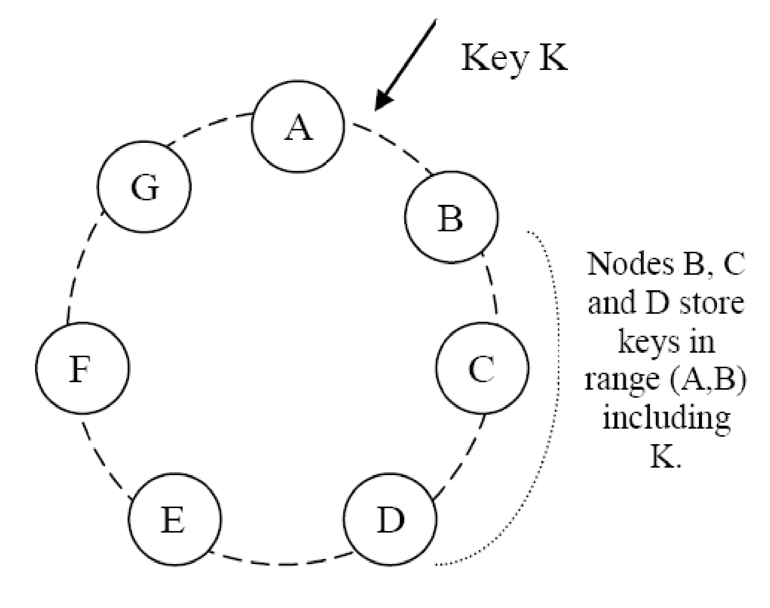
1. Partitioning – Consistent Hashing

- Consistent Hashing: the output range of a hash function is treated as a fixed circular space or “ring”; for example, K mod N
- Each node becomes responsible for the region in the ring between it and its predecessor node on the ring
- A node handling a read or write operation is known as the coordinator. (typically the first among the top N nodes in the preference list)
- Preference List: nodes storing corresponding key
- Arrival/Departure of nodes only affects neighbors



1. Partitioning – Virtual Nodes

- Virtual Nodes: each physical node can be responsible for more than one virtual node.
  - When a node become unavailable…
  - When a node become available again
  - Accounting for heterogeneity



02 2. Replication – Data Versioning

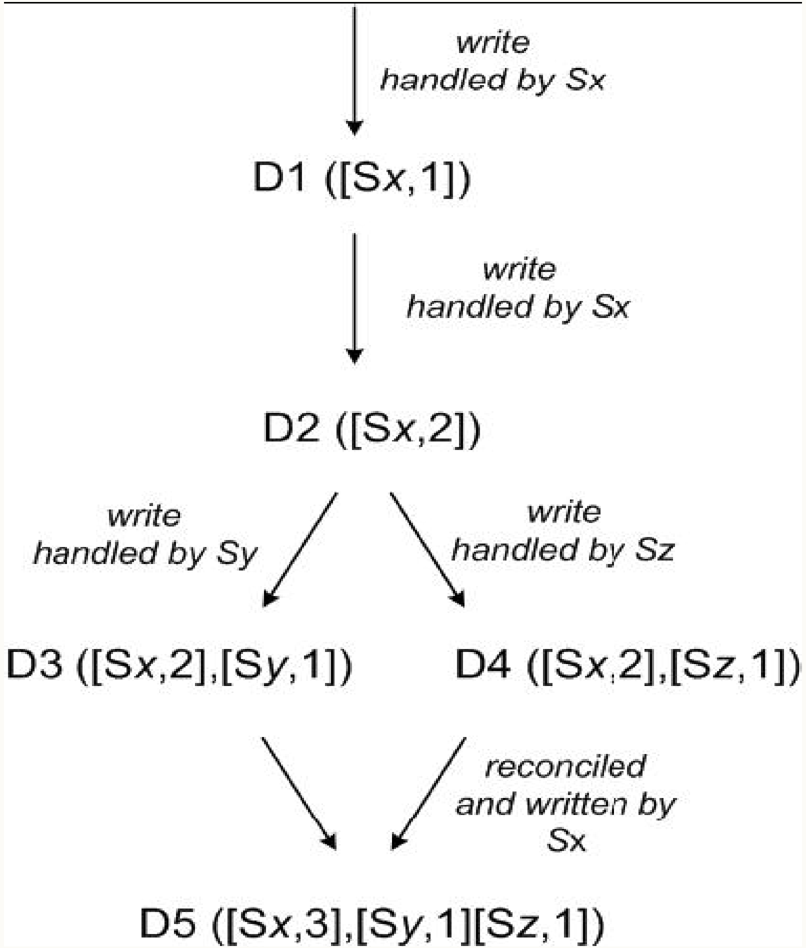
- Eventual Consistency: if no new updates are made to a given data item, eventually all accesses to that item will return the last updated value
- A put() call may return to its caller before the update has been applied at **all** the replicas
- A get() call may return **many** versions of the same object
- Challenge: reconcile different versions
- Example: Shopping Cart
- Solution: using **vector clock** to capture causality between different versions of the same object

