Amazon’s Dynamo: A Deep Dive into High Available Key-Value Stores

Hamza Asif, Dawood Abbas,

Ali Abbas,
Shayan , Faseeh

Motivation

- Build a distributed storage system:
  - Scalability
  - Simple: key-value
  - Performance and low latency
  - High available and durability
  - Security
  - Cost Efficient

System Assumptions and Requirements

- ***Query Model***: simple read and write operations to a data item that is uniquely identified by a key.
- ***ACID Properties****: Atomicity, Consistency, Isolation, Durability.*
- ***Efficiency****:* latency requirements which are in general measured at the 99.9th percentile of the distribution.
- **Other Assumptions**: operation environment is assumed to be non-hostile and there are no security related requirements such as authentication and authorization.

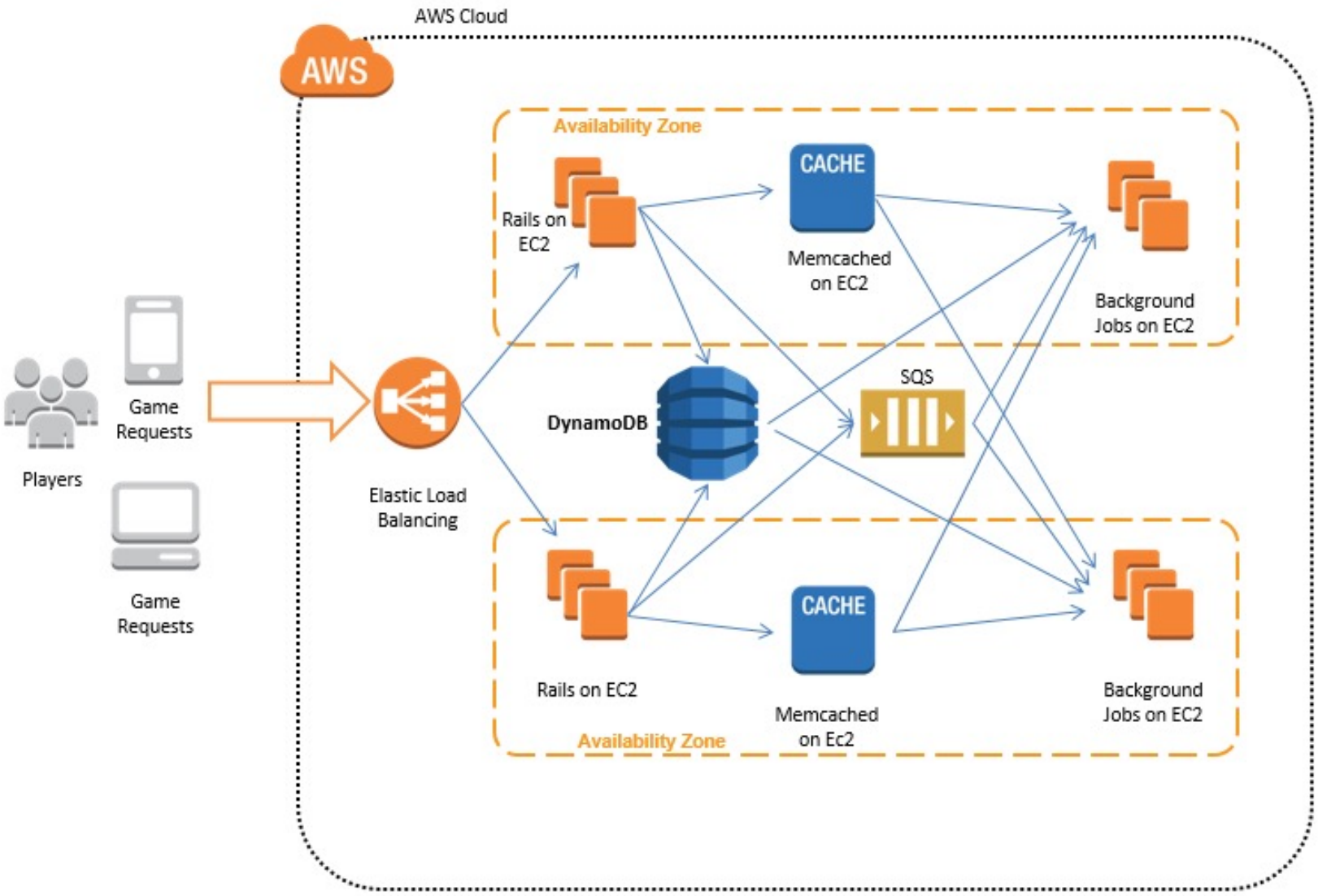
Basic DynamoDB Components

- ***Tables:*** Similar to other database systems, DynamoDB stores data in tables. A *table* is a collection of data
