Amazon Dynamo DB

No Sql database

* They are non relational database and are distributed
* No database include emongo db and Dynamo DB
* They don’t support join query or limited support
* All the data that is needed for a query is present in one row
* They don’t perform aggregations such as sum avg
* They scale horizontally

Amazon dynamo db

* Fully manage highly available with replation across multiple azs
* Not a relational database
* Scales to massive workloads , distributed database
* Millions of request per seconds and trillions of rows and 100 tb of storage
* Low latency on retrieval
* Integrated with Iam for security and authorization
* Event driven programming with dynamo db streams
* Low cost

Dynamo DB basics

* Is made of tables
* Has primary key decided at cration
* Infinite rows
* Can be null and can be added over time
* Scalar types, document types, set type

Dynamo Db – Primary keys

* Option 1 : Partition key (Hash)
  + Partition key must be unique for each item
  + Diverse so that data is distributed
* Option 2 : Partition Key + Sorty Key (Hash + Range)
  + Must be unique for each item combination
  + Data is grouped by partition key

Table types

* Dynamo Db IA
* Dynamo DB standard

**Dynamo DB read and write capacity mode**

* Provisioned Mode
  + We can specity the number of reads and write per second
  + Plan capacity before hand
  + Pay for capacity units
* On demand mode
  + Automaticy scale up and down
  + No capacity planning needed
  + More expensive

You can switch between them in 24 hours

R/W Capacity Modes: Provisioned

* Must provisioned read and write capacity units
  + RCU
  + WCU
  + Through put can be exceeded temporarily using “Brust Capacity”
  + If brust capacity is exceded we get exception called
    - ProvisionedThrouhputExceededException
  + Then we should do exponential backoff retry

DynamoDB – Write Capacity Units (WCU)

* One write capacity unit (WCU) represents one write per second for an item up to 1 kb in size
* If the items are larger than 1 kb more wcus are consumed

Strongly consistent read VS Eventually consistent read

* Eventually consistent read
  + If you want to read data immediately after, write it might not have been replicated to all servers so there is chance of
    - Data latency
    - Stale Data
* Strongly consistent read
  + What if you want to read data immediately and want it to be accurate
    - It will be expensive as it will consume more RCU

Dynamo DB – Read Capacity Units (RCU)

* 1 RCU represents one **Strongly consistent Read** per second

Or **Two Eventually consistent Read per second** for item up to 4 kb in size

* If the items are larger then 4kb then more RCU are consumed
* 2 eventually consistent read per second = 1 RCU
* Round up is mandatory

**Dynamo DB partitions Internal**

* Data is stored in partitions
* Partition keys go through a hashing algorithm to know which partition they go
* WCU’s and RCU’s are spread evenly across partitions evenly across.

Dynamo DB Throttling

* If we exceed provisioned RCU’s or WCU’s we get “**ProvisionedThroughputExceededException**”
* Reasons
  + Hot keys – one partition key is being read too many times
  + Hot partitions
  + Very large items
* Solutions:
  + Exponential back off when exception is encountered (already in SDK)
  + Distribute partition keys as much as possible
  + We can use DAX ( Dynamo DB accelerator)

R/W Capacity modes – On-Demand read and write

* Based on workload
* No capacity planning needed
* Expensive 2.5 times
* Use for unknown workloads

**Dynamo DB Wrting Data**

Put item

* Creates a new item or fully replace an old item (same primary key)
* Consumes WCUs

UpdateItem

* Edit an existing items attributes or adds a new item if it doesnot exist
* Can be used to implement Atomic Counters – a numeric attribute that’s unconditionally incremented

Conditional Writes

* Accept a write/update/delete only if conditions are met otherwise returns and error
* Help with concurrent access

**Dynamo DB Reading Data**

Get item

* Read based on Primary Key (hash or has +range)
* Eventually consistent or Strongly consistent

Query

* Return items based on key condition expression
  + Partition key value , Sort key value
* Filter Expression
  + Additional filtering based on query operation (before data return)
  + Non key attributes

Scan

* Used to scan entire table and filter out data , inefficient
* Returns up to 1 MB of data
* Consumes lot of RCU
* Limit impact using limit or reduce size of the result and pause
* For faster performance we need to use Parallel Scan