02 2. Replication – Vector Clock

- A vector clock is a list of (node, counter) pairs, for example, [(A, 1), (B, 2)]
- Every version of every object is associated with one vector clock
- If the counters on the first object's clock are less-than-or-equal to all of the nodes in the second clock, then the first is an ancestor of the second and can be forgotten, ELSE they are conflict, for example, [(A, 1), (B, 1)] < [(A, 2), (B, 1), (C,1)]
- Example

02

2. Replication – Vector Clock Example

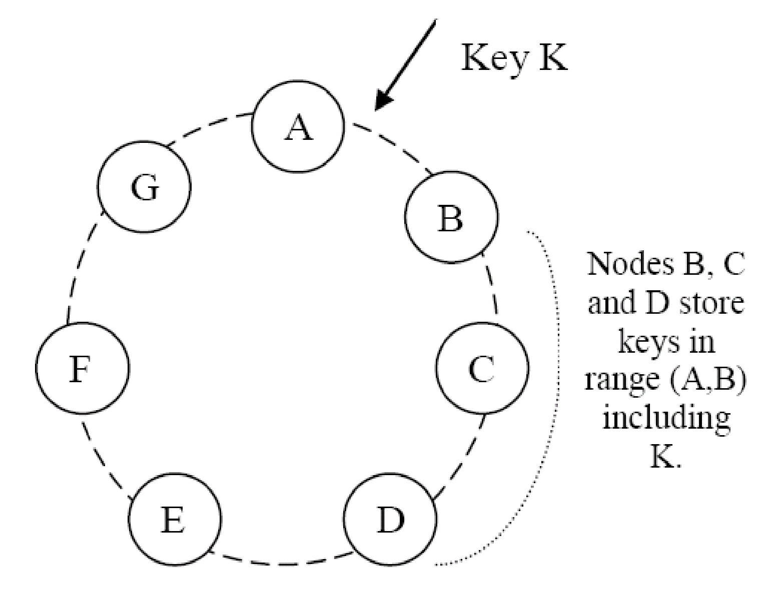


2. Replication – Vector Clock

- In case of network partitions or multiple server failures, the size of vector clock may grow
- Along with each (node, counter) pair, Dynamo uses **timestamp** to remove oldest one when the size is beyond the threshold
- However, this problem has not surfaced in production