- **Items:** Each table contains zero or more items. An *item* is a group of attributes that is uniquely identifiable among all of the other items
- **Attributes*:*** Each item is composed of one or more attributes. An *attribute* is a fundamental data element, something that does not need to be broken down any further

Primary Key Concept

- **Partition key*:*** A simple primary key, composed of one attribute known as the *partition key*
- **Partition key and sort key***:* Referred to as a *composite primary key*, this type of key is composed of two attributes. The first attribute is the *partition key*, and the second attribute is the *sort key*

DynamoDB Work Flow

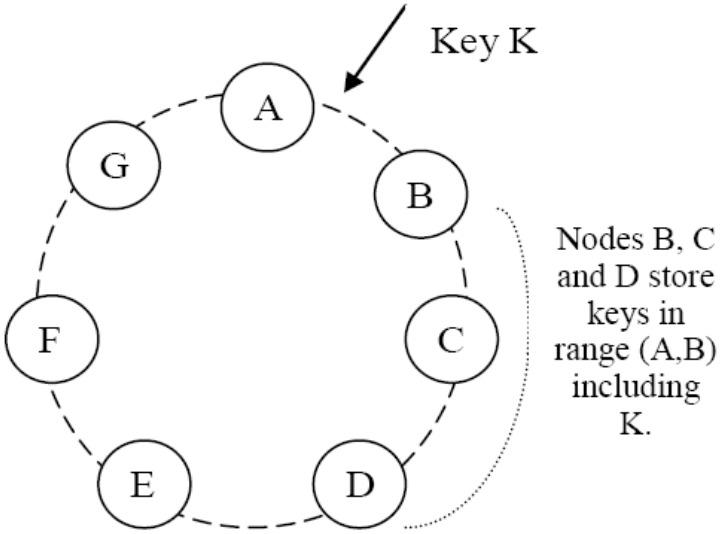


Summary of techniques used in *Dynamo* and their advantages

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.

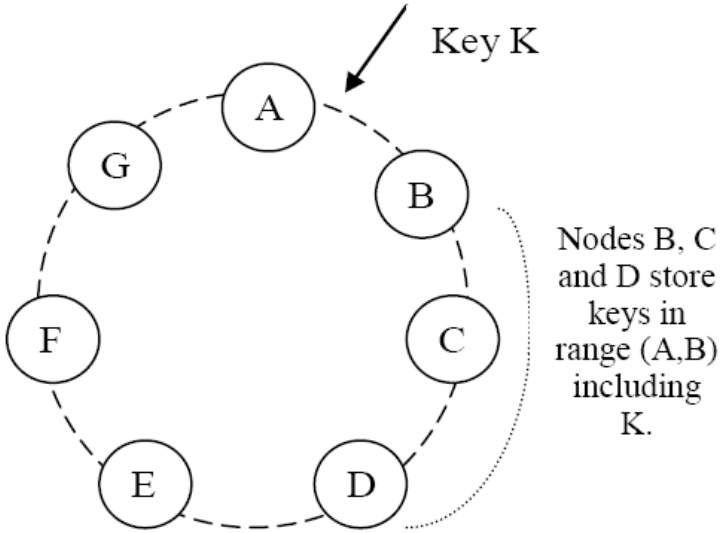
Partition Algorithm

- Consistent hashing: the output range of a hash function is treated as a fixed circular space or “ring”.
- ”Virtual Nodes”: Each node can be responsible for more than one virtual node.

