# NoSQL Databases Comparison

*By*
Yashaswi Tamta
Jugal Gandhi
Jonathan McFadden

## What is DynamoDB

- Amazon's fast, scalable, and reliable NoSQL database service for cloud.
- Management of database software and the provisioning of hardware needed to run it
- Helps us to deploy a non-relational database in minutes.

## History of DynamoDB

- DynamoDB is inspired from Dynamo, Amazon's first non relational database [1].
- Dynamo, although was the best technology at the time, but was still a software.

## History of DynamoDB

- Developers and clients preferred simplicity of a web service over the fine-grained control of a software. -- SimpleDB
- But, SimpleDB had it's own issues.

## History of DynamoDB

- Amazon combined the best parts of Original *Dynamo* (incremental scalability, predictable performance) with best parts of *SimpleDB* (ease of administration, consistency)
- Thus, forming DynamoDB'

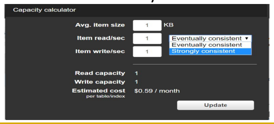
## History of DynamoDB - Evolution

- More flexibility was achieved by adding support for Global secondary indexes and being able to change them on the fly.
- Support for JSON

## Features

- Managed- NoSQL database service
  - With Amazon DynamoDB, AWS provides managed infrastructure.
  - Automatic replication of data over different regions. It enforces data replication across three availability zones for high availability, durability and read consistency. Cross-region replication option is also available.
  - Infinite scalability with data items being stored on solid-state drives, which provide high I/O performance.
  - Data is backed on S3 storage.
  - Pay-per-use model.
  - Security and access control using Amazon's IAM service.

## Features

- Predictable Performance:
  - Delivers highly predictable performance based on the quality of service you choose. You can specify how much provisioned throughput capacity you want to reserve for reads and writes
  - Two types: Strong Consistency (Read-after-Write) or Eventual Consistency.
  - 

Capacity calculator

Avg. item size: 1 KB
Item read/sec: 1 — Eventually consistent ▾ (Eventually consistent / Strongly consistent)
Item write/sec: 1

Read capacity: 1
Write capacity: 1
Estimated cost per table/index: $0.59 / month

Update

## Features

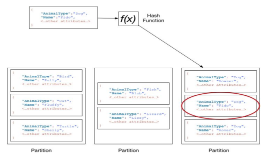
- DynamoDB Data Types:
  - DynamoDB uses three basic data model units: Tables, Items, and Attributes. Tables are collections of Items, and Items are collections of Attributes. Attributes can be of following data-types:
  - Scalar – Number, String, Binary, Boolean, and Null.
  - Multi-valued – String Set, Number Set, and Binary Set.
  - Document – List and Map.

## Features

- Amazon DynamoDB partitions
  - Stores data in partitions.
  - If your table has a simple primary key (partition key only), DynamoDB stores and retrieves each item based on its partition key value.
  - To write an item to the table, DynamoDB uses the value of the partition key as input to an internal hash function. The output value from the hash function determines the partition in which the item will be stored.

## Features

- Amazon DynamoDB partitions


{ "AnimalType":"Dog", "Name":"Fido", <...other attributes...> }
f(x)
Hash Function
{ "AnimalType": "Bird", "Name": "Polly", <...other attributes...> }
{ "AnimalType": "Cat", "Name": "Fluffy", <...other attributes...> }
{ "AnimalType": "Turtle", "Name": "Shelly", <...other attributes...> }
Partition
{ "AnimalType": "Fish", "Name": "Blub", <...other attributes...> }
{ "AnimalType": "Lizard", "Name": "Lizzy", <...other attributes...> }
Partition
{ "AnimalType": "Dog", "Name": "Bowser", <...other attributes...> }
{ "AnimalType": "Dog", "Name": "Fido", <...other attributes...> }
{ "AnimalType": "Dog", "Name": "Rover", <...other attributes...> }
Partition

- A single partition can support a maximum of 3,000 read capacity units or 1,000 write capacity units. When you create a new table, the initial number of partitions can be expressed as follows:

( readCapacityUnits / 3,000 ) + ( writeCapacityUnits / 1,000 ) = initialPartitions (rounded up)

## Features

- DynamoDB Streams and AWS Lambda Triggers
  - A DynamoDB stream is an ordered flow of information about changes to items in an Amazon DynamoDB table. When you enable a stream on a table, DynamoDB captures information about every modification to data items in the table.
  - Amazon DynamoDB is integrated with AWS Lambda so that you can create triggers—pieces of code that automatically respond to events in DynamoDB Streams. With triggers, you can build applications that react to data modifications in DynamoDB tables.
  - AWS Lambda polls the stream and invokes your Lambda function synchronously when it detects new stream records.