3. Failure Recovery – Hinted Handoff

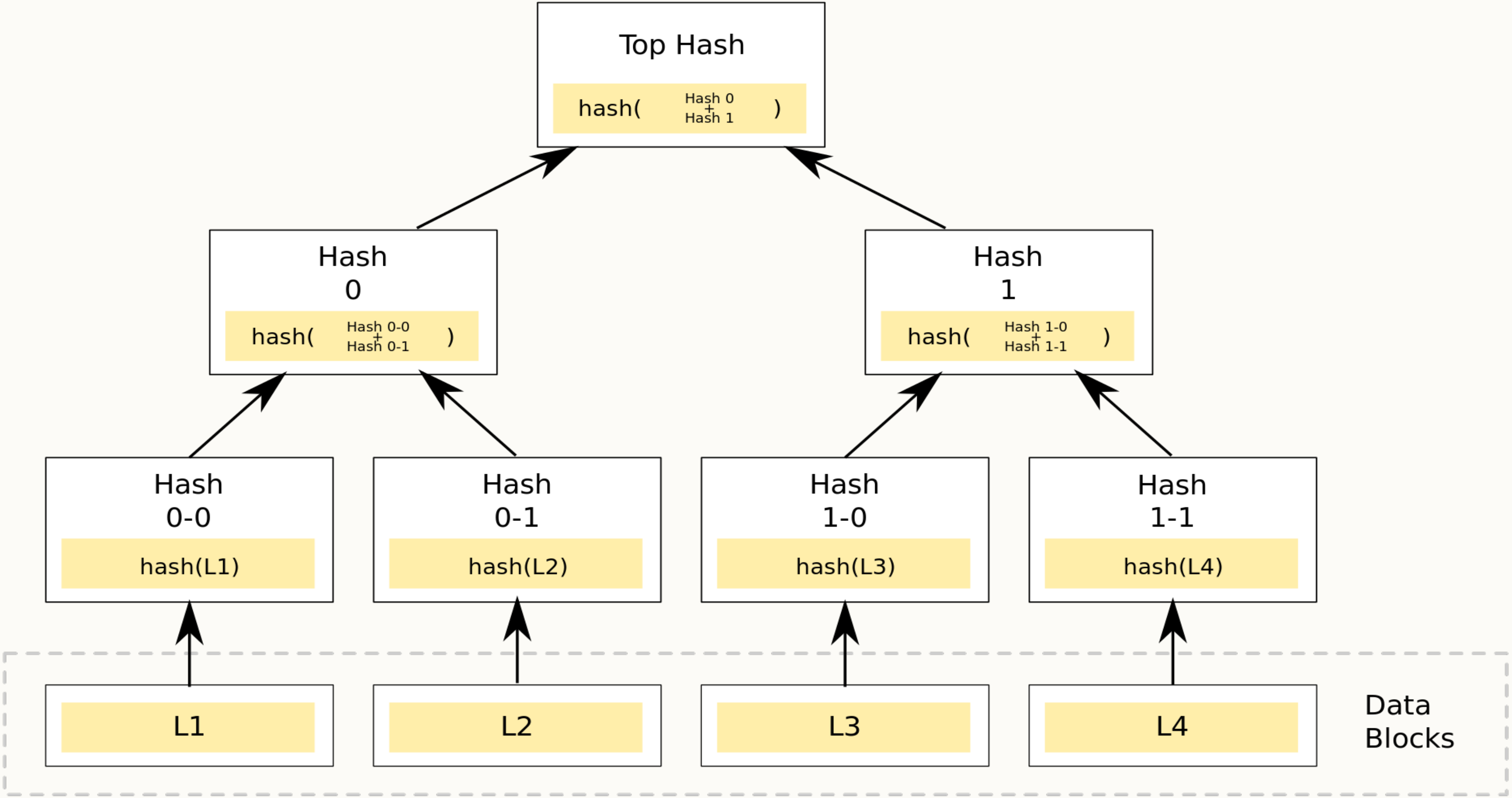
- Assume N = 3. When A is temporarily down or unreachable during a write, send replica to D
- D is hinted that the replica belongs to A and it will deliver to A when A is recovered
- Again, **Always Writable**



3. Failure Recovery – Replication Synchronization

- Merkle tree
    - A hash tree where leaves are hashes of the values of individual keys
    - Parent nodes higher in the tree are hashes of their respective children
- Advantage
    - Each branch of the tree can be checked **independently** without requiring nodes to download the entire tree.
    - Help in **reducing the amount of data** that needs to be transferred while checking for inconsistencies among replicas.
- How Dynamo uses Merkle Tree