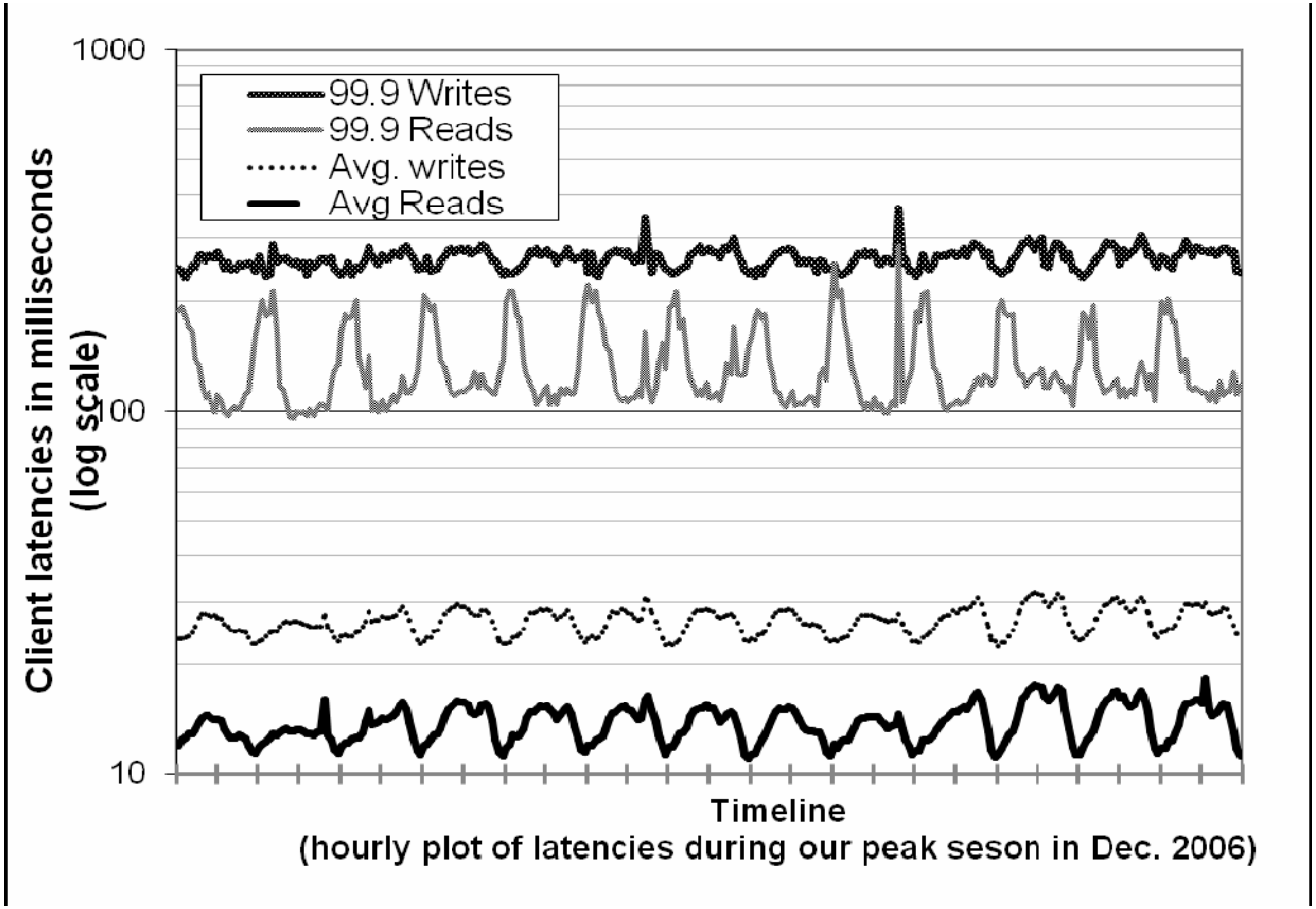


Replication

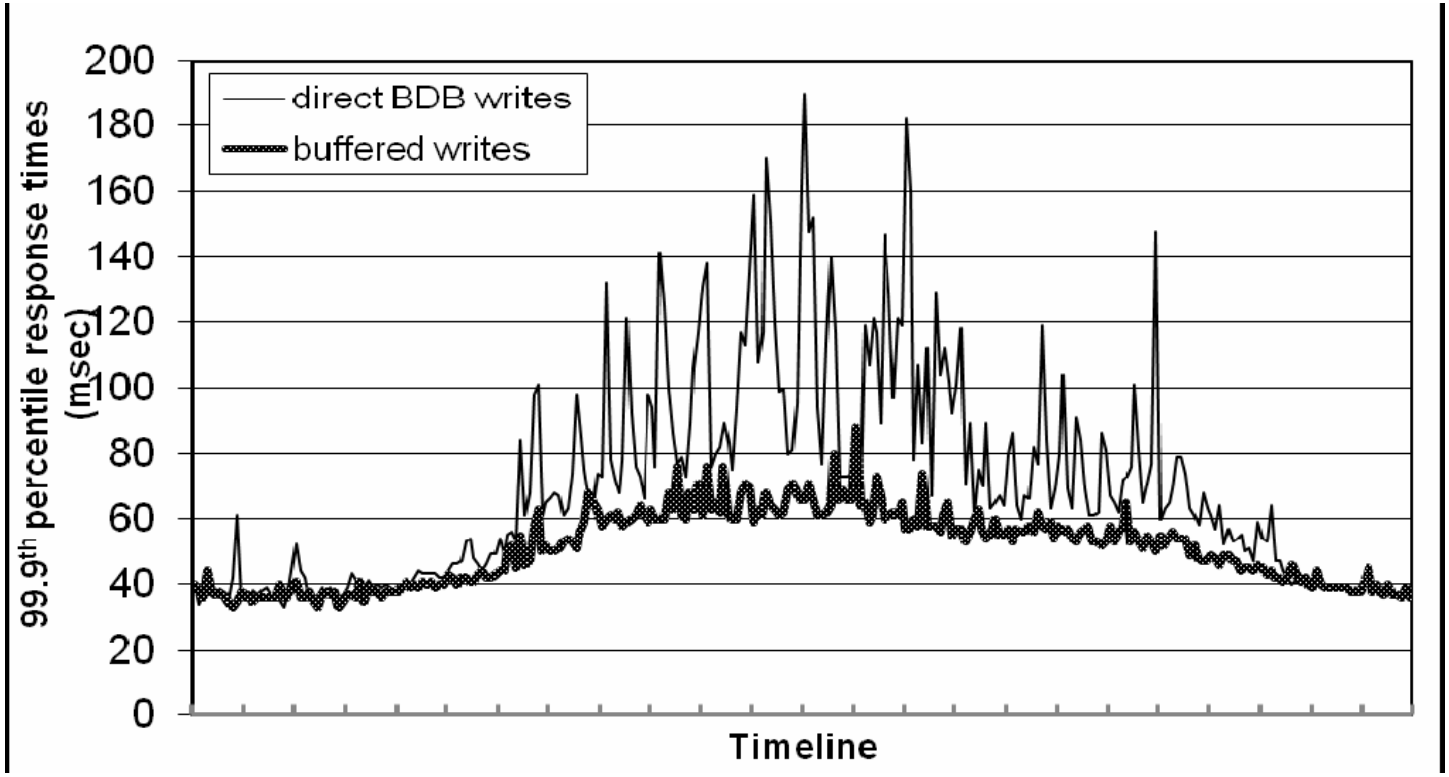
- Each data item is replicated at N hosts.
- "*preference list*": The list of nodes that is responsible for storing a particular key.



Evaluation


1000
100
10
99.9 Writes
99.9 Reads
Avg. writes
Avg Reads
Client latencies in milliseconds
(log scale)
Timeline
(hourly plot of latencies during our peak seson in Dec. 2006)

Evaluation


200
180
160
140
120
100
80
60
40
20
0
direct BDB writes
buffered writes
99.9th percentile response times (msec)
Timeline