DynamoDB

A view under the hood

Summary

- Features
- Key Concepts
- Data Plane
    - GET/PUT
    - Replication details
    - Storage Node details
    - Provisioned Throughput
- Control Plane
    - Health Checks
    - Create Table & Partition
    - Partition Split
    - Balancing

Features

Large Scale

Predictable Performance

Managed DB

Strong Consistency

Atomic Operations

Key Concepts

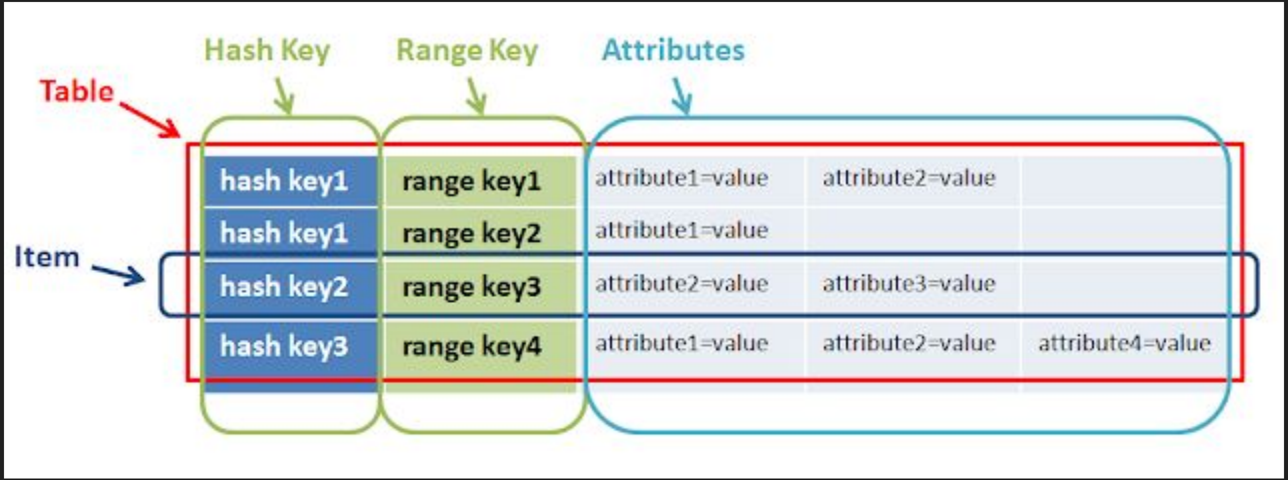
tables, items, and attributes;

primary keys;

read and write capacity;

secondary indexes, (local and global);

Tables, Items, Attributes



hash key1	range key1	attribute1=value	attribute2=value	
hash key1	range key2	attribute1=value		
hash key2	range key3	attribute2=value	attribute3=value	
hash key3	range key4	attribute1=value	attribute2=value	attribute4=value

GET/PUT

Request-Router

- Find servers that host partition for this table & key

Storage-Node

- Find Storage Manager hosting partition
- Admission Control
- Execute Request

Finding the Partition

System Tables (some of them):

- Tables (H:User-Id, R:TableName) - Table-Id, IOPS, State, ...
- Partitions (H: Table-Id, R:Starting-Primary-Key) - Partition-Id, ReplicaSet, ...
- Nodes (H: AvailabilityZone-Type, R: Node-Id) - IPaddr, CapacityIOPS, ...
- Tasks (H: Type/State, R: Task-Type-Specific) - liveness, owner, ...

Finding the Partition

Request Router Steps:

- Query(Tables, …) [cacheable]
- Query(Partitions, …) [cacheable]
- Select Node Order for Replica-Set [master hint]
- Resp = IssueRequest(StorageNode, Req)
- Resp can invalidate/update cache and we start again

PUT on the ReplicaSet

Multi-Paxos implementation

Only leader accepts PUT (or ConsistentGet) requests

only ACKs to RR after AvailabilityZone-aware quorum write of PUT on repl log.

Append Acks from remaining members are implicit leader-re-elections

Possible delay between what replication log holds and what is *applied*

PUT on the ReplicaSet

Applying the replication log to the StorageManager requires an additional LSN.

Replication heartbeat information contains node-ID, partition-ID, apply-LSN, append-LSN, IOPS-free, ...

Heartbeat information is piggybacked on append-acks.

If replicaSet is idle, heartbeats still flow through..

Heartbeats are de-multiplexed to reduce network-chatter between node-pairs that have common replica-sets.

ReplicaSets

ReplicaSet data is versioned.

Authoritative ReplicaSet information is part of the partition and replication log data.

ReplicaSet changes are metadata-PUTs on the partition. After that leader updates Metadata tables as an “anti-entropy” strategy.

Membership changes are internal requests issued to the partition leaders. (ie AddReplica, RemoveReplica, ...)

Storage Node

+12 SSD drives

Isolation of OS + application stack + debug logs on single SSD

Deep integration of OS cache, InnoDB, App. stack (design for 1 seek per req)

Bin-packing partitions onto volumes using IOPS + disk-space

In 2011 the P99.9 latency of SSDs wasn’t good enough to do RAID0 w/ all drives

We had to “shard” the SN into 5 volumes

Provisioned Throughput

This requires extreme stability in the distribution of load across two dimensions:

- Key-space
- Time

This lead to the addition of “Bursting” to DynamoDB, going away from “Provisioned IOPS”.

Can StorageNodes “Burst” IOPS and maintain “predictable performance” ?

.. predictable performance & variable workloads is still innovation space

AutoAdmin Fleet

AutoAdmin Nodes handle:

- *Control Plane* operations (create/delete table/partition, create index..)
- Automated management of the health of the fleet

AutoAdmin Nodes act on *events*

- Events are created for all major operations (create table, AddReplica, ...)
- System evolves and new event *types* are created to increase automation

HEALTH CHECKS

The whole deployment is *deep* health checked

AutoAdmin nodes elect a leader that:

- Assigns nodes table ranges for health-checks to remaining AutoAdmin nodes
- Health-Checks AutoAdmin nodes

On health-check failures, “heal” events are created on “events” table

Create Partition

- Create table
- Partition split
    - change in table provisioned throughput
    - Partition is *too large*

Candidate Storage Nodes to host a partition must be selected based on:

- Availability Zone
- Available IOPS
- Available disk space

2D *live* bin-packing problem, choose best from random subset of nodes.

Partition Split

Partition split is *re-sharding*:

- Key-space divided evenly among child partitions
- IOPS divided evenly also

ReplicaSet metadata-operations:

- AddMember
- Split (leader decides the child ReplicaSets, writes into replication log)
- ReplicaSet commits *harakiri* upon processing Split operations
    - Spawns new ReplicaSets, that inherit the rep log & BTree
    - Further maintenance operations to effectively release physical space

Balancing

Things to balance:

- Storage Space, as partitions grow or split
- Storage IOPS, as partitions grow or split
  - AddMember (which contains Storage Node selection) holds the allocation logic
- Leaders: ideally, each node should be leader of ⅓ of the ReplicaSets
- TCP Connections per Availability Zone and per RequestRouter
- Metadata Server requests to ask about health/balancing.
  - Nodes report back their “wishes”
- Bugs on *Emergent Behaviour..*