02 3. Failure Recovery – Merkle Tree Example



02

4. Membership & Failure Detection

- Gossip-based membership protocol and failure detection
- Won’t cover this in today’s talk

02 5. Execution of Put and Get

- Select a node
  - Route the request through a generic load balancer
  - A partition-aware client library that routes requests **directly** to the appropriate coordinator nodes
- Consistency Protocol: Sloppy Quorum
  - R/W is the minimum number of nodes that must participate in a successful read/write operation
  - Setting R + W > N yields a quorum-like system
  - In this model, the latency of a get (or put) operation is dictated by the slowest of the R (or W) replicas(thus usually less than N)
  - Common (N, R, W) setting is (3, 2, 2)
- Versions in Put/Get

02 5. Execution of Put and Get

- Put
  - Generate new version
  - Send it to other replicas
  - Wait for W - 1 nodes
- Get
  - Request all existing versions
  - Wait for R nodes
  - Return all versions that are not causally related
  - Reconciled by the application then written back

02 Implementation

- Local Persistent and Pluggable Store
  - Berkeley Database (BDB): object of tens of kilobytes
  - MySQL: object of > tens of kilobytes
- All in Java
- Improvement
  - Buffer for read/write
  - Durable write

01 Design Consideration

Highly Available

01

Replication, Vector Clock

Always Writable

02

Vector Clock with reconciliations during reading

Scalability

03

Partition, Hinted Handoff, Sloppy Quorum, Failure Recovery

Symmetry & Heterogeneity

04

Partition, Membership and Failure Detection

02 Summary

Problem	Technique	Advantage
Partitioning	Consistent Hashing	Incremental Scalability
High Availability for writes	Vector clocks with reconciliation during reads	Version size is decoupled from update rates.
Handling temporary failures	Sloppy Quorum and hinted handoff	Provides high availability and durability guarantee when some of the replicas are not available.
Recovering from permanent failures	Anti-entropy using Merkle trees	Synchronizes divergent replicas in the background.
Membership and failure detection	Gossip-based membership protocol and failure detection.	Preserves symmetry and avoids having a centralized registry for storing membership and node liveness information.

03

Comparison & Limitations

- Comparison
- Limitations

Name	DynamoDB	Cassandra	HBase
Description	Hosted, scalable database service by Amazon with the data stored in Amazons cloud	Wide-column store based on ideas of BigTable and DynamoDB	Wide-column store based on Apache Hadoop and on concepts of BigTable
Database model	Document Store	Wide Column Store	Wide Column Store
Cloud-based	yes	no	no
Implementation language	Java	Java	Java
Data scheme	Schema-free	Schema-free	Schema-free
Typing	yes	yes	no
Secondary indexes	yes	Restricted(equality queries)	no
SQL	no	Restricted(CQL)	no
APIs and other access methods	RESTful	CQL, Thrift	Java, RESTful, Thrift
Server-side scripts	no	no	Yes(coprocessor)
Triggers	yes	yes	yes
Partitioning methods	Sharding(consistent hashing)	Sharding(consistent hashing)	Sharding(range partition)
MapReduce	No(only EMR)	yes	yes
Consistency concepts	Eventual Consistency Immediate Consistency	Eventual Consistency Immediate Consistency	Immediate Consistency
Foreign keys	no	no	no
Transaction concepts	no	no	no

02 Limitations

- Object size can not be large
- No cross-region(data center) replication
- Backup process is not very friendly

04

DynamoDB Demo

- Demo

02 DynamoDB Demo

- Demo
- DynamoDB Portal

05

Q & A

- Q & A

05 Q & A

Reference

Reference

[1] DeCandia, Giuseppe, et al. "Dynamo: amazon's highly available key-value store." ACM SIGOPS operating systems review 41.6 (2007): 205-220.

Thank You!

THANK YOU FOR WATCHING