## Features

- Other features
  - Amazon DynamoDB integration with Amazon EMR(Elastic Map-Reduce) and Redshift: To perform analysis on datasets.
  - Amazon DynamoDB JavaScript Web Shell: AWS has introduced a web-based user interface known as the DynamoDB JavaScript Shell for local development. When you are ready to deploy your application in production, you can make some minor changes to your code so that it uses the Amazon DynamoDB web service.

## Use-cases

- Use Case 1: Product Catalog
  - To store product information in DynamoDB. Each product has its own distinct attributes, so you will need to store different information about each of these products.
  -

| Table Name | Primary Key |
|---|---|
| ProductCatalog | Partition key: Id (Number) |

## Use-cases

- Use Case 2: Forum Application
  - To build an application for message boards, or discussion forums.
  - Each AWS service has a dedicated forum. Anyone can start a new discussion thread by posting a message in a forum. Each thread might receive any number of replies. You can model this application by creating three tables: Forum, Thread, and Reply.

| Table Name | Primary Key |
|---|---|
| Forum | Partition key: Name (String) |
| Thread | Partition key: ForumName (String)<br>Sort key: Subject (String) |
| Reply | Partition key: Id (String)<br>Sort key: ReplyDateTime (String) |

## Usability

- Easy web based GUI for initial table setup, including:
  - items (*columns*)
  - data-pipelines (*for one-time or scheduled import/export*)
  - permissions
  - provisioning
- All GUI functions are available in the AWS web console and through the AWS SDK.

## Costs

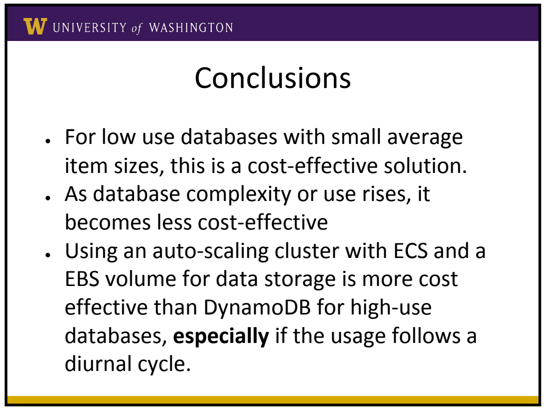
Costs are based on:

- read/sec
- writes/sec
- average item size

Costs are on a sliding scale which is NOT published; however AWS does provide an easy-to-use cost estimator. Additionally, writes are billed at twice the rate of reads.

## Possible Alternatives

- MongoDB SaaS
  - Prohibitively expensive vs. DynamoDB
  - Not easy to use MemCaching or acceleration
- DynamoDB/MongoDB/etc. Container in ECS
  - DynamoDB SaaS removes the trouble with configuring, hosting, and maintaining a container
- Native Install of MongoDB/Cassandra/etc
  - Configure and maintain host system
  - Configure, install, and maintain software

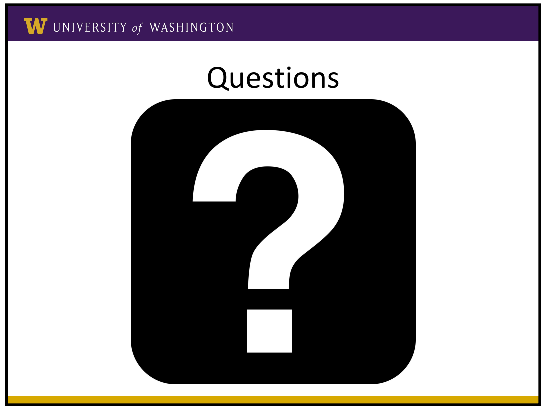

UNIVERSITY *of* WASHINGTON

## Conclusions

- For low use databases with small average item sizes, this is a cost-effective solution.
- As database complexity or use rises, it becomes less cost-effective
- Using an auto-scaling cluster with ECS and a EBS volume for data storage is more cost effective than DynamoDB for high-use databases, **especially** if the usage follows a diurnal cycle.

UNIVERSITY *of* WASHINGTON

## Demo


UNIVERSITY *of* WASHINGTON

## Questions